

[Posts and Telecommunications Institute of Technology](http://portal.ptit.edu.vn/eng/)

Team Notebook

PTIT.PHOENIX

2019

Contents

[1 Toán 2](#_Toc26195800)

[1.1 Phi hàm Euler 2](#_Toc26195801)

[1.2 Modulo trick 2](#_Toc26195802)

[1.3 Lehman 2](#_Toc26195803)

[1.4 Miller Rabin 3](#_Toc26195804)

[1.5 Extended Euclid 3](#_Toc26195805)

[1.6 Đếm các số ≤ n có k bit 1 3](#_Toc26195806)

[1.7 Duyệt các tập con khác rỗng của S 4](#_Toc26195807)

[1.8 Bunyakovsky cho 2 bộ số 4](#_Toc26195808)

[1.9 Số Catalan 4](#_Toc26195809)

[1.10 Discrete logarithm 4](#_Toc26195810)

[1.11 Discrete root 5](#_Toc26195811)

[1.12 Chinese remainder theorem 5](#_Toc26195812)

[2 Hình học 6](#_Toc26195813)

[2.1 Struct 6](#_Toc26195814)

[2.2 Ccw 6](#_Toc26195815)

[2.3 Giao điểm 2 đường thẳng 6](#_Toc26195816)

[2.4 Giao điểm giữa đường tròn và đường thẳng 7](#_Toc26195817)

[2.5 Monotone chain 7](#_Toc26195818)

[2.6 Công thức trong tam giác 7](#_Toc26195819)

[2.7 Định lí Pick 7](#_Toc26195820)

[2.8 Emo Welzl: Đường tròn nhỏ nhất chứa mọi điểm cho trước 7](#_Toc26195821)

[2.9 Đường tròn đi qua nhiều điểm nhất 8](#_Toc26195822)

[2.10 Convex Hull Trick 9](#_Toc26195823)

[3 Đồ thị 10](#_Toc26195824)

[3.1 Tarjan 10](#_Toc26195825)

[3.2 Khớp & cầu 10](#_Toc26195826)

[3.3 Sắp xếp topo 10](#_Toc26195827)

[3.4 Phát hiện chu trình trong đồ thị có hướng 10](#_Toc26195828)

[3.5 Chu trình Euler 10](#_Toc26195829)

[3.6 Bellman Ford 11](#_Toc26195830)

[3.7 Prim 11](#_Toc26195831)

[3.8 Cặp ghép cực đại không trọng số 12](#_Toc26195832)

[3.9 Dinitz 12](#_Toc26195833)

[3.10 Hungari 13](#_Toc26195834)

[4 Xử lý xâu 14](#_Toc26195835)

[4.1 Z algorithm 14](#_Toc26195836)

[4.2 Manacher 14](#_Toc26195837)

[4.3 KMP 15](#_Toc26195838)

[4.4 Suffix array và Longest common prefix 15](#_Toc26195839)

[5 Khác 15](#_Toc26195840)

[5.1 Centroid decomposition 15](#_Toc26195841)

[5.2 Heavy-light decomposition 15](#_Toc26195842)

[5.3 Treap 17](#_Toc26195843)

[6 . Trình chấm 18](#_Toc26195844)

[6.1 Comp.sh 18](#_Toc26195845)

[6.2 Main.cpp 18](#_Toc26195846)

[6.3 Soldung.cpp 18](#_Toc26195847)

[6.4 Solsai.cpp 18](#_Toc26195848)

[6.5 Lệnh biên dịch 18](#_Toc26195849)

# Toán

## Phi hàm Euler

|  |
| --- |
| ɸ(n) là số các số nguyên dương ≤ n và nguyên tố cùng nhau với n  ɸ(1) = 1  ɸ(n) = (p – 1)pk – 1 với n là luỹ thừa bậc k của số nguyên tố p  ɸ(mn) = ɸ(m) × ɸ(n) với m và n nguyên tố cùng nhau  n = p1k1…prkr với pj là các số nguyên tố phân biệt thì  ɸ(n) = (p1 – 1)p1k1 – 1…(pr – 1)prkr – 1 |
| int phi(int n) {  int res = n;  for (int i = 2; (long long)i \* i <= n; i++)  if (n % i == 0) {  while (n % i == 0) n /= i;  res -= res / i;  }  if (n > 1) res -= res / n;  return res;  } |

## Modulo trick

|  |
| --- |
| (A / B) % MOD = (A % (MOD × B)) / B  Điều kiện: không có |
| (A / B) % MOD = ((A % MOD) × (Bɸ(MOD) - 1 % MOD)) % MOD  Điều kiện: B và MOD nguyên tố cùng nhau |
| (A / B) % MOD = ((A % MOD) × (BMOD - 2 % MOD)) % MOD  Điều kiện: B và MOD nguyên tố cùng nhau, MOD nguyên tố |
| AN % MOD = AN % ɸ(MOD) % MOD  Điều kiện: A và MOD nguyên tố cùng nhau |
| Điều kiện: A và MOD nguyên tố cùng nhau |
| (Aɸ(n) – 1) % n = 0 |
| Số tự nhiên là số nguyên tố khi và chỉ khi |

## Lehman

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef unsigned long long ull;  ull lehman\_simple(ull n) {  ull n\_1\_3 = (ull ) ceil(pow(n, 1.0/3.0));  double n\_1\_6 = pow(n, 1.0/6.0);  ull ub\_d = max(n\_1\_3, (ull ) 19);  for(ull d=2; d<=ub\_d; d++)  if(n % d == 0) return d;  for(ull k=1; k<=n\_1\_3; k++) {  ull lb = ceil(2\*sqrt(k)\*sqrt(n));  ull ub = floor(2\*sqrt(k)\*sqrt(n) + n\_1\_6/(4\*sqrt(k)));  for(ull a=lb; a<=ub; a++) {  ull delta = a\*a - 4\*k\*n;  ull b = floor(sqrt(delta));  if(b\*b == delta) {  return \_\_gcd(a+b, n);  }  }  }  return n;  }  void lehman(ull n, ull & p, ull & k, ull & m) {  m = n;  do {  p = m;  m = lehman\_simple(p);  } while(m != p);  k = 0;  while(n % p == 0) {  n /= p;  ++k;  }  m = n;  }  vector<ull> factory\_prime(ull n) {  vector <ull> vt;  ull p, k, m;  lehman(n, p, k, m);  for (int i = 1; i <= k; ++i) {  vt.push\_back(p);  }  while(m != 1) {  lehman(m, p, k, m);  for (int i = 1; i <= k; ++i) {  vt.push\_back(p);  }  }  return vt;  }  int main() {  ull n;  cin >> n;  vector<ull> vt = factory\_prime(n);  for (int i = 0; i < vt.size(); i++) cout << vt[i] << " ";  }// input: 12 output: 2 2 3 |

