Mục lục

[**1.Toán** 2](#_Toc499087878)

[Phi hàm Euler 2](#_Toc499087879)

[Modulo trick 2](#_Toc499087880)

[Lehman 2](#_Toc499087881)

[Miller Rabin 3](#_Toc499087882)

[Extended Euclid: Tìm x, y sao cho ax + by = gcd(a, b) 4](#_Toc499087883)

[Đếm số các số ≤ n có k bit 1 5](#_Toc499087884)

[**2. Đồ thị** 5](#_Toc499087885)

[Tarjan: Tìm thành phần liên thông mạnh 5](#_Toc499087886)

[Sắp xếp topo 6](#_Toc499087887)

[Tìm chu trình Euler 6](#_Toc499087888)

[Floyd: Đường đi ngắn nhất giữa mọi cặp đỉnh 8](#_Toc499087889)

[Prim: Tìm cây khung nhỏ nhất 8](#_Toc499087890)

[Cặp ghép cực đại trên đồ thị 2 phía 9](#_Toc499087891)

[Phát hiện chu trình trong đồ thị có hướng 9](#_Toc499087892)

[Khớp & cầu 9](#_Toc499087893)

[Bellman Ford và xác định chu trình âm 10](#_Toc499087894)

[Dinitz: luồng cực đại trên đồ thị 10](#_Toc499087895)

[Edmonds – Karp: Lát cắt hẹp nhất trong mạng 11](#_Toc499087896)

[**3. Hình học** 12](#_Toc499087897)

[So sánh 2 số thực 12](#_Toc499087898)

[Hình học cơ bản 12](#_Toc499087899)

[Monotone chain 15](#_Toc499087900)

[Một số công thức trong tam giác 16](#_Toc499087901)

[Bao lồi Graham 16](#_Toc499087902)

[Emo Welzl – Đường tròn nhỏ nhất chứa mọi điểm cho trước 16](#_Toc499087903)

[**4. Xử lý xâu** 17](#_Toc499087904)

[Z algorithm 17](#_Toc499087905)

[Manacher: Xâu palindrome dài nhất 18](#_Toc499087906)

[KMP: So khớp chuỗi 18](#_Toc499087907)

[Suffix array và Longest common prefix 18](#_Toc499087908)

[**5. Khác** 19](#_Toc499087909)

[QTREE 19](#_Toc499087910)

[Persitent Segment Tree 21](#_Toc499087911)

[Diện tích n hình chữ nhật 22](#_Toc499087912)

[SCPC3-2017 23](#_Toc499087913)

[F- ACM Vietnam National 2017 25](#_Toc499087914)

# 1.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 result = n;**  **for(int i = 2; i \* i <= n; ++i)**  **if(n % i == 0) {**  **while(n % i == 0) n /= i;**  **result -= result / i;**  **}**  **if(n > 1)**  **result -= result / n;**  **return result;**  **}** |

## 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 n>1n>1 là số nguyên tố khi và chỉ khi  (n−1)!≡n−1 (mod n). |

## 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 số 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;  int t;  cin >> t;  while (t--) {  cin >> n >> k;  cout << f(n, k) << endl;  }  } |

# 2. Đồ thị

## Tarjan: Tìm thành phần liên thông mạnh

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 100005;  const int inf = 1e9;  int n, m, Num[N], Low[N], cnt=0;  vector<int> a[N];  stack<int> st;  int Count;  void visit(int u) {  Low[u]=Num[u]=++cnt;  st.push(u);  for (int i=0; int v=a[u][i]; i++)  if (Num[v])  Low[u]=min(Low[u], Num[v]);  else {  visit(v); Low[u]=min(Low[u], Low[v]);  }  if (Num[u]==Low[u]) { // found one  Count++;  int v;  do {  v=st.top(); st.pop();  Num[v]=Low[v]=inf; // remove v from graph  } while (v!=u);  }  }  int main(){  scanf("%d%d", &n, &m);  for (int i=1; i<=m; i++) {  int x, y; scanf("%d%d", &x, &y); a[x].push\_back(y);  }  for (int i=1; i<=n; i++) a[i].push\_back(0);  for (int i=1; i<=n; i++) if (!Num[i]) visit(i); cout << Count << endl;  } |

