1 Algorithm

1.1 LIS

```cpp
// longest increasing subsequence
int LIS(const vector<int>& s) {
// use binary search to update the smallest element
// that ends in a subsequence of length d
    vector<int> v{};
    v.push_back(s[0]);
    for (int i{1}; i < s.size(); ++i)
        if (s[i] > v.back()) v.push_back(s[i]);
        else *lower_bound(v.begin(), v.end(), s[i]) = s[i];
    return v.size();
}
```

1.2 LCS

```cpp
// longest common subsequence
int LCS(const vector<int>& a, const vector<int>& b) {
    vector<vector<int>> dp(a.size() + 1, vector<int>(b.size() + 1));
    for (int i{1}; i <= int(a.size()); ++i)
        for (int j{1}; j <= int(b.size()); ++j) {
            dp[i][j] = max(dp[i][j], max(dp[i - 1][j], dp[i][j - 1]));
            if (a[i - 1] == b[j - 1]) dp[i][j] = max(dp[i][j], dp[i - 1][j - 1] + 1);
        }
    return dp[a.size()][b.size()];
}
```

2 Basic

2.1 mod helper function

```cpp
int add(int i, int j) {
    if ((i += j) >= MOD)
        i -= MOD;
    return i;
}

int sub(int i, int j) {
    if ((i -= j) < 0)
        i += MOD;
    return i;
}
```

2.2 self-defined-pq-operator

```cpp
auto cmp = [](int a, int b) {
    return a > b;
};
priority_queue<int, vector<int>, decltype(cmp)> pq(cmp);
```

2.3 generating all subsets

```cpp
for (int b = 0; b < (1<<n); b++) {
    vector<int> subset;
    for (int i = 0; i < n; i++) {
        if (b&(1<<i)) subset.push_back(vc[i]);
    }
}
```

2.4 memset

```cpp
memset(a, 0, sizeof(a)); // 0
memset(a, 0x3f3f3f3f , sizeof(a)); // INF
```

2.5 submask enumeration

```cpp
// O(3^n)

int m = 0b10110; // binary: 10110, decimal: 22

cout << "Mask: " << bitset<5>(m) << " (" << m << ")\\n";

// Enumerate all submasks
for (int s = m; s; s = (s - 1) & m) {
    cout << bitset<5>(s) << " (" << s << ")\\n";
}

// Optionally include the empty submask
cout << bitset<5>(0) << " (0)\\n";
```

2.6 custom-hash

```cpp
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // <http://xorshift.di.unimi.it/splitmix64.c>
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

unordered_map<long long, int, custom_hash> safe_map;
gp_hash_table<long long, int, custom_hash> safe_hash_table;
```

2.7 stringstream split by comma

```cpp
while (std::getline(ss, segment, ',')) {
    segments.push_back(segment);
}
```

3 DP

3.1 deque

```cpp
/*
遊戲 DP - O(N^2)
A 與 B 將進行以下的遊戲。

最初，他們會得到一個序列 a = (a1, a2, ..., aN)。在 a 尚未為空時，兩位玩家輪流進行以下操作，從 A 開始：

從 a 的開頭或結尾移除一個元素。玩家會獲得 x 分，其中 x 為被移除的元素。
設 X 與 Y 分別為遊戲結束時 A 與 B 的總得分。A 會嘗試最大化 X-Y，而 B 會嘗試最小化 X-Y。

假設兩位玩家都採取最優策略，請求出最後的 X-Y 值。

定義 dp[i][j] 為在區間 [i, j] 上，對於 B 來說的最優分數 (X-Y)。
*/

void solve() {
    int n;
    cin >> n;
    vector<int> a(n);
    vector<vector<int>> dp(n + 1, vector<int>(n + 1, 0));

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

    for (int i = n - 1; i >= 0; --i) {
        for (int j = i + 1; j < n; ++j) {
            dp[i][j] = max(a[i] - dp[i + 1][j], a[j] - dp[i][j - 1]);
        }
    }
    cout << dp[0][n - 1] << "\n";
}
```

3.2 walk

```cpp
/*
DP on graphs - O(N^3 log K)
給定一個簡單的有向圖 G，具有 N 個頂點，編號為 1, 2, …, N。

對於任意 i, j (1 ≤ i, j ≤ N)，給定整數 a_{i, j}，表示是否存在從頂點 i 指向頂點 j 的有向邊。若 a_{i,j} = 1，則存在邊；若 a_{i, j} = 0，則不存在。

求圖中長度為 K 的不同有向路徑數目，對 10^9+7 取模。路徑可重複通過相同邊（即允許重複邊）。

注意：當我們將鄰接矩陣 m 與 m 相乘時，得到的是長度為 2 的路徑數；若取 m 的 p 次方 m^p，則其 (i, j) 元素表示從 i 到 j 的長度為 p 的路徑數。
*/

void solve() {
    int n, k;
    cin >> n >> k;
    vector<vector<int>> m(n, vector<int>(n));

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

    Matrix<int> mat(m);
    mat = power(mat, k);
    int ans = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            ans += mat.mat[i][j];
            ans %= MOD;
        }
    }
    cout << ans << "\n";
}
```

3.3 grouping

```cpp
/*
狀壓 DP - O(3^N * + 2^N * N^2)
有 N 隻兔子，編號為 1,2,…,N。

對於每一對 i,j (1≤i,j≤N)，兔子 i 與 j 的相容度由整數 a_i,j 描述。這裡 a_i,i = 0 對於每個 i (1≤i≤N)，且 a_i,j = a_j,i 對於任意 i 與 j (1≤i,j≤N)。

A 將 N 隻兔子分成若干個群組。每隻兔子必須且僅屬於一個群組。分群後，對於每一對 i 與 j (1≤i<j≤N)，若兔子 i 與 j 屬於同一群組，A 即可獲得 a_i,j 分。

求 A 能獲得的最大總分。

令 cost[S] 表示將集合 S 中的所有兔子放在同一群組時所得到的分數。此值可在 O(2^N * N^2) 時間內計算。

接著我們計算 dp[S]，表示對集合 S 中的兔子進行分群時所能得到的最大分數。
*/

void solve() {
    int n;
    cin >> n;
    vector<vector<int>> a(n, vector<int>(n));
    vector<int> cost(1<<n, 0);
    vector<int> dp(1<<n, 0);

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

    // backtrack all subset
    for (int b = 0; b < (1<<n); ++b) {
        vector<int> subset;
        for (int i = 0; i < n; ++i) {
            if (b & (1<<i)) {
                for (const int& j : subset) {
                    cost[b] += a[i][j];
                }
                subset.pb(i);
            }
        }
    }

    // dp
    for (int i = 0; i < (1<<n); ++i) {
        int j = ((1<<n) - 1) ^ i;
        for (int s = j; s != 0; s = (s - 1) & j) {
            dp[i ^ s] = max(dp[i ^ s], dp[i] + cost[s]);
        }
    }
    cout << dp[(1<<n) - 1] << "\n";
}
```

3.4 matching

```cpp
/*
Bitmask DP - O(N * 2^N)
有 N 個男人和 N 個女人，分別編號為 1, 2, …, N。

對於每個 i, j (1 ≤ i, j ≤ N)，男人 i 和女人 j 的相容性由整數 a[i][j] 給出。
如果 a[i][j] = 1，則男人 i 和女人 j 是相容的；
如果 a[i][j] = 0，則不是。

A 正在嘗試組成 N 對，每對由一個相容的男人和女人組成。在這裡，每個男人和每個女人必須恰好屬於一對。

求 A 可以組成 N 對的方法數，結果對 10^9 + 7 取模。

定義 dp[S] 為將集合 S 中的女性與前 |S| 個男性配對的方法數。
*/

const int maxn = 21;
const int MOD = 1e9 + 7;
int n;
int grid[maxn][maxn];
int dp[1 << maxn];

void solve() {
    cin >> n;
    memset(dp, 0, sizeof(dp));

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

    dp[0] = 1;
    for (int s = 0; s < (1 << n); ++s) {
        int ps = __builtin_popcount(s);
        for (int w = 0; w < n; ++w) {
            if ((s & (1 << w)) || !grid[ps][w]) {
                continue;
            }

            dp[s | (1 << w)] += dp[s];
            dp[s | (1 << w)] %= MOD;
        }
    }
    cout << dp[(1 << n) - 1] << "\n";
}
```

3.5 projects

```cpp
/*
LIS DP - O(N log N)
有 n 個你可以參加的專案。對於每個專案，你知道其開始與結束天數以及可獲得的報酬金額。
在同一天你最多只能參加一個專案。
問：你最多可以賺到多少金額？

dp[i] = 在第 i 天之前我們可以賺到的最大金額。
*/

void solve() {
    int n;
    cin >> n;
    vector<array<int, 3>> vc(n);
    map<int, int> days;

    for (int i = 0; i < n; ++i) {
        int a, b, p;
        cin >> a >> b >> p;
        days[a] = days[b] = 1;
        vc[i] = {a, b, p};
    }
    int idx = 1;
    for (auto& x : days) {
        x.second = idx++;
    }
    vector<int> dp(idx, 0);

    sort(vc.begin(), vc.end(), [](const array<int, 3>& va, const array<int, 3>& vb) {
        if (va[1] != vb[1]) return va[1] < vb[1];
        if (va[0] != vb[0]) return va[0] < vb[0];
        return va[2] > vb[2];
    });

    int i = 0;
    for (int d = 1; d < idx; ++d) {
        dp[d] = dp[d - 1];
        while (i < n && days[vc[i][1]] == d) {
            dp[d] = max(dp[d], dp[days[vc[i][0]] - 1] + vc[i][2]);
            i++;
        }
    }
    cout << dp[idx - 1] << "\n";
```

3.6 stones

```cpp
/*
遊戲 DP - O(N^2)
有一個集合 A={a1,a2,…,aN}，包含 N 個正整數。太郎和次郎將進行以下的遊戲。

一開始有一堆 K 顆石頭。兩位玩家輪流進行以下操作，從太郎開始：

選擇集合 A 中的一個元素 x，並從石堆中移除恰好 x 顆石頭。
當某位玩家無法進行操作時即輸掉比賽。假設兩位玩家都採取最優策略，請判斷誰會獲勝。

定義 dp[i] 表示當剩下 i 顆石頭時，是否有可能獲勝。
*/

void solve() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    vector<bool> dp(k + 1, 0);

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

    for (int i = 1; i <= k; ++i) {
        for (int x : a) {
            if (i >= x && !dp[i - x]) {
                dp[i] = 1;
            }
        }
    }
    cout << (dp[k] ? "First" : "Second") << "\n";
}
```

3.7 coins

```cpp
/*
機率 DP - O(N^2)
給定一個正奇數 N
有 N 枚編號為 1,2,…,N 的硬幣，第 i 枚出現正面的機率為 p，反面為 1-p。
已經拋擲所有硬幣，求正面數大於反面的機率。

定義 dp[i][j] 為拋完前 i 枚硬幣後，得到 j 次正面的機率。
*/

void solve() {
    int n;
    cin >> n;
    vector<double> a(n);
    vector<vector<double>> dp(n + 1, vector<double>(n + 1, 0.0));

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

    for (int i = 0; i <= n; ++i) {
        dp[i][0] = 1.0;
    }
```

```cpp
    int least = n / 2 + 1;

    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= least; ++j) {
            // head
            dp[i][j] = dp[i - 1][j - 1] * a[i - 1];
            // tail
            dp[i][j] += dp[i - 1][j] * (1 - a[i - 1]);
        }
    }

    cout << fixed << setprecision(10) << dp[n][least] << "\n";
}
```

3.8 elevator rides

```cpp
/*
狀壓 DP - O(2^N)
有 n 個人想要搭電梯到樓頂，建築物只有一部電梯。你知道每個人的體重以及電梯的最大允許載重。最少需要搭乘多少次電梯？

定義 dp[S] = {r, w}，其中 r 是將集合 S 中的所有人送到樓頂所需的最少電梯次數，w 是最後一次電梯所載人的總重量。
*/

void solve() {
    int n, x;
    cin >> n >> x;
    vector<int> w(n);
    vector<pii> dp(1<<n, {INF, INF});

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

    dp[0] = {1, 0};
    for (int b = 1; b < (1<<n); ++b) {
        for (int i = 0; i < n; ++i) {
            if (b & (1<<i)) {
                auto [r_prev, w_prev] = dp[b ^ (1<<i)];
                pii can;
                if (w_prev + w[i] <= x) {
                    can = {r_prev, w_prev + w[i]};
                }
                else {
                    can = {r_prev + 1, w[i]};
                }
                dp[b] = min(dp[b], can);
            }
        }
    }
    cout << dp[(1<<n) - 1].first << "\n";
}
```

3.9 slimes

```cpp
/*
Range DP - O(N^3)
有 N 個史萊姆排成一列。最初，從左邊數來第 i 個史萊姆的大小為 ai。

A 想要把所有史萊姆合併成一個更大的史萊姆。他會重複執行以下操作，直到只剩下一個史萊姆為止：

選擇兩個相鄰的史萊姆，將它們合併成一個新的史萊姆。新史萊姆的大小為 x+y，其中 x 和 y 是合併前兩個史萊姆的大小。
這時會產生 x+y 的花費。合併時，史萊姆的相對位置不會改變。
請求出合併所有史萊姆所需的最小總花費。

令 dp[i][j] 表示將第 i 個到第 j 個史萊姆合併成一個史萊姆的最小花費。
*/

const int maxn = 401;
const int INF = 1e18;
int dp[maxn][maxn];
int a[maxn];
int prefix[maxn + 1];

int f(int i, int j) {
    if (i + 1 == j) {
        return a[i] + a[j];
    }
    if (i == j) {
        return 0;
    }
    if (dp[i][j] != INF) {
        return dp[i][j];
    }
    //cerr << i << " " << j << "\n";

    int ans = INF;
    for (int k = i; k < j; ++k) {
        ans = min(ans, f(i, k) + f(k + 1, j));
    }
    return dp[i][j] = ans + (prefix[j + 1] - prefix[i]);
}
```