## Miller Rabin

|  |
| --- |
| #include <bits/stdc++.h>  typedef long long ll;  using namespace std;  ll mulmod(ll a, ll b, ll mod) {  ll x = 0,y = a % mod;  while (b > 0) {  if (b % 2 == 1) {  x = (x + y) % mod;  }  y = (y \* 2) % mod;  b /= 2;  }  return x % mod;  }  ll modulo(ll base, ll exponent, ll mod) {  ll x = 1;  ll y = base;  while (exponent > 0) {  if (exponent % 2 == 1)  x = (x \* y) % mod;  y = (y \* y) % mod;  exponent = exponent / 2;  }  return x % mod;  }  bool Miller(ll p,int iteration) {  if (p < 2) {  return false;  }  if (p != 2 && p % 2==0) {  return false;  }  ll s = p - 1;  while (s % 2 == 0) {  s /= 2;  }  for (int i = 0; i < iteration; i++) {  ll a = rand() % (p - 1) + 1, temp = s;  ll mod = modulo(a, temp, p);  while (temp != p - 1 && mod != 1 && mod != p - 1) {  mod = mulmod(mod, mod, p);  temp \*= 2;  }  if (mod != p - 1 && temp % 2 == 0) {  return false;  }  }  return true;  }  int main() {  int iteration = 5;  ll num;  cout<<"Enter integer to test primality: ";  cin>>num;  if (Miller(num, iteration))  cout<<num<<" is prime"<<endl;  else  cout<<num<<" is not prime"<<endl;  return 0;  } |

## Extended Euclid

Tìm x, y sao cho ax + by = gcd(a, b)**;** x, y thoả mãn |x| + |y| nhỏ nhất và x ≤ y.

|  |
| --- |
| #include <bits/stdc++.h>  #define X first  #define Y second  using namespace std;  typedef long long ll;  typedef pair<ll, ll> ii;  typedef pair<ll, ii> triple;  ii extended\_gcd(ll a, ll b){  ii qr, st;  if (b==0) return ii(1, 0);  else {  qr=ii(a/b, a%b);  st=extended\_gcd(b, qr.Y);  return ii(st.Y, st.X-qr.X\*st.Y);  }  }  main(){  ll p, q;  ii ww;  for (;;){  if (scanf("%lld%lld", &p, &q) < 0) return 0;  ww = extended\_gcd(p, q);  printf("%lld %lld %lld\n", ww.X, ww.Y, \_\_gcd(p, q));  }  } |

## Đếm các số ≤ n có k bit 1

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  #define LL long long  LL getBit(LL x){  LL ans = -1;  while(x) {  ans++;  x >>= 1;  }  return ans;  }  LL c[65][65], a, b;  LL calC(LL m, LL k){  if(c[m][k] != 0) return c[m][k];  if(k == 0 || k == m) return c[m][k] = 1;  return c[m][k] = calC(m - 1, k) + calC(m-1, k-1);  }  LL f(LL a, LL k){  if(k < 0) return 0LL;  LL m = getBit(a);  if(m < k) return 0LL;  return calC(m, k) + f(a & ((1ll<<m)-1ll), k-1ll);  }  int main() {  LL n, k; cin >> n >> k; cout << f(n, k) << endl;  } |

## Duyệt các tập con khác rỗng của S

|  |
| --- |
| for (int i=S; i; i=(i-1)&S) |

## Bunyakovsky cho 2 bộ số

|  |
| --- |
| Với 2 bộ số và ta có:  Dấu “=” xảy ra khi và chỉ khi  Hệ quả: |

## Số Catalan

|  |
| --- |
| Ứng dụng của Cn:  1. Số cây nhị phân có n đỉnh  2. Số xâu ngoặc đúng có n cặp dấu đóng mở ngoặc tương ứng  3. Số cách chia 1 đa giác lồi có n+2 cạnh thành các tam giác bằng cách nối các đỉnh với nhau mà không cắt nhau    4. Số cây nhị phân đầy đủ (mỗi đỉnh có 2 đỉnh con hoặc không có đỉnh con nào) có n+1 lá  5. Số lượng đường đi với 2n bước trên lưới hình chữ nhật từ điểm trái dưới (0,0) đến điểm phải trên (n,n) mà không vượt qua đường chéo chính    6. Số cách thêm n cặp dấu ngoặc (hoặc n-1) vào 1 xâu n+1 kí tự mà vẫn thoả mãn tính đúng của các dấu ngoặc  7. Số hoán vị độ dài n mà không có 3 phần tử liên tiếp nào tạo thành dãy tăng dần  8. Các số từ 1 đến n xếp lần lượt theo chiều kim đồng hồ thành vòng tròn. Cn là số cách chia tập hợp các số từ 1 đến n thành các tập hợp con khác rỗng sao cho không có 2 tập con nào tạo thành các đa giác giao nhau |

## Discrete logarithm

Tìm số nguyên x thỏa mãn ax ≡ b (mod m), trong đó a và m nguyên tố cùng nhau

|  |
| --- |
| int solve (int a, int b, int m) {  int n = (int) sqrt (m + .0) + 1;  int an = 1;  for (int i=0; i<n; ++i)  an = (an \* a) % m;  map<int,int> vals;  for (int i=1, cur=an; i<=n; ++i) {  if (!vals.count(cur))  vals[cur] = i;  cur = (cur \* an) % m;  }  for (int i=0, cur=b; i<=n; ++i) {  if (vals.count(cur)) {  int ans = vals[cur] \* n - i;  if (ans < m)  return ans;  }  cur = (cur \* a) % m;  }  return -1;  } |

