```
------------------------------------------------------------
             binary_search(upper_lower_bound)
------------------------------------------------------------
#include <bits/stdc++.h>
#define le 100010

using namespace std;
int n[le];

int fnc_lo(int a[], int len, int key){
  int hi = len - 1, lo = 0, index = -1, mid;
  while(lo <= hi){
    mid = (lo + hi) / 2;
    if(key == a[mid]){
      index = mid;
      hi = mid - 1;
    }
    else if(key < a[mid]) hi = mid - 1;
    else lo = mid + 1;
  }
  return lo;
}

int fnc_up(int a[], int len, int key){
  int hi = len - 1, lo = 0, index = -1, mid;
  while(lo <= hi){
    mid = (hi + lo) / 2;
    if(key == a[mid]){
      index = mid;
      lo = mid + 1;
    }
    else if(key < a[mid]) hi = mid - 1;
    else lo = mid + 1;
  }
  return lo;
}

int main()
{
  int t, co = 0;
  int len, x, y, q;
  scanf("%d", &t);
  while(t--){
    scanf("%d %d", &len, &q);
    for(int i = 0; i < len; i++) scanf("%d", &n[i]);
    printf("Case %d:\n", ++co);
    while(q--){
      scanf("%d %d", &x, &y);
      x = fnc_lo(n, len, x);
      y = fnc_up(n, len, y);
      y  -= x;
      printf("%d\n", y);
    }
  }

  return 0;
}
------------------------------------------------------------
                       segment_tree
------------------------------------------------------------
#include <bits/stdc++.h>
#define le 100005
using namespace std;
int n[le];
struct eg{
  int sum, maxx, minn;
};
eg arr[4 * le];
eg com(eg a, eg b){
  eg temp;
  temp.sum = a.sum + b.sum;
  temp.maxx = max(a.maxx, b.maxx);
  temp.minn = min(a.minn, b.minn);
  return temp;
}
void init(int nd, int b, int e){
  if(b == e){
    arr[nd].sum = n[b];
    arr[nd].maxx = n[b];
    arr[nd].minn = n[b];
    return;
  }
  int l = nd << 1, r = l | 1, m = (b + e) >> 1;
  init(l, b, m);
  init(r, m + 1, e);
  arr[nd] = com(arr[l], arr[r]);
}
eg query(int nd, int b, int e, int i, int j){
  if(b > j || e < i){
    eg temp;
    temp.sum = 0;
    temp.maxx = -INT_MAX;
    temp.minn = INT_MAX;
    return temp;
  }
  if(b >= i && e <= j) return arr[nd];
  int l = nd << 1, r = l | 1, m = (b + e) >> 1;
  return com(query(l, b, m, i, j), query(r, m + 1, e, i,
j));
}
void update(int nd, int b, int e, int i, int v){
  if(b > i || e < i) return;
  if(b >= i && e <= i){
    arr[nd].sum = v;
    arr[nd].maxx = v;
    arr[nd].minn = v;
    return;
  }
  int l = nd << 1, r = l | 1, m = (b + e) >> 1;
  update(l, b, m, i, v);
  update(r, m + 1, e, i, v);
  arr[nd] = com(arr[l], arr[r]);
}
int main(){
  //freopen("input.txt", "r", stdin);
  int len, q, a, b, c;
  scanf("%d %d", &len, &q);
  for(int i = 0; i < len; scanf("%d", &n[i]), i++);
  init(1, 0, len - 1);
  while(q--){
    scanf("%d %d %d", &c, &a, &b);
    if(c == 1){
      eg temp = query(1, 0, len - 1, a - 1, b - 1);
      int ve = temp.sum - temp.maxx - temp.minn;
      printf("%d\n", ve);
    }
    else update(1, 0, len - 1, a - 1, b);
  }
  return 0;
}
------------------------------------------------------------
                     segment_tree(lazy)
------------------------------------------------------------
#include <bits/stdc++.h>
#define ll long long int
#define le 400005
using namespace std;
struct info{
  ll sum, pro;
}arr[le];
void update(int nd, int b, int e, int i, int j, ll v){
  if(b > j || e < i) return;
  if(b >= i && e <= j){
    arr[nd].sum += ((e - b + 1) * v);
    arr[nd].pro += v;
    return;
  }
  int l = nd << 1, r = l | 1, m = (b + e) >> 1;
  update(l, b, m, i, j, v);
  update(r, m + 1, e, i, j, v);
  arr[nd].sum = arr[l].sum + arr[r].sum + (e - b + 1) *
arr[nd].pro;
}
ll query(int nd, int b, int e, int i, int j, ll c){
  if(b > j || e < i) return 0;
  if(b >= i && e <= j) return arr[nd].sum + c * (e - b + 1);
```

```
  int l = nd << 1, r = l | 1, m = (b + e) >> 1;
  ll v1 = query(l, b, m, i, j, c + arr[nd].pro);
  ll v2 = query(r, m + 1, e, i, j, c + arr[nd].pro);
  return v1 + v2;
}
int main(){
  //freopen("input.txt", "r", stdin);
  //freopen("output.txt", "w", stdout);
  int t, co = 0, len, q, a, b, c;
  ll d;
  for(scanf("%d", &t); t--; ){
    scanf("%d %d", &len, &q);
    memset(arr, 0, sizeof(arr));
    printf("Case %d:\n", ++co);
    while(q--){
      scanf("%d %d %d", &c, &a, &b);
      if(!c){
        scanf("%lld", &d);
        update(1, 0, len - 1, a, b, d);
      }
      else printf("%lld\n", query(1, 0, len - 1, a, b, 0));
    }
  }
  return 0;
}
```

