Index

英文字母/数字开头

A

B

C

D

E

L

M

N

P

Q

S

T

W

X

Y

Z

97

- 字符串的冒泡排序

20

- 最长公共子序列

40

- 最大子阵

3

Data Structure and Algorithms - All Codes

Week 1 20250221

A 两个整数a,b之和

```cpp
// 这题用C，CPP会超时
// https://blog.csdn.net/OOFFrankDura/article/details/79093578

#include <iostream>

using namespace std;

int main(void) {
    int n;
    cin >> n;

    int a, b;
    for (int i = 0; i < n; i++) {
        cin >> a >> b;
        cout << a + b << endl;
    }
}
```

B 斐波那契数列

```cpp
#include <iostream>

using namespace std;

int num(int n);

int main(void) {
    int n;
    cin >> n;
    cout << num(n);
}

int num(int n) {
    if (n == 1) {
```

```
        return 1;
    }
    if (n == 2) {
        return 1;
    }

    int mod = 1000000007;
    int a = 1, b = 1, c;
    for (int i = 3; i <= n; i++) {
        c = (a + b) % mod;
        a = b;
        b = c;
    } // 防超时

    return c;
}
```

C 矩阵旋转

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n, m;
    cin >> n >> m;

    int v[n][m];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> v[i][j];
        }
    }

    for (int i = 0; i < m; i++) {
        for (int j = n - 1; j >= 0; j--) {
            if (j == n - 1) {
                cout << v[j][i];
                continue;
            }
            cout << " " << v[j][i];
        }
        cout << endl;
    }
}
```

D 最大子阵

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int m, n;
    cin >> m >> n;

    vector<vector<int>> v(m, vector<int>(n));
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            cin >> v[i][j];
        }
    }

    for (int i = 0; i < m; i++) {
        for (int j = 1; j < n; j++) {
            v[i][j] += v[i][j - 1];
        }
    }

    long long b[m];
    long long ans = INT_MIN;
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            for (int k = 0; k < m; k++) {
                if (j == 0) {
                    b[k] = v[k][0];
                } else if (i != 0) {
                    b[k] = v[k][j] - v[k][i - 1];
                } else if (i == 0){
                    b[k] = v[k][j];
                }
            }

            vector<long long> dp(m, b[0]);
            long long mmax = dp[0];
            for (int k = 1; k < m; k++) {
                dp[k] = max(dp[k - 1] + b[k], b[k]);
                mmax = max(mmax, dp[k]);
            }

            ans = max(mmax, ans);
        }
```

```
    }

    cout << ans << endl;
}
```

E 四平方和

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n;
    cin >> n;
    for (int a = 0; a * a <= n; a++) { // 没什么好说的，暴力枚举
        for (int b = a; b * b <= n - a * a; b++) {
            for (int c = b; c * c <= n - a * a - b * b; c++) {
                double d = sqrt(n - a * a - b * b - c * c);
                if (!(d - (int)d)) {
                    cout << a << " " << b << " " << c << " " << (int)d << " " << endl;
                    return 0;
                }
            }
        }
    }
}
```

F A*B问题

```cpp
#include <bits/stdc++.h>

using namespace std;

vector<int> multi(vector<int> A, vector<int> B) {
    vector<int> C(A.size() + B.size(), 0);
    for (int i = 0; i < A.size(); i++) {
        for (int j = 0; j < B.size(); j++) {
            C[i + j] += A[i] * B[j];
        }
    }

    int t = 0;
    for (int i = 0; i < C.size(); i++) {
        t += C[i];
```

```
        C[i] = t % 10;
        t /= 10;
    }

    while (C.size() > 1 && C.back() == 0) {
        C.pop_back();
    }

    return C;
}

int main(void) {
    string a, b;
    cin >> a >> b;

    vector<int> A, B;
    for (int i = a.size() - 1; i >= 0; i--) {
        A.push_back(a[i] - '0');
    }
    for (int i = b.size() - 1; i >= 0; i--) {
        B.push_back(b[i] - '0');
    }

    vector<int> C = multi(A, B);

    for (int i = C.size() - 1; i >= 0; i--) {
        cout << C[i];
    }
    cout << endl;
}
```

G 得到整数 X

```
// 二进制枚举？
// 实际上也是动态规划
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n, x;
    cin >> n >> x;
    vector<int> v(n);

    for (int i = 0; i < n; i++) {
        cin >> v[i];
```

```
    }

    vector<int> dp(x + 1, 0);
    dp[0] = 1;
    for (int i = 0; i < n; i++) {
        for (int j = x; j >= v[i]; j--) {
            dp[j] += dp[j - v[i]];
        }
    }

    cout << dp[x];
}
```

Week 2 20250228

A 打印锯齿矩阵（STL）

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n, m;
    cin >> n >> m;
    vector<vector<int>> v(n);
    for (int i = 0; i < m; i++) {
        int a, b;
        cin >> a >> b;
        v[a - 1].push_back(b);
    }

    for (int i = 0; i < n; i++) {
        if (v[i].size() == 0) {
            cout << " " << endl;
            continue;
        }

        for (int j = 0; j < v[i].size(); j++) {
            cout << v[i][j] << " ";
        }
        cout << endl;
    }
}
```

C 计算集合的并集（STL）

```cpp
#include <bits/stdc++.h>

using namespace std;

bool search(vector<int> v, int target) {
    for (int i = 0; i < v.size(); i++) {
        if (v[i] == target) {
            return true;
        }
    }

    return false;
}

int main(void) {
    int n, m;
    cin >> n >> m;
    vector<int> v1(n);
    vector<int> v2(m);

    for (int i = 0; i < n; i++) {
        cin >> v1[i];
    }
    for (int i = 0; i < m; i++) {
        cin >> v2[i];
    }

    for (int i = 0; i < m; i++) {
        if (search(v1, v2[i])) {
            continue;
        } else {
            v1.push_back(v2[i]);
        }
    }

    sort(v1.begin(), v1.end());
    for (int i = 0; i < v1.size(); i++) {
        cout << v1[i] << " ";
    }
    cout << endl;
}
```

D 小明学英语（STL）

```cpp
#include <bits/stdc++.h>

using namespace std;

bool search(vector<string> v, string target) {
    for (int i = 0; i < v.size(); i++) {
        if (v[i] == target) {
            return true;
        }
    }
    return false;
}

int main(void) {
    int n;
    cin >> n;
    vector<string> v1;

    for (int i = 0; i < n; i++) {
        int status;
        cin >> status;
        if (status == 0) {
            string name;
            cin >> name;
            transform(name.begin(), name.end(), name.begin(), ::tolower);
// 转为小写
            v1.push_back(name);
        } else {
            string name;
            cin >> name;
            transform(name.begin(), name.end(), name.begin(), ::tolower);
            if (search(v1, name)) {
                cout << "Yes" << endl;
            } else {
                cout << "No" << endl;
            }
        }
    }
}
```

J islands 打炉石传说（二进制枚举）

```cpp
#include <bits/stdc++.h>

using namespace std;

```

```cpp
struct Card {
    int cost;
    bool d;
    int w;
};

int main(void) {
    int n;
    cin >> n;

    Card cards[n];
    for (int i = 0; i < n; i++) {
        cin >> cards[i].cost >> cards[i].d >> cards[i].w;
    }

    vector<int> dp(11, 0); // i点法力值时的最大攻击力
    for (int i = 0; i < n; i++) {
        if (!cards[i].d) {
            for (int j = 10; j >= cards[i].cost; j--) {
                dp[j] = max(dp[j], dp[j - cards[i].cost] + cards[i].w);
            }
        }
    }

    for (int i = 0; i < n; i++) {
        if (cards[i].d) {
            for (int j = 10; j >= cards[i].cost; j--) {
                dp[j] = max(dp[j], dp[j - cards[i].cost] + cards[i].w);
            }
        }
    }

    int max = INT_MIN;
    for (int i = 0; i < 11; i++) {
        if (dp[i] > max) {
            max = dp[i];
        }
    }

    cout << max << endl;
}
```

Q 算法提高 选择排序

```cpp
#include <bits/stdc++.h>

```

```cpp
using namespace std;

int main(void) {
    int n;
    cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i];
    }

    bool sorted = false;
    int index = 0;
    while (!sorted) {
        int a = v[index];
        int minNum = INT_MAX;
        int minIndex;
        for (int i = index; i < n; i++) {
            if (v[i] < minNum) {
                minNum = v[i];
                minIndex = i;
            }
        }
        if (minNum == a) {
            cout << "swap(a[" << index << "], a[" << index << "]):";
            for (int i = 0; i < n; i++) {
                cout << v[i] << " ";
            }
            cout << endl;
            index++;
        } else {
            swap(v[index], v[minIndex]);
            cout << "swap(a[" << index << "], a[" << minIndex << "]):";
            for (int i = 0; i < n; i++) {
                cout << v[i] << " ";
            }
            cout << endl;
            index++;
        }
        if (index == n) {
            sorted = true;
        }
    }
}
```

R 得到整数 X

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n, m;
    cin >> n >> m;
    vector<int> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i];
    }

    vector<int> dp(m + 1, 0);
    dp[0] = 1;
    for (int i = 0; i < n; i++) {
        for (int j = m; j >= v[i]; j--) {
            dp[j] += dp[j - v[i]];
        }
    }

    cout << dp[m] << endl;
}
```

Week 3 20250308

A 汉诺塔

```cpp
#include<iostream>
using namespace std;

long long moveCount(int n) {
    if (n == 1) {
        return 1;
    }
    return 1 + 2 * moveCount(n - 1);
}

long long strengthCost(int n) {
    if (n == 1) {
        return 1;
    }
    return n + 2 * strengthCost(n - 1);
}
```

```cpp

int main() {
    int n;
    cin >> n;
    cout << moveCount(n) << " " << strengthCost(n);
    return 0;
}
```

B 踏青

```cpp
#include<iostream>
using namespace std;

int px[4] = {-1, 1, 0, 0};
int py[4] = {0, 0, -1, 1};
char map[100][100];
int vst[100][100] = { 0 };
int n, m;

void grass(int x, int y) {
    vst[x][y] = 1;
    for (int i = 0; i < 4; i++) {
        int nx = x + px[i];
        int ny = y + py[i];

        if (nx >= 0 && nx < n && ny >= 0 && ny < m && vst[nx][ny] == 0 && map[nx][ny] == '#') {
            grass(nx, ny);
        }
    }
}

int main() {
    cin >> n >> m;
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> map[i][j];
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (map[i][j] == '#' && vst[i][j] == 0) {
                grass(i, j);
                count++;
            }
```

```
        }
    }
    cout << count;
    return 0;
}
```

C 金字塔数独

```
#include<iostream>
using namespace std;

const int N = 9;
int grid[N][N];
int scores[N][N] = {
    {6, 6, 6, 6, 6, 6, 6, 6, 6},
    {6, 7, 7, 7, 7, 7, 7, 7, 6},
    {6, 7, 8, 8, 8, 8, 8, 7, 6},
    {6, 7, 8, 9, 9, 9, 8, 7, 6},
    {6, 7, 8, 9, 10, 9, 8, 7, 6},
    {6, 7, 8, 9, 9, 9, 8, 7, 6},
    {6, 7, 8, 8, 8, 8, 8, 7, 6},
    {6, 7, 7, 7, 7, 7, 7, 7, 6},
    {6, 6, 6, 6, 6, 6, 6, 6, 6}
};
int maxScore = -1;

bool isValid(int x, int y, int num) {
    for (int i = 0; i < N; i++) {
        if (grid[i][y] == num || grid[x][i] == num) {
            return false;
        }
    }

    int rowStart = 3 * (x / 3);
    int colStart = 3 * (y / 3);
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (grid[rowStart + i][colStart + j] == num) {
                return false;
            }
        }
    }

    return true;
}

```

```cpp
void solve(int x, int y, int totalScore) {
    if (x == N) {
        maxScore = max(maxScore, totalScore);
        return;
    }

    int nx = y == N - 1 ? x + 1 : x;
    int ny = y == N - 1 ? 0 : y + 1;
    if (grid[x][y] == 0) {
        for (int i = 1; i < 10; i++) {
            if (!isValid(x, y, i)) {
                continue;
            }
            grid[x][y] = i;
            solve(nx, ny, totalScore + i * scores[x][y]);
        }
        grid[x][y] = 0;
    }
    else {
        solve(nx, ny, totalScore + grid[x][y] * scores[x][y]);
    }
}

int main() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cin >> grid[i][j];
        }
    }
    solve(0, 0, 0);
    if (maxScore == 0) {
        cout << -1 << endl;
    }
    else {
        cout << maxScore << endl;
    }
    return 0;
}

```

D 水果店（STL）

```cpp
#include <bits/stdc++.h>

using namespace std;

```

```cpp
int main(void) {
    int n;
    cin >> n;
    map<string, map<string, int>> m;
    for (int i = 0; i < n; i++) {
        string origin;
        string fruit;
        int sales;
        cin >> fruit >> origin >> sales;
        m[origin][fruit] += sales;
    }

    for (auto& p1 : m) {
        cout << p1.first << endl;
        for (auto& p2 : p1.second) {
            cout << "   |----" << p2.first << '(' << p2.second << ')' << endl;
        }
    }
}
```

E 小明面试（STL）

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n;
    cin >> n;
    unordered_map<int, int> m;
    for (int i = 0; i < n; i++) {
        int a;
        cin >> a;
        if (m.find(a) != m.end()) {
            m[a]++;
        } else {
            m.insert({a, 1});
        }
    }

    int imax = INT_MIN;
    for (auto& [key, value] : m) { // 注意迭代器
        imax = max(value, imax);
    }

```

```
    int mmax = INT_MIN;
    for (auto& [key, value] : m) {
        if (value == imax) {
            mmax = max(mmax, key);
        }
    }
    cout << mmax << ' ' << imax << endl;
}
```

F 网页跳转

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n;
    cin >> n;
    stack<string> backStack, forwardStack;
    string current = "Ignore";
    for (int i = 0; i < n; i++) {
        string action;
        cin >> action;
        if (action == "VISIT") {
            string url;
            cin >> url;
            cout << url << endl;
            if (current != "Ignore") {
                backStack.push(current);
            }
            current = url;
            while (!forwardStack.empty()) {
                forwardStack.pop();
            }
        } else if (action == "BACK") {
            if (backStack.empty()) {
                cout << "Ignore" << endl;
            } else {
                forwardStack.push(current);
                current = backStack.top();
                backStack.pop();
                cout << current << endl;

            }
        } else if (action == "FORWARD") {
            if (forwardStack.empty()) {
```

```
                cout << "Ignore" << endl;
            } else {
                backStack.push(current);
                current = forwardStack.top();
                forwardStack.pop();
                cout << current << endl;
            }
        }
    }
}
```

G 幼儿园买玩具（二进制枚举）

```
#include <iostream>
using namespace std;

int main() {
    int n, m, k;
    int result = 0;
    int list[101][16];
    cin >> n >> m >> k;

    for (int i = 0; i < n; i++) {
        cin >> list[i][0];
        for (int j = 1; j <= list[i][0]; j++) {
            cin >> list[i][j];
        }
    }

    for (int i = 0; i < (1 << k); i++) {
        int flag[16] = {0};
        int count = 0;
        int num = 0;
        for (int j = 0; j < k; j++) {
            if (i & (1 << j)) {
                num++;
                flag[j + 1] = 1;
            }
        }
        for (int x = 0; x < n; x++) {
            bool a = true;
            for (int y = 1; y <= list[x][0]; y++) {
                if (flag[list[x][y]] == 0) {
                    a = false;
                }
            }
```