3.10 digit sum

```cpp
/*
Digit DP - O(|K| * D)
計算在 1 到 K（含）之間，滿足其十進位數字和為 D 的倍數的整數數量，答案對 10^9+7 取模。

令 dp[i][j] 表示在已確定前 i 位數字的情況下，構成長度為 |K| 的數字且目前數字和 mod D 等於 j 的方法數。
*/

const int MOD = 1e9 + 7;
int dp[10001][101][2];

void solve() {
    string K;
    int D;
    cin >> K >> D;
    int len = K.size();
    memset(dp, 0, sizeof(dp));
    dp[0][0][1] = 1;

    for (int i = 1; i <= len; ++i) {
        int limit = K[i - 1] - '0';
        for (int s = 0; s < D; ++s) {
            for (int flag = 0; flag <= 1; ++flag) {
                int ways = dp[i - 1][s][flag];
                if (ways == 0) continue;
                int max_d = (flag ? limit : 9);
                for (int d = 0; d <= max_d; ++d) {
                    int rs = (s + d) % D;
                    int rflag = (flag && d == max_d ? 1 : 0);
                    dp[i][rs][rflag] += ways;
                    dp[i][rs][rflag] %= MOD;
                }
            }
        }
    }

    int ans = (dp[len][0][0] + dp[len][0][1]) % MOD;
    ans = (ans - 1 + MOD) % MOD;
    cout << ans << "\n";
}
```

3.11 sushi

```cpp
/*
期望值 DP - O(N^3)
有 N 盤壽司，編號從 1 到 N。第 i 盤最初有 a_i (1 ≤ a_i ≤ 3) 塊壽司。

太郎不斷擲一顆編號 1 到 N 的骰子。如果結果是第 i 盤且該盤還有壽司，他就吃掉一塊；否則什麼也不做。

請求出吃完所有壽司所需擲骰子的期望次數。

dp[x][y][z] 代表還有
x 盤剩 1 塊壽司、
y 盤剩 2 塊壽司、
z 盤剩 3 塊壽司時的期望擲骰次數。
*/

const int maxn = 301;
double dp[maxn][maxn][maxn];
int n;

double dfs(int x, int y, int z) {
    if (x < 0 || y < 0 || z < 0) return 0;
    if (x == 0 && y == 0 && z == 0) return 0;
    if (dp[x][y][z] > 0) return dp[x][y][z];
    double ans = n + x * dfs(x - 1, y, z)
                   + y * dfs(x + 1, y - 1, z)
                   + z * dfs(x, y + 1, z - 1);
    return dp[x][y][z] = ans / (x + y + z);
}

void solve() {
    cin >> n;
    vector<int> a(n);
    memset(dp, -1, sizeof(dp));
    vector<int> freq(4, 0);

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

    cout << fixed << setprecision(10) << dfs(freq[1], freq[2], freq[3]) << "\n";
}
```

3.12 candies

```cpp
/*
組合 DP - O(NK)
有 N 個小孩，編號為 1,2,…,N。

他們決定將 K 顆糖果分給自己。對於每個 i (1≤i≤N)，第 i 個小孩最多可以拿到 ai 顆糖果（包含 0 顆）。所有糖果都必須分完，不能剩下。

請問有多少種分配糖果的方法？請將答案對 10^9+7 取模。若存在某個小孩分到的糖果數不同，則視為不同的分配方式。

令 dp[i][j] 表示將 j 顆糖果分給前 i 個小孩的方法數。
*/

void solve() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    vector<int> dp(k + 1, 0), S(k + 1, 0);

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

```cpp

    dp[0] = 1;
    for (int i = 0; i < n; ++i) {
        vector<int> new_dp(k + 1, 0);
        S[0] = dp[0];
        for (int j = 1; j <= k; ++j) {
            S[j] = (S[j - 1] + dp[j]) % MOD;
        }
        for (int j = 0; j <= k; ++j) {
            if (j - a[i] - 1 >= 0) {
                new_dp[j] = (S[j] - S[j - a[i] - 1] + MOD) % MOD;
            }
            else {
                new_dp[j ] = S[j] % MOD;
            }
        }
        dp = new_dp;
    }
    cout << dp[k] << "\n";
}
```

3.13 permutation

```cpp
/*
抽象 DP - O(N^2)
設 N 為正整數。給定一個長度為 N-1 的字串 s，字元僅包含 '<' 與 '>'。

求滿足條件的排列 (p1, p2, …, pN)（即 1 到 N 的排列）數量，答案對 10^9+7 取模：

對於每個 i (1 ≤ i ≤ N-1)，若 s 的第 i 個字元為 '<'，則要求 p_i < p_{i+1}；若為 '>'，則要求 p_i > p_{i+1}。

令 dp[i][j] 表示：在考慮前 i 個比較符號（即構成長度為 i+1 的排列）且最後一個元素為 j 的有效排列數量。
*/

void solve() {
    int n;
    string s;
    cin >> n >> s;
    vector<vector<int>> dp(n + 1, vector<int>(n + 1, 0));
    vector<int> prefix(n + 1, 0);
    dp[1][0] = 1;

    for (int i = 2; i <= n; ++i) {
        for (int k = 0; k < n; ++k) {
            prefix[k + 1] = prefix[k] + dp[i - 1][k];
        }
        for (int j = 0; j < i; ++j) {
            if (s[i - 2] == '>') {
                dp[i][j] += prefix[i - 1] - prefix[j];
                dp[i][j] %= MOD;
                /*
                for (int k = j; k < i - 1; ++k) {
                    dp[i][j] += dp[i - 1][k];
                }
                */
            }
            else {
                dp[i][j] += prefix[j];
                dp[i][j] %= MOD;
                /*
                for (int k = 0; k < j; ++k) {
                    dp[i][j] += dp[i - 1][k];
                }
                */
            }
        }
    }
    int ans = 0;
    for (int j = 0; j < n; ++j) {
        ans += dp[n][j];
        ans %= MOD;
    }
    cout << ans << "\n";
}
```

3.14 Knasack2

```cpp
// 01 背包, 背包承重大 (1e9), 物品價值和較小 (1e5)

const int maxn = 101;
const int maxv = 100001;
int weight[maxn];
int cost[maxn];
int dp[maxv];

void solve() {
    int n, w;
    cin >> n >> w;

    for (int i = 0; i < n; ++i) {
        cin >> weight[i] >> cost[i];
    }
    fill(dp, dp + maxv, 1e18);

    dp[0] = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = maxv - 1; j >= 0; --j) {
            if (dp[j] + weight[i] <= w) {
                dp[j + cost[i]] = min(dp[j + cost[i]], dp[j] + weight[i]);
            }
        }
    }

    for (int i = maxv - 1; i >= 0; --i){
        if (dp[i] != 1e18) {
            cout << i << "\n";
            return;
        }
    }
}
```

3.15 flowers

```cpp
/*
LIS DP + Segment Tree - O(N log N)
有 N 朵花排成一列。對於每個 i (1 ≤ i ≤ N)，第 i 朵花的高度與美麗分別為 h_i 與 a_i。此處 h_1, h_2, …, h_N 兩兩互異。

A 會拔掉一些花，使得剩下的花從左到右的高度為單調遞增（嚴格遞增）。

求剩下花的美麗值總和的最大可能值。

令 dp[i] 表示以第 i 朵花為結尾的遞增子序列所能取得的最大美麗值。
*/

void solve() {
    int n;
    cin >> n;
    SGT<int, MergeMax> tree(n + 1, 0);
    vector<int> h(n), b(n);

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

    for (int i = 0; i < n; ++i) {
        int mx = tree.query(0, h[i]);
        tree.modify(h[i], mx + b[i]);
    }

    cout << tree.query(0, n + 1) << "\n";
}
```

3.16 independent set

```cpp
/*
DP on Trees - O(N)
有一棵含 N 個頂點的樹，頂點編號為 1,2,…,N。對於每個 i (1 ≤ i ≤ N-1)，第 i 條邊連接頂點 x_i 和 y_i。

A 決定將每個頂點塗成白色或黑色，但不允許兩個相鄰的頂點同時為黑色。

求將頂點塗色的方案數，對 10^9+7 取模。

設 dp[i][j] 表示以節點 i 為根的子樹中，在節點 i 顏色為 j 時的塗色方案數（例如 j=0 表示白色，j=1 表示黑色）。
*/

const int maxn = 100001;
vector<int> adj[maxn];
int f[maxn][2];
const int MOD = 1e9 + 7;

void dp(int u, int p) {

    for (int v : adj[u]) {
        if (v != p) {
            dp(v, u);
            f[u][0] = (f[v][0] + f[v][1]) % MOD * f[u][0] % MOD;
            f[u][1] = f[v][0] * f[u][1] % MOD;
        }
    }
}

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

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

    for (int i = 0; i < n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        adj[u].pb(v);
        adj[v].pb(u);
    }

    dp(0, -1);

    cout << (f[0][0] + f[0][1]) % MOD << "\n";
}
```

3.17 couting tower

```cpp
/*
狀態機 DP - O(N)
你的任務是建造一座寬度為 2、高度為 n 的塔。你有無限數量寬度與高度為整數的方塊。

dp[i][0] = 高度為 i 的塔中，頂層為一個寬度為 2 的方塊（即該層由跨越兩欄的單一方塊覆蓋）的塔的數量。
dp[i][1] = 高度為 i 的塔中，頂層在該層有兩個寬度為 1 的方塊（每欄各一個）的塔的數量。
*/

void solve() {
```

```cpp
    long long n;
    cin >> n;
    dp[1][0] = 1;
    dp[1][1] = 1;
    for(int i = 2;i<=n;i++){
        dp[i][0] = (4 * dp[i-1][0] + dp[i-1][1]) % mod;
        dp[i][1] = (dp[i-1][0] + 2 * dp[i-1][1]) % mod;
    }
    cout << (dp[n][0] + dp[n][1]) % mod << endl;
}
```

3.18 couting numbers

```cpp
/*
區間 DP - O(digits*10)
您的任務是計算在區間 a 到 b 之間，沒有任何相鄰兩位數字相同的整數的個數。

定義 dp[pos][prev_digit][is_tight][is_started] 為從位置 pos 到結尾的有效整數數量，
其中 prev_digit 是位置 pos-1 的數字，
is_tight 表示是否受原數字前綴的限制，
is_started 表示是否已開始構成有效數字（用以避免計入前導零）。
*/

// dp[pos][prev_digit][is_tight][is_started]
int dp[20][10][2][2];
string s;

int f(int pos, int prev_digit, bool is_tight, bool is_started) {
    if (pos == (int)s.size()) {
        return 1;
    }
    if (dp[pos][prev_digit][is_tight][is_started] != -1) {
        return dp[pos][prev_digit][is_tight][is_started];
    }

    int ans = 0;
    int max_d = is_tight ? (s[pos] - '0') : 9;
    for (int d = 0; d <= max_d; ++d) {
        if (is_started && d == prev_digit) {
            continue;
        }

        bool new_is_started = is_started || (d > 0);
        bool new_is_tight = is_tight && (d == max_d);
        ans += f(pos + 1, d, new_is_tight, new_is_started);
    }

    return dp[pos][prev_digit][is_tight][is_started] = ans;
}

int count(int x) {
    if (x < 0) return 0;
    s = to_string(x);
    memset(dp, -1, sizeof(dp));
    return f(0, 0, true, false);
}

void solve() {
    int a, b;
    cin >> a >> b;

    int ca = count(a - 1);
    int cb = count(b);
    cout << cb - ca << "\n";
}
```

4 Data Structure

4.1 undo disjoint set

```cpp
struct DisjointSet {
  // save() is like recursive
  // undo() is like return
  int n, fa[MXN], sz[MXN];
  vector<pair<int*,int>> h;
  vector<int> sp;
  void init(int tn) {
    n=tn;
    for (int i=0; i<n; i++) sz[fa[i]=i]=1;
    sp.clear(); h.clear();
  }
  void assign(int *k, int v) {
    h.PB({k, *k});
    *k=v;
  }
  void save() { sp.PB(SZ(h)); }
  void undo() {
    assert(!sp.empty());
    int last=sp.back(); sp.pop_back();
    while (SZ(h)!=last) {
      auto x=h.back(); h.pop_back();
      *x.F=x.S;
    }
  }
  int f(int x) {
    while (fa[x]!=x) x=fa[x];
    return x;
  }
  void uni(int x, int y) {
    x=f(x); y=f(y);
    if (x==y) return ;
    if (sz[x]<sz[y]) swap(x, y);
    assign(&sz[x], sz[x]+sz[y]);
    assign(&fa[y], x);
  }
}djs;
```

4.2 segment tree range update (lazy propagation)