## Discrete root

Cho số nguyên tố n và 2 số nguyên a, k, tìm mọi x thỏa mãn xk ≡ a (mod n)

|  |
| --- |
| int gcd (int a, int b) {  return a ? gcd (b%a, a) : b;  }  int powmod (int a, int b, int p) {  int res = 1;  while (b)  if (b & 1)  res = int (res \* 1ll \* a % p), --b;  else  a = int (a \* 1ll \* a % p), b >>= 1;  return res;  }  int generator (int p) {  vector<int> fact;  int phi = p-1, n = phi;  for (int i=2; i\*i<=n; ++i)  if (n % i == 0) {  fact.push\_back (i);  while (n % i == 0)  n /= i;  }  if (n > 1)  fact.push\_back (n);  for (int res=2; res<=p; ++res) {  bool ok = true;  for (size\_t i=0; i<fact.size() && ok; ++i)  ok &= powmod (res, phi / fact[i], p) != 1;  if (ok) return res;  }  return -1;  }  int main() {  int n, k, a;  cin >> n >> k >> a;  if (a == 0) {  puts ("1\n0");  return 0;  }  int g = generator (n);  int sq = (int) sqrt (n + .0) + 1;  vector < pair<int,int> > dec (sq);  for (int i=1; i<=sq; ++i)  dec[i-1] = make\_pair (powmod (g, int (i \* sq \* 1ll \* k % (n - 1)), n), i);  sort (dec.begin(), dec.end());  int any\_ans = -1;  for (int i=0; i<sq; ++i) {  int my = int (powmod (g, int (i \* 1ll \* k % (n - 1)), n) \* 1ll \* a % n);  vector < pair<int,int> >::iterator it =  lower\_bound (dec.begin(), dec.end(), make\_pair (my, 0));  if (it != dec.end() && it->first == my) {  any\_ans = it->second \* sq - i;  break;  }  }  if (any\_ans == -1) {  puts ("0");  return 0;  }  int delta = (n-1) / gcd (k, n-1);  vector<int> ans;  for (int cur=any\_ans%delta; cur<n-1; cur+=delta)  ans.push\_back (powmod (g, cur, n));  sort (ans.begin(), ans.end());  printf ("%d\n", ans.size());  for (size\_t i=0; i<ans.size(); ++i)  printf ("%d ", ans[i]);  } |

## Chinese remainder theorem

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | struct CRT{ | |  | ll a[105],M[105],Mod[105],y[105],big\_mod,n,res; | |  |  | |  | ll qp(ll a,ll b,ll mod){ | |  | if(b==0) return 1; | |  | if(b&1) return a\*qp(a,b-1,mod)%mod; | |  | return qp(a\*a%mod,b>>1,mod)%mod; | |  | } | |  | CRT(){} | |  | CRT(vi \_a,vi \_Mod){ | |  | big\_mod=1; | |  | n=\_a.size(); | |  | FTB(i,1,n){ | |  | a[i]=\_a[i-1]; | |  | Mod[i]=\_Mod[i-1]; | |  | big\_mod\*=Mod[i]; | |  | } | |  | FTB(i,1,n) M[i]=big\_mod/Mod[i]; | |  | FTB(i,1,n){ | |  | y[i]=qp(M[i],Mod[i]-2,Mod[i])\*a[i]%Mod[i]; | |  | } | |  | } | |  | void Set(vi \_a,vi \_Mod){ | |  | big\_mod=1; | |  | n=\_a.size(); | |  | FTB(i,1,n){ | |  | a[i]=\_a[i-1]; | |  | Mod[i]=\_Mod[i-1]; | |  | big\_mod\*=Mod[i]; | |  | } | |  | FTB(i,1,n) M[i]=big\_mod/Mod[i]; | |  | FTB(i,1,n){ | |  | y[i]=qp(M[i],Mod[i]-2,Mod[i])\*a[i]%Mod[i]; | |  | } | |  | } | |  | ll get(){ | |  | res=0; | |  | FTB(i,1,n){ | |  | res=(res+y[i]\*M[i]%big\_mod)%big\_mod; | |  | } | |  | return res; | |  | } | |  |  | |  | } China; | |

# Hình học

## Struct

|  |
| --- |
| **struct point {**  **double x, y;**  **};**  **struct line {**  **double a, b, c;**  **}** |

## Ccw

ccw > 0: ngược chiều kim đồng hồ

ccw < 0: theo chiều kim đồng hồ

ccw = 0: thẳng hàng

|  |
| --- |
| int ccw(point a, point b, point c) {  return cmp(a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y), 0);  } |

## Giao điểm 2 đường thẳng

|  |
| --- |
| double det (double a, double b, double c, double d) {  return a \* d - b \* c;  }  bool intersect (line m, line n, point & res) {  double zn = det (m.a, m.b, n.a, n.b);  if (abs (zn) < eps)  return false;  res.x = - det (m.c, m.b, n.c, n.b) / zn;  res.y = - det (m.a, m.c, n.a, n.c) / zn;  return true;  }  bool equivalent (line m, line n) {  return abs (det (m.a, m.b, n.a, n.b)) < eps  && abs (det (m.a, m.c, n.a, n.c)) < eps  && abs (det (m.b, m.c, n.b, n.c)) < eps;  }  bool parallel (line m, line n) {  return abs (det (m.a, m.b, n.a, n.b)) < eps;  } |

## Giao điểm giữa đường tròn và đường thẳng

Giả sử tâm đường tròn là gốc toạ độ (0,0)

|  |
| --- |
| double r, a, b, c;  double x0 = -a\*c/(a\*a+b\*b), y0 = -b\*c/(a\*a+b\*b);  if (c\*c > r\*r\*(a\*a+b\*b)+eps) puts ("no points");  else if (abs (c\*c - r\*r\*(a\*a+b\*b)) < EPS) {  puts ("1 point");  cout << x0 << ' ' << y0 << '\n';  } else {  double d = r\*r - c\*c/(a\*a+b\*b);  double mult = sqrt (d / (a\*a+b\*b));  double ax, ay, bx, by;  ax = x0 + b \* mult;  bx = x0 - b \* mult;  ay = y0 - a \* mult;  by = y0 + a \* mult;  puts ("2 points");  cout << ax << ' ' << ay << '\n' << bx << ' ' << by << '\n';  } |