```
            if (a)
                count++;
        }
        if (count > result && num <= m) {
            result = count;
        }
    }

    cout << result;
    return 0;
}
```

H 约瑟夫环问题（循环链表）

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n, m;
    cin >> n >> m;
    list<int> l;
    for (int i = 1; i <= n; i++) {
        l.push_back(i);
    }

    auto it = l.begin();
    while (l.size() > 1) {
        for (int i = 1; i < m; i++) {
            it++;
            if (it == l.end()) {
                it = l.begin();
            }
        }

        cout << *it << " ";
        it = l.erase(it);
        if (it == l.end()) {
            it = l.begin();
        }
    }
    cout << *l.begin() << endl;
}
```

I n个最小和（STL）

```cpp
#include <bits/stdc++.h>

using namespace std;

struct Node {
    int sum, i, j;
    bool operator>(const Node &other) const {
        return sum > other.sum;
    }
};

vector<int> kSmallestSums(vector<int> &A, vector<int> &B, int n) {
    sort(A.begin(), A.end());
    sort(B.begin(), B.end());

    priority_queue<Node, vector<Node>, greater<Node>> minHeap;
    set<pair<int, int>> visited;

    minHeap.push({A[0] + B[0], 0, 0});
    visited.insert({0, 0});

    vector<int> result;

    while (n--) {
        Node cur = minHeap.top();
        minHeap.pop();
        result.push_back(cur.sum);

        int i = cur.i, j = cur.j;

        if (i + 1 < A.size() && !visited.count({i + 1, j})) {
            minHeap.push({A[i + 1] + B[j], i + 1, j});
            visited.insert({i + 1, j});
        }

        if (j + 1 < B.size() && !visited.count({i, j + 1})) {
            minHeap.push({A[i] + B[j + 1], i, j + 1});
            visited.insert({i, j + 1});
        }
    }

    return result;
}

int main() {
    int n;
```

```
    cin >> n;
    vector<int> A(n), B(n);

    for (int i = 0; i < n; i++) cin >> A[i];
    for (int i = 0; i < n; i++) cin >> B[i];

    vector<int> result = kSmallestSums(A, B, n);

    for (int i = 0; i < result.size(); i++) {
        if (i > 0) cout << " ";
        cout << result[i];
    }
    cout << endl;

    return 0;
}
```

J 字符串的冒泡排序

```
#include <bits/stdc++.h>
using namespace std;

vector<int> bubbleSortTransform(string sh, const string &ch) {
    int n = sh.size();
    vector<int> steps;

    unordered_map<char, int> countSh, countCh;
    for (char c : sh) {
        countSh[c]++;
    }
    for (char c : ch) {
        countCh[c]++;
    }
    if (countSh != countCh) {
        return {-1};
    }

    for (int i = 0; i < n; i++) {
        if (sh[i] == ch[i]) {
            continue;
        }

        int pos = i;
        while (sh[pos] != ch[i]) pos++;

        for (int j = pos; j > i; j--) {
```

```
            swap(sh[j], sh[j - 1]);
            steps.push_back(j);
        }
    }

    steps.insert(steps.begin(), steps.size());
    return steps;
}

int main() {
    int n;
    string sh, ch;
    cin >> n >> sh >> ch;

    vector<int> result = bubbleSortTransform(sh, ch);

    if (result[0] == -1) {
        cout << -1 << endl;
    } else {
        for (int i = 0; i < result.size(); i++) {
            if (i > 0) cout << " ";
            cout << result[i];
        }
        cout << endl;
    }
    return 0;
}
```

K 插入排序

```
#include <bits/stdc++.h>
#define MAXN 8010
using namespace std;

struct node {
    int val;
    int num;
    bool operator>(const node b) const {
        if (this->val != b.val) return this->val > b.val;
        return this->num > b.num;
    }
    bool operator<(const node b) const {
        if (this->val != b.val) return this->val < b.val;
        return this->num < b.num;
    }
};
```

```cpp

node a[MAXN];
int n, q, type, x, v;
int order[MAXN];

void get_order() {
    for (int i = 1; i <= n; i++) {
        order[a[i].num] = i;
    }
}

void update(int x, int v) {
    a[order[x]].val = v;
    for (int i = order[x]; i < n; i++) {
        if (a[i] > a[i + 1]) {
            swap(a[i], a[i + 1]);
        }
    }
    for (int i = order[x]; i > 1; i--) {
        if (a[i] < a[i - 1]) {
            swap(a[i], a[i - 1]);
        }
    }
    get_order();
}

int main() {
    freopen("sort.in", "r", stdin);
    freopen("sort.out", "w", stdout);
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i].val);
        a[i].num = i;
    }
    sort(a + 1, a + n + 1);
    get_order();
    while (q--) {
        scanf("%d", &type);
        if (type == 1) {
            scanf("%d%d", &x, &v);
            update(x, v);
        } else {
            scanf("%d", &x);
            printf("%d\n", order[x]);
        }
    }
```

```
    return 0;
}
```

Week 4 20250314

A 一维坐标的移动

```cpp
#include <bits/stdc++.h>

using namespace std;

int minSteps(int n, int start, int end) {
    if (start == end)
        return 0;

    vector<bool> visited(n + 1, false);
    queue<pair<int, int>> q;
    q.push({start, 0});

    while (!q.empty()) {
        int pos = q.front().first;
        int steps = q.front().second;
        q.pop();

        if (pos == end) return steps;

        if (pos + 1 <= n && !visited[pos + 1]) {
            q.push({pos + 1, steps + 1});
            visited[pos + 1] = true;
        }
        if (pos - 1 >= 0 && !visited[pos - 1]) {
            q.push({pos - 1, steps + 1});
            visited[pos - 1] = true;
        }
        if (pos * 2 <= n && !visited[pos * 2]) {
            q.push({pos * 2, steps + 1});
            visited[pos * 2] = true;
        }
    }

    return -1;
}

```

```cpp
int main() {
    int n, start, end;
    cin >> n >> start >> end;

    int result = minSteps(n, start, end);
    cout << result << endl;

    return 0;
}
```

B 多机调度问题

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
    int n, m;
    vector<int> taxt;
    cin >> n >> m;
    vector<int> dp(m);

    for (int i = 0; i < n; i++) {
        int tmp;
        cin >> tmp;
        taxt.push_back(tmp);
    }
    sort(taxt.begin(), taxt.end(), greater<int>());
    if (n <= m) {
        cout << taxt[n - 1];
        return 0;
    }
    else {
        for (int j = 0; j < m; j++) {
            dp[j] = taxt[j];
        }
        for (int j = m; j < n; j++) {
            sort(dp.begin(), dp.end());
            dp[0] += taxt[j];
        }
    }

    sort(dp.begin(), dp.end(), greater<int>());
    cout << dp[0];
    return 0;
}
```

C 迷瘴

```cpp
#include <cstdio>
#include <bits/stdc++.h>
using namespace std;

int main() {
    int v, n, i;
    double w, sum = 0, p = 0, an[105] = {0};
    scanf("%d%d%lf", &n, &v, &w);
    for (i = 0; i < n; i++) {
        scanf("%lf", &an[i]);
    }
    sort(an, an + n);
    for (i = 0; i < n; i++) {
        sum += an[i];
        if (sum / (i + 1) <= w) {
            p = sum / (i + 1);
        } else {
            break;
        }
    }
    if (i == 0)
        printf("0 0.00\n");
    else
        printf("%d %.2f\n", i * v, p / 100);
    return 0;
}
```

D 活动安排

```cpp
#include <bits/stdc++.h>

using namespace std;

struct Program {
    int start;
    int end;
};

bool cmp(Program a, Program b) {
    return a.end < b.end;
}

int main() {
    int n;
```

```
    cin >> n;
    vector<Program> programs(n);

    for (int i = 0; i < n; i++) {
        cin >> programs[i].start >> programs[i].end;
    }
    sort(programs.begin(), programs.end(), cmp);

    int count = 1;
    int tmp = programs[0].end;
    for (int i = 1; i < n; i++) {
        if (programs[i].start >= tmp) {
            tmp = programs[i].end;
            count++;
        }
    }

    cout << count;
    return 0;
}
```

E 逆序对

```
#include<iostream>
using namespace std;

int nums[100000];
long long reverseCount = 0;

void merge(int sta, int mid, int end) {
    int tmp[100000];
    int i = sta;
    int j = mid + 1;
    int k = sta;

    while (i <= mid && j <= end) {
        if (nums[i] > nums[j]) {
            reverseCount += mid + 1 - i;
            tmp[k++] = nums[j++];
        }
        else {
            tmp[k++] = nums[i++];
        }
    }
    while (i <= mid) {
        tmp[k++] = nums[i++];
```

```
    }
    while (j <= end) {
        tmp[k++] = nums[j++];
    }

    for (int i = sta; i <= end; i++) {
        nums[i] = tmp[i];
    }
}

void mergeSort(int sta, int end) {
    if (sta < end) {
        int mid = sta + (end - sta) / 2;
        mergeSort(sta, mid);
        mergeSort(mid + 1, end);
        merge(sta, mid, end);
    }
}

int main() {
    int n, k;
    cin >> n >> k;
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
    }

    mergeSort(0, n - 1);
    cout << max((long long)0, reverseCount - k);
    return 0;
}
```

G Intervals

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;

struct Interval {
    int start;
    int end;
};

bool cmp(Interval a, Interval b) {
    return a.start < b.start;
}
```

```cpp

int main() {
    int n;
    int count = 0;
    cin >> n;
    vector<Interval> intervals(n);
    vector<Interval> result;

    for (int i = 0; i < n; i++) {
        cin >> intervals[i].start >> intervals[i].end;
    }
    sort(intervals.begin(), intervals.end(), cmp);

    result.push_back(intervals[0]);
    for (int i = 1; i < n; i++) {
        if (intervals[i].start <= result[count].end) {
            result[count].end = max(intervals[i].end, result[count].end);
        }
        else {
            result.push_back(intervals[i]);
            count++;
        }
    }

    for (int i = 0; i <= count; i++) {
        cout << result[i].start << " " << result[i].end << endl;
    }
    return 0;
}
```

H 逃跑

```cpp
#include<iostream>
#include<vector>
#include<queue>
#include<bitset>
using namespace std;

const int N = 110;
int dx[] = { 0, 0, 1, -1, 0 };
int dy[] = { 1, -1, 0, 0, 0 };
int pretime = -1;
int n, m, k, d;
bitset<N> vis[N];
bitset<N> sold[N];
bitset<N> go[N][N];
```

```cpp

struct Role {
    int x;
    int y;
    int time;
    int d;
};

struct Soldier {
    char c;
    int t;
    int v;
    int x;
    int y;
}soldiers[N];

struct Bullet {
    int x;
    int y;
    int v;
    char c;
};

queue<Role> roles;
vector<Bullet> bullets;

bool isValid(int x, int y) {
    return x >= 0 && x <= n && y >= 0 && y <= m;
}

void bulletUpdate(Bullet& bullet) {
    switch (bullet.c) {
        case 'N':
            bullet.x -= bullet.v;
            break;
        case 'W':
            bullet.y -= bullet.v;
            break;
        case 'E':
            bullet.y += bullet.v;
            break;
        case 'S':
            bullet.x += bullet.v;
            break;
    }
}
```

```
61
62  void update(int time) {
63      if (pretime == time) {
64          return;
65      }
66      else {
67          pretime = time;
68          for (Bullet& bullet : bullets) {
69              if (!isValid(bullet.x, bullet.y)) {
70                  continue;
71              }
72              vis[bullet.x][bullet.y] = 0;
73              bulletUpdate(bullet);
74              if (!isValid(bullet.x, bullet.y)) {
75                  continue;
76              }
77              vis[bullet.x][bullet.y] = 1;
78          }
79          for (int i = 1; i <= k; i++)
80          {
81              if (time % soldiers[i].t == 0) {
82                  bullets.push_back({ soldiers[i].x, soldiers[i].y,
    soldiers[i].v, soldiers[i].c });
83                  vis[soldiers[i].x][soldiers[i].y] = 1;
84              }
85          }
86      }
87  }
88
89  void bfs(int x, int y, int time, int d) {
90      bool flag = false;
91      roles.push({ x, y, time, d });
92      go[x][y] = true;
93
94      while (roles.size()) {
95          Role role = roles.front();
96          roles.pop();
97
98          if (role.d < m + n - role.x - role.y) {
99              continue;
100         }
101         if (go[n][m][role.time] == 1) {
102             cout << roles.back().time;
103             flag = true;
104             return;
105         }
```

```
106         for (int i = 0; i < 5; i++) {
107             int tx = role.x + dx[i];
108             int ty = role.y + dy[i];
109             update(role.time + 1);
110             if ((!isValid(tx, ty)) || vis[tx][ty] == 1 || go[tx][ty]
    [role.time] || sold[tx][ty] == 1) {
111                 continue;
112             }
113             if (!vis[tx][ty]) {
114                 go[tx][ty][role.time] = 1;
115             }
116             roles.push({ tx, ty, role.time + 1, role.d - 1 });
117         }
118     }
119 
120     if (!flag) {
121         cout << "Bad luck!";
122     }
123 }
124 
125 void solve() {
126     cin >> n >> m >> k >> d;
127     for (int i = 1; i <= k; i++) {
128         char c;
129         int t, v, x, y;
130         cin >> c >> t >> v >> x >> y;
131         soldiers[i] = { c, t, v, x, y };
132         sold[x][y] = 1;
133     }
134     for (int i = 1; i <= k; i++) {
135         bullets.push_back({ soldiers[i].x, soldiers[i].y, soldiers[i].v,
    soldiers[i].c });
136         vis[soldiers[i].x][soldiers[i].y] = 1;
137     }
138 
139     bfs(0, 0, 0, d);
140 }
141 
142 int main() {
143     int q = 1;
144     for (int i = 0; i < q; i++) {
145         solve();
146     }
147     return 0;
148 }
```

小明回家

```cpp
#include<iostream>
#include<vector>
#include<queue>
using namespace std;

struct Point {
    int x;
    int y;
    int step;
};

char map[2000][2000];
int px[4] = {-1, 1, 0, 0};
int py[4] = {0, 0, -1, 1};
int n, m;

int stepCount(Point start, Point end) {
    int vst[2000][2000] = { 0 };
    queue<Point> points;
    points.push(start);
    vst[start.x][start.y] = 1;

    while (!points.empty()) {
        Point tmp = points.front();
        int x = tmp.x;
        int y = tmp.y;
        int step = tmp.step;
        points.pop();

        if (x == end.x && y == end.y) {
            return step;
        }

        for (int i = 0; i < 4; i++) {
            int nx = x + px[i];
            int ny = y + py[i];

            if (nx >= 0 && nx < n && ny >= 0 && ny < m && vst[nx][ny] == 0 && map[nx][ny] != '#') {
                Point next;
                next.x = nx;
                next.y = ny;
                next.step = step + 1;
                points.push(next);
```

```
                vst[nx][ny] = 1;
            }
        }
    }

    return -1;
}

int main() {
    Point start, key, end;
    vector<Point> keys;
    int keyCount = 0;
    int minStep = -1;
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> map[i][j];
            if (map[i][j] == 'S') {
                start.x = i;
                start.y = j;
                start.step = 0;
            }
            if (map[i][j] == 'P') {
                key.x = i;
                key.y = j;
                keyCount++;
                keys.push_back(key);
            }
            if (map[i][j] == 'T') {
                end.x = i;
                end.y = j;
            }
        }
    }

    for (int i = 0; i < keyCount; i++) {
        if (stepCount(start, keys[i]) == -1)
            continue;
        keys[i].step = stepCount(start, keys[i]);
        if (stepCount(keys[i], end) == -1)
            continue;
        minStep = minStep == -1 ? stepCount(keys[i], end) :
min(stepCount(keys[i], end), minStep);
    }
    cout << minStep;
    return 0;
```

```
}
```