```cpp
// segment tree
// range query & range modify
class SGT {
    using value_t = int;
    using node_t = pair<value_t, int>;

    int n;
    vector<node_t> t;
    vector<optional<value_t>> lz;
    // [ tv+1 : tv+2*(tm-tl) ) -> left subtree
    int left(int tv) { return tv + 1; }
    int right(int tv, int tl, int tm) {
        return tv + 2 * (tm - tl); }
    /** differ from case to case **/
    // query is "max" and modify is "add" here
    node_t merge(const node_t& x, const node_t& y) { // associative function
        return max(x, y);
    }
    void update(int tv, int len, const value_t& x) {
        if (!lz[tv]) lz[tv] = x;
        else lz[tv] = lz[tv].value() + x;
        t[tv].fi = t[tv].fi + x;
    }
    /*****************************/
    void build(const vector<value_t>& v, int tv, int tl, int tr) {
        if (tr - tl > 1) {
            int tm{(tl + tr) / 2};
            build(v, left(tv), tl, tm);
            build(v, right(tv, tl, tm), tm, tr);
            t[tv] = merge(t[left(tv)], t[right(tv, tl, tm)]);
        } else t[tv] = {v[tl], tl};
    }
    void push(int tv, int tl, int tr) { // lazy propagation
        if (!lz[tv]) return ;
        int tm{(tl + tr) / 2};
        update(left(tv), tm - tl, lz[tv].value());
        update(right(tv, tl, tm), tr - tm, lz[tv].value());
        lz[tv].reset();
    }
    void set(int p, const value_t& x, int tv, int tl, int tr) {
        if (tr - tl > 1) {
            push(tv, tl, tr);
            int tm{(tl + tr) / 2};
            if (p < tm) set(p, x, left(tv), tl, tm);
            else set(p, x, right(tv, tl, tm), tm, tr);
            t[tv] = merge(t[left(tv)], t[right(tv, tl, tm)]);
        } else t[tv].fi = x;
    }
    void rmodify(int l, int r, const value_t& x, int tv, int tl, int tr) {
        if (!(l == tl && r == tr)) {
            push(tv, tl, tr);
            int tm{(tl + tr) / 2};
            if (r <= tm) rmodify(l, r, x, left(tv), tl, tm);
            else if (l >= tm) rmodify(l, r, x, right(tv, tl, tm), tm, tr);
            else rmodify(l, tm, x, left(tv), tl, tm),
                 rmodify(tm, r, x, right(tv, tl, tm), tm, tr);
            t[tv] = merge(t[left(tv)], t[right(tv, tl, tm)]);
        } else update(tv, tr - tl, x);
    }
    node_t rquery(int l, int r, int tv, int tl, int tr) {
        if (l == tl && r == tr) return t[tv];
        push(tv, tl, tr);
        int tm{(tl + tr) / 2};
        if (r <= tm) return rquery(l, r, left(tv), tl, tm);
        else if (l >= tm) return rquery(l, r, right(tv, tl, tm), tm, tr);
        else return merge(rquery(l, tm, left(tv), tl, tm),
                          rquery(tm, r, right(tv, tl, tm), tm, tr));
    }
public:
    explicit SGT(const vector<value_t>& v) : n{v.size()}, t(2 * n - 1), lz(2 * n - 1) { build(v, 0, 0, n); }
    void set(int p, const value_t& x) { set(p, x, 0, 0, n); }
    void rmodify(int l, int r, const value_t& x) { rmodify(l, r, x, 0, 0, n); } // [l:r)
    node_t rquery(int l, int r) { return rquery(l, r, 0, 0, n); } // [l:r)
};

int main() {
    vector<long long> a = {1, 5, 2, 4, 3};
    SGT st(a);

    auto [val, idx] = st.rquery(0, 5);
    cout << "Initial max: " << val << " at " << idx << "\\n"; // (5, 1)

    st.rmodify(1, 4, 3); // add 3 to indices 1..3
    tie(val, idx) = st.rquery(0, 5);
    cout << "After add: " << val << " at " << idx << "\\n"; // (8, 1)

    st.set(2, 10); // set a[2] = 10
    tie(val, idx) = st.rquery(0, 5);
    cout << "After set: " << val << " at " << idx << "\\n"; // (10, 2)
}
```

4.3 segment tree prefix sum lower bound

```cpp
class SGT {
    int n;
    vector<long long> t;
    int left(int tv) { return tv + 1; }
    int right(int tv, int tl, int tm) {
        return tv + 2 * (tm - tl); }
    void modify(int p, long long x, int tv, int tl, int tr) {
        if (tr - tl > 1) {
            int tm{(tl + tr) / 2};
            if (p < tm) modify(p, x, left(tv), tl, tm);
            else modify(p, x, right(tv, tl, tm), tm, tr);
            t[tv] = t[left(tv)] + t[right(tv, tl, tm)];
        } else t[tv] = x;
    }
    long long query(int l, int r, int tv, int tl, int tr) {
        if (l == tl && r == tr) return t[tv];
        int tm{(tl + tr) / 2};
        if (r <= tm) return query(l, r, left(tv), tl, tm);
        else if (l >= tm) return query(l, r, right(tv, tl, tm), tm, tr);
        else return query(l, tm, left(tv), tl, tm) +
            query(tm, r, right(tv, tl, tm), tm, tr);
    }
public:
    explicit SGT(int _n) : n{_n}, t(2 * n - 1) {}
    void modify(int p, long long x) { modify(p, x, 0, 0, n); };
    long long query(int l, int r) { return query(l, r, 0, 0, n); }
    int ps_lower_bound(long long ps) { // prefix sum lower bound
        if (ps > t[0]) return n;
        int tv{0}, tl{0}, tr{n};
        while (tr - tl > 1) {
            int tm{(tl + tr) / 2};
            if (t[left(tv)] >= ps) tv = left(tv), tr = tm;
            else ps -= t[left(tv)], tv = right(tv, tl, tm), tl = tm;
        }
        return tl;
    }
};
```

4.4 Fenwick tree (BIT)

```cpp
// 1-based
struct Fenwick {
    int n;
    vector<int> bit;
    Fenwick(int _n=0): n(_n), bit(n+1, 0) {}
    void update(int idx, int val) {
        for (; idx <= n; idx += idx & -idx)
            bit[idx] += val;
    }
    int query(int idx) {
        int res = 0;
        for (; idx > 0; idx -= idx & -idx)
            res += bit[idx];
        return res;
    }
    int query(int l, int r) {
        return query(r) - query(l-1);
    }
};

int main() {
    Fenwick fw(n);
    for (int i = 1; i < n; ++i) {
        fw.update(i, a[i]);
    }
    cout << fw.query(3, 7) << "\\n";  // range sum [3..7]
    int current = ...; // old value at idx
    int newVal  = ...; // new value you want
    fw.update(idx, newVal - current);
}
```

4.5 Trie (Prefix tree)

```cpp
struct trie {
    int n = 0;
    trie *a[2];
    trie() {
        a[0] = a[1] = nullptr;
    }

    void insert(int k) {
        trie *curr = this;
        for (int i = 63; i >= 0; --i) {
            bool bit = (k & (1LL << i)) > 0;
            if (curr->a[bit] == nullptr) {
                curr->a[bit] = new trie();
            }
            curr = curr->a[bit];
            curr->n++ ;
        }
    }

    void erase(int k) {
        trie *curr = this;
        for (int i = 63; i >= 0; --i) {
            bool bit = (k & (1LL << i)) > 0;
            curr = curr->a[bit];
            curr->n--;
        }
    }

    int query(int k) {
        trie *curr = this;
        int x = 0;
        for (int i = 63; i >= 0; --i) {
            x ^= (1LL << i) & k;
            x ^= 1LL << i;
            bool bit = (k & (1LL << i)) == 0;
            if (curr->a[bit] == nullptr || curr->a[bit]->n == 0)
                x ^= 1LL << i, bit ^= 1;
            curr = curr->a[bit];
        }
        return x;
    }
};
```

4.6 segment tree

```cpp
// Segment tree
template<typename value_t, class merge_t>
class SGT {
    int n;
    vector<value_t> t;
    value_t defa;
    merge_t merge;
public:
    explicit SGT(int _n, value_t _defa, const merge_t& _merge = merge_t{})
        : n{_n}, t(2 * n), defa{_defa}, merge{_merge} {}

    void modify(int p, const value_t& x) {
        for (t[p += n] = x; p > 1; p >>= 1)
            t[p >> 1] = merge(t[p], t[p ^ 1]);
    }

    value_t query(int l, int r) { return query(l, r, defa); }

    value_t query(int l, int r, value_t init) {
        for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
            if (l & 1) init = merge(init, t[l++]);
            if (r & 1) init = merge(init, t[--r]);
        }
        return init;
    }
};

// Custom merge for range minimum + index
struct MergeMin {
    pair<int, int> operator()(const pair<int, int>& a,
                              const pair<int, int>& b) const {
        if (a.first != b.first) return (a.first < b.first) ? a : b;
        return (a.second < b.second) ? a : b; // tie-break on index
    }
};

int main() {
    int n = 6;
    SGT<pair<int, int>, MergeMin> tree(n, {INT_MAX, -1});

    vector<int> a = {5, 3, 6, 1, 4, 2};
    for (int i = 0; i < n; ++i)
        tree.modify(i, {a[i], i});

    auto [min_val, min_idx] = tree.query(1, 5); // range [1, 5)
    cout << "Min value in [1, 5): " << min_val << ", at index " << min_idx << '\\n';

    return 0;
}
```

4.7 BIT range update point query

```cpp
// Fenwick Tree (Binary Indexed Tree) for Range Updates and Point Queries

template<typename T>
class BIT {
#define ALL(x) begin(x), end(x)
private:
    vector<T> arr;
    int n;
    inline int lowbit(int x) { return x & (-x); }
    void addInternal(int s, T v) {
        while (s > 0) {
            arr[s] += v;
            s -= lowbit(s);
        }
    }
public:
    void init(int n_) {
        n = n_;
        arr.resize(n + 1);
        fill(ALL(arr), 0);
    }
    void add(int l, int r, T v) {
        // add v to interval (l, r], 1-based
        addInternal(l, -v);
        addInternal(r, v);
    }
    T query(int x) {
        // value at index x
        T res = 0;
        while (x <= n) {
            res += arr[x];
            x += lowbit(x);
        }
        return res;
    }
#undef ALL
};

int main() {
```

```
    BIT<int> bit;
    bit.init(5);

    // add +3 to indices 1..3
    bit.add(0, 3, 3);

    // add +2 to indices 3..5
    bit.add(2, 5, 2);

    cout << "Value at 1 = " << bit.query(1) << "\n"; // expect 3
    cout << "Value at 3 = " << bit.query(3) << "\n"; // expect 3+2=5
    cout << "Value at 5 = " << bit.query(5) << "\n"; // expect 2
}
```

4.8 DSU remove node find prev next one

```
// previous/next one
class PvNx {
    vector<int> pa, sz, mn, mx;
    int find(int x) { // collapsing find
        return pa[x] == -1 ? x : pa[x] = find(pa[x]);
    }
    void unionn(int x, int y) { // weighted union
        auto rx{find(x)}, ry{find(y)};
        if (rx == ry) return ;
        if (sz[rx] < sz[ry]) swap(rx, ry);
        pa[ry] = rx, sz[rx] += sz[ry], mn[rx] = min(mn[rx], mn[ry]), mx[rx] = max(mx[rx], mx[ry]);
    }
public:
    explicit PvNx(int n) : pa(n + 1, -1), sz(n + 1, 1), mn(n + 1) { iota(mn.begin(), mn.end(), 0), mx = mn; }
    void remove(int i) { unionn(i, i + 1); }
    int prev(int i) { return mn[find(i)] - 1; }
    int next(int i) {
        int j{mx[find(i)]};
        if (i == j) j = mx[find(j + 1)];
        return j;
    }
    bool exist(int i) { return i == mx[find(i)]; }
};
```

4.9 DSU

```
// fast disjoint set union
class DSU {
    vector<int> pa, sz;
public:
    explicit DSU(int n) : pa(n, -1), sz(n, 1) {}
    int find(int x) { // collapsing find
        return pa[x] == -1 ? x : pa[x] = find(pa[x]);
    }
    void unite(int x, int y) { // weighted union
        auto rx{find(x)}, ry{find(y)};
        if (rx == ry) return ;
        if (sz[rx] < sz[ry]) swap(rx, ry);
        pa[ry] = rx, sz[rx] += sz[ry];
    }
};
```

5 Geometry

5.1 cross. product

```
// If cross(a,b,c) > 0, c is to the left of line ab
// If cross(a,b,c) < 0, c is to the right of line ab
// If cross(a,b,c) = 0, a, b, c are collinear

// point struct version
// 2D Point structure
struct Point {
    double x, y;
};

// Cross product (scalar in 2D)
double cross(const Point& a, const Point& b, const Point& c) {
    // Computes (b - a) × (c - a)
    return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
}

// std::complex version
typedef std::complex<double> point;
#define x real()
#define y imag()

double cross(const point &a, const point &b) {
    return (std::conj(a) * b).imag();
}
```

5.2 line intersect

```
// 2D Point structure
struct Point {
    double x, y;
};

// Cross product (scalar in 2D)
double cross(const Point& a, const Point& b, const Point& c) {
    // Computes (b - a) × (c - a)
    return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
}

// Check if value is between two others (with endpoints allowed)
bool between(double a, double b, double x) {
    return min(a, b) <= x && x <= max(a, b);
}

// Check if point c lies on segment ab
bool onSegment(const Point& a, const Point& b, const Point& c) {
    return cross(a, b, c) == 0 &&
           between(a.x, b.x, c.x) &&
           between(a.y, b.y, c.y);
}

// Main intersection check
bool segmentsIntersect(const Point& A, const Point& B,
                       const Point& C, const Point& D) {
    double c1 = cross(A, B, C);
    double c2 = cross(A, B, D);
    double c3 = cross(C, D, A);
    double c4 = cross(C, D, B);

    // General case
    if (c1 * c2 < 0 && c3 * c4 < 0) return true;