## Monotone chain

|  |
| --- |
| struct point {  double x, y;  };  bool cmp(point a, point b) {  return a.x < b.x || a.x == b.x && a.y < b.y;  }  bool cw(point a, point b, point c) {  return a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y) < 0;  }  bool ccw(point a, point b, point c) {  return a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y) > 0;  }  void convex\_hull(vector<point> &a) {  if (a.size() == 1)  return;  sort(a.begin(), a.end(), &cmp);  point p1 = a[0], p2 = a.back();  vector<point> up, down  up.push\_back(p1);  down.push\_back(p1);  for (size\_t i=1; i<a.size(); ++i) {  if (i==a.size()-1 || cw (p1, a[i], p2)) {  while (up.size()>=2 && !cw (up[up.size()-2], up[up.size()-1], a[i]))  up.pop\_back();  up.push\_back(a[i]);  }  if (i==a.size()-1 || ccw (p1, a[i], p2)) {  while (down.size()>=2 && !ccw (down[down.size()-2], down[down.size()-1], a[i]))  down.pop\_back();  down.push\_back(a[i]);  }  }  a.clear();  for (size\_t i=0; i<up.size(); ++i)  a.push\_back(up[i]);  for (size\_t i=down.size()-2; i>0; --i)  a.push\_back(down[i]);  } |

## Công thức trong tam giác

|  |
| --- |
| Độ dài trung tuyến:  Độ dài đường phân giác:  Bán kính đường tròn nội tiếp:  Bán kính đường tròn ngoại tiếp: |

## Định lí Pick

|  |
| --- |
| S: Diện tích đa giác lồi  I: Số điểm toạ độ nguyên trong đa giác  B: Số điểm toạ độ nguyên trên cạnh đa giác |

## Emo Welzl: Đường tròn nhỏ nhất chứa mọi điểm cho trước

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef pair<double, double> point;  typedef pair<point, double> circle;  #define X first  #define Y second  point operator + (point a, point b) { return point(a.X+b.X, a.Y+b.Y); }  point operator - (point a, point b) { return point(a.X-b.X, a.Y-b.Y); }  point operator / (point a, double x) { return point(a.X/x, a.Y/x); }  double abs(point a) { return sqrt(a.X\*a.X+a.Y\*a.Y); }  point center\_from(double bx, double by, double cx, double cy) {  double B=bx\*bx+by\*by, C=cx\*cx+cy\*cy, D=bx\*cy-by\*cx;  return point((cy\*B-by\*C)/(2\*D), (bx\*C-cx\*B)/(2\*D));  }  circle circle\_from(point A, point B, point C) {  point I = center\_from(B.X-A.X, B.Y-A.Y, C.X-A.X, C.Y-A.Y);  return circle(I+A, abs(I));  }  const int N = 100005;  int n, x[N], y[N];  point a[N];  circle f(int n, vector<point> T) {  if (T.size()==3 || n==0) {  if (T.size()==0) return circle(point(0, 0), -1);  if (T.size()==1) return circle(T[0], 0);  if (T.size()==2) return circle((T[0]+T[1])/2, abs(T[0]-T[1])/2);  return circle\_from(T[0], T[1], T[2]);  }  random\_shuffle(a+1, a+n+1);  circle Result = f(0, T);  for (int i=1; i<=n; i++)  if (abs(Result.X - a[i]) > Result.Y+1e-9) {  T.push\_back(a[i]);  Result = f(i-1, T);  T.pop\_back();  }  return Result;  }  int main() {  scanf("%d", &n);  for (int i=1; i<=n; i++) {  scanf("%d%d", &x[i], &y[i]);  a[i] = point(x[i], y[i]);  }  circle C = f(n, vector<point>());  (cout << fixed).precision(2);  cout << 2\*C.Y << endl;  } |

## Đường tròn đi qua nhiều điểm nhất

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const double eps = 1e-9;  struct point {  double x, y;  };  struct line {  double a, b, c;  };  point P[101];  double dist(point A, point B) {  return sqrt((A.x - B.x)\*(A.x - B.x) + (A.y - B.y)\*(A.y - B.y));  }  bool eq(double A, double B) {  return fabs(A - B) < eps;  }  line extract(point p1, point p2) {  line res;  res.a = p1.y - p2.y;  res.b = p2.x - p1.x;  res.c = -res.a \* p1.x - res.b \* p1.y;  return res;  }  line create(point p, double A, double B) {  line res;  if (eq(A, 0)) {  res.a = 1;  res.b = 0;  res.c = -p.x;  }  else if (eq(B, 0)) {  res.a = 0;  res.b = 1;  res.c = -p.y;  }  else {  res.a = -1/(A/B);  res.b = 1;  res.c = -res.a \* p.x - res.b \* p.y;  }  return res;  }  line midper(point p1, point p2) {  line tmp = extract(p1, p2);  point ct;  ct.x = (p1.x + p2.x) / 2;  ct.y = (p1.y + p2.y) / 2;  tmp = create(ct, tmp.a, tmp.b);  return tmp;  }  bool intersect(line l1, line l2, point &p) {  double det = l1.a \* l2.b - l1.b \* l2.a;  if (eq(det, 0)) return false;  p.x = -(l1.c \* l2.b - l2.c \* l1.b) / det;  p.y = -(l2.c \* l1.a - l1.c \* l2.a) / det;  return true;  }  int calc(int n) {  if (n <= 2) return n;  int res = 2;  for (int i = 1; i < n - 1; i++) {  for (int j = i + 1; j < n; j++) {  for (int k = j + 1; k <= n; k++) {  line l1 = midper(P[i], P[j]);  line l2 = midper(P[j], P[k]);  point its;  bool ok = intersect(l1, l2, its);  if (ok) {  int sum = 3;  double r = dist(its, P[i]);  for (int l = k + 1; l <= n; l++) {  sum += eq(r, dist(its, P[l]));  }  res = max(res, sum);  }  }  }  }  return res;  }  int main() {  int n;  while (scanf("%d", &n) and n) {  for (int i = 1; i <= n; i++) scanf("%lf %lf", &P[i].x, &P[i].y);  printf("%d\n", calc(n));  }  } |

## Convex Hull Trick

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | struct convex\_hull\_trick{  ll top=0;  ii z[maxn];  bool del(ll i,ll j,ll k){  return 1.\*(z[j].fi-z[i].fi)\*(z[k].se-z[i].se)>=1.\*(z[k].fi-z[i].fi)\*(z[j].se-z[i].se);  }  void add(ll a,ll b){  z[++top]={a,b};  while(top>=3&&del(top-2,top-1,top)){  z[top-1]=z[top];  top--;  }  }  ll query(ll x){  ll ans,l=1,r=top;  while(l<r){  ll mid=(l+r)/2;  ll p1=z[mid].fi\*x+z[mid].se;  ll p2=z[mid+1].fi\*x+z[mid+1].se;  if(p1>p2){  l=mid+1;  }  else{  r=mid;  }  }  return z[l].fi\*x+z[l].se;  }  }; | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |

# Đồ thị

## Tarjan

|  |
| --- |
| void dfs(int u) {  low[u] = num[u] = ++cnt; s.push(u); ins[u] = true;  for (int v: G[u]) {  if (num[v] == 0) {  dfs(v);  low[u] = min(low[u], low[v]);  } else if (ins[v]) low[u] = min(low[u], num[v]);  }  if (low[u] == num[u]) {  int v; // all v are in one SCC  do {  v = s.top();  s.pop();  ins[u] = false;  } while (v != u);  }  }  int main() {  for (int i = 1; i <= n; i++) if (num[i] == 0) dfs(i);  } |

## Khớp & cầu

|  |
| --- |
| vector<pair<int, int> > bridge; bool articulated[nn];  int root, child, cnt;  void dfs(int u, int p) {  low[u] = num[u] = ++cnt;  for (int v: G[u]) {  if (v == p) continue;  if (num[v] > 0) {  low[u] = min(low[u], num[v]);  } else {  dfs(v, u); child += (u == root);  low[u] = min(low[u], low[v]);  if (low[v] > num[u]) bridge.push\_back({u, v});  articulated[u] |= low[v] >= num[u];  }  }  }  int main() {  for (int i = 1; i <= n; i++) if (num[i] == 0) {  root = i; child = 0; dfs(i, 0);  articulated[root] = child > 1;  }  } |

## Sắp xếp topo

|  |
| --- |
| ll in[300005];// luu ban bac vao  bool ak[300005];//danh dau cac canh da xoa  int n,m;//n dinh va m canh  bool toposort(){  queue<LL> q;  FTB(i,1,n) if(in[i]==0) q.push(i);  while(q.size()){  ll u=q.front();q.pop();  for(auto z:ke[u]){  ll v=z.fi,g=z.se;  if(ak[g]) continue;  in[v]--;m--;  ak[g]=true;  if(in[v]==0) q.push(v);  }    }  if(m) return false;  return true;  } |

## Phát hiện chu trình trong đồ thị có hướng

|  |
| --- |
| cycle = false;  void dfs(int u) {  visit[u] = 1;  for(int v : a[u])  if (visit[v] == 0) dfs(v);  else if (visit[v] == 1) cycle = true;  visit[u] = 2;  } |

## Chu trình Euler

|  |
| --- |
| /\*  Đường đi Euler là đường đi trên đồ thị mà mỗi cạnh đi qua đúng 1 lần. Chu trình Euler là một đường đi Euler mà đỉnh đầu trùng đỉnh cuối. Một đồ thị vô hướng có chu trình Euler khi tất cả các đỉnh có bậc chẵn, và tất cả các đỉnh có bậc dương thuộc cùng một thành phần liên thông. Một đồ thị có hướng có chu trình Euler khi tất cả các đỉnh có bậc ra bằng bậc vào, và tất cả các đỉnh có bậc dương thuộc cùng một thành phần liên thông mạnh. Độ phức tạp o(V+E).  \*/  #include <bits/stdc++.h>  using namespace std;  const int MAXN = 100000;  vector<int> euler\_cycle\_directed(vector<int> adj[], int u) {  vector<int> stack, res, cur\_edge(MAXN);  stack.push\_back(u);  while (!stack.empty()) {  u = stack.back();  stack.pop\_back();  while (cur\_edge[u] < (int)adj[u].size()) {  stack.push\_back(u);  u = adj[u][cur\_edge[u]++];  }  res.push\_back(u);  }  reverse(res.begin(), res.end());  return res;  }  vector<int> euler\_cycle\_undirected(vector<int> adj[], int u) {  vector<vector<bool> > used(MAXN, vector<bool>(MAXN, false));  vector<int> stack, res, cur\_edge(MAXN);  stack.push\_back(u);  while (!stack.empty()) {  u = stack.back();  stack.pop\_back();  while (cur\_edge[u] < (int)adj[u].size()) {  int v = adj[u][cur\_edge[u]++];  if (!used[min(u, v)][max(u, v)]) {  used[min(u, v)][max(u, v)] = 1;  stack.push\_back(u);  u = v;  }  }  res.push\_back(u);  }  reverse(res.begin(), res.end());  return res;  }  int main() {  int nodes, edges, u, v;  vector<int> g1[5], g2[5], cycle;  cin >> nodes >> edges;  for (int i = 0; i < edges; i++) {  cin >> u >> v;  g1[u].push\_back(v);  g2[u].push\_back(v);  g2[v].push\_back(u);  }  cycle = euler\_cycle\_directed(g1, 0);  cout << "Eulerian cycle from 0 (directed): ";  for (int i = 0; i < (int)cycle.size(); i++)  cout << " " << cycle[i];  cout <<"\n";  cycle = euler\_cycle\_undirected(g2, 2);  cout << "Eulerian cycle from 2 (undirected): ";  for (int i = 0; i < (int)cycle.size(); i++)  cout << " " << cycle[i];  cout << "\n";  return 0;  }  /\*  input: output:  5 6 Eulerian cycle from 0 (directed): 0 1 3 4 1 2 0  0 1 Eulerian cycle from 2 (undirected): 2 1 3 4 1 0 2  1 2  2 0  1 3  3 4  4 1  \*/ |

## Bellman Ford

|  |
| --- |
| //MBF  l(s) = 0, l(v) = infinity if v is not s, pred(v) = NULL for all v  For i from 1 to n-1 do  // At iteration i, l(v) is the length of the shortest path from s to v using at most i edges  For all edges (u,v) in E do  if (l(u) + w(u,v) < l(v))  Set l(v) to l(u) + w(u,v)  Set pred(v) to u  // Detect negative cycle  Apply the MBF algorithm to the graph  For all edges in E do  if (l(u) + w(u,v) < l(v)) then  Output TRUE  Output FALSE |