Week 5 20250321

A 一元三次方程求解

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

double A, B, C, D;

double f(double x) {
    return A*x*x*x + B*x*x + C*x + D;
}

int main() {
    cin >> A >> B >> C >> D;
    for (int i = -10000; i <= 10000; i++) {
        if (f((i - 0.5) / 100) * f((i + 0.5) / 100) < 0 || f((i - 0.5) / 100) == 0)
            printf("%.2lf ", i / 100.0);
    }
    return 0;
}
```

B 循环比赛日程表

```cpp
#include <iostream>
#include <vector>
#include <iomanip>
using namespace std;

int main() {
    int M;
    cin >> M;
    int N = 1 << M;
    vector<vector<int>> table(N, vector<int>(N, 0));

    table[0][0] = 1;
    int half = 1;

    for (int k = 0; k < M; k++) {
```

```
        for (int i = 0; i < (1 << k); i++) {
            for (int j = 0; j < (1 << k); j++) {
                table[i][j + half] = table[i][j] + half;
                table[i + half][j] = table[i][j] + half;
                table[i + half][j + half] = table[i][j];
            }
        }
        half *= 2;
    }

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            cout << setw(3) << table[i][j];
        }
        cout << endl;
    }

    return 0;
}
```

C Entropy

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int freq_arr[27];
    string s;
    while (cin >> s && s != "END")
    {
        int total = 0;
        int sum = 0;
        priority_queue<int, vector<int>, greater<int> > q;
        memset(freq_arr, 0, sizeof(freq_arr));
        int n = s.size();
        for (int i = 0; i < n; i++)
        {
            if (s[i] == '_')
            {
                freq_arr[26]++;
            }
            else
            {
                freq_arr[s[i] - 'A']++;
            }
```

```
        }
        for (int i = 0; i < 27; i++)
        {
            if (freq_arr[i])
            {
                q.push(freq_arr[i]);
            }
        }
        if (q.size() == 1)
        {
            total = q.top();
            printf("%d %d %.1lf\n", n * 8, n * 1, (double)8/1);
        }
        else
        {
            while (q.size() > 1)
            {
                sum = 0;
                sum += q.top();
                q.pop();
                sum += q.top();
                q.pop();
                q.push(sum);
                total += sum;
            }
            printf("%d %d %.1lf\n", n * 8, total, (double)n * 8 / total);
        }
    }

    return 0;
}
```

D 小明的购物袋1

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int V;
    int n;
    cin >> V >> n;

    vector<int> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i];
```

```
    }

    vector<vector<int>> dp(n + 1, vector<int>(V + 1, 0));
    for (int i = 1; i < n + 1; i++) {
        for (int j = 0; j <= V; j++) {
            if (v[i - 1] <= j) {
                dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - v[i - 1]] +
v[i - 1]);
            } else {
                dp[i][j] = dp[i - 1][j];
            }
        }
    }
    cout << V - dp[n][V] << endl;
}
```

E 小明的购物袋2

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int V, n;
    cin >> V >> n;
    vector<pair<int, int>> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i].first >> v[i].second;
    }

    vector<vector<int>> dp(n + 1, vector<int>(V + 1, 0));

    for (int i = 1; i < n + 1; i++) {
        for (int j = 0; j < V + 1; j++) {
            if (j < v[i - 1].first) {
                dp[i][j] = dp[i - 1][j];
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - v[i -
1].first] + v[i - 1].second);
            }
        }
    }

    cout << dp[n][V] << endl;
}
```

F 小明的购物袋3

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int V, n;
    cin >> n >> V;
    vector<pair<int, int>> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i].first >> v[i].second;
    }

    vector<int> dp(V + 1, 0);

    for (int i = 0; i < n; i++) {
        for (int j = v[i].first; j <= V; j++) {
            dp[j] = max(dp[j], dp[j - v[i].first] + v[i].second);
        }
    }

    cout << dp[V] << endl;
}
```

Week 6 20250328

A 小明跳木桩

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n;
    cin >> n;
    vector<int> h(n);
    for (int i = 0; i < n; i++) {
        cin >> h[i];
    }

    vector<int> dp(n, 1);
    for (int i = 0; i < n; i++) {
```

```
        for (int j = 0; j < i; j++) {
            if (h[j] >= h[i]) {
                dp[i] = max(dp[i], dp[j] + 1);
            }
        }
    }

    int mmax = INT_MIN;
    for (int i = 0; i < n; i++) {
        mmax = max(mmax, dp[i]);
    }

    cout << mmax << endl;
}
```

B 删除最少的元素

```
#include<iostream>
using namespace std;

int main(){
    int n;
    cin >> n;
    int dp1[1009];
    int dp2[1009];
    int a[1009];
    int ans = 0;

    for(int i = 0; i < n; i++){
        cin >> a[i];
    }

    for(int i = 0; i < n; i++){
        dp1[i] = 1;
        dp2[i] = 1;
    }

    for(int i = 1; i < n; i++){
        for(int j = 0; j < i; j++){
            if(a[i] <= a[j]){
                dp1[i] = max(dp1[i], dp1[j] + 1);
            }
        }
    }

    for(int i = n - 2; i >= 0; i--){
```

```
        for(int j = n - 1; j > i; j--){
            if(a[i] <= a[j]){
                dp2[i] = max(dp2[i], dp2[j] + 1);
            }
        }
    }

    for(int i = 0; i < n; i++){
        ans = max(ans, dp1[i] + dp2[i] - 1);
    }

    cout << n - ans;

    return 0;
}
```

C 最长公共子序列

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    string a, b;
    cin >> a >> b;
    vector<vector<int>> dp(a.length() + 1, vector<int>(b.length() + 1, 0));

    for (int i = 1; i <= a.size(); i++) {
        for (int j = 1; j <= b.size(); j++) {
            if (a[i - 1] == b[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }


    int mmax = INT_MIN;
    for (int i = 0; i < a.size() + 1; i++) {
        for (int j = 0; j < b.size() + 1; j++) {
            mmax = max(mmax, dp[i][j]);
        }
    }

```

```
    cout << mmax << endl;
}
```

D 回文串

```
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    string s;
    cin >> s;
    string s2 = s;
    reverse(s2.begin(), s2.end());

    vector<vector<int>> dp(s.size() + 1, vector<int>(s.size() + 1));
    for (int i = 1; i <= s.size(); i++) {
        for (int j = 1; j <= s.size(); j++) {
            if (s[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }

    // l = len - lcs
    cout << s.length() - dp[s.size()][s.size()] << endl;
}
```

E 灌溉机器人

```
#include <bits/stdc++.h>
using namespace std;
#define max_Heap(x) priority_queue<x, vector<x>, less<x>>
#define min_Heap(x) priority_queue<x, vector<x>, greater<x>>
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> PII;
typedef pair<long long, long long> PLL;
const double PI = acos(-1);

int n, m;
char field[106][16];
vector<int> s[106];
```

```cpp
int dp[106][106][106];

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    unordered_map<int, int> mp;

    cin >> n >> m;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cin >> field[i][j];
        }
    }

    for (int i = 1; i <= n; i++)
    {
        for (int j = 0; j < (1 << m); j++)
        {
            bool ok = 1;
            for (int k = 0; k < m; k++)
            {
                if (((j >> k) & 1) && (field[i][k] == 'H'))
                {
                    ok = 0;
                    break;
                }
            }
            if ((j & (j << 1)) || (j & (j << 2)) || (j & (j >> 1)) || (j & (j >> 2)))
            {
                ok = 0;
            }
            if (ok)
                s[i].push_back(j);
        }
    }

    for (int i = 0; i < (1 << m); i++)
    {
        int cnt = 0;
        for (int j = 0; j < m; j++)
```

```cpp
        {
            if ((i >> j) & 1)
                cnt++;
        }
        mp[i] = cnt;
    }

    for (int i = 0; i < s[1].size(); i++)
    {
        dp[1][i][0] = mp[s[1][i]];
    }

    s[0].push_back(0);

    for (int i = 1; i <= n; i++)
    {
        for (int num3 = 0; num3 < s[i].size(); num3++)
        {
            int s3 = s[i][num3];
            for (int num2 = 0; num2 < s[i - 1].size(); num2++)
            {
                int s2 = s[i - 1][num2];
                for (int num1 = 0; num1 < s[i - 2].size(); num1++)
                {
                    int s1 = s[i - 2][num1];
                    if (!(s1 & s2) && !(s1 & s3) && !(s2 & s3))
                        dp[i][num3][num2] = max(dp[i][num3][num2], dp[i - 1][num2][num1] + mp[s3]);
                }
            }
        }
    }

    int ans = 0;
    for (int i = 0; i < s[n].size(); i++)
    {
        for (int j = 0; j < s[n - 1].size(); j++)
        {
            ans = max(ans, dp[n][i][j]);
        }
    }
    cout << ans;

    return 0;
}
```

F 小明的积木

```cpp
#include <bits/stdc++.h>
using namespace std;
#define max_Heap(x) priority_queue<x, vector<x>, less<x>>
#define min_Heap(x) priority_queue<x, vector<x>, greater<x>>
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int, int> PII;
typedef pair<long long, long long> PLL;
const double PI = acos(-1);

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    int capacity, n;
    cin >> capacity >> n;
    int dp[(1 << n)];
    memset(dp, 0x3f, sizeof(dp));
    vector<int> s(n);
    vector<int> w(n);
    for (int i = 0; i < n; i++)
    {
        cin >> s[i] >> w[i];
    }
    for (int i = 0; i < (1 << n); i++)
    {
        int max_s = 0;
        int sum_w = 0;
        for (int j = 0; j < n; j++)
        {
            if ((i >> j) & 1)
            {
                max_s = max(max_s, s[j]);
                sum_w += w[j];
            }
        }
        if (sum_w <= capacity)
        {
            dp[i] = max_s;
        }
        else
        {
```

```
            for (int j = ((i - 1) & i); j > 0; j = ((j - 1) & i))
            {
                dp[i] = min(dp[i], dp[j] + dp[j ^ i]);
            }
        }
    }
    cout << dp[(1 << n) - 1];

    return 0;
}
```

G 消除字符串

```
#include <bits/stdc++.h>
using namespace std;

string str;

bool isReverse(int i) {
    string str1;
    string str2;

    for (int j = 0; j < 16; j++) {
        if ((i >> j) & 1) {
            str1 += str[j];
        }
    }
    str2 = str1;
    reverse(str1.begin(), str1.end());

    return str1 == str2;
}

int main() {
    cin >> str;
    int length = str.size();
    vector<int> dp(1 << length, 0);

    for (int i = 1; i < (1 << length); i++) {
        dp[i] = isReverse(i) ? 1 : 16;
        for (int j = i; j; j = (j - 1) & i) {
            dp[i] = min(dp[i], dp[j] + dp[j ^ i]);
        }
    }
    cout << dp[(1 << length) - 1];
    return 0;
```

```
}
```

Week 8 20250411

A 二叉树中的最低公共祖先

```
#include<bits/stdc++.h>

using namespace std;
struct Node {
    int val;
    Node* lchild;
    Node* rchild;
    Node(int val) : val(val), lchild(NULL), rchild(NULL) {}
};

vector<int> inorder;
vector<int> preorder;
map<int, Node*> record;

Node* createTree(vector<int> preorder, vector<int> inorder, int preL, int preR, int inL, int inR) {
    if (preL > preR) {
        return NULL;
    }

    Node* root = new Node(preorder[preL]);
    record[preorder[preL]] = root;
    int k = inL;
    while (inorder[k] != preorder[preL]) {
        k++;
    }
    int numLeft = k - inL;

    root->lchild = createTree(preorder, inorder, preL + 1, preL + numLeft, inL, k - 1);
    root->rchild = createTree(preorder, inorder, preL + numLeft + 1, preR, k + 1, inR);
    return root;
}

Node* LCA(Node* root, Node* first, Node* second) {
    if (root == NULL || root == first || root == second) {
```

```
35          return root;
36      }
37
38      Node* left = LCA(root->lchild, first, second);
39      Node* right = LCA(root->rchild, first, second);
40      if (left && right) return root;
41      else if (left) return left;
42      else return right;
43  }
44
45  int main() {
46      int M, N;
47      cin >> M >> N;
48      for (int i = 0; i < N; i++) {
49          int order;
50          cin >> order;
51          inorder.push_back(order);
52      }
53      for (int i = 0; i < N; i++) {
54          int order;
55          cin >> order;
56          preorder.push_back(order);
57      }
58
59      Node* root = createTree(preorder, inorder, 0, N - 1, 0, N - 1);
60      while (M--) {
61          int u, v;
62          cin >> u >> v;
63          if (record.count(u) && record.count(v)) {
64              Node* lca = LCA(root, record[u], record[v]);
65              if (lca->val == u) {
66                  cout << u << " is an ancestor of " << v << "." << endl;
67              }
68              else if (lca->val == v) {
69                  cout << v << " is an ancestor of " << u << "." << endl;
70              }
71              else {
72                  cout << "LCA of " << u << " and " << v << " is " << lca-
    >val << "." << endl;
73              }
74          }
75          else if (record.count(u)) {
76              cout << "ERROR: " << v << " is not found." << endl;
77          }
78          else if (record.count(v)) {
79              cout << "ERROR: " << u << " is not found." << endl;
```

```
        }
        else {
            cout << "ERROR: " << u << " and " << v << " are not found."
<< endl;
        }
    }
    return 0;
}
```

B FBI 树

```
#include <bits/stdc++.h>

using namespace std;

struct Node {
    char val;
    Node* lchild;
    Node* rchild;
    Node(char val) : val(val), lchild(NULL), rchild(NULL) {}
};

char classify(string str) {
    bool contain1 = false;
    bool contain0 = false;

    for (int i = 0; i < str.size(); i++) {
        if (str[i] == '0') {
            contain0 = true;
        }
        else {
            contain1 = true;
        }
    }
    if (contain0 && contain1) {
        return 'F';
    }
    else if (contain0){
        return 'B';
    }
    else {
        return 'I';
    }
}

Node* creatTree(string str) {
```

```
    char val = classify(str);
    int len = str.size();

    if (len == 0) {
        return NULL;
    }

    Node* node = new Node(val);
    if (len >= 2) {
        node->lchild = creatTree(str.substr(0, len / 2));
        node->rchild = creatTree(str.substr(len / 2, len / 2));
    }
    return node;
}

void postorder(Node* node) {
    if (node) {
        postorder(node->lchild);
        postorder(node->rchild);
        cout << node->val;
    }
}

int main() {
    int N;
    string str;
    cin >> N >> str;