## Sắp xếp topo

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 1e5 + 8;  vector<int> G[N];  bool trv[N], done[N], DAG = true;  int topo[N], n, m, cnt;  void dfs(int u) {  if (trv[u]) {  DAG = false;  return;  }  if (done[u]) return;  trv[u] = true;  for (int v: G[u]) dfs(v);  trv[u] = false;  done[u] = true;  topo[cnt--] = u;  }  void toposort() {  // nếu có nhiều cách sắp xếp, in ra cách có số đầu tiên nhỏ nhất,  // nếu có nhiều cách như vậy, in ra cách có số thứ 2 nhỏ nhất,...  for (int i = 1; i <= n; i++) {  sort(G[i].begin(), G[i].end());  reverse(G[i].begin(), G[i].end());  }  cnt = n;  for (int i = n; i >= 1; i--) if (!done[i]) dfs(i);  }  int main()  {  cin >> n >> m;  for (int i = 1; i <= m; i++) {int u, v; scanf("%d %d", &u, &v); G[u].push\_back(v);}  toposort();  if (DAG) for (int i = 1; i <= n; i++) printf("%d ", topo[i]);  else cout << "NOT DAG";  } |

## Tìm 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  \*/ |

## Floyd: Đường đi ngắn nhất giữa mọi cặp đỉnh

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int inf = 1e9;  int a[300][300];  int n, m;  void minimize(int &a, int b){ if (a>b) a=b; }  main(){  int i,j,k, p,q,w;  scanf("%d%d", &n, &m);  for (i=1; i<=n; i++)  for (j=1; j<=n; j++)  a[i][j] = inf;  for (i=1; i<=n; i++) a[i][i] = 0;  for (i=1; i<=m; i++) {  scanf("%d%d%d", &p, &q, &w);  a[p][q] = a[q][p] = w;  }  for (k=1; k<=n; k++)  for (i=1; i<=n; i++)  for (j=1; j<=n; j++)  minimize(a[i][j], a[i][k] + a[k][j]);  } |

## Prim: Tìm cây khung nhỏ nhất

|  |
| --- |
| #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 trên đồ thị 2 phía

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 102;  int n, m, Assigned[N];  int Visited[N], t=0;  vector<int> a[N];  bool visit(int u) {  if (Visited[u]!=t)  Visited[u]=t;  else  return false;  for (int i=0; int v=a[u][i]; i++)  if (!Assigned[v] || visit(Assigned[v])) {  Assigned[v]=u;  return true;  }  return false;  }  main() {  scanf("%d%d", &m, &n);  int x, y;  while (scanf("%d%d", &x, &y) > 0)  a[x].push\_back(y);  for (int i=1; i<=m; i++)  a[i].push\_back(0);  int Count = 0;  for (int i=1; i<=m; i++) {  t++;  Count += visit(i);  }  printf("%d\n", Count);  for (int i=1; i<=n; i++)  if (int j=Assigned[i])  printf("%d %d\n", j, i);  } |

## 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;  } |

## Khớp & cầu

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 100005;  int n, m;  vector<int> a[N];  int CriticalEdge=0;  bool CriticalNode[N];  int Num[N], Low[N], Time=0;  void visit(int u, int p) {  int NumChild = 0;  Low[u] = Num[u] = ++Time;  for (int i=0; int v=a[u][i]; i++)  if (v!=p) {  if (Num[v]!=0)  Low[u] = min(Low[u], Num[v]);  else {  visit(v, u);  NumChild++;  Low[u] = min(Low[u], Low[v]);  if (Low[v] >= Num[v])  CriticalEdge++;  if (u==p) {  if (NumChild >= 2)  CriticalNode[u] = true;  } else {  if (Low[v] >= Num[u])  CriticalNode[u] = true;  }  }  }  }  main() {  scanf("%d%d", &n, &m);  for (int i=1; i<=m; i++) {  int x, y;  scanf("%d%d", &x, &y);  a[x].push\_back(y);  a[y].push\_back(x);  }  for (int i=1; i<=n; i++)  a[i].push\_back(0);  for (int i=1; i<=n; i++)  if (!Num[i]) visit(i, i);  int Count = 0;  for (int i=1; i<=n; i++)  if (CriticalNode[i]) Count++;  printf("%d %d\n", Count, CriticalEdge);  } |

## Bellman Ford và xác định chu trình âm

|  |
| --- |
| //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 |

## Dinitz: luồng cực đại trên đồ thị

Độ phức tạp: O(n2m)

