Standard Code Library

mobbb

SMU

May 10, 2024

Contents

一切的开始

宏定义

- 需要 C++11

```
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(x...) do { cout << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... A>
void err(T<t> a, A... x) { for (auto v: a) cout << v << ' '; err(x...); }
template<typename T, typename... A>
void err(T a, A... x) { cout << a << ' '; err(x...); }
#else
#define dbg(...)
#endif
// -----------------------------------------------------------------------------
```

数据结构

ST 表

- 二维

```
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

数学

long * long 整数 mould

```
ll mul(ll x,ll y,ll m){
    x%=m,y%=m;
    ll d = ((long double)x * y / m);
    d = x * y - d * m;
    if(d >= m) d-=m;
    if(d < 0) d +=m;
    return d;
}
```

快速幂

```
ll qkm(ll a,ll b){
    ll x = a , res = 1;
    while (b){
        if (b & 1)
            res *= x , res %= P;
        b >>= 1;
        x *= x , x %= P;
    }
    return res;
}
```

扩展欧几里得

```
int exgcd(int a,int b,int &x,int &y){
    if(b == 0){
        y = 0;
        x = 1;
        return a;
    }
    int d = exgcd(b,a%b,y,x);
    y -= a/b*x;
    return d;
}
```

线性筛

```
struct Euler{
    vector<int> p,pri;
    Euler (int n){
        p.resize(n + 1);
        pri.resize(n + 1);
        int cnt = 0;
        for (int i = 2;i <= n;i++){
            if (!p[i]) p[i] = i,pri[++cnt] = i;
            for (int j = 1;j <= cnt && i * pri[j] <= n;j++){
                p[i * pri[j]] = pri[j];
                if (i == pri[j])break;
            }
        }
    }
};
```

整数分块

```
void solution(){
    ll n;cin>>n;
    unsigned ll sum = 0;
    for(ll l = 1;l<=n;l++){
        ll d = n/l,r = n/d;
        sum += (r-l+1)*d;
        l = r;
    }
}
```

最大质因数

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

using namespace std;

typedef long long ll;

int t;
long long max_factor, n;

long long gcd(long long a, long long b) {
  if (b == 0) return a;
  return gcd(b, a % b);
}

long long quick_pow(long long x, long long p, long long mod) {  // 快速幂
  long long ans = 1;
  while (p) {
    if (p & 1) ans = (__int128)ans * x % mod;
    x = (__int128)x * x % mod;
    p >>= 1;
  }
  return ans;
}

bool Miller_Rabin(long long p) {  // 判断素数
  if (p < 2) return 0;
  if (p == 2) return 1;
  if (p == 3) return 1;
  long long d = p - 1, r = 0;
  while (!(d & 1)) ++r, d >>= 1;  // 将 d 处理为奇数
  for (long long k = 0; k < 10; ++k) {
    long long a = rand() % (p - 2) + 2;
    long long x = quick_pow(a, d, p);
    if (x == 1 || x == p - 1) continue;
    for (int i = 0; i < r - 1; ++i) {
      x = (__int128)x * x % p;
      if (x == p - 1) break;
    }
    if (x != p - 1) return 0;
  }
  return 1;
}

long long Pollard_Rho(long long x) {
  long long s = 0, t = 0;
  long long c = (long long)rand() % (x - 1) + 1;
  int step = 0, goal = 1;
  long long val = 1;
  for (goal = 1;; goal *= 2, s = t, val = 1) {  // 倍增优化
    for (step = 1; step <= goal; ++step) {
      t = ((__int128)t * t + c) % x;
      val = (__int128)val * abs(t - s) % x;
      if ((step % 127) == 0) {
        long long d = gcd(val, x);
        if (d > 1) return d;
      }
    }
    long long d = gcd(val, x);
    if (d > 1) return d;
  }
}

void fac(long long x) {
  if (x <= max_factor || x < 2) return;
  if (Miller_Rabin(x)) {              // 如果 x 为质数
    max_factor = max(max_factor, x);  // 更新答案
    return;
  }
  long long p = x;
  while (p >= x) p = Pollard_Rho(x);  // 使用该算法
  while ((x % p) == 0) x /= p;
  fac(x), fac(p);  // 继续向下分解 x 和 p
}

int main() {
  scanf("%d", &t);
  while (t--) {
    srand((unsigned)time(NULL));
    max_factor = 0;
    scanf("%lld", &n);
    fac(n);
    if (max_factor == n)  // 最大的质因数即自己
```

```
      printf("Prime\n");
    else
      printf("%lld\n", max_factor);
  }
  return 0;
}