    Node* root = creatTree(str);
    postorder(root);

    return 0;
}
```

C 食物链

```
#include <bits/stdc++.h>

using namespace std;

class FoodUnion {
private:
    int parent[50000];
    int relation[50000];
    int fakeCount;
    int total;
```

```cpp

public:
    FoodUnion(int n) {
        for (int i = 1; i <= n; ++i) {
            parent[i] = i;
            relation[i] = 0;
        }
        fakeCount = 0;
        total = n;
    }

    int find(int x) {
        if (x != parent[x]) {
            int root = find(parent[x]);
            relation[x] += relation[parent[x]];
            parent[x] = root;
        }
        return parent[x];
    }

    void unite(int x, int y, int relType) {
        relType--;
        if (x > total || y > total) {
            fakeCount++;
            return;
        }
        if (relType == 1 && x == y) {
            fakeCount++;
            return;
        }

        int rootX = find(x);
        int rootY = find(y);

        if (rootX == rootY) {
            int delta = ((relation[x] - relation[y]) % 3 + 3) % 3;
            if (delta != relType) {
                fakeCount++;
            }
        } else {
            parent[rootX] = rootY;
            relation[rootX] = relation[y] - relation[x] + relType;
        }
    }

    int getFakeCount() const {
```

```
        return fakeCount;
    }
};

int main() {
    int n, k;
    cin >> n >> k;
    FoodUnion uf(n);

    for (int i = 0; i < k; ++i) {
        int type, a, b;
        cin >> type >> a >> b;
        uf.unite(a, b, type);
    }

    cout << uf.getFakeCount() << endl;
    return 0;
}
```

D 求二叉树高度

```
#include<string>
#include<iostream>
#include <queue>
#include <algorithm>
#include <sstream>
using namespace std;

struct TreeNode {
     int val;
     TreeNode *lchild;
     TreeNode *rchild;
     TreeNode(int x) : val(x), lchild(NULL), rchild(NULL) {}
 };

class Solution {
public:
    int maxDepth(TreeNode* p) {
        int lh,rh,hi;
        lh = rh = hi = 0;
        if(p != nullptr){
            if(p ->lchild == NULL) lh = 0 ; else lh= maxDepth(p->lchild);
            if(p->rchild == NULL)  rh= 0 ; else rh= maxDepth(p->rchild);
            if (lh > rh) hi = lh + 1; else hi = rh + 1;
        }
        else hi = 0;
```

```
        return hi;
    }
};

void trimLeftTrailingSpaces(string &input) {
    input.erase(input.begin(), find_if(input.begin(), input.end(), [](int
ch) {
        return !isspace(ch);
    }));
}

void trimRightTrailingSpaces(string &input) {
    input.erase(find_if(input.rbegin(), input.rend(), [](int ch) {
        return !isspace(ch);
    }).base(), input.end());
}

TreeNode* stringToTreeNode(string input) {
    trimLeftTrailingSpaces(input);
    trimRightTrailingSpaces(input);
    input = input.substr(1, input.length() - 2);
    if (!input.size()) {
        return nullptr;
    }

    string item;
    stringstream ss;
    ss.str(input);

    getline(ss, item, ',');
    TreeNode* root = new TreeNode(stoi(item));
    queue<TreeNode*> nodeQueue;
    nodeQueue.push(root);

    while (true) {
        TreeNode* node = nodeQueue.front();
        nodeQueue.pop();

        if (!getline(ss, item, ',')) {
            break;
        }

        trimLeftTrailingSpaces(item);
        if (item != "null") {
            int leftNumber = stoi(item);
            node->lchild = new TreeNode(leftNumber);
```

```
            nodeQueue.push(node->lchild);
        }

        if (!getline(ss, item, ',')) {
            break;
        }

        trimLeftTrailingSpaces(item);
        if (item != "null") {
            int rightNumber = stoi(item);
            node->rchild = new TreeNode(rightNumber);
            nodeQueue.push(node->rchild);
        }
    }
    return root;
}

int main() {
    string line;
    getline(cin, line);
    TreeNode* root = stringToTreeNode(line);
    int ret = Solution().maxDepth(root);
    string out = to_string(ret);
    cout << out << endl;
    return 0;
}
```

E 新二叉树

```
#include <bits/stdc++.h>

using namespace std;

struct TreeNode {
    char val;
    TreeNode *left;
    TreeNode *right;
    TreeNode(char val) : val(val), left(nullptr), right(nullptr) {}
};

TreeNode *traversal(TreeNode *node, char val) {
    if (node == nullptr) {
        return nullptr;
    }
    if (node->val == val) {
        return node;
```

```
    }

    TreeNode *p = traversal(node->left, val);
    if (p == nullptr) {
        return traversal(node->right, val);
    }
    return p;
}

void traversal2(TreeNode *node) {
    if (node == nullptr) {
        return;
    }
    cout << node->val;
    traversal2(node->left);
    traversal2(node->right);
}

int main(void) {
    int n;
    cin >> n;

    TreeNode *root = nullptr;
    TreeNode *p = root;
    for (int i = 0; i < n; i++) {
        char a, b, c;
        cin >> a >> b >> c;
        if (i == 0) {
            root = new TreeNode(a);
        }
        p = traversal(root, a);
        if (b != '*') {
            p->left = new TreeNode(b);
        }
        if (c != '*') {
            p->right = new TreeNode(c);
        }
    }
    traversal2(root);
    cout << endl;
}
```

F 二叉树遍历

```
#include <bits/stdc++.h>

```

```cpp
using namespace std;

struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode(int val) : val(val), left(nullptr), right(nullptr) {}
};

TreeNode* traversal(vector<int>& pre, vector<int>& middle) {
    if (pre.size() == 0) {
        return nullptr;
    }
    int rootVal = pre[0];
    TreeNode *root = new TreeNode(rootVal);

    if (pre.size() == 1) {
        return root;
    }

    int index = -1;
    for (int i = 0; i < middle.size(); i++) {
        if (middle[i] == rootVal) {
            index = i;
            break;
        }
    }

    vector<int> leftMiddle(middle.begin(), middle.begin() + index);
    vector<int> rightMiddle(middle.begin() + index + 1, middle.end());

    vector<int> leftPre(pre.begin() + 1, pre.begin() + leftMiddle.size()
+ 1);
    vector<int> rightPre(pre.begin() + leftMiddle.size() + 1, pre.end());

    root->left = traversal(leftPre, leftMiddle);
    root->right = traversal(rightPre, rightMiddle);

    return root;
}

void traversal2(TreeNode *root) {
    if (root == nullptr) {
        return;
    }

```

```
    traversal2(root->left);
    traversal2(root->right);
    cout << root->val << ' ';
}

int main(void) {
    int n;
    cin >> n;
    vector<int> pre(n);
    vector<int> middle(n);

    for (int i = 0; i < n; i++) {
        cin >> pre[i];
    }

    for (int i = 0; i < n ; i++) {
        cin >> middle[i];
    }

    TreeNode *root = traversal(pre, middle);

    traversal2(root);
    cout << endl;
}
```

G 朋友

```
#include<bits/stdc++.h>

using namespace std;

class UnionFind {
private:
    int father[5000];
    int height[5000];

public:
    UnionFind(int n) {
        for (int i = 1; i <= n; i++) {
            father[i] = i;
            height[i] = 1;
        }
    }

    int get(int x) {
        if (x == father[x]) {
```

```
            return x;
        }
        return father[x] = get(father[x]);
    }

    void merge(int x, int y) {
        x = get(x);
        y = get(y);

        if (x == y) {
            return;
        }
        else {
            if (height[x] == height[y]) {
                height[x]++;
                father[y] = x;
            }
            else if (height[x] > height[y]) {
                father[y] = x;
            }
            else {
                father[x] = y;
            }
        }
    }
};

int main() {
    int n, m, p;
    cin >> n >> m >> p;
    UnionFind friends(n);

    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
        friends.merge(x, y);
    }
    for (int i = 0; i < p; i++) {
        int x, y;
        cin >> x >> y;
        if (friends.get(x) == friends.get(y)) {
            cout << "Yes" << endl;
        }
        else {
            cout << "No" << endl;
        }
```

```
    }

    return 0;
}
```

H 网络交友

```
#include <bits/stdc++.h>

using namespace std;

class DisjointSet {
private:
    unordered_map<string, string> parent;
    unordered_map<string, int> size;
    unordered_map<string, int> depth;

public:
    void insert(const string& person) {
        if (parent.count(person)) return;
        parent[person] = person;
        size[person] = 1;
        depth[person] = 1;
    }

    string find(const string& person) {
        if (parent[person] != person) {
            parent[person] = find(parent[person]);
        }
        return parent[person];
    }

    void unite(const string& a, const string& b) {
        string rootA = find(a);
        string rootB = find(b);

        if (rootA == rootB) return;

        if (depth[rootA] < depth[rootB]) {
            parent[rootA] = rootB;
            size[rootB] += size[rootA];
        } else {
            parent[rootB] = rootA;
            size[rootA] += size[rootB];
            if (depth[rootA] == depth[rootB]) {
                depth[rootA]++;
```

```
            }
        }
    }

    int getSize(const string& person) {
        return size[find(person)];
    }
};

int main() {
    int relationCount;
    cin >> relationCount;

    DisjointSet network;

    while (relationCount--) {
        string person1, person2;
        cin >> person1 >> person2;
        network.insert(person1);
        network.insert(person2);
        network.unite(person1, person2);
        cout << network.getSize(person1) << endl;
    }

    return 0;
}
```

Week 9

A 找出所有谎言

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 50010;
int parent[MAXN], rel[MAXN];

int findf(int x) {
    if (parent[x] != x) {
        int p = parent[x];
        parent[x] = findf(p);
        rel[x] = (rel[x] + rel[p]) % 3;
    }
```

```cpp
    return parent[x];
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    int n, k;
    cin >> n >> k;
    for (int i = 1; i <= n; i++) {
        parent[i] = i;
        rel[i] = 0;
    }
    int ans = 0;
    while (k--) {
        int d, x, y;
        cin >> d >> x >> y;
        if (x < 1 || x > n || y < 1 || y > n || (d == 2 && x == y)) {
            ans++;
            continue;
        }
        int fx = findf(x), fy = findf(y);
        if (fx == fy) {
            if (d == 1) {
                if ((rel[x] - rel[y] + 3) % 3 != 0) ans++;
            } else {
                if ((rel[x] - rel[y] + 3) % 3 != 1) ans++;
            }
        } else {
            if (d == 1) {
                parent[fx] = fy;
                rel[fx] = (rel[y] - rel[x] + 3) % 3;
            } else {
                parent[fx] = fy;
                rel[fx] = (rel[y] + 1 - rel[x] + 3) % 3;
            }
        }
    }
    cout << ans << endl;
    return 0;
}
```

B 接龙

```cpp
#include <cstdio>
#include <cstdlib>
```

```cpp
#include <algorithm>
using namespace std;

const int MAXN = 30001;

int parent[MAXN];
int distance_[MAXN];
int size_[MAXN];

void init() {
    for (int i = 1; i <= 30000; ++i) {
        parent[i] = i;
        distance_[i] = 0;
        size_[i] = 1;
    }
}

int find(int x) {
    if (parent[x] != x) {
        int old_parent = parent[x];
        find(old_parent);
        distance_[x] += distance_[old_parent];
        parent[x] = parent[old_parent];
    }
    return parent[x];
}

void merge(int i, int j) {
    int x = find(i);
    int y = find(j);
    if (x != y) {
        parent[x] = y;
        distance_[x] = size_[y];
        size_[y] += size_[x];
    }
}

int main() {
    init();
    int T;
    scanf("%d", &T);
    char op;
    int a, b;
    while (T--) {
        scanf(" %c %d %d", &op, &a, &b);
        if (op == 'M') {
```

```
            merge(a, b);
        } else {
            int x_root = find(a);
            int y_root = find(b);
            if (x_root != y_root) {
                printf("-1\n");
            } else {
                int diff = abs(distance_[a] - distance_[b]);
                printf("%d\n", diff - 1);
            }
        }
    }
    return 0;
}
```

C 在二叉树上移动

```
#include <bits/stdc++.h>

using namespace std;

string handle(string &s) {
    string result;
    int i = 0;
    while (i < s.size()) {
        if (i == 0) {
            result.push_back(s[i]);
            i++;
            continue;
        }
        if (s[i] == 'U') {
            if (result.back() == 'L' || result.back() == 'R') {
                result.pop_back();
                i++;
                continue;
            }
        }

        result.push_back(s[i]);
        i++;
    }

    return result;
}

int main(void) {
```

```
    ios::sync_with_stdio(false);
    cin.tie(0);
    unsigned long long N, X;
    cin >> N >> X;
    string s;
    cin >> s;
    s = handle(s);
    for (char c : s) {
        if (c == 'U'){
            X /= 2;
        } else if (c == 'L') {
            X *= 2;
        } else {
            X = X * 2 + 1;
        }
    }
    cout << X << endl;
}
```

D 二叉搜索树

```
#include <iostream>
using namespace std;

struct Node {
    int value;
    Node* left;
    Node* right;
    Node(int v) : value(v), left(nullptr), right(nullptr) {}
};

Node* createBST(const string& sequence) {
    if (sequence.empty()) return nullptr;

    Node* root = new Node(sequence[0] - '0');

    for (int i = 1; i < sequence.size(); ++i) {
        int num = sequence[i] - '0';
        Node* current = root;
        Node* previous = nullptr;

        while (current != nullptr) {
            previous = current;
            if (num < current->value)
                current = current->left;
            else
```

```
                current = current->right;
            }

            Node* newNode = new Node(num);
            if (num < previous->value)
                previous->left = newNode;
            else
                previous->right = newNode;
        }

        return root;
    }

    bool areIdentical(Node* a, Node* b) {
        if (!a && !b) return true;
        if (!a || !b) return false;

        return (a->value == b->value) &&
               areIdentical(a->left, b->left) &&
               areIdentical(a->right, b->right);
    }

    int main() {
        int m;
        string baseSequence;
        cin >> m >> baseSequence;

        Node* referenceTree = createBST(baseSequence);

        while (m--) {
            string testSequence;
            cin >> testSequence;

            Node* testTree = createBST(testSequence);
            cout << (areIdentical(referenceTree, testTree) ? "YES" : "NO") <<
    endl;
        }

        return 0;
    }
```

E The order of a Tree

```
#include <iostream>
#include <cstring>
using namespace std;
```

```cpp

const int N = 100010;

int value[N], leftChild[N], rightChild[N], preorderResult[N];
int nodeCount, totalNodes;

void insertNode(int rootIndex, int val) {
    if (val <= value[rootIndex]) {
        if (leftChild[rootIndex])
            insertNode(leftChild[rootIndex], val);
        else
            leftChild[rootIndex] = totalNodes;
    } else {
        if (rightChild[rootIndex])
            insertNode(rightChild[rootIndex], val);
        else
            rightChild[rootIndex] = totalNodes;
    }
}

void preorderTraversal(int index) {
    preorderResult[nodeCount++] = value[index];
    if (leftChild[index]) preorderTraversal(leftChild[index]);
    if (rightChild[index]) preorderTraversal(rightChild[index]);
}

int main() {
    int n;
    while (cin >> n) {
        memset(leftChild, 0, sizeof leftChild);
        memset(rightChild, 0, sizeof rightChild);
        int rootIndex = -1, x;
        totalNodes = 0;

        for (int i = 0; i < n; ++i) {
            cin >> x;
            if (rootIndex == -1) {
                value[++rootIndex] = x;
            } else {
                value[++totalNodes] = x;
                insertNode(rootIndex, x);
            }
        }

        nodeCount = 0;
        preorderTraversal(rootIndex);
```

```
50
51          cout << preorderResult[0];
52          for (int i = 1; i < nodeCount; ++i)
53              cout << " " << preorderResult[i];
54          cout << endl;
55      }
56
57      return 0;
58  }
```