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 1003, oo = 0x3c3c3c3c;  int n, m, S, T;  int d[N], c[N][N], f[N][N];  int Dfs[N], t=0;  vector<int> a[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: a[u])  if (!d[v] && f[u][v]<c[u][v])  { qu.push(v); d[v]=d[u]+1; }  }  return false;  }  int visit(int u, int Min) {  if (u==T) return Min;  if (Dfs[u]!=t) Dfs[u]=t;  else return 0;    for (int v: a[u])  if (f[u][v]<c[u][v])  if (Dfs[v]!=t && d[v]==d[u]+1)  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, z; scanf("%d%d%d", &x, &y, &z);  a[x].push\_back(y);  a[y].push\_back(x);  c[x][y] += z;  }  int Sum = 0;  while (bfs(S, T)) {  while (int x = (t++, visit(S, oo))) {  Sum += x;  //printf("Sum=%d\n", Sum);  }  }  cout << Sum << endl;  } |

## Edmonds – Karp: Lát cắt hẹp nhất trong mạng

Độ phức tạp: O(nm2)

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  void minimize(int &a, int b){  if (a>b) a=b;  }  int n, m;  vector<int> a[12309];  int start, target;  int c[123][123];  int f[123][123];  int d[12309];  bool bfs(int start, int target){  queue<int> qu;  int u, i, v;  for (i=1; i<=n; i++) d[i]=0;  d[start] = -1;  qu.push(start);  while (qu.size()){  u=qu.front(); qu.pop();  if (u==target) return true;  for (i=0; v=a[u][i]; i++)  if (d[v]==0 && f[u][v]<c[u][v]){  d[v]=u;  qu.push(v);  }  }  return false;  }  int mincut(bool tracing=false){  int u, i, v, r=0;  for (u=1; u<=n; u++)  for (i=0; v=a[u][i]; i++)  if (d[u] && !d[v]) {  r += c[u][v];  if (tracing) printf("%d %d\n", u, v);  }  return r;  }  void enlarge(){  int i;  int delta=1000111000;  for (i=target; i!=start; i=d[i])  minimize(delta, c[d[i]][i]-f[d[i]][i]);  for (i=target; i!=start; i=d[i]){  f[d[i]][i] += delta;  f[i][d[i]] -= delta;  }  }  main(){  int i, p, q, w;  for (;;){  scanf("%d%d", &n, &m);  if (n==0) return 0;  for (i=1; i<=n; i++) a[i].clear();  for (p=1; p<=n; p++)  for (q=1; q<=n; q++)  c[p][q]=f[p][q]=0;  start=1, target=2;  for (i=1; i<=m; i++){  scanf("%d%d%d", &p, &q, &w);  a[p].push\_back(q);  a[q].push\_back(p);  c[p][q]=c[q][p]=w;  }  for (i=1; i<=n; i++) a[i].push\_back(0);  while (bfs(start, target)) enlarge();  mincut(true); printf("\n");  }  } |

# 3. Hình học

## **So sánh 2 số thực**

|  |
| --- |
| const double eps = 1e-8;  int cmp(double A, double B) {  if (A - B < -eps) return -1;// A < B  if (A - B > eps) return 1;// A > B  return 0;// A = B  } |