    // Special cases: collinear + overlapping
    if (c1 == 0 && onSegment(A, B, C)) return true;
    if (c2 == 0 && onSegment(A, B, D)) return true;
    if (c3 == 0 && onSegment(C, D, A)) return true;
    if (c4 == 0 && onSegment(C, D, B)) return true;

    return false;
}

int main() {
    Point A{0, 0}, B{4, 4};
    Point C{0, 4}, D{4, 0};

    if (segmentsIntersect(A, B, C, D))
        cout << "Segments intersect\n";
    else
        cout << "Segments do not intersect\n";

    return 0;
}
```

5.3 Polygon area

```
/**
 * Author: Ulf Lundstrom
 * Date: 2009-03-21
 * License: CC0
 * Source: tinyKACTL
 * Description: Returns twice the signed area of a polygon.
 *  Clockwise enumeration gives negative area. Watch out for overflow if using int as T!
 * Status: Stress-tested and tested on kattis:polygonarea
 */
#pragma once

#include "Point.h"

template<class T>
T polygonArea2(vector<Point<T>>& v) {
  T a = v.back().cross(v[0]);
  rep(i,0,sz(v)-1) a += v[i].cross(v[i+1]);
  return a;
}

int main() {
    // Example: square with vertices (0,0), (0,1), (1,1), (1,0)
    vector<Point<long long>> poly = {
        {0,0}, {0,1}, {1,1}, {1,0}
    };

    long long area2 = polygonArea2(poly);
    cout << "Twice signed area = " << area2 << "\n";
    cout << "Absolute area = " << abs(area2) / 2.0 << "\n";

    return 0;
}
```

5.4 using std::complex

```
#include <iostream>
#include <complex>
#include <cmath>

typedef std::complex<double> point;
#define x real()
#define y imag()

constexpr double PI = std::acos(-1.0);
// conj(a) = a 的共軛 · reflection of a across x-axis

// Basic operations
double dot(const point &a, const point &b) {
    return (std::conj(a) * b).real(); }
double cross(const point &a, const point &b) { return (std::conj(a) * b).imag(); }

```

```cpp
// Projections / reflections / geometry helpers
point project_onto_vector(const point &p, const point &v) {
    return v * (dot(p, v) / std::norm(v));
}

point project_onto_line(const point &p, const point &a, const point &b) {
    return a + (b - a) * (dot(p - a, b - a) / std::norm(b - a));
}

point reflect_across_line(const point &p, const point &a, const point &b) {
    return a + std::conj((p - a) / (b - a)) * (b - a);
}

point intersection(const point &a, const point &b, const point &p, const point &q) {
    double c1 = cross(p - a, b - a), c2 = cross(q - a, b - a);
    return (c1 * q - c2 * p) / (c1 - c2); // undefined if parallel (divide by zero)
}

double angle_between(const point &a, const point &b) {
    // angle of vector b - a
    return std::arg(b - a);
}

double angle_ABC(const point &a, const point &b, const point &c) {
    double r = std::remainder(std::arg(a - b) - std::arg(c - b), 2.0 * PI);
    return std::abs(r);
}

int main() {
    // sample
    point a = 2.0;          // (2,0)
    point b(3.0, 7.0);      // (3,7)
    std::cout << a << ' ' << b << '\n'; // (2,0) (3,7)
    std::cout << a + b << '\n';         // (5,7)

    // usage examples
    point p1(3, 2), p2(2, -7);
    std::cout << "p1 + p2 = " << p1 + p2 << '\n';   // (5,-5)
    std::cout << "p1 - p2 = " << p1 - p2 << '\n';   // (1,9)
    std::cout << "3.0 * p1 = " << 3.0 * p1 << '\n'; // (9,6)
    std::cout << "p1 / 5.0 = " << p1 / 5.0 << '\n'; // (0.6,0.4)

    // dot / cross via complex
    std::cout << "dot(p1,p2) = " << dot(p1, p2) << '\n';
    std::cout << "cross(p1,p2) = " << cross(p1, p2) << '\n';

    // distances, angle, rotation, polar/cartesian
    std::cout << "squared dist = " << std::norm(p1 - p2) << '\n';
    std::cout << "euclid dist = " << std::abs(p1 - p2) << '\n';
    std::cout << "angle p1->p2 = " << angle_between(p1, p2) << '\n';
    std::cout << "angle ABC = " << angle_ABC(point(1,0), point(0,0), point(0,1)) << '\n';

    // project / reflect / intersect examples
    point v(1, 1);
    point proj = project_onto_vector(p1, v);
    std::cout << "project p1 onto v = " << proj << '\n';

    point lineA(0,0), lineB(2,0);
    std::cout << "project p1 onto line = " << project_onto_line(p1, lineA, lineB) << '\n';
    std::cout << "reflect p1 across line = " << reflect_across_line(p1, lineA, lineB) << '\n';

    // intersection example
    point r1a(0,0), r1b(1,1);
    point r2a(0,1), r2b(1,0);
    std::cout << "intersection = " << intersection(r1a, r1b, r2a, r2b) << '\n';

    // polar/cartesian and rotation
    point polar_pt = std::polar(5.0, PI/4);
    std::cout << "polar(5,PI/4) = " << polar_pt << '\n';
    point rotated = p1 * std::polar(1.0, PI/2); // rotate p1 by 90 degrees about origin
    std::cout << "rotate p1 by 90deg = " << rotated << '\n';

    return 0;
}
```

5.5 point distance from a line

```cpp
// calculate area using cross product and divide by base length
double point_line_distance(const point &p, const point &a, const point &b) {
    // area = 0.5 * |cross(b - a, p - a)|
    // base = |b - a|
    // height = 2 * area / base = |cross(b - a, p - a)| / |b - a|
    std::abs(cross(b - a, p - a)) / std::abs(b - a);
}
```

6 Graph

6.1 Euler tour+RMQ

```cpp
// Euler Tour Technique
class LCA {
    const vector<vector<int>>& adj;
    int n;
    vector<int> d, first, euler{}, log2{};
    vector<vector<int>> st{};
    void dfs(int u, int w = -1, int dep = 0) {
        d[u] = dep;
        first[u] = euler.size();
        euler.push_back(u);
        for (auto& v : adj[u]) {
            if (v == w) continue;
            dfs(v, u, dep + 1);
            euler.push_back(u);
        }
    }
public:
    LCA(const vector<vector<int>>& _adj, int root) : adj{_adj}, n{adj.size()}, d(n), first(n) {
        dfs(root);

        int tn{euler.size()};
        log2.resize(tn + 1);
        log2[1] = 0;
        for (int i{2}; i <= tn; ++i) log2[i] = log2[i / 2] + 1;

        st.assign(tn, vector<int>(log2[tn] + 1));
        for (int i{tn - 1}; i >= 0; --i) {
            st[i][0] = euler[i];
            for (int j{1}; i + (1 << j) <= tn; ++j) {
                auto& x{st[i][j - 1]};
                auto& y{st[i + (1 << (j - 1))][j - 1]};
                st[i][j] = d[x] <= d[y] ? x : y;
            }
        }
    }
    int operator()(int u, int v) {
        int l{first[u]}, r{first[v]};
        if (l > r) swap(l, r);
        ++r; // make the interval left closed right open

        int j{log2[r - l]};
        auto& x{st[l][j]};
        auto& y{st[r - (1 << j)][j]};
        return d[x] <= d[y] ? x : y;
    }
};

int main() {
    int n, q;
    cin >> n >> q;
    vector<vector<int>> adj(n);

    for (int i = 1; i < n; ++i) {
        int u;
        cin >> u;
        u--;
        adj[u].pb(i);
        adj[i].pb(u);
    }

    LCA lca(adj, 0);

    while (q--) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        cout << lca(u, v) + 1 << "\\n";
    }
    return 0;
}
```

6.2 Prim

```cpp
// Prim's algorithm
vector<tuple<int, int, long long>> Prim(const vector<vector<pair<int, long long>>>& adj) {
    const auto& n = adj.size();
    vector<tuple<int, int, long long>> mst{};

    vector<bool> found(n, false);
    using ti = tuple<long long, int, int>;
    priority_queue<ti, vector<ti>, greater<ti>> pq{};
    found[0] = true;
    for (auto& [v, w] : adj[0]) pq.emplace(w, 0, v);

    for (int i = 0; i < n - 1; ++i) {
        int mn, u, v;
        do {
            tie(mn, u, v) = pq.top(), pq.pop();
        } while (found[v]);
        found[v] = true, mst.emplace_back(u, v, mn);
        for (auto& [x, w] : adj[v]) pq.emplace(w, v, x);
    }
    return mst;
}
```

6.3 Eulerian cycle

```cpp
// Eulerian cycle in an undirected graph

vector<int> euler_cycle(vector<vector<pair<int, int>>>& adj, int w = 0) {
    int n{adj.size()}, m{};
```

```cpp
    for (int v{0}; v < n; ++v) m += adj[v].size();
    m /= 2;

    vector<int> res{};
    stack<pair<int, int>> stk{};
    stk.emplace(w, -1);
    vector<int> nxt(n);
    vector<bool> usd(m);
    while (!stk.empty()) {
        auto [u, i]{stk.top()};
        while (nxt[u] < adj[u].size() && usd[adj[u][nxt[u]].second]) ++nxt[u];
        if (nxt[u] < adj[u].size()) {
            auto [v, j]{adj[u][nxt[u]]};
            ++nxt[u], usd[j] = true, stk.emplace(v, j);
        } else {
            if (i != -1) res.push_back(i);
            stk.pop();
        }
    }
    return res;
}

int main() {
    int n = 4; // number of vertices
    vector<vector<pair<int, int>>> adj(n);

    // Add edges with edge IDs
    int eid = 0;
    auto add_edge = [&](int u, int v) {
        adj[u].push_back({v, eid});
        adj[v].push_back({u, eid});
        ++eid;
    };

    add_edge(0, 1);
    add_edge(1, 2);
    add_edge(2, 3);
    add_edge(3, 0);

    vector<int> cycle = euler_cycle(adj, 0);

    cout << "Euler cycle (edge IDs in order): ";
    for (int id : cycle) cout << id << " ";
    cout << "\n";

    return 0;
}
```

6.4 Floyd-Warshall

```cpp
// Floyd-Warshall algorithm
template<typename T>
vector<vector<optional<T>>> Floyd_Warshall(const vector<vector<optional<T>>>& adj) {
    const auto& n{adj.size()};
    auto d{adj};

    for (int i{0}; i < n; ++i) d[i][i] = 0;
    for (int k{0}; k < n; ++k)
        for (int i{0}; i < n; ++i)
            for (int j{0}; j < n; ++j) {
                if (!d[i][k] || !d[k][j])
                    continue; // no value
                if (!d[i][j] || d[i][j] > d[i][k].value() + d[k][j].value())
                    d[i][j] = d[i][k].value() + d[k][j].value();
            }

    return d;
}
```

6.5 MST

```cpp
// Kruskal' s algorithm
vector<tuple<int, int, long long>> Kruskal(int n, vector<tuple<long long, int, int>>& edges) {
    vector<tuple<int, int, long long>> mst{};
    DSU dsu{n};

    sort(edges.begin(), edges.end());
    for (auto& [w, u, v] : edges)
        if (dsu.find(u) != dsu.find(v))
            dsu.unionn(u, v), mst.emplace_back(u, v, w);
    return mst;
}
```

6.6 all longest path dfs

```cpp
// all longest path (generalization of the tree diameter problem)
vector<tuple<int, int, int>> dp{};
// [mx1, x, mx2] the path of mx1 goes through x
int dfs1(int u, int w = -1) {
    int mx{0};
    for (auto& v : adj[u])
        if (v != w) {
            auto l{1 + dfs1(v, u)};
            mx = max(mx, l);

            auto& [mx1, x, mx2]{dp[u]};
            if (l >= mx1) mx2 = mx1, mx1 = l, x = v;
            else if (l > mx2) mx2 = l;
        }
    return mx;
}
void dfs2(int u, int w = -1) {
    if (w != -1) {
        int tmx;
        { // calculate the longest path through parent
            auto& [mx1, x, mx2]{dp[w]};
            if (x != u) tmx = mx1 + 1;
            else tmx = mx2 + 1;
        }
        { // update the path
            auto& [mx1, x, mx2]{dp[u]};
            if (tmx >= mx1) mx2 = mx1, mx1 = tmx, x = w;
            else if (tmx > mx2) mx2 = tmx;
        }
    }
    for (auto& v : adj[u])
        if (v != w) dfs2(v, u);
}
void all_longest_path() {
    dfs1(0), dfs2(0);
}
```

6.7 all longest path top sort

```cpp
// all longest path in DAG
// 1. topological sort
vector<int> in(n, 0);
for (int i = 0; i < m; ++i) {
    int a, b, w;
    cin >> a >> b >> w;
    adj[a].emplace_back(b, w);
    in[b]++;
}

vector<int> topo; // sequence of top sort
queue<int> q;
for (int i = 0; i < n; ++i) {
    if (in[i] == 0) {
        q.push(i);
    }
}
while (!q.empty()) {
    int pa = q.front();
    q.pop();
    topo.push_back(pa);
    for (auto& [child, w] : adj[pa]) {
        in[child]--;
        if (in[child] == 0) {
            q.push(child);
        }
    }
}

// all longest path
vector<int> dist(n, INT_MIN);
vector<vector<int>> parents(n);
dist[0] = process[0];

for (int u : topo) {
    for (auto& [v, w] : adj[u]) {
        if (dist[v] < dist[u] + process[v] + w) {
            dist[v] = dist[u] + process[v] + w;
            parents[v] = {u};
        }
        else if (dist[v] == dist[u] + process[v] + w) {
            parents[v].push_back(u);
        }
    }
}
cout << dist[n - 1];
```

6.8 Dijkstra

