算法竞赛模板

by ChaomengOrion

最后修改于2024 年 10 月 31 日

目录

1 前期准备

1.1 cpp 文件一键编译测试

Windows - **build.bat**

```bat
@echo off
set exe=.\output\%~n1.exe
if not exist output mkdir output
g++ %1 -o %exe% -std=c++17 -Wall -Wextra -g3 -O2 -D_GLIBCXX_DEBUG && (
    echo [build done to %exe%]
    %exe%
)
```

1.2 cpp 模板

注意题目是不是多组样例

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

using i64 = long long;

#define LOG(...) std::cerr << "DEBUG: " << __VA_ARGS__ << std::endl;
#define LOGV(_vec, _size)                         \
    std::cerr << #_vec << " = " << '[';           \
    for (int _i = 0; _i < (_size); _i++) {        \
        std::cerr << (_vec)[_i];                  \
        if (_i != (_size) - 1) std::cerr << ", "; \
    }                                             \
    std::cerr << ']' << std::endl;

void solve() {}

int main()
{
    std::cin.tie(nullptr)->sync_with_stdio(false);
    int t;
    std::cin >> t;
    while (t--) solve();
    return 0;
}
```

2 算法

2.1 素数筛

```cpp
bool isPrime(int num)
{
    if (num == 1) return 0;
    if (num == 2 || num == 3) return 1;
    if (num % 6 != 1 && num % 6 != 5) return 0;
    int tmp = sqrt(num);
    for (int i = 5; i <= tmp; i += 6)
        if (num % i == 0 || num % (i + 2) == 0) return 0;
    return 1;
}
```

2.2 二维差分

```cpp
void best()
{
    int N, M; std::cin >> N >> M;
    std::vector map(N + 1, std::vector<int>(N + 1, 0));
    std::vector diff(N + 2, std::vector<int>(N + 2, 0));
    for (int i = 1; i <= N; i++) for (int j = 1; j <= N; j++) std::cin >> map[i][j];

    while (M--) {
        int x1, x2, y1, y2;
        std::cin >> x1 >> y1 >> x2 >> y2;
        diff[x1][y1]++;
        diff[x1][y2 + 1]--;
        diff[x2 + 1][y1]--;
        diff[x2 + 1][y2 + 1]++;
    }
//  修改
//  ###+@@@-
//  ###@@@@#
//  ###-###+
    for (int i = 1; i <= N; i++)
        for (int j = 1; j <= N; j++)
            diff[i][j] += diff[i - 1][j];

    for (int i = 1; i <= N; i++)
        for (int j = 1; j <= N; j++)
            diff[i][j] += diff[i][j - 1];

    for (int i = 1; i <= N; i++) {
        for (int j = 1; j <= N; j++) {
            std::cout << diff[i][j] + map[i][j] << ' ';
        }
        std::cout << '\n';
    }
}
```

2.3 并查集

```cpp
        void solve()
        {
            int N, M;
            std::cin >> N >> M;
            std::vector<int> fa(N + 1);
            for (int i = 0; i <= N; i++) {
                fa[i] = i;
            }
            auto find = [&](auto&& find, int x) -> int {
                return x == fa[x] ? x : fa[x] = find(find, fa[x]);
            };

            while (M--) {
                int Z, X, Y;
                std::cin >> Z >> X >> Y;
                if (Z == 1) {
                    fa[find(find, Y)] = fa[find(find, X)];
                } else if (Z == 2) {
                    std::cout << (fa[find(find, Y)] == fa[find(find, X)] ? 'Y' : 'N') << std::endl;
                }
            }
        }
```

2.4 最短路 (Dijkstra)