## **Hình học cơ bản**

|  |
| --- |
| **struct point{**  **double x;**  **double y;**  **point(double \_x = 0, double \_y = 0) {**  **x = \_x; y = \_y;**  **}**  **bool operator == (const point& that) const{**  **return (cmp(x, that.x) == 0 && cmp(y, that.y) == 0);**  **}**  **bool operator < (const point& that) const {**  **if(cmp(x, that.x) != 0) return cmp(x, that.x) < 0;**  **return cmp(y, that.y) < 0;**  **}**  **};**  // 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);  }  **// Hai đoạn thẳng thực sự cắt nhau**  **int isRealCut(point p0, point p1, point p2, point p3){**  **return ccw(p0,p1,p2)\*ccw(p0,p1,p3)<0 &&**  **ccw(p2,p3,p0)\*ccw(p2,p3,p1)<0;**  **}**  **// Vị trí của p0 so với đoạn thẳng p1p2**  **int segmentPos(point p0, point p1, point p2){**  **if(p1 == p2) return -2;**  **else if(ccw(p0,p1,p2) != 0) return -1; // không thẳng hàng**  **else if (p2 < p1) swap(p1,p2);**  **if(p0 < p1) return 1; // nằm ngoài gần phía p1**  **else if(p2 < p0) return 2; // nằm ngoài gần phía p2**  **return 0; // nằm trong đoạn**  **}**  **// 2 đoạn thẳng cắt nhau**  **int isSegmentCut(point p0, point p1, point p2, point p3){**  **if(isRealCut(p0,p1,p2,p3)) return 1;**  **if(segmentPos(p0,p2,p3) || segmentPos(p1,p2,p3)) return 1;**  **if(segmentPos(p2,p0,p1) || segmentPos(p3,p0,p1)) return 1;**  **return 0;**  **}**  **// Phương trình đường thẳng**  **int getLine(point p0, point p1, double &a, double &b, double &c){**  **if(p0 == p1) return 0;**  **a = p1.y - p0.y;**  **b = p0.x - p1.x;**  **c = -(a\*p0.x + b\*p0.y);**  **return 1;**  **}**  **// Khoảng cách giữa hai điểm**  **double dist(point p0, point p1){**  **double dx = p1.x - p0.x, dy = p1.y - p0.y;**  **return sqrt(dx \* dx + dy \* dy);**  **}**  **// Giao điểm của hai đường thẳng**  **int getIntersection(point p0, point p1, point p2, point p3, point &p4){**  **double a0,b0,c0,a1,b1,c1;**  **getLine(p0,p1,a0,b0,c0);**  **getLine(p2,p3,a1,b1,c1);**  **double d = a0\*b1 - a1\*b0, dx = b0\*c1 - b1\*c0, dy = -a0\*b1 + a1\*c0;**  **if(cmp(d,0) == 0){**  **if(cmp(dx,0) == 0 && cmp(dy,0) == 0) return -1; // trùng nhau**  **else return 0; // song song**  **}**  **p4.x = dx/d; p4.y = dy/d;**  **return 1;**  **}**  **// Diện tích đa giác**  **double areaPolygon(point P[], int n){**  **P[n+1] = P[1];**  **double ans = 0;**  **for (int i = 1; i <= n; i++) ans += P[i].x\*P[i+1].y - P[i+1].x\*P[i].y;**  **return fabs(ans/2);**  **}**  **// Kiểm tra một điểm nằm trong đa giác o(n)**  **int insidePolygon(point P[], int n, point p0){**  **P[n+1] = P[1];**  **/\* truong hop da giac khong phai da giac loi**  **for (int i = 1; i <= n; i++)**  **if(segmentPos(p0, P[i], P[i+1]) == 0) return 1;**  **int dem = 0;**  **point Z; Z.x = 1000000007; Z.y = 1000000008;**  **while(1){**  **int ok = 1;**  **for (int i = 1; i <= n; i++) if(ccw(Z,P[i],P[i+1]) == 0) {**  **ok = 0;**  **Z.y++;**  **}**  **if(ok == 1) break;**  **}**  **for (int i = 1; i <= n; i++) if(isRealCut(p0,Z,P[i],P[i+1])) dem++;**  **return (dem%2);**  **\*/**  **int x1 = 0, x2 = 0;**  **for (int i = 1; i <= n; i++){**  **if(ccw(P[i], P[i + 1], p0) == 0) return 0;**  **else if(ccw(P[i], P[i + 1], p0) == -1) x1++;**  **else x2++;**  **}**  **return (!x1 || !x2);**  **}**  **// Kiểm tra 2 đa giác có điểm trung**  **int intersectionPolygon(point P[], int n1, point Q[], int n2){**  **P[n1+1] = P[1]; Q[n2+1] = Q[1];**  **// check a point in a polygon**  **for (int i = 1; i <= n1; i++) if(insidePolygon(Q,n2,P[i])) return 1;**  **for (int i = 1; i <= n2; i++) if(insidePolygon(P,n1,Q[i])) return 1;**  **// check line intersect**  **for (int i = 1; i <= n1; i++)**  **for (int j = 1; j <= n2; j++)**  **if(isRealCut(P[i],P[i+1],Q[j],Q[j+1]) == 1) return 1;**  **// in case same point**  **for (int i = 1; i <= n1; i++)**  **for (int j = 1; j <= n2; j++)**  **if(isSegmentCut(P[i],P[i+1],Q[j],Q[j+1])) return 1;**  **return 0;**  **}**  **// Diện tích tam giác**  **double areaTriangle(point A, point B, point C){**  **double ans = abs(A.x\*(B.y-C.y) + B.x\*(C.y-A.y) + C.x\*(A.y-B.y));**  **return ans/2;**  **}**  **// Tính góc BAC theo radian**  **double getAngle(point A, point B, point C){**  **double a = dist(B,C);**  **double b = dist(C,A);**  **double c = dist(A,B);**  **double agoc = (b\*b + c\*c - a\*a) / (2\*b\*c);**  **return acos(agoc);**  **}**  **// Kiểm tra điểm p0 nằm trong tam giác ABC**  **int insideTriangle(point p0, point A, point B, point C){**  **double S1 = areaTriangle(p0,A,B);**  **double S2 = areaTriangle(p0,B,C);**  **double S3 = areaTriangle(p0,C,A);**  **double Sum = areaTriangle(A,B,C);**  **if(cmp(Sum, S1+S2+S3) == 0) return 1;**  **return 0;**  **}**  **// Kiểm tra điểm p0 nằm trong góc tạo bởi tia AB, AC**  **int insideAngle(point p0, point A, point B, point C){**  **if(p0 == A) return 1;**  **if(ccw(p0,A,B) \* ccw(p0,A,C) > 0) return 0;**  **//return (getAngle(A,p0,B) + getAngle(A,p0,C) < PI+eps);**  **if(cmp(getAngle(A,B,C), getAngle(A,p0,B) + getAngle(A,p0,C)) == 0) return 1;**  **return 0;**  **}** |