```cpp
// Dijkstra algorithm
template<typename T>
vector<optional<T>> Dijkstra(const vector<vector<pair<int, T>>>& adj, int s) {
    const auto& n{adj.size()};
    vector<optional<T>> d(n, nullopt);
    d[s] = 0;

    vector<bool> found(n, false);
    using pq_t = pair<T, int>;
    priority_queue<pq_t, vector<pq_t>, greater<pq_t>> pq{};
    pq.emplace(0, s);
    while (!pq.empty()) {
        auto [_, u]{pq.top()}; pq.pop();
        if (found[u]) continue;
        found[u] = true;
        for (auto& [v, w] : adj[u])
            if (!d[v] || d[v] > d[u].value() + w) {
                d[v] = d[u].value() + w;
                pq.emplace(d[v].value(), v);
            }
    }

    return d;
}
```

6.9 binary lifting

```cpp
// binary lifting
class LCA {
    const vector<vector<int>>& adj;
    int n;
    vector<int> d;
    vector<int> log2;
    vector<vector<int>> an{};
    void dfs(int u, int w = -1, int dep = 0) {
        d[u] = dep;
        an[u][0] = w;
        for (int i{1}; i <= log2[n - 1] && an[u][i - 1] != -1; ++i)
            an[u][i] = an[an[u][i - 1]][i - 1]; // 走 2^(i-1) 再走 2^(i-1) 步

```

```cpp
        // 因為計算 an 會用到祖先的資訊，所以先計算再繼續往下
        for (auto& v : adj[u]) {
            if (v == w) continue; // parent
            dfs(v, u, dep + 1);
        }
    }
public:
    LCA(const vector<vector<int>>& _adj, int root)
    : adj{_adj}, n{adj.size()}, d(n), log2(n) {
        log2[1] = 0;
        for (int i{2}; i < log2.size(); ++i)
            log2[i] = log2[i / 2] + 1;
        an.assign(n, vector<int>(log2[n - 1] + 1, -1));
        dfs(root);
    }
    int operator()(int u, int v) {
        if (d[u] > d[v]) swap(u, v);

        for (int i{log2[d[v] - d[u]]}; i >= 0; --i)
            if (d[v] - d[u] >= (1 << i)) v = an[v][i];
        // v 先走到跟 u 同高度
        if (u == v) return u;

        for (int i{log2[d[u]]}; i >= 0; --i)
            if (an[u][i] != an[v][i]) u = an[u][i], v = an[v][i];
        // u, v 一起走到 lca(u, v) 的下方

        return an[u][0];
        // 回傳 lca(u, v)
    }
};

int main() {
    int n, q;
    cin >> n >> q;
    vector<vector<int>> adj(n);

    for (int i = 1; i < n; ++i) {
        int u;
        cin >> u;
        u--;
        adj[u].pb(i);
        adj[i].pb(u);
    }

    // adj, root
    LCA lca(adj, 0);

    while (q--) {
        int u, v;
        cin >> u >> v;
        u--, v--;
        cout << lca(u, v) + 1 << "\\n";
    }
    return 0;
}
```

6.10 topological sort

```cpp
// topological sort 1
optional<vector<int>> top_sort(vector<vector<int>>& adj) {
    vector<int> res{};
    int n{static_cast<int>(adj.size())};
    vector<int> cnt(n, 0); // predecessor count
    for (int u = 0; u < n; ++u)
        for (auto& v : adj[u]) ++cnt[v];

    queue<int> qu{};
    for (int u = 0; u < n; ++u) if (!cnt[u]) qu.push(u);
    while (!qu.empty()) {
        auto u = qu.front();
        qu.pop();
        res.push_back(u);

        for (auto& v : adj[u])
            if (!--cnt[v]) qu.push(v);
    }

    if (res.size() != adj.size()) return nullopt;
    return res;
}
```

6.11 tree diameter

```cpp
int diam = 0;

int dfs(int u, int p = -1) {
    int mx = 0;
    for (int v : adj[u]) {
        if (v != p) {
            int len = 1 + dfs(v, u);
            diam = max(diam, mx + len);
            mx = max(mx, len);
        }
    }
    return mx;
}
```

6.12 all longest path

```cpp
int fir[maxn]; // length of the longest downward path from u into its subtree.
int sec[maxn]; // length of the second longest downward path from u
int res[maxn];

void dfs1(int u, int p) {
    for (int v : adj[u]) {
        if (v != p) {
            dfs1(v, u);
            if (fir[v] + 1 > fir[u]) {
                sec[u] = fir[u];
                fir[u] = fir[v] + 1;
            }
            else if (fir[v] + 1 > sec[u]) {
                sec[u] = fir[v] + 1;
            }
        }
    }
}

// to_p: the best path length that comes from the parent' s side
void dfs2(int u, int p, int to_p) {
    res[u] = max(to_p, fir[u]);

    for (int v : adj[u]) {
        if (v != p) {
            if (fir[v] + 1 == fir[u]) {
                dfs2(v, u, max(to_p, sec[u]) + 1);
            }
            else {
                dfs2(v, u, res[u] + 1);
            }
        }
    }
}

// usage
dfs1(1, 0);
dfs2(1, 0, 0);
// Now res[i] is the maximum distance from node i to any other node
for (int i = 1; i <= n; i++) {
    cout << res[i] << " ";
}
```

6.13 tree diameter (len,end)

```cpp
array<int, 2> dfs(int u, int w = -1) {
    array<int, 2> mx{0, u};  // {length, farthest leaf}
    for (auto& v : adj[u]) {
        if (v == w) continue;
        auto [len, leaf]{dfs(v, u)};
        mx = max(mx, {len + 1, leaf});
    }
    return mx;
}

array<int, 3> tree_diameter(int a = 0) {
    auto b{dfs(a)[1]};      // farthest node from 'a'
    auto [l, c]{dfs(b)};    // farthest node from 'b'
    return {l, b, c};       // {diameter length, endpoint1, endpoint2}
}
```

6.14 Bellman-Ford

```cpp
// Bellman-Ford algorithm
template<typename T>
optional<vector<optional<T>>> Bellman_Ford(const vector<vector<pair<int, T>>>& adj, int s) {
    const auto& n{adj.size()};
    vector<optional<T>> d(n, nullopt);
    d[s] = 0;

    queue<int> qu{}, qu2{};
    vector<bool> in(n, false), in2(n, false);
    qu.push(s), in[s] = true;
    for (int i{0}; i < n; ++i) { // at most n-1 edges
        while (!qu.empty()) {
            int u{qu.front()};
            qu.pop(), in[u] = false;
            for  (auto& [v, w] : adj[u])
                if (!d[v] || d[v] > d[u].value() + w) { // relax
                    d[v] = d[u].value() + w;
                    if (!in2[v]) qu2.push(v), in2[v] = true;
                }
        }
        qu.swap(qu2), in.swap(in2);
    }

    if (qu.empty()) return d;
    return nullopt; // if negative cycle
}
```

7 Language

7.1 CNF

```cpp
#define MAXN 55
struct CNF{
  int s,x,y;//s->xy | s->x, if y==-1
  int cost;
  CNF(){}
  CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
};
int state;//規則數量
map<char,int> rule;//每個字元對應到的規則，小寫字母為終端字符
vector<CNF> cnf;
void init(){
  state=0;
  rule.clear();
  cnf.clear();
}
void add_to_cnf(char s,const string &p,int cost){
  //加入一個s -> <p>的文法，代價為cost
  if(rule.find(s)==rule.end())rule[s]=state++;
```

```cpp
  for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
  if(p.size()==1){
    cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
  }else{
    int left=rule[s];
    int sz=p.size();
    for(int i=0;i<sz-2;++i){
      cnf.push_back(CNF(left,rule[p[i]],state,0));
      left=state++;
    }
    cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],cost));
  }
}
vector<long long> dp[MAXN][MAXN];
vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會有無限小的情形
void relax(int l,int r,const CNF &c,long long cost,bool neg_c=0){
  if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][c.s])){
    if(neg_c||neg_INF[l][r][c.x]){
      dp[l][r][c.s]=0;
      neg_INF[l][r][c.s]=true;
    }else dp[l][r][c.s]=cost;
  }
}
void bellman(int l,int r,int n){
  for(int k=1;k<=state;++k)
    for(auto c:cnf)
      if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
}
void cyk(const vector<int> &tok){
  for(int i=0;i<(int)tok.size();++i){
    for(int j=0;j<(int)tok.size();++j){
      dp[i][j]=vector<long long>(state+1,INT_MAX);
      neg_INF[i][j]=vector<bool>(state+1,false);
    }
    dp[i][i][tok[i]]=0;
    bellman(i,i,tok.size());
  }
  for(int r=1;r<(int)tok.size();++r){
    for(int l=r-1;l>=0;--l){
      for(int k=l;k<r;++k)
        for(auto c:cnf)
          if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c.cost);
      bellman(l,r,tok.size());
    }
  }
}
```

8 Number Theory

8.1 Linear Sieve

```cpp
// Calculate the smallest divisor of integers in [2, maxn) in O(maxn)
vector<int> min_div{[] {
    constexpr int maxn = 400000 + 10;

    vector<int> v(maxn), p;

    for (int i = 2; i < maxn; ++i) {
        if (!v[i]) {
            v[i] = i;
            p.push_back(i);
        }
        for (int j = 0; p[j] * i < maxn; ++j) {
            v[p[j] * i] = p[j];
            if (p[j] == v[i]) break;
        }
    }

    return v;
}()};
```

8.2 C(n,m)

```cpp
ll Cnm(ll n, ll m) {
    if (m > n / 2) m = n - m;
    ll r{1};
    for (ll i{1}, j{n}; i <= m; ++i, --j) r *= j, r /= i;
    return r;
}
```

8.3 derangement (Principle of Inclusion-Exclusion)

```cpp
// 1. Principle of Inclusion-Exclusion
// n! = n! * Σ (from k=0 to n) [((-1)^k) / (k!)]

mint c = 1;
for (int i = 1; i <= n; i++) {
    c = (c * i) + (i % 2 == 1 ? -1 : 1);
    cout << c.val() << ' ';
}
```

8.4 matrix template (with fast power)

```cpp
template<class T> struct Matrix {
    T **mat; int a, b;

    Matrix() : a(0), b(0) {}
    Matrix(int a_, int b_) : a(a_), b(b_) {
        int i, j;
        mat = new T*[a];
        for (i = 0; i < a; ++i) {
            mat[i] = new T[b];
            for (j = 0; j < b; ++j ){
                mat[i][j] = 0;
            }
        }
    }
    Matrix(const vector<vector<T>>& vt) {
        int i, j;

        *this = Matrix((int)vt.size(), (int)vt[0].size());
        for (i = 0; i < a; ++i) {
            for (j = 0; j < b; ++j) {
                mat[i][j] = vt[i][j];
            }
        }
    }
    Matrix operator*(const Matrix& m) {
        int i, j, k;

        assert(b == m.a);
        Matrix r(a, m.b);
        for (i = 0; i < a; ++i) {
            for (j = 0; j < m.b; ++j) {
                for (k = 0; k < b; ++k) {
                    r.mat[i][j] += mul(mat[i][k], m.mat[k][j]);
                    r.mat[i][j] %= MOD;
                }
            }
        }
        return r;
    }
    Matrix& operator*=(const Matrix& m) {
        return *this = (*this) * m;
    }
    friend Matrix power(Matrix m, long long p) {
        int i;

        assert(m.a == m.b);
        Matrix r(m.a, m.b);
        for (i = 0; i < m.a; ++i) {
            r.mat[i][i] = 1;
        }
        for (; p > 0; p >>= 1, m *= m) {
            if (p & 1) {
                r *= m;
            }
        }
        return r;
    }
};

int main() {
    Matrix<int> mat(adj);
    mat = power(mat, k); // mat^k
    cout << mat.mat[i][j];
}
```

8.5 Sieve of Eratosthenes (with big num)

```cpp
const int MX = 100000;
bool np[MX + 1];
vector<int> prime_numbers;

int init = []() {
    np[0] = np[1] = true;
    for (int i = 2; i <= MX; i++) {
        if (!np[i]) {
            prime_numbers.push_back(i);
            for (int j = i; j <= MX / i; j++) { // 避免溢出的写法
                np[i * j] = true;
            }
        }
    }
    return 0;
}();

bool is_prime(long long n) {
    if (n <= MX) {
        return !np[n];
    }
    for (long long p : prime_numbers) {
        if (p * p > n) {
            break;
        }
        if (n % p == 0) {
            return false;
        }
    }
    return true;
}
```

8.6 mod inv

```cpp
// Modular inverse when mod is prime
long long mod_inverse(long long a, long long mod) {
    return power_mod(a, mod - 2, mod);  // Fermat's Little Theorem
}
```

8.7 derangement (DP)

```cpp
// 2. DP
// !n = (n - 1) * (!(n - 1) + !(n - 2)), with !0 = 1, !1 = 0

mint a = 1, b = 0;
cout << 0 << ' ';

```

```cpp
for (int i = 2; i <= n; i++) {
    mint c = (i - 1) * (a + b);
    cout << c.val() << ' ';
    a = b;
    b = c;
}
```

8.8 fast power

```cpp
/* Iterative Function to calculate (a^b) % mod in O(log b) */
long long power_mod(long long a, long long b, long long mod) {
    long long res{1};
    while (b > 0) {
        if (b & 1) res = res * a % mod;
        b >>= 1;
        a = (a * a) % mod;
    }
    return res;
}
```

8.9 first and second mex

