Editorial OLP 2025

Số vui vẻ

Nhận xét chung, ta thấy N sau mỗi bước giảm rất nhanh, nên sẽ không có quá nhiều giá trị khác biệt của N.

Subtask 1:

- Do N cuối cùng luôn là 1. Nên khi gặp N là 1 thì chúng ta dừng quá trình (vì 1^2 = 1).

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int N;
    cin >> N;
    cout << N << ' ';
    while (N != 1) {
        int temp = 0;
        while (N > 0) {
            temp += (N % 10) * (N % 10);
            N /= 10;
        }
        N = temp;
        cout << N << ' ';
    }
    return 0;
}
```

Subtask 2:

- Ta cần kiểm tra xem N hiện tại đã xuất hiện trước đó chưa.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
```

```
    int N;
    cin >> N;
    cout << N << ' ';
    vector<int> exists;
    exists.push_back(N);
    while (true) {
        int temp = 0;
        while (N > 0) {
            temp += (N % 10) * (N % 10);
            N /= 10;
        }
        N = temp;
        for (int i : exists) {
            if (i == N) {
                return 0;
            }
        }
        exists.push_back(N);
        cout << N << ' ';
    }
    return 0;
}
```

Subtask 3:

- Ta thấy sau lần biến đổi đầu tiên, thì N tối đa 9 *9* 50 = 4050, nên vẫn giống Subtask 2, ta có thể dùng kiểu dữ liệu string để lưu những số này.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    string N;
    cin >> N;
    cout << N << ' ';
    vector<string> exists;
    exists.push_back(N);
    while (true) {
        int temp = 0;
        for (char c : N) {
            temp += (c - '0') * (c - '0');
        }
        N = to_string(temp);
```

```
        for (string i : exists) {
            if (i == N) {
                return 0;
            }
        }
        exists.push_back(N);
        cout << N << ' ';
    }
    return 0;
}
```

Xoá chuỗi

Subtask 1:

- Ta có thể sử dụng hàm find() và erase() với string để thực hiện các thao tác, độ phức tạp sẽ là $O(3 * N^2)$ = $O(N^2)$, với 3 là độ dài "UTE", N là độ dài của S.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    string S;
    cin >> S;
    while (S.find("UTE") != string::npos) {
        int pos = S.find("UTE");
        S.erase(pos, 3);
    }
    cout << (S.empty() ? "empty" : S);
    return 0;
}
```

Subtask 2:

- Ta sử dụng thêm một chuỗi T ảo để lưu chuỗi trong quá trình duyệt qua S, bất cứ khi nào 3 ký tự cuối cùng của T tạo nên "UTE", ta xoá 3 ký tự đó đi. Chuỗi cuối cùng cũng là giá trị cuối cùng của T.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
```

```
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    string S;
    cin >> S;
    string T = "";
    for (char c : S) {
        T += c;
        int n = T.size();
        if (n >= 3 && T[n - 3] == 'U' && T[n - 2] == 'T' && T[n - 1] == 'E') {
            T.pop_back();
            T.pop_back();
            T.pop_back();
        }
    }
    cout << (T.empty() ? "empty" : T);
    return 0;
}
```

Tô màu

Subtask 1:

- Ta dùng quay lui hoặc bitmask để duyệt toàn bộ cách tô màu cho N đỉnh. Độ phức tạp là $O(M * 2^N)$

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int N, M;
    cin >> N >> M;
    vector<int> u(M), v(M), c(M);
    for (int i = 0; i < M; i++) {
        cin >> u[i] >> v[i] >> c[i];
        u[i]--;
        v[i]--;
    }
    for (int i = 0; i < (1 << N); i++) {
        vector<int> color(N);
        for (int j = 0; j < N; j++) {
            if ((i >> j) & 1) {
                color[j] = 1;
            }
```

```
        }
        int ok = 1;
        for (int j = 0; j < M; j++) {
            if (color[u[j]] + color[v[j]] != c[j]) {
                ok = 0;
            }
        }
        if (ok) {
            for (int j : color) cout << j << ' ';
            return 0;
        }
    }
    cout << -1;
    return 0;
}
```

Subtask 2:

- Do không có cạnh số nào nối 2 đỉnh gồm 1 trắng và 1 đen. Do đó nếu gặp cạnh 2 thì ta gán 2 đỉnh màu đen, nếu gặp cạnh 0 thì ta gán 2 đỉnh màu trắng, còn những đỉnh chưa tô màu thì ta gán màu bất kỳ. Sau đó ta duyệt lại một lần nữa xem cách tô màu đó có thoả mãn không. Có thể chứng minh được nếu cách tô màu đó không thoả mãn, thì không còn cách nào thoả mãn.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int N, M;
    cin >> N >> M;
    vector<int> color(N), u(M), v(M), c(M);
    for (int i = 0; i < M; i++) {
        cin >> u[i] >> v[i] >> c[i];
        u[i]--;
        v[i]--;
    }
    for (int i = 0; i < M; i++) {
        if (c[i] == 2) {
            color[u[i]] = color[v[i]] = 1;
        }
    }
    for (int i = 0; i < M; i++) {
        if (color[u[i]] + color[v[i]] != c[i]) {
```

```
            cout << -1;
            return 0;
        }
    }
    for (int i : color) cout << i << ' ';
    return 0;
}
```

Subtask 3:

- Ta vẫn gán màu cạnh 2 và 0 như subtask 2. Với cạnh 1, ta nhận thấy thành phần liên thông mà tạo bởi các cạnh 1 là đồ thị 2 phía (đồ thị không có chu trình lẻ). Do đó ta duyệt những đỉnh đã tô màu, và thực hiện dfs sang những đỉnh chưa tô màu khác với màu khác đỉnh hiện tại (do bây giờ chúng ta chỉ chứa các cạnh 1). Sau đó nếu còn những đỉnh chưa tô màu, thì những đỉnh này hoàn toàn nối với nhau bởi cạnh 1. Ta gán 1 đỉnh bất kỳ với màu bất kỳ trong thành phần liên thông, sau đó thực hiện dfs gán màu cho những đỉnh còn lại trong thành phần liên thông đó.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int N, M;
    cin >> N >> M;
    vector<int> color(N, -1), u(M), v(M), c(M);
    vector<vector<int>> adj(N);
    for (int i = 0; i < M; i++) {
        cin >> u[i] >> v[i] >> c[i];
        u[i]--;
        v[i]--;
    }
    for (int i = 0; i < M; i++) {
        if (c[i] == 2) {
            color[u[i]] = color[v[i]] = 1;
        }
        else if (c[i] == 0) {
            color[u[i]] = color[v[i]] = 0;
        }
        else {
            adj[u[i]].push_back(v[i]);
            adj[v[i]].push_back(u[i]);
        }
    }
```

```
    vector<int> vis(N);
    function<void(int)> dfs1 = [&](int U) {
        vis[U] = 1;
        for (int V : adj[U]) {
            if (color[V] == -1) {
                color[V] = (color[U] == 1 ? 0 : 1);
                dfs1(V);
            }
        }
    };
    for (int i = 0; i < N; i++) {
        if (!vis[i] && color[i] != -1) {
            dfs1(i);
        }
    }
    function<void(int)> dfs2 = [&](int U) {
        for (int V : adj[U]) {
            if (color[V] == -1) {
                color[V] = (color[U] == 1 ? 0 : 1);
                dfs2(V);
            }
        }
    };
    for (int i = 0; i < N; i++) {
        if (color[i] == -1) {
            color[i] = 1;
            dfs2(i);
        }
    }
    for (int i = 0; i < M; i++) {
        if (color[u[i]] + color[v[i]] != c[i]) {
            cout << -1;
            return 0;
        }
    }
    for (int i : color) cout << i << ' ';
    return 0;
}
```

Tổng đường đi

Nhận xét chung cho Subtask 1 và Subtask 2, do mỗi đỉnh có tối đa 2 đỉnh kề, ta có thể suy ra cây này bây giờ thực chất là một mảng, nên ta chỉ cần trải thẳng cây ra là được. Ta có thể gán lại số của các đỉnh theo thứ tự 1 đến N cho dễ code.
Subtask 1:

- Ta duyệt từng cặp $i\ j$ $(i < j)$ và tính giá trị của đoạn này. Độ phức tạp $O(N^2)$

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    vector<vector<pair<int, int>>> adj(n);
    vector<int> cnt(n);
    for (int i = 1; i < n; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[--u].emplace_back(--v, w);
        adj[v].emplace_back(u, w);
        cnt[u]++, cnt[v]++;
    }
    const int mod = 1e9 + 7;
    int ans = 0;
    int u = min_element(cnt.begin(), cnt.end()) - cnt.begin();
    vector<int> vis(n);
    vis[u] = 1;
    vector<int> ord;
    queue<int> q;
    q.push(u);
    while (!q.empty()) {
        u = q.front();
        q.pop();
        ord.push_back(u);
        for (auto [v, w] : adj[u]) {
            if (!vis[v]) {
                vis[v] = 1;
                q.push(v);
            }
        }
    }
    for (int i = 0; i < n; i++) {
        long long f = 1;
        for (int j = i + 1; j < n; j++) {
            for (auto [v, w] : adj[ord[j]]) {
                if (v == ord[j - 1]) {
                    f *= w;
                    f %= mod;
                }
```

```
            }
            ans += f;
            ans %= mod;
        }
    }
    cout << ans;
    return 0;
}
```

Subtask 2:

- Ta gọi $dp[i]$ là tổng các đường đi mà kết thúc tại bắt đầu tại j và kết thúc tại i với mọi $j < i$. Ta thấy $dp[i+1]$ sẽ có thêm 1 đường đi mới là từ i tới $i+1$ và toàn bộ đường đi kết thúc tại i bây giờ nối thêm cạnh i tới $i+1$ vào. Ta có công thức : $dp[i+1] = dp[i] * w + w$, với w là trọng số cạnh nối đỉnh i và $i+1$.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    vector<vector<pair<int, int>>> adj(n);
    vector<int> cnt(n);
    for (int i = 1; i < n; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[--u].emplace_back(--v, w);
        adj[v].emplace_back(u, w);
        cnt[u]++, cnt[v]++;
    }
    const int mod = 1e9 + 7;
    int ans = 0;
    int u = min_element(cnt.begin(), cnt.end()) - cnt.begin();
    vector<int> vis(n);
    vis[u] = 1;
    vector<int> ord;
    queue<int> q;
    q.push(u);
    while (!q.empty()) {
        u = q.front();
        q.pop();
        ord.push_back(u);
```

```
        for (auto [v, w] : adj[u]) {
            if (!vis[v]) {
                vis[v] = 1;
                q.push(v);
            }
        }
    }
    long long f = 0;
    for (int i = 1; i < n; i++) {
        for (auto [v, w] : adj[ord[i]]) {
            if (v == ord[i - 1]) {
                f = (f + 1) * w % mod;
            }
        }
        ans += f;
        ans %= mod;
    }
    cout << ans;
    return 0;
}
```

Subtask 3:

- Ý tưởng cũng giống subtask 2, nhưng bây giờ là với cây, ta sẽ cố định $dp[i]$ là tổng đường đi với i là đỉnh nằm trên trong cấu trúc cây đoạn i j với i là tổ tiên của j. Đáp án của chúng ta là tổng của mảng dp và tổng giá trị các đường đi giữa hai đỉnh u và v với u và v đều khác i và u và v có tổ tiên chung nhỏ nhất là i. Phần sau chúng ta cần phải tính toán cẩn thận để tránh tính trùng đường đi.

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
int32_t main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int n;
    cin >> n;
    vector<vector<pair<int, int>>> adj(n);
    vector<int> cnt(n);
    for (int i = 1; i < n; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[--u].emplace_back(--v, w);
        adj[v].emplace_back(u, w);
```