## Prim

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef pair<int, int> ii;  const int N=100005, oo=0x3c3c3c3c;  int n, m, d[N];  vector<int> a[N], b[N];  int prim(int u) {  int Sum = 0;  priority\_queue<ii> qu;  for (int i=1; i<=n; i++) d[i]=oo;  qu.push(ii(0, u)); d[u]=0;  while (qu.size()) {  ii Pop=qu.top(); qu.pop();  int u=Pop.second, du=-Pop.first;  if (du!=d[u]) continue;  Sum+=d[u]; d[u]=0;  for (int i=0; int v=a[u][i]; i++)  if (d[v] > b[u][i]) {  d[v]=b[u][i];  qu.push(ii(-d[v], v));  }  }  return Sum;  }  main() {  scanf("%d%d", &n, &m);  for (int i=1; i<=m; i++) {  int x, y, z;  scanf("%d%d%d", &x, &y, &z);  a[x].push\_back(y);  b[x].push\_back(z);  a[y].push\_back(x);  b[y].push\_back(z);  }  for (int i=1; i<=n; i++)  a[i].push\_back(0);  cout << prim(1) << endl;  } |

## Cặp ghép cực đại không trọng số

|  |
| --- |
| LL m,n,x[104],y[103],ok[103],res;  VI ke[104];  bool fin(LL i){  if(ok[i]) return false;  ok[i]=true;  FT(j,0,ke[i].size()){  LL v=ke[i][j];  if(y[v]==0){  x[i]=v;  y[v]=i;  return true;  }  if(fin(y[v])){  x[i]=v;  y[v]=i;  return true;  }  }  return false;  }  int main(){  ios::sync\_with\_stdio(false);cin.tie(0);  cin>>m>>n;  LL u,v;  while(cin>>u){  cin>>v;  ke[u].PB(v);  }  FTB(i,1,m){  memset(ok,0,sizeof(ok));  if(fin(i)) res++;  }  cout<<res<<endl;  FTB(i,1,m){  if(x[i]) cout<<i<<" "<<x[i]<<endl;  }  return 0;  } |

## Dinitz

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 1234;  const int oo = 123456789;  int n, m, S, T;  int c[N][N], f[N][N], Dfs[N], d[N], t = 0;  vector<int> e[N];  bool bfs(int S, int T) {  memset(d, 0, sizeof d);  queue<int> qu;  qu.push(S); d[S] = 1;  while (qu.size()) {  int u = qu.front(); qu.pop();  if (u == T) return true;  for (int v : e[u]) {  if (!d[v] && f[u][v] < c[u][v]) {  d[v] = d[u] + 1;  qu.push(v);  }  }  }  return false;  }  int visit(int u, int Min) {  if (u == T) return Min;  if (Dfs[u] == t) return 0;  else Dfs[u] = t;  for (int v : e[u]) {  if (Dfs[v] != t && d[v] == d[u] + 1 && f[u][v] < c[u][v]) {  if (int x = visit(v, min(Min, c[u][v] - f[u][v]))) {  f[u][v] += x;  f[v][u] -= x;  return x;  }  }  }  return 0;  }  int main() {  cin >> n >> m >> S >> T;  for (int i = 1; i <= m; ++i) {  int x, y, p;  cin >> x >> y >> p;  e[x].push\_back(y);  e[y].push\_back(x);  c[x][y] += p;  }  int ans = 0;  while (bfs(S, T)) {  while (int x = (++t, visit(S, oo))) {  ans += x;  }  }  cout << ans;  return 0;  } |

## Hungari

|  |
| --- |
| // Mincost  // O(n^3)  // Maxcost: d[i] = oo, f[i] = oo, fx-=, fy+=, c[i][j] = 0, getC = fx + fy - c[u][v]  #include <bits/stdc++.h>  using namespace std;  const int MAXN = 1005;  const long long oo = 12345678987654321;  int n, m, k, N;  long long c[MAXN][MAXN], fx[MAXN], fy[MAXN];  int mx[MAXN], my[MAXN], trace[MAXN], d[MAXN], args[MAXN];  queue<int> qu;  void init() {  N = max(n, m);  for (int i = 1; i <= N; ++i) {  fx[i] = fy[i] = mx[i] = my[i] = 0;  for (int j = 1; j <= N; ++j) {  c[i][j] = oo;  }  }  }  void add\_edge(int u, int v, long long cost) {  c[u][v] = min(c[u][v], cost);  }  long long getC(int u, int v) {  return c[u][v] - fx[u] - fy[v];  }  void initBfs(int start) {  while (qu.size()) {  qu.pop();  }  qu.push(start);  for (int v = 1; v <= N; ++v) {  trace[v] = 0;  d[v] = getC(start, v);  args[v] = start;  }  }  int bfs(int start) {  initBfs(start);  while (qu.size()) {  int u = qu.front(); qu.pop();  for (int v = 1; v <= N; ++v) {  if (!trace[v]) {  long long w = getC(u, v);  if (!w) {  trace[v] = u;  if (!my[v]) {  return v;  }  else {  qu.push(my[v]);  }  }  if (d[v] > w) {  d[v] = w;  args[v] = u;  }  }  }  }  return 0;  }  void enlarge(int u) {  int x, y;  while (y = u) {  x = trace[y];  u = mx[x];  mx[x] = y;  my[y] = x;  }  }  int subX\_addY(int start) {  long long delta = oo;  for (int v = 1; v <= N; ++v) {  if (trace[v] == 0 && d[v] < delta) {  delta = d[v];  }  }  fx[start] += delta;  for (int v = 1; v <= N; ++v) {  if (trace[v]) {  int u = my[v];  fx[u] += delta;  fy[v] -= delta;  }  else {  d[v] -= delta;  }  }  for (int v = 1; v <= N; ++v) {  if (trace[v] == 0 && d[v] == 0) {  trace[v] = args[v];  if (my[v] == 0) {  return v;  }  qu.push(my[v]);  }  }  return 0;  }  long long hungarian() {  long long ans = 0;  for (int i = 1; i <= N; ++i) {  int u;  do {  u = bfs(i);  if (u == 0) {  u = subX\_addY(i);  }  } while (u == 0);  enlarge(u);  }  for (int i = 1; i <= N; ++i) {  if (c[i][mx[i]] < oo) {  ans += c[i][mx[i]];  }  }  return ans;  }  int main() {  cin >> n >> m >> k;  init();  while (k--) {  int u, v, w;  cin >> u >> v >> w;  add\_edge(u, v, w);  }  cout << hungarian() << "\n";  for (int i = 1; i <= N; ++i) {  if (c[i][mx[i]] < oo) {  cout << i << " " << mx[i] << "\n";  }  }  return 0;  } |