F 二叉树

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   struct TreeNode {
6       int val;
7       TreeNode *left;
8       TreeNode *right;
9       TreeNode(int val) : val(val), left(nullptr), right(nullptr) {}
10  };
11
12  TreeNode *traversal(vector<int>& in, vector<int>& pre, int inLeft, int
    inRight, int preLeft, int preRight) {
13      if (inLeft >= inRight) {
14          return nullptr;
15      }
16      int rootVal = pre[preLeft];
17      TreeNode *root = new TreeNode(rootVal);
18      if (inRight - inLeft == 1) {
19          return root;
20      }
21      int i = 0;
22      for (i = inLeft; i < inRight; i++) {
23          if (in[i] == rootVal) {
24              break;
25          }
26      }
27
28      root->left = traversal(in, pre, inLeft, i, preLeft + 1, preLeft + i -
    inLeft);
29      root->right = traversal(in, pre, i + 1, inRight, preLeft + i - inLeft
    + 1, preRight);
30      return root;
31  }
```

```cpp

vector<int> v;

void traversal2(TreeNode *node) {
    if (node == nullptr) {
        return;
    }
    traversal2(node->left);
    traversal2(node->right);
    v.push_back(node->val);
}

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(0);

    int n;
    while (cin >> n) {
        v.clear();
        vector<int> pre(n);
        for (int i = 0; i < n; i++) {
            cin >> pre[i];
        }
        vector<int> in(n);
        for (int i = 0; i < n; i++) {
            cin >> in[i];
        }

        TreeNode *root = traversal(in, pre, 0, in.size(), 0, pre.size());
        traversal2(root);
        for (int i = 0; i < n - 1; i++) {
            cout << v[i] << " ";
        }

        cout << v[n - 1] << endl;
    }
}
```

G 线索二叉树的中序遍历

```cpp
#include<string>
#include<iostream>
#include <queue>
#include <algorithm>
#include <sstream>
using namespace std;
```

```
7
8   struct TreeNode {
9        int val;
10       TreeNode *lchild;
11       TreeNode *rchild;
12       int ltag;
13       int rtag;
14       TreeNode(int x) : val(x),ltag(0),rtag(0), lchild(NULL), rchild(NULL)
    {}
15    };
16
17
18  class Solution {
19  public:
20      void inorderthread(TreeNode * thr) {
21          TreeNode * p = thr;
22          while (p) {
23              while(p->ltag == 0 && p->lchild) p = p->lchild;
24              cout << p->val << endl;
25              while (p->rtag == 1 && p->rchild) {
26                  p = p->rchild;
27                  cout << p->val << endl;
28              }
29              p = p->rchild;
30          }
31      }
32  };
33
34
35
36
37  void trimLeftTrailingSpaces(string &input) {
38      input.erase(input.begin(), find_if(input.begin(), input.end(), [](int
    ch) {
39          return !isspace(ch);
40      }));
41  }
42
43  void trimRightTrailingSpaces(string &input) {
44      input.erase(find_if(input.rbegin(), input.rend(), [](int ch) {
45          return !isspace(ch);
46      }).base(), input.end());
47  }
48
49
50  void inThread(TreeNode *p, TreeNode *&pre){
```

```
    if(p){
        inThread(p->lchild, pre); //线索化左子树
        if(p->lchild == NULL){
            p->lchild = pre;
            p->ltag = 1;
        }
        if(pre != NULL && pre->rchild == NULL){
            pre->rchild = p;
            pre->rtag = 1;
        }
        pre = p;
        inThread(p->rchild, pre);
    }
}

void createThreadTree(TreeNode * root){
    TreeNode * pre = NULL;
    if(root){
        inThread(root, pre);
    }
}


TreeNode* stringToTreeNode(string input) {
    trimLeftTrailingSpaces(input);
    trimRightTrailingSpaces(input);
    input = input.substr(1, input.length() - 2);
    if (!input.size()) {
        return nullptr;
    }

    string item;
    stringstream ss;
    ss.str(input);

    getline(ss, item, ',');
    TreeNode* root = new TreeNode(stoi(item));
    queue<TreeNode*> nodeQueue;
    nodeQueue.push(root);

    while (true) {
        TreeNode* node = nodeQueue.front();
        nodeQueue.pop();

        if (!getline(ss, item, ',')) {
            break;
```

```
        }

        trimLeftTrailingSpaces(item);
        if (item != "null") {
            int leftNumber = stoi(item);
            node->lchild = new TreeNode(leftNumber);
            nodeQueue.push(node->lchild);
        }

        if (!getline(ss, item, ',')) {
            break;
        }

        trimLeftTrailingSpaces(item);
        if (item != "null") {
            int rightNumber = stoi(item);
            node->rchild = new TreeNode(rightNumber);
            nodeQueue.push(node->rchild);
        }
    }
    return root;
}

int main() {
    string line;
    getline(cin, line);
    TreeNode* root = stringToTreeNode(line);
    createThreadTree(root);
    Solution().inorderthread(root);
    //TravIn_Thread_BT(root);
    return 0;
}
```

Week 10 课上

A Trees on the level

```
#include<iostream>
#include<cstring>
#include<queue>
#include<stdio.h>
#include<string>
using namespace std;
```

```cpp
#define maxn 300

struct node {
    int val;
    bool haveVal;
    int l,r;
} tree[300];
int root,len;
bool ok;

bool read_input() {
    string s;
    ok = true;
    root = 0,len=1,tree[0].l=-1,tree[0].r=-1,tree[0].haveVal=false;
    for (;;) {
        if (!(cin>>s))return false;
        if (s=="()")return true;
        int v,now=root;
        sscanf(&s[1], "%d", &v);
        int i=s.find(',')+1;
        while (s[i] != ')') {
            if (s[i] == 'L') {
                if (tree[now].l == -1) {
                    tree[now].l = len++;

tree[tree[now].l].l=-1,tree[tree[now].l].r=-1,tree[tree[now].l].haveVal=fa
                }
                now = tree[now].l;
            }
            else{
                if (tree[now].r == -1){
                    tree[now].r = len++;

tree[tree[now].r].l=-1,tree[tree[now].r].r=-1,tree[tree[now].r].haveVal=fa
                }
                now = tree[now].r;
            }
            i++;
        }
        if (tree[now].haveVal) {
            ok = false;
        }
        tree[now].val = v;
        tree[now].haveVal = true;
    }
}
```

```
51
52  int main() {
53      while (read_input()) {
54          if (!ok) {
55              cout << "not complete" << endl;
56              continue;
57          }
58          string s="";
59          queue<int> q;
60          q.push(root);
61          int now;
62          while (q.size()) {
63              now = q.front(); q.pop();
64              if (!tree[now].haveVal) {
65                  s= "not complete";
66                  break;
67              }
68              else {
69                  if (s == "")
70                      s += to_string(tree[now].val);
71                  else
72                      s += " "+to_string(tree[now].val);
73              }
74              if(tree[now].l!=-1)q.push(tree[now].l);
75              if(tree[now].r!=-1)q.push(tree[now].r);
76          }
77          cout << s << endl;
78      }
79      return 0;
80  }
```

C Shaolin

```
1   #include <iostream>
2   #include <cstdio>
3   #include <cstring>
4   #include <vector>
5   #include <set>
6   #include <algorithm>
7   #include <cmath>
8   #include <map>
9   using namespace std;
10
11  #define MAX_N 100006
12  int n;
13  map<int, int> distanceMap;
```

```cpp
map<int, int>::iterator lowerIt, prevIt;

int main() {
    while (scanf("%d", &n) == 1 && n) {
        distanceMap.clear();
        distanceMap[1000000000] = 1;

        for (int i = 0; i < n; i++) {
            int x, y;
            scanf("%d %d", &x, &y);

            lowerIt = distanceMap.lower_bound(y);

            if (lowerIt == distanceMap.begin()) {
                printf("%d %d\n", x, lowerIt->second);
            } else {
                prevIt = lowerIt;
                prevIt--;

                if (abs(y - prevIt->first) <= abs(y - lowerIt->first)) {
                    printf("%d %d\n", x, prevIt->second);
                } else {
                    printf("%d %d\n", x, lowerIt->second);
                }
            }

            distanceMap[y] = x;
        }
    }
    return 0;
}
```

D Robotic Sort

```cpp
#include<stdio.h>
#include<algorithm>
#include<string.h>
using namespace std;
#define N 500006
#define lc (tr[id].c[0])
#define rc (tr[id].c[1])
#define key (tr[tr[root].c[1]].c[0])
struct tree
{
    int fa,sum,c[2],lz,v;
}tr[N];
```

```
struct point
{
    int v,id;
    bool operator<(const point a)const
    {
        if(a.v==v)return id<a.id;
        else return v<a.v;
    }
}so[N/5];
int tot,root,n;
int xia[N];
int newpoint(int d,int f)
{
    tr[tot].sum=1;
    tr[tot].v=d;
    tr[tot].c[0]=tr[tot].c[1]=-1;
    tr[tot].lz=0;
    tr[tot].fa=f;
    return tot++;
}
void push(int id)
{
    int lsum,rsum;
    if(lc==-1)lsum=0;
    else lsum=tr[lc].sum;
    if(rc==-1)rsum=0;
    else rsum=tr[rc].sum;
    tr[id].sum=lsum+rsum+1;
}
int build(int l,int r,int v)
{
    if(r<l)return -1;
    int mid=(r+l)>>1;
    int ro=newpoint(mid,v);
    xia[mid]=ro;
    tr[ro].c[0]=build(l,mid-1,ro);
    tr[ro].c[1]=build(mid+1,r,ro);
    push(ro);
    return ro;
}
void lazy(int id)
{
    if(tr[id].lz)
    {
        swap(lc,rc);
        tr[lc].lz^=1;
```

```
59          tr[rc].lz^=1;
60          tr[id].lz=0;
61      }
62  }
63
64  void xuanzhuan(int x,int k)
65  {
66      if(tr[x].fa==-1)return ;
67      int fa=tr[x].fa;
68      int w;
69      lazy(fa);
70      lazy(x);
71      tr[fa].c[!k]=tr[x].c[k];
72      if(tr[x].c[k]!=-1)tr[tr[x].c[k]].fa=fa;
73      tr[x].fa=tr[fa].fa;
74      tr[x].c[k]=fa;
75      if(tr[fa].fa!=-1)
76      {
77          w=tr[tr[fa].fa].c[1]==fa;
78          tr[tr[fa].fa].c[w]=x;
79      }
80      tr[fa].fa=x;
81      push(fa);
82      push(x);
83  }
84
85  void splay(int x,int goal)
86  {
87      if(x==-1)return ;
88      lazy(x);
89      while(tr[x].fa!=goal)
90      {
91          int y=tr[x].fa;
92          lazy(tr[y].fa);
93          lazy(y);
94          lazy(x);
95          bool w=(x==tr[y].c[1]);
96          if(tr[y].fa!=goal&&w==(y==tr[tr[y].fa].c[1]))xuanzhuan(y,!w);
97          xuanzhuan(x,!w);
98      }
99      if(goal==-1)root=x;
100     push(x);
101 }
102 int next(int id)
103 {
104     lazy(id);
```

```
    int p=tr[id].c[1];
    if(p==-1)return id;
    lazy(p);
    while(tr[p].c[0]!=-1)
    {
        p=tr[p].c[0];
        lazy(p);
    }
    return p;
}
int main()
{
    while(scanf("%d",&n),n)
    {
        for(int i=1;i<=n;i++)
        {
            scanf("%d",&so[i].v);
            so[i].id=i;
        }
        sort(so+1,so+n+1);
        so[0].id=0;
        tot=0;
        int d,l;
        root=build(0,n+1,-1);
        for(int i=1;i<=n;i++)
        {
            int ro=xia[so[i].id];
            int ne;
            splay(ro,-1);
            d=tr[tr[root].c[0]].sum;
            l=xia[so[i-1].id];
            ne=next(ro);
            splay(l,-1);
            splay(ne,root);
            lazy(root);
            lazy(tr[root].c[1]);
            tr[key].lz^=1;
            if(i!=1)printf(" ");
            printf("%d",d);
        }
        printf("\n");
    }
    return 0;
}
```

E Looploop 可能过不了

```cpp
#include <functional>
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <map>
#include <cmath>
using namespace std;
typedef long long ll;
typedef long  double ld;
#define key_value ch[ch[root][1]][0]
const int maxn = 200010;

int ch[maxn][2];
int pre[maxn],siz[maxn],num[maxn];
int rev[maxn],key[maxn];
int add[maxn];
int Min[maxn],a[maxn];
int tot,tp;
int root,n;
void push_up(int r)
{
    int lson = ch[r][0],rson = ch[r][1];
    siz[r] = siz[lson] + siz[rson] + 1;
    Min[r] = min(key[r],min(Min[lson],Min[rson]));
}

void update_add(int r,int val)
{
    if(!r) return;
    key[r] += val;
    add[r] += val;
    Min[r] += val;
}

void inOrder(int r)
{
    if(!r)
        return;
    inOrder(ch[r][0]);
    printf("%d ",key[r]);
    inOrder(ch[r][1]);
}

void debug()
```

```
{
    inOrder(root);
    cout <<endl;
}

void update_rev(int r)
{
    if(!r)return ;
    swap(ch[r][0],ch[r][1]);
    rev[r] ^= 1;
}

void push_down(int r)
{
    if(rev[r])
    {
        update_rev(ch[r][0]);
        update_rev(ch[r][1]);
        rev[r] = 0;
    }
    if(add[r])
    {
        update_add(ch[r][0],add[r]);
        update_add(ch[r][1],add[r]);
        add[r] = 0;
    }
}


void NewNode(int &r,int far,int k)
{
    r = ++tot;
    pre[r] = far;
    ch[r][0] = ch[r][1] = 0;
    siz[r] = 1;
    Min[r] = k;
    key[r] = k;
    rev[r] = 0;
    add[r] = 0;
}


void rotat(int x,int kind)
{
    int y = pre[x];
    push_down(y);
```

```
93      push_down(x);
94      ch[y][!kind] = ch[x][kind];
95      pre[ch[x][kind]] = y;
96      if(pre[y])
97          ch[pre[y]][ch[pre[y]][1]==y] = x;
98      pre[x] = pre[y];
99      ch[x][kind] = y;
100     pre[y] = x;
101     push_up(y);
102 }
103
104 void build(int &x,int l,int r,int far)
105 {
106     if(l > r) return ;
107     int mid = (l+r) >>1;
108     NewNode(x,far,a[mid]);
109     build(ch[x][0],l,mid-1,x);
110     build(ch[x][1],mid+1,r,x);
111     push_up(x);
112 }
113
114 void splay(int r,int goal)
115 {
116     push_down(r);
117     while(pre[r] != goal)
118     {
119         if(pre[pre[r]] == goal)
120         {
121             push_down(pre[r]);
122             push_down(r);
123             rotat(r,ch[pre[r]][0] == r);
124         }
125         else
126         {
127             push_down(pre[pre[r]]);
128             push_down(pre[r]);
129             push_down(r);
130             int y = pre[r];
131             int kind = ch[pre[y]][0] == y;
132             if(ch[y][kind] == r)
133             {
134                 rotat(r,!kind);
135                 rotat(r,kind);
136             }
137             else
138             {
```