```
    }
    const int mod = 1e9 + 7;
    int ans = 0;
    vector<int> dp(n);
    function<void(int, int)> dfs = [&](int u, int p) {
        for (auto [v, w] : adj[u]) {
            if (v == p) continue;
            dfs(v, u);
            dp[u] = (dp[u] + w + dp[v] * 1ll * w) % mod;
        }
        ans += dp[u];
        ans %= mod;
        int temp = 0;
        for (auto [v, w] : adj[u]) {
            if (v == p) continue;
            ans += (1ll * temp * (w + w * dp[v] % mod) % mod);
            ans %= mod;
            temp += (w + 1ll * w * dp[v]) % mod;
            temp %= mod;
        }
    };
    dfs(0, 0);
    cout << ans;
    return 0;
}
```

Hình vuông

Nhận xét chung, với một tập các cây mà chúng ta xét chi phí, thì chi phí sẽ là $max(max_x - min_x, max_y - min_y)$ với max_x là toạ độ x lớn nhất, tương tự là min_x, max_y, min_y.

Chi phí để rào N cây luôn lớn hơn hoặc bằng chi phí để rào $N-1$ cây.

Subtask 1:

- Ta xét chọn từng cây trong đoạn $[L, R]$ để bỏ qua, sau đó lấy $max_x, min_x, max_y, min_y$ của những cây còn lại và tính chi phí, ta lấy chi phí nhỏ nhất. Độ phức tạp $O(N^3)$

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n, q;
```

```
    cin >> n >> q;
    vector<pair<int, int>> a(n);
    for (auto &[x, y] : a) cin >> x >> y;
    while (q--) {
        int l, r;
        cin >> l >> r;
        l--, r--;
        int ans = 2e9 + 1;
        for (int i = l; i <= r; i++) {
            int min_x = 1e9, max_x = -1e9;
            int min_y = 1e9, max_y = -1e9;
            for (int j = l; j <= r; j++) {
                if (j == i) continue;
                auto [x, y] = a[j];
                min_x = min(min_x, x);
                max_x = max(max_x, x);
                min_y = min(min_y, y);
                max_y = max(max_y, y);
            }
            ans = min(ans, max(max_x - min_x, max_y - min_y));
        }
        cout << ans << '\n';
    }
    return 0;
}
```

Nhận xét chung cho subtask 2 và subtask 3, thực tế, ta chỉ cần xét việc loại bỏ với tối đa 4 cây. Cây có max_x, hoặc min_x, hoặc max_y, hoặc min_y. Vậy nếu những điểm x và y đều là biên thì sao? . Không phải vấn đề, vì nếu như vậy, 4 điểm chúng ta xét cũng bao gồm nó rồi. Việc chứng minh coi như là bài tập cho bạn đọc.

Subtask 2:

- Ta duyệt qua và cố định điểm loại bỏ, có thể biến tạm để lưu trữ giá trị 4 điểm biên hiện tại nếu bỏ điểm đó ra hoặc dùng RMQ trong $O(1)$ để tìm.

```
#include <bits/stdc++.h>
using namespace std;
// RMQ with sparse table O<NlogN, 1>
template<typename T>
struct RMQ
{
    vector<vector<T>> rmq;
    function<T(T, T)> f;
    T res;
```

```
    int n;
    RMQ() : n(0) {}
    RMQ(vector<T> & dat, T id, function<T(T, T)> g)
    {
        res = id;
        f = g;
        n = dat.size();
        assert(n > 0);
        rmq.resize(n, vector<T>(__lg(n) + 1));
        for (int i = 0; i < n; i++) rmq[i][0] = dat[i];
        for (int i = 1; i <= __lg(n); i++)
        {
            for (int j = 0; j + (1 << i) <= n; j++)
            {
                rmq[j][i] = f(rmq[j][i - 1], rmq[j + (1 << (i - 1))][i - 1]);
            }
        }
    }
    T query(int l, int r)
    {
        if (l > r) return res;
        int d = __lg(r - l + 1);
        return f(rmq[l][d], rmq[r - (1 << d) + 1][d]);
    };
};
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n, q;
    cin >> n >> q;
    auto MIN = [&](int a, int b) -> int { return a < b ? a : b; };
    auto MAX = [&](int a, int b) -> int { return a > b ? a : b; };
    const int inf = 1e9;
    vector<int> x(n), y(n);
    for (int i = 0; i < n; i++) cin >> x[i] >> y[i];
    RMQ<int> treex_min(x, inf, MIN);
    RMQ<int> treex_max(x,-inf, MAX);
    RMQ<int> treey_min(y, inf, MIN);
    RMQ<int> treey_max(y,-inf, MAX);
    while (q--) {
        int l, r;
        cin >> l >> r;
        l--, r--;
        int ans = 2 * inf;
        for (int i = l; i <= r; i++) {
```

```
            int min_x = min(treex_min.query(l, i - 1), treex_min.query(i + 1,
r)), max_x = max(treex_max.query(l, i - 1), treex_max.query(i + 1, r));
            int min_y = min(treey_min.query(l, i - 1), treey_min.query(i + 1,
r)), max_y = max(treey_max.query(l, i - 1), treey_max.query(i + 1, r));
            ans = min(ans, max(max_x - min_x, max_y - min_y));
        }
        cout << ans << '\n';
    }
    return 0;
}
```