## **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]);  } |

## Một số 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: |

## Bao lồi Graham

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef pair<int, int> ii;  #define X first  #define Y second  ii origin;  void operator -= (ii &A, ii B){ A.X-=B.X; A.Y-=B.Y; }  bool ccw(ii O, ii A, ii B){ A-=O, B-=O; return A.X\*B.Y > A.Y\*B.X; }  bool cmp(ii A, ii B){ return ccw(origin, A, B); }  int n;  ii a[12309];  int main(){  int i, t;  scanf("%d", &n);  for (i=1; i<=n; i++)  scanf("%d%d", &a[i].X, &a[i].Y);    sort(a+1, a+n+1);  origin = a[1];  sort(a+2, a+n+1, cmp);  a[0]=a[n]; a[n+1]=a[1];  int j=1;  for (i=1; i<=n+1; i++){ // a[1] and a[n+1] will be both added  while (j>2 && !ccw(a[j-2], a[j-1], a[i])) j--;  a[j++]=a[i];  }  n=j-2;  for (i=1; i<=n; i++) printf("%d %d\n", a[i].X, a[i].Y);  } |

## 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;  } |

# 4. 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: Xâu palindrome dài nhất

|  |
| --- |
| const char DUMMY = '\*';  int manacher(string s) {  int n = s.size() \* 2 - 1;  vector <int> f = vector <int>(n, 0);  string a = string(n, DUMMY);  for (int i = 0; i < n; i += 2) a[i] = s[i / 2];  int l = 0, r = -1, center, res = 0;  for (int i = 0, j = 0; i < n; i++) {  j = (i > r ? 0 : min(f[l + r - i], r - i)) + 1;  while (i - j >= 0 && i + j < n && a[i - j] == a[i + j]) j++;  f[i] = --j;  if (i + j > r) {  r = i + j;  l = i - j;  }  int len = (f[i] + i % 2) / 2 \* 2 + 1 - i % 2;  if (len > res) {  res = len;  center = i;  }  }  return res;  } |

## KMP: So khớp chuỗi

|  |
| --- |
| 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] là vị trí của hậu tố có thứ tự từ điển i

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

|  |
| --- |
| 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;  }  }  // Số xâu con phân biệt:  // Điều kiện để 1 xâu độ dài q xuất hiện k lần trong xâu T là trong mảng lcp(T) tồn tại k-1 số liên tiếp ≥ q |