```
                rotat(y,kind);
                rotat(r,kind);
            }
        }
    }
    push_up(r);
    if(goal == 0)
        root = r;
}


int get_kth(int r,int k)
{
    push_down(r);
    int t = siz[ch[r][0]] + 1;
    if(k == t)return r;
    if(t > k) return get_kth(ch[r][0],k);
    else return get_kth(ch[r][1],k-t);
}

int get_next(int r)
{
    push_down(r);
    if(ch[r][1] == 0)return -1;
    r = ch[r][1];
    while(ch[r][0])
    {
        r = ch[r][0];
        push_down(r);
    }
    return r;
}

void Reverse(int l,int r)
{
    splay(get_kth(root,l),0);
    splay(get_kth(root,r+2),root);
    update_rev(key_value);
    push_up(ch[root][1]);
    push_up(root);
}

void Add(int l,int r,int val)
{
    splay(get_kth(root,l),0);
    splay(get_kth(root,r+2),root);
```

```
185     update_add(key_value,val);
186     push_up(ch[root][1]);
187     push_up(root);
188 }
189
190 void ini(int n)
191 {
192     tot = root = 0;
193     ch[root][0] = ch[root][1] = pre[root] = siz[root] = num[root] = 0;
194     Min[root] = 0x3f3f3f3f;
195     rev[root] = add[root] = 0;
196     NewNode(root,0,-1);
197     NewNode(ch[root][1],root,-1);
198     for(int i=1; i <= n; i++)
199     {
200         scanf("%d",&a[i]);
201     }
202     build(key_value,1,n,ch[root][1]);
203
204     push_up(ch[root][1]);
205     push_up(root);
206 }
207
208 int get_min(int r)
209 {
210     push_down(r);
211     while(ch[r][0])
212     {
213         r = ch[r][0];
214         push_down(r);
215     }
216     return r;
217 }
218
219 int Delete(int r)
220 {
221     int t = get_kth(root,r+1);
222     splay(t,0);
223     if(ch[root][0] == 0 || ch[root][1] == 0)
224     {
225         root = ch[root][0] + ch[root][1];
226         pre[root] = 0;
227         return key[t];
228     }
229     int k = get_min(ch[root][1]);
230     splay(k,root);
```

```
231      ch[ch[root][1]][0] = ch[root][0];
232      root = ch[root][1];
233      pre[ch[root][0]] = root;
234      pre[root] = 0;
235      push_up(root);
236      n--;
237      return key[t];
238  }
239
240  void Insert(int x,int y)
241  {
242      splay(get_kth(root,x),0);
243      splay(get_kth(root,x+1),root);
244      NewNode(key_value,ch[root][1],y);
245      push_up(ch[root][1]);
246      push_up(root);
247      n++;
248  }
249
250  void Move(int x)
251  {
252     if(x == 1)
253     {
254         int t = Delete(n);
255         Insert(1,t);
256  //         debug();
257     }
258     else
259     {
260         int t = Delete(1);
261         Insert(n+1,t);
262     }
263  }
264
265
266  int main()
267  {
268      int p,k1,k2;
269      int cas = 1;
270      while(scanf("%d%d%d%d",&n,&p,&k1,&k2) != EOF)
271      {
272          if(!n && !p && !k1 && !k2)
273              break;
274          printf("Case #%d:\n",cas++);
275          ini(n);
276          char opr[10];
```

```
277             int x;
278             for(int i =1; i <= p; i++)
279             {
280                 scanf("%s",opr);
281                 if(opr[0] == 'a')
282                 {
283                     scanf("%d",&x);
284                     Add(1,k2,x);
285                 }
286                 else if(opr[0] == 'm')
287                 {
288                     scanf("%d",&x);
289                     Move(x);
290                 }
291                 else if(opr[0] == 'r')
292                 {
293                     Reverse(1,k1);
294                 }
295                 else if(opr[0] == 'q')
296                 {
297                     printf("%d\n",key[get_kth(root,2)]);
298                 }
299                 else if(opr[0] == 'i')
300                 {
301                     scanf("%d",&x);
302                     Insert(2,x);
303                 }
304                 else if(opr[0] == 'd')
305                 {
306                     Delete(1);
307                 }
308             }
309         }
310         return 0;
311     }
```

Week 10 课下

B 普通平衡树

```
1   #include <iostream>
2   #include <cstring>
3   using namespace std;
```

```cpp

const int N = 1e5 + 10, INF = 1e8;
struct Node {
    int son[2];
    int key, val;
    int cnt, size;
} tr[N];
int n, root, idx;

int get_node(int key) {
    tr[++idx].key = key;
    tr[idx].val = rand();
    tr[idx].cnt = tr[idx].size = 1;
    return idx;
}

void pushup(int p) {
    tr[p].size = tr[tr[p].son[0]].size + tr[tr[p].son[1]].size +
tr[p].cnt;
}

void rotate(int& p, int d) {
    int q = tr[p].son[d ^ 1];
    tr[p].son[d ^ 1] = tr[q].son[d], tr[q].son[d] = p, p = q;
    pushup(tr[p].son[d]);
}

void insert(int& p, int key) {
    if (!p) p = get_node(key);
    else if (key < tr[p].key) {
        insert(tr[p].son[0], key);
        if (tr[tr[p].son[0]].val > tr[p].val) rotate(p, 1);
    } else if (key > tr[p].key) {
        insert(tr[p].son[1], key);
        if (tr[tr[p].son[1]].val > tr[p].val) rotate(p, 0);
    } else tr[p].cnt++;

    pushup(p);
}

void remove(int& p, int key) {
    if (!p) return;
    if (key < tr[p].key) remove(tr[p].son[0], key);
    else if (key > tr[p].key) remove(tr[p].son[1], key);
    else if (tr[p].cnt > 1) tr[p].cnt--;
    else if (!tr[p].son[0] && !tr[p].son[1]) p = 0;
```

```
49      else if (!tr[p].son[1] || (tr[p].son[0] && tr[tr[p].son[0]].val >
    tr[tr[p].son[1]].val)) {
50          rotate(p, 1);
51          remove(tr[p].son[1], key);
52      } else {
53          rotate(p, 0);
54          remove(tr[p].son[0], key);
55      }
56
57      pushup(p);
58  }
59
60  int get_rank_by_key(int p, int key) {
61      if (!p) return 0;
62      if (key < tr[p].key) return get_rank_by_key(tr[p].son[0], key);
63      else if (key > tr[p].key) return tr[tr[p].son[0]].size + tr[p].cnt +
    get_rank_by_key(tr[p].son[1], key);
64      else return tr[tr[p].son[0]].size + 1;
65  }
66
67  int get_key_by_rank(int p, int rank) {
68      if (!p) return INF;
69      else if (rank <= tr[tr[p].son[0]].size) return
    get_key_by_rank(tr[p].son[0], rank);
70      else if (rank > tr[tr[p].son[0]].size + tr[p].cnt) return
    get_key_by_rank(tr[p].son[1], rank - tr[tr[p].son[0]].size - tr[p].cnt);
71      else return tr[p].key;
72  }
73
74  int get_prev(int p, int key) {
75      if (!p) return -INF;
76      if (key <= tr[p].key) return get_prev(tr[p].son[0], key);
77      else return max(tr[p].key, get_prev(tr[p].son[1], key));
78  }
79
80  int get_next(int p, int key) {
81      if (!p) return INF;
82      if (key >= tr[p].key) return get_next(tr[p].son[1], key);
83      else return min(tr[p].key, get_next(tr[p].son[0], key));
84  }
85
86  int main() {
87      scanf("%d", &n);
88      while (n--) {
89          int op, x;
90          scanf("%d%d", &op, &x);
```

```
91          if (op == 1) insert(root, x);
92          else if (op == 2) remove(root, x);
93          else if (op == 3) printf("%d\n", get_rank_by_key(root, x));
94          else if (op == 4) printf("%d\n", get_key_by_rank(root, x));
95          else if (op == 5) printf("%d\n", get_prev(root, x));
96          else if (op == 6) printf("%d\n", get_next(root, x));
97      }
98
99      return 0;
100 }
```

D 对称二叉树

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int N=1000010;
4   long long a[N],l[N],r[N],n,ans=0;
5   inline bool dfs(long long l2,long long r2){
6       if(l2==-1&&r2==-1)
7           return true;
8       if(l2==-1||r2==-1||a[l2]!=a[r2])
9           return false;
10      return dfs(l[l2],r[r2])&&dfs(l[r2],r[l2]);
11  }
12  inline long long get(long long x){
13      if(x==-1)
14          return 0;
15      else
16          return get(l[x])+get(r[x])+1;
17  }
18  int main(){
19      cin>>n;
20      for(long long i=1;i<=n;i++)
21          cin>>a[i];
22      for(long long i=1;i<=n;i++)
23          cin>>l[i]>>r[i];
24      for(long long i=1;i<=n;i++)
25          if(dfs(i,i))
26              ans=max(ans,get(i));
27      cout<<ans;
28      return 0;
29  }
```

E 完全二叉树的权值

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

typedef long long LL;

const int N = 1e5 + 5;

LL tr[N];
LL cnt[50];

int main()
{
    int n, k = 1;scanf("%d", &n);
    for(int i = 1; i <= n; ++ i) scanf("%lld", &tr[i]);

    for(int i = 1, j = 1; i <= n; ++ i)
    {
        cnt[k] += tr[i];

        if(i - j >= (1 << (k - 1)) - 1)
        {
            j = i + 1;
            k ++;
        }
    }

    int ans = 0, maxx = 1-1e6;
    for(int i = 1; i <= k ; ++ i)
        if(cnt[i] > maxx)
        {
            maxx = cnt[i];
            ans = i;
        }
    cout << ans <<endl;
    return 0;

}
```

Week 12 课上

A Root of AVL Tree

```cpp
#include <iostream>
using namespace std;

struct Node {
    int key;
    Node *left, *right;
    int height;
    Node(int k) : key(k), left(nullptr), right(nullptr), height(1) {}
};

int height(Node* n) {
    return n ? n->height : 0;
}

void updateHeight(Node* n) {
    n->height = max(height(n->left), height(n->right)) + 1;
}

int balanceFactor(Node* n) {
    return height(n->left) - height(n->right);
}

Node* rightRotate(Node* y) {
    Node* x = y->left;
    Node* T2 = x->right;
    x->right = y;
    y->left = T2;
    updateHeight(y);
    updateHeight(x);
    return x;
}

Node* leftRotate(Node* x) {
    Node* y = x->right;
    Node* T2 = y->left;
    y->left = x;
    x->right = T2;
    updateHeight(x);
    updateHeight(y);
    return y;
}

Node* insert(Node* node, int key) {
    if (!node)
```

```
        return new Node(key);
    if (key < node->key)
        node->left = insert(node->left, key);
    else
        node->right = insert(node->right, key);

    updateHeight(node);

    int bf = balanceFactor(node);

    if (bf > 1 && key < node->left->key)
        return rightRotate(node);
    if (bf < -1 && key > node->right->key)
        return leftRotate(node);
    if (bf > 1 && key > node->left->key) {
        node->left = leftRotate(node->left);
        return rightRotate(node);
    }
    if (bf < -1 && key < node->right->key) {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }

    return node;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int N;
    cin >> N;
    Node* root = nullptr;
    for (int i = 0; i < N; i++) {
        int x;
        cin >> x;
        root = insert(root, x);
    }
    if (root)
        cout << root->key << "\n";
    return 0;
}
```

B B树的插入遍历和查找

```
#include<iostream>
```

```cpp
#include<vector>
using namespace std;

struct BTreeNode
{
    int *keys; // 元素数组
    int t; // 最小度t
    BTreeNode **C; // 子节点数组
    int n;      // 当前节点元素个数
    bool leaf; // 是否是叶子节点

    BTreeNode(int _t, bool _leaf); // 构造函数
    void insertNonFull(int k); // 当前节点未满，插入
    void splitChild(int i, BTreeNode *y); // 分裂节点
    void traverse(); // 遍历
    BTreeNode *search(int k); // 查找
};

class BTree
{
    BTreeNode *root; // B树根节点
    int t; // 最小度t
public:
    BTree(int _t)
    {  root = NULL;  t = _t; }

    void traverse()
    {  if (root != NULL) root->traverse(); }

    BTreeNode* search(int k)
    {  return (root == NULL)? NULL : root->search(k); }

    void insert(int k);
};

BTreeNode::BTreeNode(int t1, bool leaf1)
{
    t = t1;
    leaf = leaf1;

    keys = new int[2*t-1];
    C = new BTreeNode *[2*t];

    n = 0;
}

```

```
void BTreeNode::traverse()
{
    int i;
    for (i = 0; i < n; i++)
    {
        if (leaf == false)
            C[i]->traverse();
        cout << keys[i] << endl;
    }
    if (leaf == false)
        C[i]->traverse();
}

BTreeNode *BTreeNode::search(int k)
{
    int i = 0;
    while (i < n && k > keys[i])
        i++;
    if (i < n && keys[i] == k)
        return this;
    if (leaf)
        return NULL;
    return C[i]->search(k);
}

void BTree::insert(int k)
{
    if (root == NULL)
    {
        root = new BTreeNode(t, true);
        root->keys[0] = k;
        root->n = 1;
    }
    else
    {
        if (root->n == 2*t-1)
        {
            BTreeNode *s = new BTreeNode(t, false);
            s->C[0] = root;
            s->splitChild(0, root);
            int i = 0;
            if (s->keys[0] < k)
                i++;
            s->C[i]->insertNonFull(k);
            root = s;
        }
```

```
        else
            root->insertNonFull(k);
    }
}

void BTreeNode::insertNonFull(int k)
{
    int i = n-1;

    if (leaf == true)
    {
        while (i >= 0 && keys[i] > k)
        {
            keys[i+1] = keys[i];
            i--;
        }

        keys[i+1] = k;
        n = n+1;
    }
    else
    {
        while (i >= 0 && keys[i] > k)
            i--;

        if (C[i+1]->n == 2*t-1)
        {
            splitChild(i+1, C[i+1]);

            if (keys[i+1] < k)
                i++;
        }
        C[i+1]->insertNonFull(k);
    }
}

void BTreeNode::splitChild(int i, BTreeNode *y)
{
    BTreeNode *z = new BTreeNode(y->t, y->leaf);
    z->n = t - 1;

    for (int j = 0; j < t-1; j++)
        z->keys[j] = y->keys[j+t];

    if (y->leaf == false)
    {
```

```
        for (int j = 0; j < t; j++)
            z->C[j] = y->C[j+t];
    }

    y->n = t - 1;
    for (int j = n; j >= i+1; j--)
        C[j+1] = C[j];