Subtask 3:

- Ta dùng Segment Tree $O(logn)$ hoặc RMQ $O(1)$ để tìm 4 điểm đó, và các giá trị sau khi loại bỏ từng điểm trong những điểm đó.

```
#include <bits/stdc++.h>
using namespace std;
// RMQ with sparse table O<NlogN, 1>
template<typename T>
struct RMQ
{
    vector<vector<T>> rmq;
    function<T(T, T)> f;
    int n;
    RMQ() : n(0) {}
    void init(vector<T> & dat, function<T(T, T)> g)
    {
        f = g;
        n = dat.size();
        assert(n > 0);
        rmq.resize(n, vector<T>(__lg(n) + 1));
        for (int i = 0; i < n; i++) rmq[i][0] = dat[i];
        for (int i = 1; i <= __lg(n); i++)
        {
            for (int j = 0; j + (1 << i) <= n; j++)
            {
                rmq[j][i] = f(rmq[j][i - 1], rmq[j + (1 << (i - 1))][i - 1]);
            }
        }
    }
    T query(int l, int r)
    {
        assert(l <= r);
        int d = __lg(r - l + 1);
```

```
        return f(rmq[l][d], rmq[r - (1 << d) + 1][d]);
    };
};
template<typename T, typename F>
struct segtree {
    const size_t sz; const T ID; F f{};
    vector<T> tree;
    segtree(size_t n, T ID) : segtree(n, ID, F{}) {}
    explicit segtree(size_t n, T ID, const F& f) :
        sz(Log2(n)), ID(ID), f(f),
        tree(sz << 1, ID) {}
    static size_t Log2(size_t n) {
        n--;
        for (int i = 0; i < 5; i++) n |= n >> (1 << i);
        return n + 1;
    }
    void update(int i, T val) {
        --i |= sz; tree[i] = val;
        while (i >>= 1) tree[i] = f(tree[i << 1], tree[i << 1 | 1]);
    }
    T query(int l, int r) const {
        T L = ID, R = ID; --l |= sz, --r |= sz;
        while (l <= r) {
            if (l & 1) L = f(L, tree[l++]);
            if (~r & 1) R = f(tree[r--], R);
            l >>= 1, r >>= 1;
        }
        return f(L, R);
    }
};
pair<int, int> f0(pair<int, int> a, pair<int, int> b) {
    return a.first > b.first ? a : b;
}
pair<int, int> f1(pair<int, int> a, pair<int, int> b) {
    return a.first < b.first ? a : b;
}
pair<int, int> f2(pair<int, int> a, pair<int, int> b) {
    return a.second > b.second ? a : b;
}
pair<int, int> f3(pair<int, int> a, pair<int, int> b) {
    return a.second < b.second ? a : b;
}
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
```

```
    int n, q;
    cin >> n >> q;
    vector<pair<int, int>> dat(n);
    for (auto &[x, y] : dat) cin >> x >> y;
    RMQ<pair<int, int>> rmq[8];
    rmq[0].init(dat, f0);
    rmq[1].init(dat, f1);
    rmq[2].init(dat, f2);
    rmq[3].init(dat, f3);
    const int inf = 1e9 + 1;
    auto MIN = [&](int a, int b) -> int { return a < b ? a : b; };
    auto MAX = [&](int a, int b) -> int { return a > b ? a : b; };
    segtree<int, decltype(MIN)> tmnx(n + 1, inf, MIN);
    segtree<int, decltype(MAX)> tmxx(n + 1, -inf, MAX);
    segtree<int, decltype(MIN)> tmny(n + 1, inf, MIN);
    segtree<int, decltype(MAX)> tmxy(n + 1, -inf, MAX);
    map<pair<int, int>, int> mp;
    for (int i = 0; i < n; i++) {
        auto [x, y] = dat[i];
        tmnx.update(i + 1, x);
        tmxx.update(i + 1, x);
        tmny.update(i + 1, y);
        tmxy.update(i + 1, y);
        mp[dat[i]] = i;
    }
    while (q--) {
        int l, r;
        cin >> l >> r;
        int ans = 2 * inf;
        ans = min(ans, max(tmxx.query(l, r) - tmnx.query(l, r), tmxy.query(l,
r) - tmny.query(l, r)));
        l--, r--;
        for (int i = 0; i < 4; i++) {
            auto p = mp[rmq[i].query(l, r)];
            auto [x, y] = dat[p];
            tmnx.update(p + 1,  inf);
            tmxx.update(p + 1, -inf);
            tmny.update(p + 1,  inf);
            tmxy.update(p + 1, -inf);
            l++, r++;
            ans = min(ans, max(tmxx.query(l, r) - tmnx.query(l, r),
tmxy.query(l, r) - tmny.query(l, r)));
            l--, r--;
            tmnx.update(p + 1, x);
            tmxx.update(p + 1, x);
            tmny.update(p + 1, y);
```

```
            tmxy.update(p + 1, y);
        }
        cout << ans << '\n';
    }
    return 0;
}
```