# 5. Khác

## QTREE

|  |
| --- |
| //1 cây n đỉnh  //CHANGE u v: đổi trọng số cạnh thứ u thành v  //QUERY u v: tìm trọng số lớn nhất của các cạnh trên đường đi từ u đến v  #include <bits/stdc++.h>  using namespace std;  const int N = 1e4 + 8;  int pos[N], d[N], p[N], t[N << 2], cn[N], cr[N], pre[N], Next[N], cid, tid, n, eu[N], ev[N], cost[N];  char s[8];  struct data {  int v, w;  };  vector<data> a[N];  void clear\_data() {  for (int i = 1; i <= n; i++) a[i].clear();  for (int i = 1; i <= cid; i++) cr[i] = 0;  memset(t, 0, sizeof(t));  cid = 1;  tid = -1;  p[1] = 1;  }  void dfs(int u, int pre\_) {  d[u] = 1;  for (int i = 0; i < a[u].size(); i++) {  int v = a[u][i].v;  if (v != pre\_) {  pre[v] = u;  p[v] = p[u] + 1;  dfs(v, u);  }  d[u] += d[v];  }  }  void hld(int u) {  if (cr[cid] == 0) cr[cid] = u;  cn[u] = cid;  pos[u] = ++tid;  int id = 0, Max = 0;  for (int i = 0; i < a[u].size(); i++) {  if (a[u][i].v != pre[u] and d[a[u][i].v] > Max) {  Max = d[a[u][i].v];  id = a[u][i].v;  }  }  if (id > 0) {  Next[u] = id;  hld(id);  }  for (int i = 0; i < a[u].size(); i++) {  if (a[u][i].v == pre[u] or a[u][i].v == id) continue;  cid++;  hld(a[u][i].v);  }  }  int lca(int u, int v) {  while (cn[u] != cn[v]) {  if (p[cr[cn[u]]] > p[cr[cn[v]]]) u = pre[cr[cn[u]]];  else v = pre[cr[cn[v]]];  }  if (p[u] < p[v]) return u;  return v;  }  void update(int k, int l, int r, int x, int v) {  if (l == x and r == x) {  t[k] = v;  return;  }  if (l > x or r < x) return;  int m = (l + r) >> 1;  update(k << 1, l, m, x, v);  update((k << 1) + 1, m + 1, r, x, v);  t[k] = max(t[k << 1], t[(k << 1) + 1]);  }  int get(int k, int l, int r, int x, int y) {  if (l > y or r < x) return 0;  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 getpoint(int x) {  if (p[eu[x]] > p[ev[x]]) return eu[x];  return ev[x];  }  void query1(int x, int v) {  int id = getpoint(x - 1);  update(1, 0, n - 1, pos[id], v);  }  int calc(int u, int w) {  if (p[u] < p[w]) return 0;  int res = 0;  while (cn[u] != cn[w]) {  res = max(res, get(1, 0, n - 1, pos[cr[cn[u]]], pos[u]));  u = pre[cr[cn[u]]];  }  res = max(res, get(1, 0, n - 1, pos[w], pos[u]));  return max(res, get(1, 0, n - 1, pos[w], pos[u]));  }  int query2(int u, int v) {  if (u == v) return 0;  int w = lca(u, v);  w = Next[w];  return max(calc(u, w), calc(v, w));  }  int main() {  int test;  scanf("%d", &test);  while (test--) {  clear\_data();  scanf("\n%d", &n);  for (int i = 1; i < n; i++) {  int u, v, w;  scanf("%d %d %d", &u, &v, &w);  data tmp;  tmp.v = v;  tmp.w = w;  a[u].push\_back(tmp);  tmp.v = u;  a[v].push\_back(tmp);  eu[i - 1] = u;  ev[i - 1] = v;  cost[i - 1] = w;  }  dfs(1, 0);  hld(1);  for (int i = 0; i < n - 1; i++) {  int id = getpoint(i);  update(1, 0, n - 1, pos[id], cost[i]);  }  while (scanf("%s", &s)) {  if (s[0] == 'D') break;  if (s[0] == 'C') {  int x, v;  scanf("%d %d\n", &x, &v);  query1(x, v);  }  else {  int u, v;  scanf("%d %d\n", &u, &v);  printf("%d\n", query2(u, v));  }  }  }  } |