```cpp
// Calculate first and second MEX
pair<int, int> calculate_mexes(vector<int>& nums) {
    int n = nums.size();
    vector<bool> seen(n + 2, false);

    for (int num : nums) {
        if (num >= 0 && num < seen.size()) {
            seen[num] = true;
        }
    }

    int first_mex = -1;
    int second_mex = -1;

    for (int i = 0; i < seen.size(); ++i) {
        if (!seen[i]) {
            if (first_mex == -1) {
                first_mex = i;
            } else {
                second_mex = i;
                break;
            }
        }
    }

    return {first_mex, second_mex};
}
```

8.10 Chinese Remainder Theorem

```cpp
// coprime (p^k)
pair<ll, ll> CRT(const vector<pair<ll, ll>>& congruences) {
    ll M{1}, sol{};
    for (auto& [m, a] : congruences) M *= m;
    for (auto& [m, a] : congruences) {
        ll x{M / m}, y{MI(x, m)};
        sol = MA(sol, MM(MM(a, x, M), y, M), M);
    }
    return {M, sol};
}
```

8.11 mod inv (not prime)

```cpp
/* exists when a and mod are coprime */
/* but mod is not prime */
long long MI(long long a, long long mod) {
    return power_mod(a, euler_phi(mod) - 1, mod);
}
```

8.12 Euler Totient precompute

```cpp
constexpr int maxn{100000};
vector<int> phi{[] {
    vector<int> v(maxn + 1); v[1] = 1;
    for (int i{2}; i <= maxn; ++i) {
        if (v[i]) continue;
        v[i] = i;
        for (int j{i}; j <= maxn; j += i) {
            if (!v[j]) v[j] = j;
            v[j] = v[j] / i * (i - 1);
        }
    }
    return v;
}()};
```

8.13 mod inv (not coprime)

```cpp
/* a and mod are not coprime */
long long MI(long long a, long long mod) {
    long long d, x, y;
    extEcu(a, mod, d, x, y);
    return d == 1 ? (x + mod) % mod : -1;
}
```

8.14 Euler Totient

```cpp
int euler_phi(int n) {
    int res{n};
    for (int i{2}; i * i <= n; ++i) {
        if (n % i) continue;
        while (n % i == 0) n /= i;
        res = res / i * (i - 1);
    }
    if (n > 1) res = res / n * (n - 1);
    return res;
}
```

8.15 C(n,k) mod inverse

```cpp
fac[0] = 1;
for (int i = 1; i <= n; ++i) {
    fac[i] = fac[i - 1] * i % MOD;
}

inv_fac[n] = power_mod(fac[n], MOD - 2, MOD);
for (int i = n - 1; i >= 0; --i) {
    inv_fac[i] = inv_fac[i + 1] * (i + 1) % MOD;
}

// C(n, k) = fac[n] * inv_fac[k] * inv_fac[n - k];
```

8.16 擴展歐基里德

```cpp
/* solve x, y for ax + by = gcd(a, b) = g */
template<typename T>
void extEcu(T a, T b, T &g, T &x, T &y) {
    if (b) extEcu(b, a % b, g, y, x), y -= x * (a / b);
    else g = a, x = 1, y = 0;
}
```

8.17 Sieve of Eratosthenes

```cpp
void sieve(vector<int>& primes) {
    vector<int> is_prime(INF + 1, 0);
    is_prime[0] = 1;
    is_prime[1] = 1;

    int sq = sqrt(INF);

    for (int i = 2; i <= sq; ++i) {
        if (!is_prime[i]) {
            primes.push_back(i);
            for (int j = i * i; j <= INF; j += i) {
                is_prime[j] = 1;
            }
        }
    }
}
```

8.18 C(n,k) DP

```cpp
long long binomial(long long n, long long k, long long p) {
    // dp[i][j] = iCj
    vector<vector<long long>> dp(n + 1, vector<long long>(k + 1, 0));

    for (int i = 0; i <= n; ++i) {
        dp[i][0] = 1;
        if (i <= k) dp[i][i] = 1;
    }

    for (int i = 0; i <= n; ++i) {
        for (int j = 1; j <= min(i, k); ++j) {
            if (i != j) {
                dp[i][j] = (dp[i - 1][j - 1] + dp[i - 1][j]) % p;
            }
        }
    }

    return dp[n][k];
}
```

9 String

9.1 Z

```cpp
// 计算并返回 z 数组，其中 z[i] = |LCP(s[i:], s)|
vector<int> calc_z(const string& s) {
    int n = s.size();
    vector<int> z(n);
    int box_l = 0, box_r = 0;
    for (int i = 1; i < n; i++) {
        if (i <= box_r) {
            z[i] = min(z[i - box_l], box_r - i + 1);
        }
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            box_l = i;
            box_r = i + z[i];
            z[i]++;
        }
    }
    z[0] = n;
    return z;
}
```

9.2 manacher

```cpp
class Solution {
public:
    int countSubstrings(string s) {
        int l1 = s.size(), l2 = l1 * 2 + 1;
        string ch = "#";
        for(char c: s) {
            ch = ch + c + "#";
        }

        int c = 0, r = 0, cnt = 0;
        vector<int> p(l2);
        for(int i = 0; i < l2; i++) {
            p[i] = (i < r)? min(p[2 * c - i], r - i): 1;
            while(i + p[i] < l2 && i - p[i] >= 0 && ch[i + p[i]] == ch[i - p[i]]) p[i]++;
            if(i + p[i] > r) {
                r = i + p[i];
                c = i;
            }
            int l = p[i] - 1;
            if(l % 2 == 0) cnt += l / 2;
            else cnt += l / 2 + 1;
        }
        return cnt;
    }
};
```

9.3 AC 自動機

```cpp
template<char L='a',char R='z'>
class ac_automaton{
  struct joe{
    int next[R-L+1],fail,efl,ed,cnt_dp,vis;
    joe():ed(0),cnt_dp(0),vis(0){
      for(int i=0;i<=R-L;++i)next[i]=0;
    }
  };
public:
  std::vector<joe> S;
  std::vector<int> q;
  int qs,qe,vt;
  ac_automaton():S(1),qs(0),qe(0),vt(0){}
  void clear(){
    q.clear();
    S.resize(1);
    for(int i=0;i<=R-L;++i)S[0].next[i]=0;
    S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
  }
  void insert(const char *s){
    int o=0;
    for(int i=0,id;s[i];++i){
      id=s[i]-L;
      if(!S[o].next[id]){
        S.push_back(joe());
        S[o].next[id]=S.size()-1;
      }
      o=S[o].next[id];
    }
    ++S[o].ed;
  }
  void build_fail(){
    S[0].fail=S[0].efl=-1;
    q.clear();
    q.push_back(0);
    ++qe;
    while(qs!=qe){
      int pa=q[qs++],id,t;
      for(int i=0;i<=R-L;++i){
        t=S[pa].next[i];
        if(!t)continue;
        id=S[pa].fail;
        while(~id&&!S[id].next[i])id=S[id].fail;
        S[t].fail=~id?S[id].next[i]:0;
        S[t].efl=S[S[t].fail].ed?S[t].fail:S[S[t].fail].efl;
        q.push_back(t);
        ++qe;
      }
    }
  }
  /*DP出每個前綴在字串s出現的次數並傳回所有字串被s匹配成功的次數O(N+M)*/
  int match_0(const char *s){
    int ans=0,id,p=0,i;
    for(i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      ++S[p].cnt_dp;/*匹配成功則它所有後綴都可以被匹配(DP計算)*/
    }
    for(i=qe-1;i>=0;--i){
      ans+=S[q[i]].cnt_dp*S[q[i]].ed;
      if(~S[q[i]].fail)S[S[q[i]].fail].cnt_dp+=S[q[i]].cnt_dp;
    }
    return ans;
  }
  /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)*/
  int match_1(const char *s)const{
    int ans=0,id,p=0,t;
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed)ans+=S[p].ed;
      for(t=S[p].efl;~t;t=S[t].efl){
        ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
      }
    }
    return ans;
  }
  /*枚舉(s的子字串∩A)的所有相異字串各恰一次並傳回次數O(N*M^(1/3))*/
  int match_2(const char *s){
    int ans=0,id,p=0,t;
    ++vt;
    /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成false
    這種利用vt的方法可以O(1)歸零vis陣列*/
    for(int i=0;s[i];++i){
      id=s[i]-L;
      while(!S[p].next[id]&&p)p=S[p].fail;
      if(!S[p].next[id])continue;
      p=S[p].next[id];
      if(S[p].ed&&S[p].vis!=vt){
        S[p].vis=vt;
        ans+=S[p].ed;
      }
      for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
        S[t].vis=vt;
        ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
      }
    }
    return ans;
  }
  /*把AC自動機變成真的自動機*/
  void evolution(){
    for(qs=1;qs!=qe;){
      int p=q[qs++];
      for(int i=0;i<=R-L;++i)
        if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];
    }
  }
};
```

9.4 KMP

```cpp
// 在文本串 text 中查找模式串 pattern，返回所有成功匹配的位置（pattern[0] 在 text 中的下标）
vector<int> kmp(const string& text, const string& pattern) {
    int m = pattern.size();
    vector<int> pi(m);
    int cnt = 0;
    for (int i = 1; i < m; i++) {
        char b = pattern[i];
        while (cnt && pattern[cnt] != b) {
            cnt = pi[cnt - 1];
        }
        if (pattern[cnt] == b) {
            cnt++;
        }
        pi[i] = cnt;
    }

    vector<int> pos;
    cnt = 0;
    for (int i = 0; i < text.size(); i++) {
        char b = text[i];
        while (cnt && pattern[cnt] != b) {
            cnt = pi[cnt - 1];
        }
        if (pattern[cnt] == b) {
            cnt++;
        }
        if (cnt == m) {
            pos.push_back(i - m + 1);
            cnt = pi[cnt - 1];
        }
    }
    return pos;
}
```

9.5 suffix array lcp

```cpp
#define radix_sort(x,y){\
  for(i=0;i<A;++i)c[i]=0;\
  for(i=0;i<n;++i)c[x[y[i]]]++;\
  for(i=1;i<A;++i)c[i]+=c[i-1];\
  for(i=n-1;~i;--i)sa[--c[x[y[i]]]]=y[i];\
}
#define AC(r,a,b)\
  r[a]!=r[b]||a+k>=n||r[a+k]!=r[b+k]
void suffix_array(const char *s,int n,int *sa,int *rank,int *tmp,int *c){
  int A='z'+1,i,k,id=0;
  for(i=0;i<n;++i)rank[tmp[i]=i]=s[i];
  radix_sort(rank,tmp);
  for(k=1;id<n-1;k<<=1){
    for(id=0,i=n-k;i<n;++i)tmp[id++]=i;
    for(i=0;i<n;++i)
      if(sa[i]>=k)tmp[id++]=sa[i]-k;
    radix_sort(rank,tmp);
    swap(rank,tmp);
    for(rank[sa[0]]=id=0,i=1;i<n;++i)
      rank[sa[i]]=id+=AC(tmp,sa[i-1],sa[i]);
    A=id+1;
  }
}
//h:高度數組 sa:後綴數組 rank:排名
void suffix_array_lcp(const char *s,int len,int *h,int *sa,int *rank){
  for(int i=0;i<len;++i)rank[sa[i]]=i;
  for(int i=0,k=0;i<len;++i){
    if(rank[i]==0)continue;
    if(k)--k;
    while(s[i+k]==s[sa[rank[i]-1]+k])++k;
    h[rank[i]]=k;
  }
  h[0]=0;// h[k]=lcp(sa[k],sa[k-1]);
}
```

9.6 hash

```cpp
#define MAXN 1000000
#define mod 1073676287
/*mod 必須要是質數*/
typedef long long T;
char s[MAXN+5];
T h[MAXN+5];/*hash陣列*/
T h_base[MAXN+5];/*h_base[n]=(prime^n)%mod*/
void hash_init(int len,T prime){
  h_base[0]=1;
```

```cpp
  for(int i=1;i<=len;++i){
    h[i]=(h[i-1]*prime+s[i-1])%mod;
    h_base[i]=(h_base[i-1]*prime)%mod;
  }
}
T get_hash(int l,int r){/*閉區間寫法，設編號為0 ~ len-1*/
  return (h[r+1]-(h[l]*h_base[r-l+1])%mod+mod)%mod;
}
```

9.7 minimal string rotation

```cpp
int min_string_rotation(const string &s){
  int n=s.size(),i=0,j=1,k=0;
  while(i<n&&j<n&&k<n){
    int t=s[(i+k)%n]-s[(j+k)%n];
    ++k;
    if(t){
      if(t>0)i+=k;
      else j+=k;
      if(i==j)++j;
      k=0;
    }
  }
  return min(i,j);//最小循環表示法起始位置
}
```

9.8 reverseBWT

```cpp
const int MAXN = 305, MAXC = 'Z';
int ranks[MAXN], tots[MAXC], first[MAXC];
void rankBWT(const string &bw){
  memset(ranks,0,sizeof(int)*bw.size());
  memset(tots,0,sizeof(tots));
  for(size_t i=0;i<bw.size();++i)
    ranks[i] = tots[int(bw[i])]++;
}
void firstCol(){
  memset(first,0,sizeof(first));
  int totc = 0;
  for(int c='A';c<='Z';++c){
    if(!tots[c]) continue;
    first[c] = totc;
    totc += tots[c];
  }
}
string reverseBwt(string bw,int begin){
  rankBWT(bw), firstCol();
  int i = begin; //原字串最後一個元素的位置
  string res;
  do{
    char c = bw[i];
    res = c + res;
    i = first[int(c)] + ranks[i];
  }while( i != begin );
  return res;
}
```

10 default

10.1 debug

```cpp
#ifdef DEBUG
#define dbg(...) {\
  fprintf(stderr,"%s - %d : (%s) = ",__PRETTY_FUNCTION__,__LINE__,#__VA_ARGS__);\
  _DO(__VA_ARGS__);\
}
template<typename I> void _DO(I&&x){cerr<<x<<endl;}
template<typename I,typename...T> void _DO(I&&x,T&&...tail){cerr<<x<<", ";_DO(tail...);}
#else
#define dbg(...)
#endif
```

10.2 template