Mảng con có tổng lớn nhất

Subtask 1:

- Nhận xét : Mọi đường đi luôn có $N-1$ lần qua phải và $N-1$ lần xuống dưới.
- Ta dùng quay lui hoặc bitmask để check toàn bộ những đường đi này. Độ phức tạp $O(2^{N-2})$ hoặc $O((2*N-1)*2^{N-2})$.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n;
    long long k;
    cin >> n >> k;
    vector<vector<int>> a(n, vector<int>(n));
    for (auto &x : a) {
        for (int &i : x) cin >> i;
    }
    vector<int> mask;
    for (int i = 0; i < n - 1; i++) mask.push_back(0);
    for (int i = 0; i < n - 1; i++) mask.push_back(1);
    long long ans = 0;
    do {
        long long mx = a[0][0], sum = a[0][0];
        int r = 0, c = 0;
        for (int i : mask) {
            if (i) r++;
            else c++;
            sum = max(sum, 0LL) + a[r][c];
            mx = max(mx, sum);
        }
        ans += k == mx;
    } while(next_permutation(mask.begin(), mask.end()));
    cout << ans;
```

```
    return 0;
}
```

Kỹ thuật để giải subtask 2 và subtask 3, chúng ta dùng Meet in the middle, 1 nửa đi từ ô (1, 1) đi phải hoặc xuống dưới và 1 nửa đi từ ô (N, N) đi trái hoặc đi lên.

Subtask 2:

- Do mảng chỉ toàn số nguyên không âm, nên đáp án là số lượng đường đi có tổng bằng K.
  Độ phức tạp $O(2^{(N-2)/2} * (N-2)/2)$
  Subtask 3:
- Phần gộp 2 nửa lại chúng ta phải xét các trường hợp:
  - Nửa trên có tổng lớn nhất là K và nửa dưới có tổng lớn nhất là K và gộp hai nửa lại tổng lớn nhất vẫn là K.
  - Nửa trên có tổng lớn nhất là K và nửa dưới có tổng lớn nhất nhỏ hơn K và gộp hai nửa lại tổng lớn nhất vẫn là K.
  - Nửa trên có tổng lớn nhất nhỏ hơn K và nửa dưới có tổng lớn nhất là K và gộp hai nửa lại tổng lớn nhất vẫn là K.
  - Nửa trên có tổng nhỏ hơn K và nửa dưới có tổng nhỏ hơn K và gộp hai nửa lại tổng lớn nhất vẫn là K.
  Độ phức tạp $O(2^{(N-2)/2} * (N-2)/2)$.