```

图论

LCA

- 倍增

```
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

二维几何：点与向量

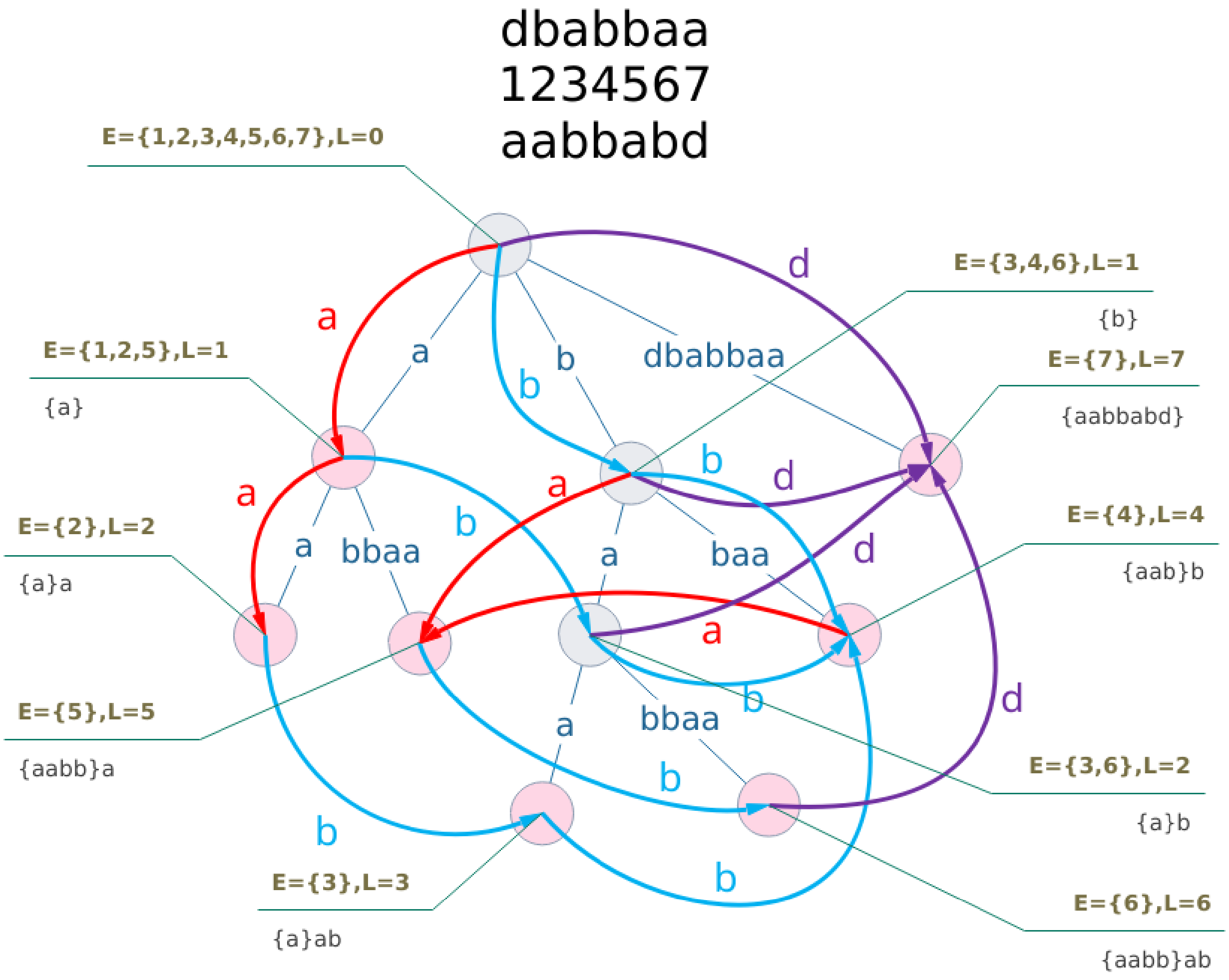
```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// --------------------------------------------
```

字符串

后缀自动机



杂项

STL

- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```