```
void solve()
{
    int N, M, S; // 点 边 出发点
    std::cin >> N >> M >> S;
    std::vector<std::vector<std::pair<int, int>>> graph(N + 1);
    for (int i = 0; i < M; i++) {
        int u, v, len;
        std::cin >> u >> v >> len;
        graph[u].emplace_back(v, len);
        //graph[v].to.emplace_back(u, len);
    }

    std::priority_queue<std::pair<int, int>, std::vector<std::pair<int, int>>, std::greater<>> pq;
    std::vector<int> dis(N + 1, INT_MAX);
    dis[S] = 0;
    pq.emplace(0, S);
    while(!pq.empty()) {
        auto [d, i] = pq.top();
        pq.pop();
        if (d != dis[i]) continue;

        for (auto [nid, cost] : graph[i]) {
            if (dis[nid] > dis[i] + cost) {
                dis[nid] = dis[i] + cost;
                pq.emplace(dis[nid], nid);
            }
        }
    }

    for (int i = 1; i <= N; i++) {
        std::cout << dis[i] << ' ';
    }
}
```

3 数据结构

3.1 树状数组

```cpp
#include <bits/stdc++.h>
using i64 = long long;
class TreeArray
{
private:
    int size;
    std::vector<int> arr;
public:
    TreeArray(int len) : size(len), arr(len + 1, 0) { }

    TreeArray(std::vector<int>& source) : size(source.size() - 1), arr(size + 1, 0)
    {
        for (int i = 1; i <= size; i++) {
            add(i, source[i]);
        }
    }
    inline static int lowbit(int x) { return x & -x; }
    void add(int pos, int value) // 第 pos 项加 value
    {
        while (pos <= size) {
            arr[pos] += value;
            pos += lowbit(pos);
        }
    }
    i64 query(int pos) // 查询 [1,pos] 项的和
    {
        i64 sum = 0;
        while (pos >= 1) {
            sum += arr[pos];
            pos -= lowbit(pos);
        }
        return sum;
    }
    i64 query(int l, int r) { return query(r) - query(l - 1); } // 查询 [l,r] 项的和
};
```

3.2 ST 表

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

template<typename T>
class ST {
private:
    std::vector<std::vector<T>> dp;
    T (*get) (T, T);
// [](int x, int y) { return std::gcd(x, y); }
// [](int x, int y) { return std::max(x, y); }
// [](int x, int y) { return std::min(x, y); }
public:
    ST(const std::vector<T>& inputs, auto getFunc) {
        get = getFunc;
        size_t len = inputs.size();
        int exp = log2(len);
        // dp[s][k] 代表从 s 出发走 2^k 步内的最值
        dp.resize(len, std::vector<T>(exp + 1, 0));
        for (size_t s = 0; s < len; s++) {
            dp[s][0] = inputs[s];
        }

        for (int k = 1; k <= exp; k++) {
            for (size_t s = 0; s + (1 << k) <= len; s++) {
                dp[s][k] = get(dp[s][k - 1], dp[s + (1 << (k - 1))][k - 1]);
            }
        }
    }

    T query(size_t start, size_t end) const {
        if (start > end) throw std::invalid_argument("start should be less than or
        ↪  equal to end");
        int exp = log2(end - start + 1);
        return get(dp[start][exp], dp[end - (1 << exp) + 1][exp]);
    }
};
```

4 数学相关

4.1 模运算

```
i64 add(i64 a, i64 b, i64 p) { // 加
    return (a % p + b % p) % p;
}

i64 sub(i64 a, i64 b, i64 p) { // 减
    return (a % p - b % p) % p;
}

i64 mul(i64 a, i64 b, i64 p) { // a > p 乘
    a %= p;
    b %= p;
    i64 ans = 0;
    while (b > 0) {
        if (b & 1) {
            ans += a;
            ans %= p;
        }
        a <<= 1;
        a %= p;
        b >>= 1;
    }
    return ans;
}
```

5 杂项

5.1 快速读入

```
inline int read()
{
    int x = 0, sgn = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-') sgn = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = x * 10 + ch - '0';
        ch = getchar();
    }
    return x * sgn;
}
```