```
#include <bits/stdc++.h>
using namespace std;
#define all(x) x.begin(), x.end()
#define N 20
vector<long long> eq[N][N], lt[N][N];
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n;
    long long k;
    cin >> n >> k;
    vector<vector<int>> a(n, vector<int>(n));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> a[i][j];
        }
    }
    if (n == 1) {
        cout << (a[0][0] == k);
        return 0;
    }
```

```
    if (a[0][0] > k || a[n - 1][n - 1] > k) {
        cout << 0;
        return 0;
    }
    int n1 = n - 1;
    int n2 = n - 2;
    for (int i = 0; i < 1 << n1; i++) {
        int x = 0, y = 0;
        long long sum = a[0][0], mx = a[0][0];
        for (int j = 0; j < n1; j++) {
            if ((i >> j) & 1) {
                x++;
            }
            else {
                y++;
            }
            sum = max(0LL, sum) + a[x][y];
            mx = max(mx, sum);
        }
        if (mx == k) eq[x][y].push_back(sum);
        else if (mx < k) lt[x][y].push_back(sum);
    }
    for (int i = 0; i < 1 << n2; i++) {
        int x = n - 1, y = n - 1;
        long long sum = a[n - 1][n - 1], mx = a[n - 1][n - 1];
        for (int j = 0; j < n2; j++) {
            if ((i >> j) & 1) {
                x--;
            }
            else {
                y--;
            }
            sum = max(0LL, sum) + a[x][y];
            mx = max(mx, sum);
        }
        if (mx == k) eq[x][y].push_back(sum);
        else if (mx < k) lt[x][y].push_back(sum);
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (eq[i][j].size()) sort(all(eq[i][j]));
            if (lt[i][j].size()) sort(all(lt[i][j]));
        }
    }
    long long ans = 0;
    // MSS_L = K and MSS_R = K
```

```
    for (int x = 0; x < n; x++) {
        int y = n - 1 - x;
        if (x + 1 < n) {
            for (auto i : eq[x][y]) {
                ans += (upper_bound(all(eq[x + 1][y]), k - i) - eq[x + 1]
[y].begin());
            }
        }
        if (y + 1 < n) {
            for (auto i : eq[x][y]) {
                ans += (upper_bound(all(eq[x][y + 1]), k - i) - eq[x][y +
1].begin());
            }
        }
    }
    // MSS_L = K
    for (int x = 0; x < n; x++) {
        int y = n - 1 - x;
        if (x + 1 < n) {
            for (auto i : eq[x][y]) {
                ans += (upper_bound(all(lt[x + 1][y]), k - i) - lt[x + 1]
[y].begin());
            }
        }
        if (y + 1 < n) {
            for (auto i : eq[x][y]) {
                ans += (upper_bound(all(lt[x][y + 1]), k - i) - lt[x][y +
1].begin());
            }
        }
    }
    // MSS_R = K
    for (int x = 0; x < n; x++) {
        int y = n - 1 - x;
        if (x + 1 < n) {
            for (auto i : lt[x][y]) {
                ans += (upper_bound(all(eq[x + 1][y]), k - i) - eq[x + 1]
[y].begin());
            }
        }
        if (y + 1 < n) {
            for (auto i : lt[x][y]) {
                ans += (upper_bound(all(eq[x][y + 1]), k - i) - eq[x][y +
1].begin());
            }
        }
```

```
    }

    // UNION = K
    for (int x = 0; x < n; x++) {
        int y = n - 1 - x;
        if (x + 1 < n) {
            for (auto i : lt[x][y]) {
                ans += (upper_bound(all(lt[x + 1][y]), k - i) -
lower_bound(all(lt[x + 1][y]), k - i));
            }
        }
        if (y + 1 < n) {
            for (auto i : lt[x][y]) {
                ans += (upper_bound(all(lt[x][y + 1]), k - i) -
lower_bound(all(lt[x][y + 1]), k - i));
            }
        }
    }
    cout << ans;
    return 0;
}
```