# Xử lý xâu

## Z algorithm

Z[i] là độ dài chuỗi con lớn nhất bắt đầu tại S[i] và là tiền tố của S.

|  |
| --- |
| int L = 0, R = 0;  Z[0] = n;  for (int i = 1; i < n; i++)  if (i > R) {  L = R = i;  while (R < n && S[R] == S[R - L]) R++;  Z[i] = R - L; R--;  }  else {  int k = i - L;  if (Z[k] < R - i + 1) Z[i] = Z[k];  else {  L = i;  while (R < n && S[R] == S[R - L]) R++;  Z[i] = R - L; R--;  }  } |

## Manacher

|  |
| --- |
| void manacher(const vector<int> &k) {  int p = k.size() \* 2 - 1;  vector<int> f(p, 0); vector<int> a(p, -8);  for (int i = 0; i < p; i += 2) a[i] = k[i / 2];  int l = 0, r = -1;  for (int i = 0, j = 0; i < p; i++) {  j = (i > r ? 0 : min(f[l + r - i], r - i)) + 1;  while (i - j >= 0 && i + j < p && a[i - j] == a[i + j]) j++;  f[i] = --j;  if (i + j > r) {  r = i + j;  l = i - j;  }  if (i & 1 ^ 1) res += f[i] / 2 + 1;  else res += (f[i] + 1) / 2;  }  } |

## KMP

|  |
| --- |
| void buildPi(string& p, vector<int>& pi) {  pi = vector<int> (p.length());  int k = -2;  for (int i = 0; i < p.length(); i++) {  while(k >= -1 && p[k+1] != p[i]) k = (k == -1) ? -2 : pi[k]; pi[i] = ++k;  }  }  int KMP(string& t, string& p) {  vector<int> pi;  buildPi(p, pi);  int k = -1;  for (int i = 0; i < t.length(); i++) {  while(k >= -1 && p[k+1] != t[i]) k = (k == -1) ? -2 : pi[k];  k++;  if (k == p.length() - 1) { // p matches t[i-m+1, ..., i]  cout << "matched at index " << i-k << ": ";  cout << t.substr(i-k, p.length()) << endl;  k = (k == -1) ? -2 : pi[k];  }  }  return 0;  } |

## Suffix array và Longest common prefix

sa[i]: vị trí của hậu tố có thứ tự từ điển i

lcp[i]: độ dài tiền tố chung dài nhất của hậu tố sa[i] và sa[i-1]

Số xâu con phân biệt: .

1 xâu độ dài q xuất hiện k lần trong xâu T nếu trong mảng lcp(T) tồn tại k-1 số liên tiếp ≥ q.

|  |
| --- |
| const int MAXN = 1e5;  int N, gap;  int sa[MAXN], pos[MAXN], tmp[MAXN], lcp[MAXN];  string S;  bool sufCmp(int i, int j) {  if (pos[i] != pos[j])  return pos[i] < pos[j];  i += gap;  j += gap;  return (i < N && j < N) ? pos[i] < pos[j] : i > j;  }  void buildSA() {  N = S.length();  for (int i = 0; i < N; i++) sa[i] = i, pos[i] = S[i];  for (gap = 1;; gap \*= 2) {  sort(sa, sa + N, sufCmp);  for (int i = 0; i < N - 1; i++) tmp[i + 1] = tmp[i] + sufCmp(sa[i], sa[i + 1]);  for (int i = 0; i < N; i++) pos[sa[i]] = tmp[i];  if (tmp[N - 1] == N - 1) break;  }  }  void buildLCP() {  for (int i = 0, k = 0; i < N; ++i) if (pos[i] != N - 1) {  for (int j = sa[pos[i] + 1]; S[i + k] == S[j + k];)  ++k;  lcp[pos[i]] = k;  if (k)--k;  }  } |

# Khác

## Centroid decomposition

Codeforces 321C:

Cho 1 cây, có thể gán bậc từ A-Z cho 1 đỉnh (bậc A > B). Nếu 2 đỉnh được gán cùng bậc thì trên đường đi ngắn nhất giữa 2 đỉnh tồn tại 1 đỉnh có bậc lớn hơn bậc của 2 đỉnh đó. Tìm một cách gán thoả mãn.

|  |
| --- |
| void centroid(ll i,ll p,ll s) {  for (auto it:ke[i]) {  if (con[it]>(s/2) && par[it]<0) {  con[i]=s-con[it];  centroid(it,p,s);  return;  }  }  par[i]=p;  for (auto it:ke[i]) {  if (par[it]<0) centroid(it,i,con[it]);  }  } |