    C[i+1] = z;

    for (int j = n-1; j >= i; j--)
        keys[j+1] = keys[j];

    keys[i] = y->keys[t-1];

    n = n + 1;
}

int main()
{
    int t, n, m;
    cin>>t>>n>>m;
    BTree tree(t);
    vector<int> a(n);
    for (int i = 0; i < n; i++)
    {
        cin>>a[i];
        tree.insert(a[i]);
    }

    tree.traverse();

    for (int i = 0; i < m; i++)
    {
        int q;
        cin>>q;
        BTreeNode *res = tree.search(q);
        if (res)
        {
            for (int j = 0; j < res->n; j++)
            {
                cout << res->keys[j];
                if (j < res->n - 1) cout << ' ';
            }
            cout << endl;
        }
```

```
    }
    return 0;
}
```

C 邻接矩阵的使用

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m;
    cin >> n >> m;

    vector<vector<int>> adj_matrix(n, vector<int>(n, 0));

    for (int k = 0; k < m; ++k) {
        int type, u, v;
        cin >> type >> u >> v;
        if (type == 0) {
            adj_matrix[u][v] = 1;
        } else {
            adj_matrix[u][v] = 1;
            adj_matrix[v][u] = 1;
        }
    }

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            cout << adj_matrix[i][j] << (j == n - 1 ? "" : " ");
        }
        cout << "\n";
    }

    return 0;
}
```

D 邻接表的使用

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 100 + 5;
```

```cpp
vector<int> G[MAXN];

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;
    int a, x, y;
    for (int i = 0; i < m; i++) {
        cin >> a >> x >> y;
        if (a == 0) {
            G[x].push_back(y);
        } else {
            G[x].push_back(y);
            G[y].push_back(x);
        }
    }

    for (int i = 0; i < n; i++) {
        cout << i << ":";
        for (int j = (int)G[i].size() - 1; j >= 0; j--) {
            cout << " " << G[i][j];
        }
        cout << "\n";
    }

    return 0;
}
```

E 画图游戏

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    vector<int> deg(n);
    long long sum = 0;
    for (int i = 0; i < n; i++) {
        cin >> deg[i];
        sum += deg[i];
    }
    if (sum % 2) {
```

```
        cout << "None\n";
        return 0;
    }
    vector<vector<int>> adj(n, vector<int>(n, 0));
    vector<pair<int,int>> v(n);
    for (int i = 0; i < n; i++) {
        v[i] = {deg[i], i};
    }
    for (int round = 0; round < n; round++) {
        sort(v.begin() + round, v.end(), [&](auto &a, auto &b) {
            return a.first > b.first;
        });
        int d = v[round].first;
        int u = v[round].second;
        if (d < 0 || d > n - 1 - round) {
            cout << "None\n";
            return 0;
        }
        for (int k = 1; k <= d; k++) {
            int vi = v[round + k].second;
            if (v[round + k].first <= 0) {
                cout << "None\n";
                return 0;
            }
            adj[u][vi] = adj[vi][u] = 1;
            v[round + k].first--;
        }
        v[round].first = 0;
    }
    for (int i = 0; i < n; i++) {
        if (v[i].first != 0) {
            cout << "None\n";
            return 0;
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << adj[i][j] << (j + 1 < n ? ' ' : '\n');
        }
    }
    return 0;
}
```

Week 12 课下

也就多了一题

A AVL Tree

```cpp
#include <bits/stdc++.h>
using namespace std;

int a[46], n;

int main() {
    a[0] = 1;
    a[1] = 2;
    for (int i = 2; i <= 45; i++)
        a[i] = a[i - 1] + a[i - 2] + 1;
    while (~scanf("%d", &n), n) {
        int ans = 0;
        while (a[ans] <= n) ans++;
        printf("%d\n", --ans);
    }
    return 0;
}
```

Week 13 课上

A 找出星型图的中心节点

```cpp
#include <bits/stdc++.h>

using namespace std;

int main(void) {
    int n, m;
    int count[100055] = {0};
    cin >> n >> m;

    for (int i = 0; i < m; i++) {
        int a, b;
        cin >> a >> b;
        count[a]++;
        count[b]++;
        if (count[a] == m) {
            cout << a;
            return 0;
```

```
        }
        if (count[b] == m) {
            cout << b;
            return 0;
        }
    }
    for (int i = 1; i <= n; i++) {
        if (count[i] == n - 1) {
            cout << i;
            return 0;
        }
    }
}
```

B 修建大桥

```
#include <bits/stdc++.h>

using namespace std;

vector<int> graph[1000];
bool vst[1000];

void dfs(int node) {
    vst[node] = true;

    for (int i = 0; i < graph[node].size(); i++) {
        int nextNode = graph[node][i];

        if (!vst[nextNode]) {
            dfs(nextNode);
        }
    }
}

int main() {
    int n, m;
    memset(vst, false, sizeof(vst));
    cin >> n >> m;

    for (int i = 0; i < m; i++) {
        int a, b;
        cin >> a >> b;
        graph[a].push_back(b);
        graph[b].push_back(a);
    }
```

```
    int components = 0;
    for (int i = 1; i <= n; i++) {
        if (!vst[i]) {
            components++;
            dfs(i);
        }
    }
    cout << components - 1;

    return 0;
}
```

C 闯关游戏

```
#include <bits/stdc++.h>
using namespace std;

struct Node {
    int ki;
    vector<int> adj;
};

int main() {
    int n;
    cin >> n;
    vector<Node> nodes(n);
    for (int i = 0; i < n; ++i) {
        int ki, m;
        cin >> ki >> m;
        vector<int> adj(m);
        for (int j = 0; j < m; ++j) {
            cin >> adj[j];
        }
        nodes[i] = {ki, adj};
    }

    if (n == 1) {
        cout << "Yes" << endl;
        return 0;
    }

    vector<vector<int>> adj_forward(n + 1);
    for (int u = 1; u <= n; ++u) {
        for (int v : nodes[u - 1].adj) {
            adj_forward[u].push_back(v);
```

```
        }
    }

    vector<vector<int>> adj_backward(n + 1);
    for (int u = 1; u <= n; ++u) {
        for (int v : adj_forward[u]) {
            adj_backward[v].push_back(u);
        }
    }

    bool in_reachable_from_start[101] = {false};
    queue<int> q;
    q.push(1);
    in_reachable_from_start[1] = true;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (int v : adj_forward[u]) {
            if (!in_reachable_from_start[v]) {
                in_reachable_from_start[v] = true;
                q.push(v);
            }
        }
    }

    bool in_can_reach_end[101] = {false};
    queue<int> q2;
    q2.push(n);
    in_can_reach_end[n] = true;
    while (!q2.empty()) {
        int v = q2.front();
        q2.pop();
        for (int u : adj_backward[v]) {
            if (!in_can_reach_end[u]) {
                in_can_reach_end[u] = true;
                q2.push(u);
            }
        }
    }

    vector<pair<int, int>> edges;
    for (int u = 1; u <= n; ++u) {
        if (in_reachable_from_start[u] && in_can_reach_end[u]) {
            for (int v : nodes[u - 1].adj) {
                if (in_can_reach_end[v]) {
                    edges.emplace_back(u, v);
```

```
                }
            }
        }
    }

    int initial = 100 + nodes[0].ki;
    if (initial <= 0) {
        cout << "No" << endl;
        return 0;
    }

    vector<int> max_energy(n + 1, INT_MIN);
    max_energy[1] = initial;

    for (int i = 0; i < n; ++i) {
        bool updated = false;
        for (auto& e : edges) {
            int u = e.first, v = e.second;
            if (max_energy[u] > 0) {
                int new_energy = max_energy[u] + nodes[v - 1].ki;
                if (new_energy > max_energy[v] && new_energy > 0) {
                    max_energy[v] = new_energy;
                    updated = true;
                }
            }
        }
        if (!updated) break;
    }

    bool has_positive_cycle = false;
    for (auto& e : edges) {
        int u = e.first, v = e.second;
        if (max_energy[u] > 0) {
            int new_energy = max_energy[u] + nodes[v - 1].ki;
            if (new_energy > max_energy[v] && new_energy > 0) {
                has_positive_cycle = true;
                break;
            }
        }
    }

    if (has_positive_cycle) {
        cout << "Yes" << endl;
        return 0;
    }

```

```
    if (max_energy[n] > 0) {
        cout << "Yes" << endl;
    } else {
        cout << "No" << endl;
    }

    return 0;
}
```

D 圣诞树

```
#include <bits/stdc++.h>

using namespace std;

using i64 = long long;
typedef long long ll;

const int maxn = 5e4 + 6;
const int maxm = 5e4 + 6;

struct edge {
    int to, len;
};

vector<edge> e[maxn];

struct node {
    i64 dis;
    int num;
    bool operator>(const node &a) const {
        return dis > a.dis;
    }
};

i64 minDis[maxn];
bool vis[maxn];
priority_queue<node, vector<node>, greater<node>> pq;

void dijkstra(int n, int s) {
    for (int i = 1; i <= n; i++) {
        minDis[i] = 1e10;
    }
    minDis[s] = 0;
    pq.push({0, s});
    while (!pq.empty()) {
```

```
        int u = pq.top().num;
        pq.pop();
        if (vis[u]) continue;
        vis[u] = 1;
        for (edge eg : e[u]) {
            int v = eg.to;
            int w = eg.len;
            if (minDis[v] > minDis[u] + w) {
                minDis[v] = minDis[u] + w;
                pq.push({minDis[v], v});
            }
        }
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, s;
    cin >> n >> m;
    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
    s = 1;
    int u, v, w;
    while (m--) {
        cin >> u >> v >> w;
        e[u].push_back({v, w});
        e[v].push_back({u, w});
    }
    dijkstra(n, s);
    i64 ans = 0;
    bool ok = 1;
    for (int i = 1; i <= n; i++) {
        if (minDis[i] == 1e10) {
            ok = 0;
            break;
        }
        ans += a[i] * minDis[i];
    }
    if (ok)
        cout << ans << '\n';
    else
        cout << "No Answer\n";
```

```
    return 0;
}
```

E 网络延时

```
#include<bits/stdc++.h>

using namespace std;

vector<int> graph[1000];
int dist[1000];

void shortestPath(int start) {
    memset(dist, -1, sizeof(dist));
    queue<int> nodes;
    nodes.push(start);
    dist[start] = 0;

    while (!nodes.empty()) {
        int cur = nodes.front();
        nodes.pop();

        for (int i = 0; i < graph[cur].size(); i++) {
            int next = graph[cur][i];

            if (dist[next] == -1) {
                nodes.push(next);
                dist[next] = dist[cur] + 1;
            }
        }
    }
}

int main() {
    int N;
    int maxNode;
    int maxPath = 0;
    cin >> N;

    for (int i = 0; i < N - 1; i++) {
        int a, b;
        cin >> a >> b;
        graph[a].push_back(b);
        graph[b].push_back(a);
    }
```

```
    shortestPath(1);
    for (int i = 1; i <= N; i++) {
        if (dist[i] > maxPath) {
            maxPath = dist[i];
            maxNode = i;
        }
    }
    shortestPath(maxNode);
    for (int i = 1; i <= N; i++) {
        if (dist[i] > maxPath) {
            maxPath = dist[i];
        }
    }

    cout << maxPath << endl;

    return 0;
}
```

Week 13 课下

Week 14 课上

A 小明的训练室

```
#include <iostream>
#include <vector>
#include <queue>
#include <limits>
using namespace std;

const int INF = numeric_limits<int>::max();

struct Edge {
    int to;
    int h;
};

void dijkstra(int s, int v, const vector<vector<Edge>>& g, vector<int>&
mh) {
```

```
    priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int, int>>> pq;
    pq.push({0, s});
    mh[s] = 0;

    while (!pq.empty()) {
        int u = pq.top().second;
        int curr_h = pq.top().first;
        pq.pop();

        if (curr_h > mh[u]) continue;

        for (const auto& e : g[u]) {
            int next_v = e.to;
            int next_h = e.h;

            if (mh[next_v] > max(mh[u], next_h)) {
                mh[next_v] = max(mh[u], next_h);
                pq.push({mh[next_v], next_v});
            }
        }
    }
}

int main() {
    int n, m, t;
    cin >> n >> m >> t;

    vector<vector<Edge>> g(n + 1);

    for (int i = 0; i < m; ++i) {
        int s, e, h;
        cin >> s >> e >> h;
        g[s].push_back({e, h});
    }

    while (t--) {
        int a, b;
        cin >> a >> b;

        vector<int> mh(n + 1, INF);

        dijkstra(a, n, g, mh);

        int res = mh[b];
        if (res == INF) cout << "-1" << endl;
```

```
        else cout << res << endl;
    }

    return 0;
}
```

B 节点的最近公共祖先

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

const int MAXN = 10001;
const int MAXLOG = 15;

vector<int> adj[MAXN];
int parent[MAXN];
int depth[MAXN];
int ancestor[MAXN][MAXLOG];

void dfs(int node, int par, int dep) {
    parent[node] = par;
    depth[node] = dep;
    for (int i = 0; i < adj[node].size(); ++i) {
        int child = adj[node][i];
        if (child != par) {
            dfs(child, node, dep + 1);
        }
    }
}

void binaryLift(int N) {
    for (int i = 1; i <= N; ++i) {
        ancestor[i][0] = parent[i];
    }

    for (int j = 1; (1 << j) <= N; ++j) {
        for (int i = 1; i <= N; ++i) {
            if (ancestor[i][j - 1] != -1) {
                ancestor[i][j] = ancestor[ancestor[i][j - 1]][j - 1];
            }
        }
    }
}

```

```cpp
int findLCA(int u, int v) {
    if (depth[u] < depth[v]) swap(u, v);

    int log;
    for (log = 1; (1 << log) <= depth[u]; ++log);
    log--;

    for (int i = log; i >= 0; --i) {
        if (depth[u] - (1 << i) >= depth[v]) {
            u = ancestor[u][i];
        }
    }

    if (u == v) return u;

    for (int i = log; i >= 0; --i) {
        if (ancestor[u][i] != -1 && ancestor[u][i] != ancestor[v][i]) {
            u = ancestor[u][i];
            v = ancestor[v][i];
        }
    }

    return parent[u];
}

int main() {
    int N, M, S;
    cin >> N >> M >> S;

    for (int i = 1; i <= N; ++i) {
        adj[i].clear();
        parent[i] = -1;
        depth[i] = 0;
        for (int j = 0; j < MAXLOG; ++j) {
            ancestor[i][j] = -1;
        }
    }

    for (int i = 1; i < N; ++i) {
        int x, y;
        cin >> x >> y;
        adj[x].push_back(y);
        adj[y].push_back(x);
    }

    dfs(S, -1, 0);
```

```
    binaryLift(N);

    for (int i = 0; i < M; ++i) {
        int a, b;
        cin >> a >> b;
        int lca = findLCA(a, b);
        cout << lca << endl;
    }

    return 0;
}
```

C 布设光纤

```
#include <iostream>
#include <vector>
#include <climits>

using namespace std;

int main() {
    int n;
    cin >> n;
    vector<vector<int>> adj(n, vector<int>(n));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            cin >> adj[i][j];
        }
    }

    vector<int> lowcost(n);
    vector<bool> visited(n, false);
    for (int i = 0; i < n; ++i) {
        lowcost[i] = adj[0][i];
    }
    visited[0] = true;
    int sum = 0;

    for (int cnt = 1; cnt < n; ++cnt) {
        int min_val = INT_MAX;
        int min_idx = -1;
        for (int j = 0; j < n; ++j) {
            if (!visited[j] && lowcost[j] < min_val) {
                min_val = lowcost[j];
                min_idx = j;
            }
```