```cpp
// alias g++='g++ -std=c++14 -fsanitize=undefined -Wall -Wextra -Wshadow -D LOCAL'

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

#ifdef LOCAL
void dbg() { cerr << '\n'; }
template<class T, class ...U> void dbg(T a, U ...b) { cerr << a << ' ', dbg(b...); }
template<class T> void org(T l, T r) { while (l != r) cerr << *l++ << ' '; cerr << '\n'; }
#define debug(args...) (dbg("#> (" + string(#args) + ") = (", args, ")"))
#define orange(args...) (cerr << "#> [" + string(#args) + ") = ", org(args))
#else
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#define debug(...) ((void)0)
#define orange(...) ((void)0)
#endif

#define int long long
#define pii pair<int, int>
#define ff first
#define ss second
#define pb push_back
#define SPEEDY ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);

void solve() {

}

signed main() {
    SPEEDY;

    return 0;
}
```

11 other

11.1 Nim game

```
a1^a2^a3^...^an != 0 ? A win : B win
```

11.2 找小於 n 所有出現的 1 數量

```
current == 0 higher * factor
current == 1 higher * factor + lower + 1
other current (higher + 1) * factor
```

12 other language

12.1 python heap

```python
import heapq

heap = [7,1,2,2]
heapq.heapify(heap)
print(heap) # [1, 2, 2, 7]
heapq.heappush(heap, 5)
print(heap) # [1, 2, 2, 7, 5]
print(heapq.heappop(heap)) # 1
print(heap) # [2, 2, 5, 7]
```

12.2 java

12.2.1 文件操作

```java
import java.io.*;
import java.util.*;
import java.math.*;
import java.text.*;

public class Main{

  public static void main(String args[]){ throws FileNotFoundException, IOException
    Scanner sc = new Scanner(new FileReader("a.in"));
    PrintWriter pw = new PrintWriter(new FileWriter("a.out"));
    int n,m;
    n=sc.nextInt();//读入下一个INT
    m=sc.nextInt();

    for(ci=1; ci<=c; ++ci){
      pw.println("Case #"+ci+": easy for output");
    }

    pw.close();//关闭流并释放，这个很重要，否则是没有输出的
    sc.close();//关闭流并释放
  }
}
```

12.2.2 优先队列

```java
PriorityQueue queue = new PriorityQueue( 1, new Comparator(){
  public int compare( Point a, Point b ){
  if( a.x < b.x || a.x == b.x && a.y < b.y )
    return -1;
  else if( a.x == b.x && a.y == b.y )
    return 0;
  else return 1;
  }
});
```

12.2.3 Map

```java
Map map = new HashMap();
map.put("sa","dd");
String str = map.get("sa").toString;

for(Object obj : map.keySet()){
  Object value = map.get(obj );
}
```

12.2.4 sort

```java
static class cmp implements Comparator{
  public int compare(Object o1,Object o2){
  BigInteger b1=(BigInteger)o1;
  BigInteger b2=(BigInteger)o2;
  return b1.compareTo(b2);
  }
}
public static void main(String[] args) throws IOException{
  Scanner cin = new Scanner(System.in);
  int n;
  n=cin.nextInt();
  BigInteger[] seg = new BigInteger[n];
  for (int i=0;i<n;i++)
  seg[i]=cin.nextBigInteger();
```

```
  Arrays.sort(seg,new cmp());
}
```

12.3 python output

```python
hello = 'Hello'
world = 7122
print(f'{hello} {world}') # Hello 7122

import math
print(f'PI is approximately {math.pi:.3f}.')
# PI is approximately 3.142.

print('AAA {} BBB "{}!"'.format('Jin', 'Kela'))
# AAA Jin BBB "Kela!"

hello = 'hello, world\n'
hellos = repr(hello)
print(hellos) # 'hello, world\n'

x = 32.5
y = 40000
print(repr((x, y, ('spam', 'eggs'))))
# "(32.5, 40000, ('spam', 'eggs'))"

x = 7
print(eval('3 * x')) # 21
```

12.4 python 大數因數分解

```python
# 大數因數分解 (使用 Pollard's Rho 與 Miller-Rabin)
import sys, random
from math import gcd

# Miller-Rabin 檢定（機率性質數判定）
def is_probable_prime(n, k=12):
    if n < 2:
        return False
    # 先檢查一些小質數
    small_primes = [2,3,5,7,11,13,17,19,23,29]
    for p in small_primes:
        if n % p == 0:
            return n == p
    # 把 n-1 寫成 d * 2^s
    d = n - 1
    s = 0
    while d % 2 == 0:
        d //= 2
        s += 1
    # 重複 k 次隨機測試
    for _ in range(k):
        a = random.randrange(2, n - 1)  # 隨機挑一個測試基數
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue
        composite = True
        for __ in range(s - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                composite = False
                break
        if composite:
            return False
    return True

# Pollard's Rho 演算法（找非平凡因數）
def pollards_rho(n):
    if n % 2 == 0:
        return 2
    if n % 3 == 0:
        return 3
    # 隨機多項式 (x^2 + c) mod n
    while True:
        c = random.randrange(1, n-1)      # 隨機挑選常數 c
        x = random.randrange(2, n-1)      # 起始點
        y = x
        d = 1
        while d == 1:
            x = (pow(x, 2, n) + c) % n    # x -> f(x)
            y = (pow(y, 2, n) + c) % n    # y -> f(f(y))，走兩步
            y = (pow(y, 2, n) + c) % n
            d = gcd(abs(x - y), n)        # 計算兩者差的 gcd
            if d == n:                    # 失敗就重試
                break
        if d > 1 and d < n:               # 找到非平凡因數
            return d

# 遞迴分解
def factor(n, out):
    if n == 1:
        return
    if is_probable_prime(n):
        out.append(n)
    else:
        d = pollards_rho(n)
        while d is None or d == n:  # 偶爾失敗就重試
            d = pollards_rho(n)
        factor(d, out)
        factor(n // d, out)

def main():
    data = sys.stdin.read().strip().split()
    if not data:
        return
    # 每個 token 當作一個數字
    for token in data:
        try:
            n = int(token)
        except:
            continue
        if n <= 1:
            print(n)
            continue
        facs = []
        factor(n, facs)
        facs.sort()
        # 輸出因數
        print(" ".join(str(x) for x in facs))

if __name__ == "__main__":
    random.seed()  # 使用系統時間作為隨機種子
    main()
```

12.5 decimal

```python
# 使用 decimal 模組來處理高精度小數運算
from decimal import *
setcontext(Context(prec=MAX_PREC, Emax=MAX_EMAX, rounding=ROUND_FLOOR))
print(Decimal(input()) * Decimal(input()))

# 將小數轉成分數，方便做近似或理論分析，且可以限制分母大小。
from fractions import Fraction
Fraction('3.14159').limit_denominator(10).numerator # 22

# 設定精確度
from decimal import Decimal, getcontext

# 精確位數設定
getcontext().prec = 70

n = 100
# 指定n為高精確度的物件
n = Decimal(n)
n /= 7
print(n)

# 將小數轉成分數
from fractions import Fraction

n = 1.5654
# 建立一個轉換物件
n = Fraction(n)

print(n)
```

12.6 python 大數排序

```python
# 大數排序
# line one n : 多少數字
# next line : 依序輸入每行一個
# sort : sort + lambda
from sys import stdin

data = stdin.read().splitlines()

i = 0

while(i < len(data)):
    T = int(data[i].strip())
    i += 1
    temp = [int(data[i + index].strip()) for index in range(T)]
    i += T
    temp = sorted(temp, key = lambda x : (x))
    for ele in temp:
        print(ele)
```

12.7 python 大數計算 2

```python
# 單行輸入
# format : n1, operation, n2
from sys import stdin

data = stdin.read().splitlines()

limit = len(data)
i = 0

while(i < limit):
    a, operation, b = map(str, data[i].split())
    a, b = int(a), int(b)
    i += 1
    if(operation == '+'):
        print(int(a + b))
    elif(operation == '-'):
        print(int(a - b))
    elif(operation == '*'):
        print(int(a * b))
    else:
        print(int(a // b))
```

12.8 python input

```python
ans = sum(map(float, input().split()))
# input: 1.1 2.2 3.3 4.4 5.5
print(ans) # 16.5

(n, m) = map(int, input().split()) # 300 200
print(n * m) # 60000

Arr = list(map(int, input().split()))
# input: 1 2 3 4 5
print(Arr) # [1, 2, 3, 4, 5]
```

12.9 python 大數計算

```python
# 讀取多行輸入
# line one first number
```

```python
# line two operation
# line three second number
from sys import stdin

data = stdin.read().splitlines()

limit = len(data)
i = 0
while(i < limit):
    a = int(data[i].strip())
    i += 1
    operation = data[i].strip()
    i += 1
    b = int(data[i].strip())
    i += 1
    if(operation == '*'):
        print(int(a * b))
    else:
        print(int(a / b))
```

13 zformula

13.1 formula

13.1.1 Pick 公式

給定頂點坐標均是整點的簡單多邊形，面積 = 內部格點數 + 邊上格點數/2-1

13.1.2 圖論

1. 對於平面圖，$F = E - V + C + 1$，C 是連通分量數
2. 對於平面圖，$E < 3V - 6$
3. 對於連通圖 G，最大獨立點集的大小設為 I(G)，最大匹配大小設為 M(G)，最小點覆蓋設為 Cv(G)，最小邊覆蓋設為 Ce(G)。對於任意連通圖：
   (a) $I(G) + Cv(G) = |V|$
   (b) $M(G) + Ce(G) = |V|$
4. 對於連通二分圖：
   (a) $I(G) = Cv(G)$
   (b) $M(G) = Ce(G)$
5. 最大權閉合圖：
   (a) $C(u, v) = \infty, (u, v) \in E$
   (b) $C(S, v) = W_v, W_v > 0$
   (c) $C(v, T) = -W_v, W_v < 0$
   (d) ans=$\sum_{W_v > 0} W_v - flow(S, T)$
6. 最大密度子圖：
   (a) 求 $max\left(\frac{W_e + W_v}{|V'|}\right), e \in E', v \in V'$
   (b) $U = \sum_{v \in V} 2W_v + \sum_{e \in E} W_e$
   (c) $C(u, v) = W_{(u,v)}, (u, v) \in E$，雙向邊
   (d) $C(S, v) = U, v \in V$
   (e) $D_u = \sum_{(u,v) \in E} W_{(u,v)}$
   (f) $C(v, T) = U + 2g - D_v - 2W_v, v \in V$
   (g) 二分搜 g：
   $l = 0, r = U, eps = 1/n^2$
   if$((U \times |V| - flow(S, T))/2 > 0)$ $l = mid$
   else $r = mid$
   (h) ans=$min\_cut(S, T)$
   (i) $|E| = 0$ 要特殊判斷
7. 弦圖：
   (a) 點數大於 3 的環都要有一條弦
   (b) 完美消除序列從後往前依次給每個點染色，給每個點染上可以染的最小顏色
   (c) 最大團大小 = 色數
   (d) 最大獨立集: 完美消除序列從前往後能選就選
   (e) 最小團覆蓋: 最大獨立集的點和他延伸的邊構成
   (f) 區間圖是弦圖
   (g) 區間圖的完美消除序列: 將區間按造又端點由小到大排序
   (h) 區間圖染色: 用線段樹做

13.1.3 dinic 特殊圖複雜度

1. 單位流：$O\left(min\left(V^{3/2}, E^{1/2}\right) E\right)$
2. 二分圖：$O\left(V^{1/2} E\right)$

13.1.4 0-1 分數規劃

$x_i = \{0, 1\}$，x_i 可能會有其他限制，求 $max\left(\frac{\sum B_i x_i}{\sum C_i x_i}\right)$

1. $D(i, g) = B_i - g \times C_i$
2. $f(g) = \sum D(i, g) x_i$
3. $f(g) = 0$ 時 g 為最佳解，$f(g) < 0$ 沒有意義
4. 因為 $f(g)$ 單調可以二分搜 g
5. 或用 Dinkelbach 通常比較快