## Persitent Segment Tree

|  |
| --- |
| struct node {  int sum;  node \*lc, \*rc;  node() {  sum = 0;  lc = NULL;  rc = NULL;  }  };  node\* init(int l, int r) {  node \*t = new node();  if (l == r) return t;  int m = (l + r) >> 1;  t->lc = init(l, m);  t->rc = init(m + 1, r);  return t;  }  node\* update(node\* k, int l, int r, int x) {  if (l > x or r < x) return k;  node\* t = new node();  if (l == x and r == x) {  t->sum = 1;  return t;  }  int m = (l + r) >> 1;  t->lc = update(k->lc, l, m, x);  t->rc = update(k->rc, m + 1, r, x);  t->sum = t->lc->sum + t->rc->sum;  return t;  }  int get(node\* k, int l, int r, int x, int y) {  if (l > y or r < x) return 0;  if (l >= x and r <= y) return k->sum;  int m = (l + r) >> 1;  return get(k->lc, l, m, x, y) + get(k->rc, m + 1, r, x, y);  } |

## Diện tích n hình chữ nhật

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 1e5 + 8;  const int M = 3e5 + 8;  struct data {  int x, l, r, t;  };  vector<data> a;  int t[M<<2], cnt[M<<2], f[M<<2];  bool cmp(const data &A, const data &B) {  return A.x < B.x;  }  void init(int k, int l, int r) {  cnt[k] = r - l + 1;  if (l == r) return;  int m = (l + r) >> 1;  init(k << 1, l, m);  init(k << 1 ^ 1, m + 1, r);  }  void update(int k, int l, int r, int x, int y, int v) {  int k1 = k<<1;  int k2 = k<<1^1;  if (f[k] != 0) {  t[k] += f[k];  if (l != r) {  f[k1] += f[k];  f[k2] += f[k];  }  f[k] = 0;  }  if (l > y or r < x) return;  if (l >= x and r <= y) {  t[k] += v;  if (l != r) {  f[k1] += v;  f[k2] += v;  }  return;  }  int m = (l + r) >> 1;  update(k1, l, m, x, y, v);  update(k2, m + 1, r, x, y, v);  t[k] = min(t[k1], t[k2]);  if (t[k1] < t[k2]) cnt[k] = cnt[k1];  else if (t[k1] > t[k2]) cnt[k] = cnt[k2];  else cnt[k] = cnt[k1] + cnt[k2];  }  void get(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;  }  }  int main() {  int n; scanf("%d", &n);  for (int i = 1; i <= n; i++) {  int x1, y1, x2, y2;  scanf("%d %d %d %d", &x1, &y1, &x2, &y2);  data tmp;  tmp.l = y1;  tmp.r = y2;  tmp.x = x1;  tmp.t = 1;  a.push\_back(tmp);  tmp.x = x2;  tmp.t = -1;  a.push\_back(tmp);  }  sort(a.begin(), a.end(), cmp);  int px = 0, res = 0;  init(1, 0, M - 1);  for (int i = 0; i < n\*2; i++) {  int py = M;  get(1, 0, M - 1);  if (t[1] == 0) py -= cnt[1];  res += (a[i].x - px) \* py;  data tmp = a[i];  update(1, 0, M - 1, a[i].l, a[i].r - 1, a[i].t);  px = a[i].x;  }  cout << res;  } |