```
        }
        if (min_idx == -1) {
            break;
        }
        sum += min_val;
        visited[min_idx] = true;
        for (int j = 0; j < n; ++j) {
            if (!visited[j] && adj[min_idx][j] < lowcost[j]) {
                lowcost[j] = adj[min_idx][j];
            }
        }
    }

    cout << sum << endl;

    return 0;
}
```

D 丁香之路

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int MAXN = 2505;
struct Edge { int u, v, w; };
vector<Edge> G;
int f[MAXN], degree[MAXN], n, m, s, u, v, sum;
int ff[MAXN];

void init(int n) {
    for (int i = 1; i <= n; ++i) f[i] = i;
}

int findf(int i) {
    return f[i] == i ? f[i] : f[i] = findf(f[i]);
}

void merge(int i, int j) {
    f[findf(i)] = findf(j);
}

bool cmp(const Edge &a, const Edge &b) {
    return a.w < b.w;
}

int main() {
```

```
    ios::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    cin >> n >> m >> s;
    init(n);
    for (int i = 1; i <= m; ++i) {
        cin >> u >> v;
        degree[u]++, degree[v]++;
        sum += abs(u - v);
        merge(u, v);
    }
    degree[s]++;
    for (int i = 1; i <= n; ++i) ff[i] = f[i];
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) f[j] = ff[j];
        degree[i]++;
        int ans = sum, pre = 0;
        vector<int> V;
        for (int j = 1; j <= n; ++j) {
            if (degree[j]) V.push_back(j);
        }
        for (int j = 1; j <= n; ++j) {
            if (degree[j] & 1) {
                if (!pre) pre = j;
                else {
                    ans += (j - pre);
                    for (int k = pre + 1; k <= j; ++k) merge(k, k - 1);
                    pre = 0;
                }
            }
        }
        G.clear();
        for (int j = 0; j + 1 < V.size(); ++j) {
            if (findf(V[j]) != findf(V[j + 1])) {
                G.push_back({V[j], V[j + 1], V[j + 1] - V[j]});
            }
        }
        sort(G.begin(), G.end(), cmp);
        for (const auto &e : G) {
            if (findf(e.u) != findf(e.v)) {
                merge(e.u, e.v); ans += 2 * e.w;
            }
        }
        degree[i]--;
        cout << ans << " ";
    }
    return 0;
```

```
}
```

E 威虎山上的分配

```
#include<bits/stdc++.h>
using namespace std;

const int MAXN = 10001;
const int MAXM = 20001;

int cnt, money[MAXN];
bool vis[MAXN];
struct Edge {
    int v, next;
    int len;
} E[MAXM];

int p[MAXN], eid = 1;

void insert(int u, int v) {
    E[eid].v = v;
    E[eid].next = p[u];
    p[u] = eid++;
}

int n, m;
int indegree[MAXN];

void topo() {
    queue<int> q;
    for (int i = 1; i <= n; i++) {
        if (indegree[i] == 0) {
            q.push(i);
            vis[i] = true;
        }
    }
    while (!q.empty()) {
        int now = q.front();
        q.pop();
        cnt++;
        for (int i = p[now]; i; i = E[i].next) {
            int v = E[i].v;
            if (--indegree[v] == 0) {
                q.push(v);
                vis[v] = true;
                money[v] = money[now] + 1;
```

```
            }
        }
    }
}

int main() {
    memset(indegree, 0, sizeof(indegree));
    memset(money, 0, sizeof(money));
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        insert(v, u);
        indegree[u]++;
    }

    topo();
    int ans = 0;
    for (int i = 1; i <= n; i++) {
        if (!vis[i]) {
            cout << "Unhappy!" << endl;
            return 0;
        }
        ans += money[i];
    }

    cout << ans + n * 100 << endl;
    return 0;
}
```

F 判定欧拉回路

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1001;

vector<int> adj[MAXN];
int degree[MAXN];
bool visited[MAXN];

void dfs(int v) {
    visited[v] = true;
    for (int u : adj[v]) {
        if (!visited[u]) {
            dfs(u);
```

```
        }
    }
}

bool isConnected(int N) {
    memset(visited, false, sizeof(visited));
    int start = -1;
    for (int i = 1; i <= N; ++i) {
        if (degree[i] > 0) {
            start = i;
            break;
        }
    }
    if (start == -1) return true;
    dfs(start);

    for (int i = 1; i <= N; ++i) {
        if (degree[i] > 0 && !visited[i]) {
            return false;
        }
    }
    return true;
}

int main() {
    int N, M;
    cin >> N >> M;

    for (int i = 0; i < M; ++i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
        degree[u]++;
        degree[v]++;
    }

    if (!isConnected(N)) {
        cout << 0 << endl;
        return 0;
    }

    for (int i = 1; i <= N; ++i) {
        if (degree[i] % 2 != 0) {
            cout << 0 << endl;
            return 0;
```

```
        }
    }

    cout << 1 << endl;

    return 0;
}
```

G 商业信息共享

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<vector<int>> adj(n), adj_t(n);
    for (int i = 0; i < n; ++i) {
        int j;
        while (cin >> j && j != 0) {
            j--; // 转换为0-based索引
            adj[i].push_back(j);
            adj_t[j].push_back(i);
        }
    }

    vector<bool> visited(n, false);
    vector<int> order;
    function<void(int)> dfs1 = [&](int u) {
        visited[u] = true;
        for (int v : adj[u]) {
            if (!visited[v]) {
                dfs1(v);
            }
        }
        order.push_back(u);
    };

    for (int i = 0; i < n; ++i) {
        if (!visited[i]) {
            dfs1(i);
        }
    }

    reverse(order.begin(), order.end());

```

```cpp
    vector<int> component(n, -1);
    int current_component = 0;
    visited.assign(n, false);
    function<void(int)> dfs2 = [&](int u) {
        visited[u] = true;
        component[u] = current_component;
        for (int v : adj_t[u]) {
            if (!visited[v]) {
                dfs2(v);
            }
        }
    };

    for (int u : order) {
        if (!visited[u]) {
            dfs2(u);
            current_component++;
        }
    }

    int k = current_component;
    vector<int> indegree(k, 0), outdegree(k, 0);
    vector<vector<bool>> added(k, vector<bool>(k, false));

    for (int u = 0; u < n; ++u) {
        for (int v : adj[u]) {
            if (component[u] != component[v]) {
                int cu = component[u];
                int cv = component[v];
                if (!added[cu][cv]) {
                    added[cu][cv] = true;
                    outdegree[cu]++;
                    indegree[cv]++;
                }
            }
        }
    }

    int c1 = 0, c2 = 0;
    for (int i = 0; i < k; ++i) {
        if (indegree[i] == 0) c1++;
        if (outdegree[i] == 0) c2++;
    }

    int ans1 = c1;
    int ans2 = (k == 1) ? 0 : max(c1, c2);
```

```
83
84      cout << ans1 << endl;
85      cout << ans2 << endl;
86
87      return 0;
88  }
```

Week 14 课下

Week 15 课上

A 阿里天池的新任务

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int MAXN = 1000005;
4   int w[MAXN], nextArr[MAXN];
5   char s[MAXN], pattern[MAXN];
6   int total = 0;
7
8   void computeLPS(const char *pat, int m, int *lps) {
9       int len = 0;
10      lps[0] = 0;
11      int i = 1;
12      while (i < m) {
13          if (pat[i] == pat[len]) {
14              len++;
15              lps[i] = len;
16              i++;
17          } else {
18              if (len != 0) {
19                  len = lps[len - 1];
20              } else {
21                  lps[i] = 0;
22                  i++;
23              }
24          }
25      }
26  }
27
28  void kmpSearch(const char *text, const char *pat) {
```

```
    int n = strlen(text);
    int m = strlen(pat);
    computeLPS(pat, m, nextArr);
    int i = 0, j = 0;
    while (i < n) {
        if (pat[j] == text[i]) {
            i++;
            j++;
        }
        if (j == m) {
            total++;
            j = nextArr[j - 1];
        } else if (i < n && pat[j] != text[i]) {
            if (j != 0) {
                j = nextArr[j - 1];
            } else {
                i++;
            }
        }
    }
}

int main() {
    int n, a, b, l, r;
    cin >> n >> a >> b >> l >> r;
    cin >> pattern;
    for (int i = 0; i < n; ++i) {
        if (i == 0) {
            w[i] = b;
        } else {
            w[i] = (w[i - 1] + a) % n;
        }
        if (w[i] >= l && w[i] <= r) {
            s[i] = (w[i] % 2 == 0) ? 'A' : 'T';
        } else {
            s[i] = (w[i] % 2 == 0) ? 'G' : 'C';
        }
    }
    kmpSearch(s, pattern);
    cout << total;
    return 0;
}
```

B 猴子打字

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

const int ALPHABET_SIZE = 26;

struct Node {
    Node *children[ALPHABET_SIZE];
    Node *fail;
    int cnt;
    Node() {
        for (int i = 0; i < ALPHABET_SIZE; i++) {
            children[i] = nullptr;
        }
        fail = nullptr;
        cnt = 0;
    }
};

Node *root;

void insert(string word) {
    Node *p = root;
    for (char c : word) {
        int idx = c - 'a';
        if (p->children[idx] == nullptr) {
            p->children[idx] = new Node();
        }
        p = p->children[idx];
    }
    p->cnt++;
}

void build() {
    queue<Node*> q;
    root->fail = root;
    for (int i = 0; i < ALPHABET_SIZE; i++) {
        if (root->children[i] != nullptr) {
            root->children[i]->fail = root;
            q.push(root->children[i]);
        } else {
            root->children[i] = root;
        }
    }

    while (!q.empty()) {
        Node *u = q.front();
        q.pop();
```

```
        for (int i = 0; i < ALPHABET_SIZE; i++) {
            if (u->children[i] != nullptr) {
                Node *v = u->children[i];
                v->fail = u->fail->children[i];
                q.push(v);
            } else {
                u->children[i] = u->fail->children[i];
            }
        }
    }
}

long long query(string article) {
    long long ans = 0;
    Node *p = root;
    for (char c : article) {
        int idx = c - 'a';
        p = p->children[idx];
        Node *temp = p;
        while (temp != root) {
            ans += temp->cnt;
            temp = temp->fail;
        }
    }
    return ans;
}

int main() {
    int n;
    cin >> n;
    root = new Node();
    string word;
    for (int i = 0; i < n; i++) {
        cin >> word;
        insert(word);
    }
    build();
    string article;
    cin >> article;
    cout << query(article) << endl;
    return 0;
}
```

C 糟糕的 Bug

```
#include <iostream>
```

```cpp
#include <cstdio>
#include <cstring>

using namespace std;

#define MAX_N 2333430
#define MAX_C 26

struct Trie {
    int *ch[MAX_N];
    int tot;
    int cnt[MAX_N];

    Trie() {
        tot = 0;
        memset(ch, 0, sizeof(ch));
        memset(cnt, 0, sizeof(cnt));
    }

    void insert(const char *str) {
        int p = 0;
        for (int i = 0; str[i]; ++i) {
            if (ch[p] == NULL) {
                ch[p] = new int[MAX_C];
                memset(ch[p], -1, sizeof(int) * MAX_C);
            }
            if (ch[p][str[i] - 'a'] == -1) {
                ch[p][str[i] - 'a'] = ++tot;
            }
            p = ch[p][str[i] - 'a'];
        }
        cnt[p]++;
    }

    bool find(const char *str) {
        int p = 0;
        for (int i = 0; str[i]; ++i) {
            if (cnt[p] != 0) return true;
            if (ch[p] == NULL) return false;
            if (ch[p][str[i] - 'a'] == -1) return false;
            p = ch[p][str[i] - 'a'];
        }
        return false;
    }
};

```

```
char s[MAX_N][15];
Trie trie;

int main() {
    int n;
    scanf("%d", &n);
    getchar();
    bool ans = false;
    for (int i = 0; i < n; ++i) {
        scanf("%s", s[i]);
        getchar();
        trie.insert(s[i]);
    }
    for (int i = 0; i < n; ++i) {
        if (trie.find(s[i])) {
            ans = true;
            break;
        }
    }
    if (ans) {
        puts("Bug!");
    } else {
        puts("Good Luck!");
    }
    return 0;
}
```

D 首尾相接

```
#include <bits/stdc++.h>
using namespace std;

vector<int> getNext(const string& p) {
    int m = p.size();
    vector<int> next(m + 1, -1);
    int i = 0, j = -1;
    while (i < m) {
        if (j == -1 || p[i] == p[j]) {
            i++;
            j++;
            next[i] = j;
        } else {
            j = next[j];
        }
    }
    return next;
```

```cpp
}

int main() {
    string s1, s2;
    getline(cin, s1);
    getline(cin, s2);

    int n = s2.size();
    int m = s1.size();
    if (m == 0 || n == 0) {
        cout << 0 << endl;
        return 0;
    }

    vector<int> next = getNext(s1);
    int j = 0, i = 0;
    int overlap = 0;

    while (i < n) {
        if (j == m) {
            j = next[j];
        }
        if (j == -1 || s2[i] == s1[j]) {
            j++;
            i++;
            if (i == n) {
                overlap = j;
            }
        } else {
            j = next[j];
        }
    }

    if (overlap > 0) {
        cout << s1.substr(0, overlap) << " " << overlap << endl;
    } else {
        cout << 0 << endl;
    }

    return 0;
}
```

Week 15 课下

E 旋转数字

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
    string s;
    cin >> s;
    int n = s.size();

    vector<int> next(n, 0);
    for (int i = 1; i < n; i++) {
        int j = next[i-1];
        while (j > 0 && s[i] != s[j]) {
            j = next[j-1];
        }
        if (s[i] == s[j]) {
            j++;
        }
        next[i] = j;
    }

    int T = n;
    if (next[n-1] != 0 && n % (n - next[n-1]) == 0) {
        T = n - next[n-1];
    }

    string ss = s + s;
    int n2 = 2 * n;
    vector<int> Z(n2, 0);
    if (n2 > 0) {
        Z[0] = n2;
        int l = 0, r = 0;
        for (int i = 1; i < n2; i++) {
            if (i <= r) {
                Z[i] = min(r - i + 1, Z[i - l]);
            }
            while (i + Z[i] < n2 && ss[Z[i]] == ss[i + Z[i]]) {
                Z[i]++;
            }
            if (i + Z[i] - 1 > r) {
                l = i;
                r = i + Z[i] - 1;
            }
        }
```

```
    }

    int cnt_less = 0, cnt_equal = 0, cnt_greater = 0;
    for (int k = 0; k < T; k++) {
        int start = (n - k) % n;
        if (start == 0) {
            cnt_equal++;
        } else {
            if (Z[start] >= n) {
                cnt_equal++;
            } else {
                if (ss[start + Z[start]] < ss[Z[start]]) {
                    cnt_less++;
                } else {
                    cnt_greater++;
                }
            }
        }
    }

    cout << cnt_less << " " << cnt_equal << " " << cnt_greater << endl;
    return 0;
}
```