```cpp
binary_search(){
  while(r-l>eps){
    g=(l+r)/2;
    for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
    找出一組合法x[i]使f(g)最大;
    if(f(g)>0) l=g;
    else r=g;
  }
  Ans = r;
}
Dinkelbach(){
  g=任意狀態(通常設為0);
  do{
    Ans=g;
    for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
    找出一組合法x[i]使f(g)最大;
    p=0,q=0;
    for(i:所有元素)
      if(x[i])p+=B[i],q+=C[i];
    g=p/q;//更新解，注意q=0的情況
  }while(abs(Ans-g)>EPS);
  return Ans;
}
```

13.1.5 學長公式

1. $\sum_{d|n} \phi(n) = n$
2. $g(n) = \sum_{d|n} f(d) => f(n) = \sum_{d|n} \mu(d) \times g(n/d)$
3. Harmonic series $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
4. $\gamma = 0.57721566490153286060651209008240243104215$
5. 格雷碼 $= n \oplus (n >> 1)$
6. $SG(A + B) = SG(A) \oplus SG(B)$
7. 選轉矩陣 $M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$

13.1.6 基本數論

1. $\sum_{d|n} \mu(n) = [n == 1]$
2. $g(m) = \sum_{d|m} f(d) \Leftrightarrow f(m) = \sum_{d|m} \mu(d) \times g(m/d)$
3. $\sum_{i=1}^{n} \sum_{j=1}^{m}$ 互質數量 $= \sum \mu(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
4. $\sum_{i=1}^{n} \sum_{j=1}^{n} lcm(i, j) = n \sum_{d|n} d \times \phi(d)$

13.1.7 排組公式

1. k 卡特蘭 $\frac{C_n^{kn}}{n(k-1)+1}$，$C_m^n = \frac{n!}{m!(n-m)!}$
2. $H(n, m) \cong x_1 + x_2 \ldots + x_n = k, num = C_k^{n+k-1}$
3. Stirling number of 2^{nd},n 人分 k 組方法數目
   (a) $S(0, 0) = S(n, n) = 1$
   (b) $S(n, 0) = 0$
   (c) $S(n, k) = kS(n-1, k) + S(n-1, k-1)$
4. Bell number,n 人分任意多組方法數目
   (a) $B_0 = 1$
   (b) $B_n = \sum_{i=0}^{n} S(n, i)$
   (c) $B_{n+1} = \sum_{k=0}^{n} C_k^n B_k$
   (d) $B_{p+n} \equiv B_n + B_{n+1} mod p$, p is prime
   (e) $B_{p^m+n} \equiv mB_n + B_{n+1} mod p$, p is prime
   (f) From $B_0 : 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$
5. Derangement, 錯排, 沒有人在自己位置上
   (a) $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \ldots + (-1)^n \frac{1}{n!})$
   (b) $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
   (c) From $D_0 : 1, 0, 1, 2, 9, 44, 265, 1854, 14833, 133496$
6. Binomial Equality
   (a) $\sum_k \binom{r}{m+k}\binom{s}{n-k} = \binom{r+s}{m+n}$
   (b) $\sum_k \binom{l}{m+k}\binom{s}{n+k} = \binom{l+s}{l-m+n}$
   (c) $\sum_k \binom{l}{m+k}\binom{s+k}{n}(-1)^k = (-1)^{l+m}\binom{s-m}{n-l}$
   (d) $\sum_{k \le l} \binom{l-k}{m}\binom{s}{k-n}(-1)^k = (-1)^{l+m}\binom{s-m-1}{l-n-m}$
   (e) $\sum_{0 \le k \le l} \binom{l-k}{m}\binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
   (f) $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$
   (g) $\binom{r}{m}\binom{m}{k} = \binom{r}{k}\binom{r-k}{m-k}$
   (h) $\sum_{k \le n} \binom{r+k}{k} = \binom{r+n+1}{n}$
   (i) $\sum_{0 \le k \le n} \binom{k}{m} = \binom{n+1}{m+1}$
   (j) $\sum_{k \le m} \binom{m+r}{k} x^k y^k = \sum_{k \le m} \binom{-r}{k}(-x)^k (x+y)^{m-k}$

13.1.8 冪次, 冪次和

1. $a^{b\%}P = a^{b\%\varphi(p)+\varphi(p)}, b \ge \varphi(p)$
2. $1^3 + 2^3 + 3^3 + \ldots + n^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
3. $1^4 + 2^4 + 3^4 + \ldots + n^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
4. $1^5 + 2^5 + 3^5 + \ldots + n^5 = \frac{n^6}{6} + \frac{n^5}{2} + \frac{5n^4}{12} - \frac{n^2}{12}$
5. $0^k + 1^k + 2^k + \ldots + n^k = P(k), P(k) = \frac{(n+1)^{k+1} - \sum_{i=0}^{k-1} C_i^{k+1} P(i)}{k+1}, P(0) = n + 1$
6. $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^{n} C_k^{n+1} B_k m^{n+1-k}$
7. $\sum_{j=0}^{m} C_j^{m+1} B_j = 0, B_0 = 1$
8. 除了 $B_1 = -1/2$，剩下的奇數項都是 0
9. $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$

13.1.9 Burnside's lemma

1. $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
2. $X^g = t^{c(g)}$
3. G 表示有幾種轉法，X^g 表示在那種轉法下，有幾種是會保持對稱的，t 是顏色數，$c(g)$ 是循環節不動的面數。
4. 正立方體塗三顏色，轉 0 有 3^6 個元素不變，轉 90 有 6 種，每種有 3^3 不變，180 有 3×3^4，120(角) 有 8×3^2，180(邊) 有 6×3^3，全部 $\frac{1}{24}\left(3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3\right) = 57$

13.1.10 Count on a tree

1. Rooted tree: $s_{n+1} = \frac{1}{n} \sum_{i=1}^{n} (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$
2. Unrooted tree:
   (a) Odd:$a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$
   (b) Even:$Odd + \frac{1}{2} a_{n/2}(a_{n/2} + 1)$
3. Spanning Tree
   (a) 完全圖 $n^n - 2$
   (b) 一般圖 (Kirchhoff's theorem)$M[i][i] = degree(V_i)$,$M[i][j] = -1$,if have $E(i, j)$,0 if no edge. delete any one row and col in A, $ans = det(A)$

13.1.11 循環小數轉分數

1. 若 $x = 0.\overline{a}$，則

$$x = \frac{a}{\underbrace{99\ldots9}_{k\text{ digits}}}$$

其中 a 為循環節，k 為循環節的位數。

2. 例子：

$$0.\overline{37} = \frac{37}{99}$$

$$0.\overline{5} = \frac{5}{9}$$

13.1.12 循環小數轉分數

1. 純循環小數：若 $x = 0.\overline{a}$，其中 a 為循環節、長度為 k，

$$x = \frac{a}{\underbrace{99\ldots9}_{k\text{ digits}}}$$

例：

$$0.\overline{37} = \frac{37}{99}, \quad 0.\overline{5} = \frac{5}{9}$$

2. 混循環小數：若 $x = 0.b\overline{a}$，其中 b 為前綴、長度 m，a 為循環節、長度 k，

$$x = \frac{(b \cdot 10^k + a) - b}{10^m(10^k - 1)}$$

例：

$$0.12\overline{3} = \frac{(12 \cdot 10^1 + 3) - 12}{10^2(10^1 - 1)} = \frac{123 - 12}{100 \cdot 9} = \frac{111}{900} = \frac{37}{300}$$

13.1.13 常見級數與組合公式

1. 平方和公式：

$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$4^2 + 6^2 + \cdots + (2n)^2 = \frac{(2n)(n+1)(2n+1)}{3}$$

$$1^2 + 3^2 + \cdots + (2n+1)^2 = \frac{n(2n-1)(2n+1)}{3}$$

2. 立方和公式：

$$1^3 + 2^3 + \cdots + n^3 = \frac{n^4 + 2n^3 + n^2}{4}$$

3. 四次方和公式：

$$1^4 + 2^4 + \cdots + n^4 = \frac{6n^5 + 15n^4 + 10n^3 - n}{30}$$

4. 五次方和公式：

$$1^5 + 2^5 + \cdots + n^5 = \frac{2n^6 + 6n^5 + 5n^4 - n^2}{12}$$

5. 六次方和公式：

$$1^6 + 2^6 + \cdots + n^6 = \frac{6n^7 + 21n^6 + 21n^5 - 7n^3 + n}{42}$$

6. 七次方和公式：

$$1^7 + 2^7 + \cdots + n^7 = \frac{3n^8 + 12n^7 + 14n^6 - 7n^4 + 2n^2}{24}$$

7. 八次方和公式：

$$1^8 + 2^8 + \cdots + n^8 = \frac{10n^9 + 45n^8 + 60n^7 - 42n^5 + 20n^3 - 3n}{90}$$

8. 九次方和公式：

$$1^9 + 2^9 + \cdots + n^9 = \frac{2n^{10} + 10n^9 + 15n^8 - 14n^6 + 10n^4 - 3n^2}{20}$$

9. 十次方和公式：

$$1^{10} + 2^{10} + \cdots + n^{10} = \frac{6n^{11} + 33n^{10} + 55n^9 - 66n^7 + 66n^5 - 33n^3 + 5n}{66}$$

10. 等比級數：

$$S = a \cdot \frac{r^n - 1}{r - 1}$$

11. 二項式係數恆等式：

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} = 2^n$$

$$\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n} = 2^n - 1$$

$$\binom{n}{1} + \binom{n}{3} + \cdots + \binom{n}{n-k} = 2^{n-1}$$

12. 分配問題（玩具分給小孩）：

(a) n 個玩具，k 位小孩，可以有人沒拿到：

$$\binom{n+k-1}{n} = \binom{n+k-1}{k-1}$$

(b) n 個玩具，k 位小孩，每個人至少一個：

$$\binom{n-1}{k-1}$$

13.1.14 位元運算

(a) 位元條件：

$$(x + k) \,\&\, (y + k) = 0$$

(b) 加法恆等式（利用 XOR 與 AND）：

$$a + b = (a \oplus b) + 2 \cdot (a \,\&\, b)$$

(c) OR 與 AND 的關係：

$$a \mid b = a + b - (a \,\&\, b)$$

(d) 交換兩數：

$$a = a \oplus b, \quad b = a \oplus b, \quad a = a \oplus b$$

(e) 取得最低位的 1：

$$x \,\&\, (-x)$$

(f) 清除最低位的 1：

$$x \,\&\, (x - 1)$$

13.1.15 數論公式

13. Bezout's identity：

$$ax + by = \gcd(a, b) \quad \text{（必定存在整數解 } x, y\text{）}$$

14. 模指數運算（冪的冪）:

$$a^{b^c} \equiv a^{\text{power\_mod}(b,c,\,\text{MOD}-1)} \pmod{\text{MOD}}$$

海龍公式：

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}, \quad s = \frac{a+b+c}{2}$$

14 Интернационал

14.1 保佑

```
//                       _oo0oo_
//                      o8888888o
//                      88" . "88
//                      (| -_- |)
//                      0\  =  /0
//                    ___/`---'\___
//                  .' \\|     |// '.
//                 / \\|||  :  |||// \
//                / _||||| -:- |||||- \
//               |   | \\\  -  /// |   |
//               | \_|  ''\---/''  |_/ |
//               \  .-\__  '-'  ___/-. /
//             ___'. .'  /--.--\  `. .'___
//          ."" '<  `.___\_<|>_/___.' >' "".
//         | | :  `- \`.;`\ _ /`;.`/ - ` : | |
//         \  \ `_.   \_ __\ /__ _/   .-` /  /
//     =====`-.____`.___ \_____/___.-`___.-'=====
//                       `=---='
//
//     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//            佛祖保佑             永無BUG
```

```
#       ┌─┐       ┌─┐
#    ┌──┘ ┴───────┘ ┴──┐
#    │                 │
#    │       ───       │
#    │  ─┬─┘     └─┬─  │
#    │                 │
#    │       ─┴─       │
#    │                 │
#    └───┐         ┌───┘
#        │         │
#        │         │
#        │         │
#        │         └──────────────┐
#        │                        │
#        │                        ├─┐
#        │                        ┌─┘
#        │                        │
#        └─┐  ┐  ┌───────┬──┐  ┌──┘
```

```
#         │ ─┤ ─┤        │ ─┤ ─┤
#         └──┴──┘        └──┴──┘
#             神獸保佑  永無BUG!


// ##              ##################
// ##              ##
// ##              ##
// ##              ##
// ##              ##
// ##              ##
// ##              ##
// ##              ##
// ################################
//                 ##              ##
//                 ##              ##
//                 ##              ##
//                 ##              ##
//                 ##              ##
//                 ##              ##
//                 ##              ##
// ################                ##
//
//         元首保佑  永無BUG

//                  __
// _____ __ __   < @ \    _  __ _____
// \____\\_\\_)   /**|   (_//_//____/
//  \____\(_)\*\/****\/*/(_)/____/
//   \____\_/_\*\****/*/_\_/____/
//           u/u/u|****|u\u\u
//            u/u/|****|\u\u
//                |*||*|
//                 |  |
//                /+--+\
//               ||  卐  ||
//                \\==//
//
//           神獸保佑  永無BUG
```

ACM ICPC Team Reference - BogoSort

Contents

ACM ICPC Judge Test - BogoSort

C++ Resource Test

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

namespace system_test {

const size_t KB = 1024;
const size_t MB = KB * 1024;
const size_t GB = MB * 1024;

size_t block_size, bound;
void stack_size_dfs(size_t depth = 1) {
  if (depth >= bound)
    return;
  int8_t ptr[block_size]; // 若無法編譯將
      block_size 改成常數
  memset(ptr, 'a', block_size);
  cout << depth << endl;
  stack_size_dfs(depth + 1);
}

void stack_size_and_runtime_error(size_t
    block_size, size_t bound = 1024) {
  system_test::block_size = block_size;
  system_test::bound = bound;
  stack_size_dfs();
}

double speed(int iter_num) {
  const int block_size = 1024;
  volatile int A[block_size];
  auto begin = chrono::high_resolution_clock
      ::now();
  while (iter_num--)
    for (int j = 0; j < block_size; ++j)
      A[j] += j;
  auto end = chrono::high_resolution_clock::
      now();
  chrono::duration<double> diff = end -
      begin;
  return diff.count();
}

void runtime_error_1() {
  // Segmentation fault
  int *ptr = nullptr;
  *(ptr + 7122) = 7122;
}

void runtime_error_2() {
  // Segmentation fault
  int *ptr = (int *)memset;
  *ptr = 7122;
}

void runtime_error_3() {
  // munmap_chunk(): invalid pointer
  int *ptr = (int *)memset;
  delete ptr;
}

void runtime_error_4() {
  // free(): invalid pointer
  int *ptr = new int[7122];
  ptr += 1;
  delete[] ptr;
}

void runtime_error_5() {
  // maybe illegal instruction
  int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_6() {
  // floating point exception
  volatile int a = 7122, b = 0;
  cout << (a / b) << endl;
}

void runtime_error_7() {
  // call to abort.
  assert(false);
}

} // namespace system_test

#include <sys/resource.h>
void print_stack_limit() { // only work in
    Linux
  struct rlimit l;
  getrlimit(RLIMIT_STACK, &l);
  cout << "stack_size = " << l.rlim_cur << "
      byte" << endl;
}
```