algorithm template

zinan.xu.dev@gmail.com

December 22, 2025

Contents

数论

素性检测

```cpp
#include<vector>
namespace PrimeTest {
    long long mul(long long a, long long b, long long mod){
        return (__int128) a * b % mod;
    }

    long long Pow(long long a, long long b, long long mod){
        //mod <= 10^18.
        long long res = 1;
        while(b){
            if (b&1) res = mul(res, a, mod);
            b >>= 1;
            a = mul(a, a, mod);
        }
        return res;
    }

    std::vector<long long> pr = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
    ↪  37};

    bool rabin_test(long long a, long long n, long long s, long long d){
        long long u = Pow(a, d, n);
        if (u == 1 or u == n - 1) return false;

        for(long long i = 1; i < s; i++){
            u = mul(u, u, n);
            if (u == n - 1) return false;
        }
        return true;
    }

    bool rabin_miller(long long n){
        if (n < 2) return false;
        if (n % 2 == 0) return n==2;
        long long res = 1;
        long  s = 0, d = n-1;
        while(d%2==0) {
            s++;
            d>>=1;
        }

```

```
41          for(long long i = 0;i<pr.size();i++){
42              if (n%pr[i] == 0) {
43                  return n == pr[i];
44              }
45              if (rabin_test(pr[i], n, s, d)){
46                  return false;
47              }
48          }
49          return true;
50      }
51  }
```

树

树的直径

```cpp
#include <vector>
#include <tuple>
namespace TreeDiameter {
    /*
     * 无向正权树的最大直径，限制：
     * 1. 直径需要<= LONG_LONG_MAX
     * 2. 单颗树，而非森林
     */
    using namespace std;
    using Graph = vector<vector<pair<int, long long>>>; // 起点对应的边(终点 & 权值)

    /*
     * Input: 树，起始点
     * Output: 离起始点最大的距离，对应的点
     * 复杂度: O(边数)
     */
    pair<long long, int> dfs(const vector<vector<pair<int, long long>>> &g, int cur, int par = -1) {
        pair<long long, int> ret(0, cur);
        for (auto e : g[cur]) {
            if (e.first == par) continue;
            auto cost = dfs(g, e.first, cur);
            cost.first += e.second;
            ret = max(ret, cost);
        }
        return ret;
    }
    /*
     * Input: 树
     * Output: 直径起点，直径终点，直径长度
     */
    tuple<int, int, long long> tree_diameter(const vector<vector<pair<int, long long>>> &g) {
        auto u = dfs(g, 0, -1).second;
        long long dist;
        int v;
        tie(dist, v) = dfs(g, u, -1);
        return make_tuple(u, v, dist);
    }

```

```
39      /*
40       * 会搜索出一条从cur到goal的路径，结果会放在path里面
41       * Input: 树
42       * Output: 路径
43       * 复杂度：O(边数)
44       */
45      void path_restoration(const vector<vector<pair<int, long long>>> &g,
        ↪  vector<int> &path, int cur, int par, int &goal) {
46          path.push_back(cur);
47          if (cur == goal) {
48              goal = -1;
49              return;
50          }
51
52          for (auto e : g[cur]) {
53              int nxt = e.first;
54              if (nxt == par) continue;
55              path_restoration(g, path, nxt, cur, goal);
56              if (goal == -1) return;
57          }
58
59          if (goal == -1) {
60              return;
61          }
62          path.pop_back();
63      }
64  }
65
```