## Heavy-light decomposition

|  |
| --- |
| // add u v: Cộng trọng số các đỉnh thuộc cây con gốc u một giá trị v  // query u v: Giá trị trọng số nhỏ nhất trên đường đi từ u đến v  #include <bits/stdc++.h>  using namespace std;  const int nn = 1e5 + 8;  const int Max = 2e9;  int cRoot[nn], cId[nn];  int t[nn << 2], f[nn << 2], sz[nn], d[nn];  int pre[nn];  int L[nn], R[nn];  int cnum = 1, n;  vector<int> G[nn];  char qr[5];  bool cmp(int u, int v) {  return sz[u] > sz[v];  }  void dfs(int u, int pa) {  sz[u] = 1;  for (int v: G[u]) if (v != pa) {  d[v] = d[u] + 1;  pre[v] = u;  dfs(v, u);  sz[u] += sz[v];  }  }  void hld(int u, int pa, int &m) {  if (!cRoot[cnum]) cRoot[cnum] = u;  cId[u] = cnum;  L[u] = m;  int first = 0;  for (int v: G[u])  {  if (v == pa) continue;  cnum += first;  first |= 1;  hld(v, u, ++m);  }  R[u] = m;  }  int lca(int u, int v) {  while (cId[u] != cId[v]) {  if (d[cRoot[cId[u]]] > d[cRoot[cId[v]]]) u = pre[cRoot[cId[u]]];  else v = pre[cRoot[cId[v]]];  }  if (d[u] < d[v]) return u;  return v;  }  void lazy(int k, int l, int r) {  if (f[k] != 0) {  t[k] += f[k];  if (l < r) {  f[k << 1] += f[k];  f[k << 1 ^ 1] += f[k];  }  f[k] = 0;  }  }  void update(int k, int l, int r, int x, int y, int v) {  lazy(k, l, r);  if (l > y or r < x) return;  if (l >= x and r <= y) {  t[k] += v;  if (l < r) {  f[k << 1] += v;  f[k << 1 ^ 1] += v;  }  return;  }  int m = (l + r) >> 1;  update(k << 1, l, m, x, y, v);  update(k << 1 ^ 1, m + 1, r, x, y, v);  t[k] = max(t[k << 1], t[k << 1 ^ 1]);  }  int get(int k, int l, int r, int x, int y) {  lazy(k, l, r);  if (l > y or r < x) return -Max;  if (l >= x and r <= y) return t[k];  int m = (l + r) >> 1;  return max(get(k << 1, l, m, x, y), get(k << 1 ^ 1, m + 1, r, x, y));  }  int maxPath(int v, int u) {  if (d[v] < d[u]) return 0;  int res = -Max;  while (cId[v] != cId[u]) {  res = max(res, get(1, 1, n, L[cRoot[cId[v]]], L[v]));  v = pre[cRoot[cId[v]]];  }  int tmp = get(1, 1, n, L[u], L[v]);  return max(res, get(1, 1, n, L[u], L[v]));  }  int query2(int u, int v) {  int w = lca(u, v);  return max(maxPath(u, w), maxPath(v, w));  }  int main() {  scanf("%d", &n);  for (int i = 1; i < n; i++) {  int u, v;  scanf("%d %d", &u, &v);  G[u].push\_back(v);  G[v].push\_back(u);  }  dfs(1, 0);  for (int i = 1; i <= n; i++) {  sort(G[i].begin(), G[i].end(), cmp);  }  int m = 1;  hld(1, 0, m);  int q;  scanf("%d", &q);  while (q--) {  int u, v;  scanf("%s %d %d", qr, &u, &v);  if (qr[0] == 'a') {  update(1, 1, n, L[u], R[u], v);  }  else {  printf("%d\n", query2(u, v));  }  }  } |

## Treap

Cho dãy số A1, A2, …, An. Có 4 truy vấn:

1 i j x (i ≤ j): gán các phần từ thứ i đến thứ j trong dãy bằng x;

2 i j x (i ≤ j): cộng phần tử thứ i cho x, thứ i+1 cho 2x,... , thứ j cho (j-i+1)x;

3 i x: chèn x vào trước phần tử thứ i của dãy hiện thời;

4 i j: (i ≤ j) Tính tổng từ phần tử thứ i đến phần tử thứ j.

|  |
| --- |
| ll n,l,r,sum;    struct node{  ll val,cnt=1,prio;  node \*lef=NULL,\*rig=NULL;  // cnt la so con cua cay  node(ll val): val(val),prio(rand()){};  };    ll get(node \*a){  return a?a->cnt:0;  }    node\* rotate\_righ(node \*root){  node \*a=root->lef,\*b=a->rig;  a->rig=root;  root->lef=b;  ll tmp=root->cnt;  root->cnt=tmp-a->cnt+get(b);  a->cnt=tmp;  return a;  }    node\* rotate\_left(node \*root){  node \*a=root->rig,\*b=a->lef;  a->lef=root;  root->rig=b;  ll tmp=root->cnt;  root->cnt=tmp-a->cnt+get(b);  a->cnt=tmp;  return a;  }    node\* ins(node \*root,ll val){  if(!root) return new node(val);  root->cnt++;  if(val>root->val){  root->rig=ins(root->rig,val);  if(root->prio<root->rig->prio) root=rotate\_left(root);  }  else if(val<root->val){  root->lef=ins(root->lef,val);  if(root->prio<root->lef->prio) root=rotate\_righ(root);  }  return root;  }    ll get\_less(node \*a,ll val){  if(!a) return 0;  if(val==a->val) return get(a->lef);  if(val<a->val) return get\_less(a->lef,val);  return a->cnt-get(a->rig)+get\_less(a->rig,val);  }      void Solve(){  srand(time(NULL));  cin>>n>>l>>r;  ll res=0;  node \*root=NULL;  root=ins(root,0);  while(n--){  ll x;cin>>x;  sum+=x;  res+=get\_less(root,sum-l+1)-get\_less(root,sum-r);  root=ins(root,sum);  }  cout<<res;  } |

# . Trình chấm

## Comp.sh

|  |
| --- |
| g++ -static -O2 -std=c++11 dung.cpp -o dung.out  g++ -static -O2 -std=c++11 sai.cpp -o sai.out  g++ -static -O2 -std=c++11 main.cpp -o main.out  ./main.out |

## Main.cpp

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  // ham rand() co may cham la max 2e9, co may max la 32k, check may  // truoc khi dung ham Rand(1,n), neu la 2e9 thi khong can nhan nhieu // RAND\_MAX, chi can nhan 2 cai de kieu unsigned  long long Rand(long long l, long long h)  {  return l + ((long long)rand() \* (RAND\_MAX + 1) \* (RAND\_MAX + 1) \* (RAND\_MAX + 1) +  (long long)rand() \* (RAND\_MAX + 1) \* (RAND\_MAX + 1) +  (long long)rand() \* (RAND\_MAX + 1) +  rand()) % (h - l + 1);  }  void sinhtest()  {  srand(time(NULL));  ofstream inp("inp.txt");  inp << Rand(1,1000);  inp.close();  }  int main()  {    for (int i = 1; i <= 100; i++)  {  cout << "chay test" << i << endl;  system("./dung.out");  system("./sai.out");  if(system("diff dung.txt sai.txt") != 0){  cout<<"WA\n";  break;  }  }  } |

## Soldung.cpp

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int main()  {  freopen("inp.txt", "r", stdin);  freopen("dung.txt", "w", stdout);  int n;  cin >> n;  cout << n;  return 0;  } |

## Solsai.cpp

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int main()  {  freopen("inp.txt", "r", stdin);  freopen("sai.txt", "w", stdout);  int n;  cin >> n;  cout << n+1;  return 0;  } |

## Lệnh biên dịch

|  |
| --- |
| ./Comp.sh |