## SCPC3-2017

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  const int N = 200;  const int M = N \* N;  int n, m;  vector<pair<int, int> > row[M], col[M];  int a[N][N], r[N][N], c[N][N], rowTrv[M], colTrv[M];  vector<int> rowList, colList;  bool dfs(int odd, int u, int type) {  if (!odd) {  rowTrv[u] = type;  for (int i = 0; i < row[u].size(); i++) {  int v = row[u][i].first;  int w = row[u][i].second;  if (colTrv[v] != -1 and w == 0) return false;  if (colTrv[v] == -1) {  if (!dfs(odd ^ 1, v, w ^ 1 ^ type)) return false;  }  }  return true;  }  else {  colTrv[u] = type;  for (int i = 0; i < col[u].size(); i++) {  int v = col[u][i].first;  int w = col[u][i].second;  if (rowTrv[v] != -1 and w == 0) return false;  if (rowTrv[v] == -1) {  if (!dfs(odd ^ 1, v, w ^ 1 ^ type)) return false;  }  }  return true;  }  }  bool calc() {  for (int i = 0; i < rowList.size(); i++) {  if (rowTrv[rowList[i]] == -1) {  vector<int> rowBackup;  vector<int> colBackup;  for (int i = 0; i < rowList.size(); i++) rowBackup.push\_back(rowTrv[rowList[i]]);  for (int i = 0; i < colList.size(); i++) colBackup.push\_back(colTrv[colList[i]]);  rowTrv[rowList[i]] = 0;  if (!dfs(0, rowList[i], 0)) {  for (int i = 0; i < rowList.size(); i++) rowTrv[rowList[i]] = rowBackup[i];  for (int i = 0; i < colList.size(); i++) colTrv[colList[i]] = colBackup[i];  rowTrv[rowList[i]] = 1;  if (!dfs(0, rowList[i], 1)) return false;  }  }  }  return true;  }  void printRes(vector<int> &A, int B[], char C) {  for (int i = 0; i < A.size(); i++) {  if (B[A[i]] == 1) {  int x = A[i] / 100;  int y = A[i] % 100;  cout << C;  if (x < 10) cout << 0;  cout << x;  if (y < 10) cout << 0;  cout << y << ' ';  }  }  cout << endl;  }  void reset() {  memset(rowTrv, 0, sizeof(rowTrv));  memset(colTrv, 0, sizeof(colTrv));  rowList.clear();  colList.clear();  }  int main() {  int test; cin >> test;  for (int I = 1; I <= test; I++) {  reset();  cin >> n >> m;  for (int i = 1; i <= n; i++) for (int j = 1; j <= m; j++) {  cin >> a[i][j] >> r[i][j] >> c[i][j];  row[i \* 100 + r[i][j]].push\_back(make\_pair(j \* 100 + c[i][j], a[i][j]));  rowList.push\_back(i \* 100 + r[i][j]);  col[j \* 100 + c[i][j]].push\_back(make\_pair(i \* 100 + r[i][j], a[i][j]));  colList.push\_back(j \* 100 + c[i][j]);  }  cout << "Case #" << I << '\n';  if (calc()) {  printRes(rowList, rowTrv, 'R');  printRes(colList, colTrv, 'C');  }  else cout << "Impossible\n";  }  } |

## F- ACM Vietnam National 2017

|  |
| --- |
| **Đề bài:** Cho A, B, d, đếm số dãy tăng chặt độ dài k thoả mãn các phần tử nằm trong đoạn [A,B] và có đúng d chữ số khác nhau từ mọi số trong dãy.  1 ≤ A ≤ B ≤ 1018; 2 ≤ k ≤ 10; 0 ≤ d ≤ 10.  **Bước 1**  Tính số lượng số trong khoảng [A, B] mà gồm các chữ số nằm đúng trong tập S (S là tập con của tập {0, 1, 2, ..., 9})   * Số số nằm trong khoảng [A, B] = (số số nằm trong [0, B]) - (số số trong [0, A]). Do đó khi quy hoạch động ta chỉ cần quan tâm đến cận trên của các số. * Giống với các bài toán quy hoạch động chữ số, xuất phát từ số 0, ta lần lượt thêm các chữ số vào, và tính f(len, mask, lower, positive) với:   + len là độ dài của số ta đang xây dựng.   + mask là tập hợp các chữ số của số ta đang xây dựng.   + lower = 1 nếu số ta đang xây dựng đã nhỏ hơn cận trên B, = 0 trong trường hợp ngược lại.   + positive = 1 nếu số ta đang xây dựng đã lớn hơn 0.   Cài đặt:  f[0][0][0][0] = 1;  // Xuất phát từ số 0  for (int len = 0; len < độ dài số B; len++) {  for (int mask = 0; mask < 1023; mask++) { // dùng bitmask lưu S.  for (int lower = 0; lower < 1; lower++) {  for (int positive = 0; positive = 1; positive++) {  // Thêm 1 chữ số  for (int new\_digit = 0; new\_digit < 10; new\_digit++) {  // Đảm bảo <= cận trên  if (lower == 0 && new\_digit > chữ số (len+1) của B)  continue;  // Tính mask2, lower2, positive2 là các giá trị của số  // mới sau khi thêm chữ số new\_digit  int positive2 = positive || (new\_digit > 0);  int lower2 = lower || (new\_digit < chữ số (len+1) của B).  int mask2 = mask;  if (positive2) mask2 |= 1<<new\_digit;  f[len+1][mask2][lower2][positive2] += f[len][mask][lower][positive];  }  }  }  }  }  **Bước 2**  Với mỗi tập S, tính xem có bao nhiêu số trong [A, B] có S là tập con của tập các chữ số của nó. Bạn có thể giải phần này trong O(3^10) bằng cách duyệt mọi tập con.  **Bước 3**  Dùng tổ hợp để đếm số bộ k. |