---

seive

```
#include <bits/stdc++.h>
#define le 100000008
using namespace std;
bool n[le >> 1];
vector<int> v;
// 1, 2 && even numbers have to handel carefully
void se(){  // don't forget to call this fnc :)
    int rt = sqrt(le) + 1, k;
    for(int i = 3; i < rt; i += 2) if(!n[i >> 1]) for(int j
= i * i, k = i << 1; j < le; j += k) n[j >> 1] = 1;
}

int main(){
    se();
    // test print
    for(int i = 3; i < 33; i += 2) if(!n[i >> 1])
printf("%d\n", i);
    return 0;
}
```

---

bit_seive

```
#define le 100000008
int n[le >> 6];
#define ck(ve) (n[ve >> 6] & (1 << ((ve >> 1) & 31)))
#define st(ve) (n[ve >> 6] |= (1 << ((ve >> 1) & 31)))
void se(){
    int rt = sqrt(le) + 1, k;
    for(int i = 3; i < rt; i += 2) if(!ck(i)) for(int j = i
* i, k = i << 1; j < le; j += k) st(j);
}
```

---

segment_sieve

```
#include <bits/stdc++.h>
#define l 1000000009
#define le 32004
#define ll long long int
using namespace std;
int n[le >> 6];
vector<int> v;
#define ck(ve) (n[ve >> 6] & (1 << ((ve >> 1) & 31)))
#define st(ve) (n[ve >> 6] |= (1 << ((ve >> 1) & 31)))
void se(){
    int rt = sqrt(le) + 1, k;
    for(int i = 3; i < rt; i += 2) if(!ck(i)) for(int j = i
* i, k = i << 1; j < le; j += k) st(j);
    v.push_back(2);
    for(int i = 3; i < le; i += 2) if(!ck(i))
v.push_back(i);
}
void segse(ll a, ll b){
  bool arr[b - a + 1];
  memset(arr, true, sizeof(arr));
  if(a == 1) arr[0] = false;
  for(int i = 0; i < v.size() && v[i] * v[i] <= b; i++){
    ll base = (a / v[i]) * v[i];
    if(base < a) base += v[i];
    for(ll j = base; j <= b; j += v[i]) arr[j - a] = false;
    if(base == v[i]) arr[base - a] = true;
  }
  bool f = false;
  for(int i = 0; i < b - a + 1; i++){
    if(arr[i]) printf("%lld\n", i + a);
  }
}
```

---

big_mod

```
int m;
int fnc(int a, int b){
    if(b == 0) return 1 % m;
    if(b % 2 == 0){
        int rt = fnc(a, b / 2);
        return ((rt % m) * (rt % m) % m);
    }
    return ((a % m) * fnc(a, b - 1) % m) % m;
}
```

Euler'sTotientFunction

```
#include <bits/stdc++.h>
#define ll long long int
using namespace std;
ll fnc(ll a){
    ll f = a;
    if(a % 2 == 0){
        f /= 2;
        while(a % 2 == 0) a /= 2;
    }
    for(ll i = 3; i * i <= a; i += 2){
        if(a % i == 0){
            while(a % i == 0) a /= i;
            f = (f * (i - 1)) / i;
        }
    }
    if(a > 1) f = (f * (a - 1)) / a;
    return f;
}
int main(){
    ll a;
    while(scanf("%lld", &a) != EOF && a){
        printf("%lld\n", fnc(a));
    }
    return 0;
}
```

Euler'sTotientFunction (Sieve)

```
#include <bits/stdc++.h>
#define le 1000006
#define ll long long int
using namespace std;
ll n[le];
bool m[le];
void fi(){
    for(int i = 1; i < le; n[i] = i, i++);
    n[1] = 1;
    m[1] = 1;
    for(int i = 2; i < le; i++){
        if(!m[i]){
            for(int j = i; j < le; j += i){
                m[j] = 1;
                n[j] = (n[j] * (i - 1)) / i;
            }
        }
    }
}
```

```
int main(){
    fi();
    int t, a;
    for(scanf("%d", &t); t--; ){
        scanf("%d", &a);
        printf("%d\n", n[a]);
    }
    return 0;
}
```

e_gcd

```
#include <bits/stdc++.h>
#define ll long long int
using namespace std;
int egcd(int a, int b, int &x, int &y){
  if(a == 0){
    x = 0;
    y = 1;
    return b;
  }
  int x1, y1;
  int g = egcd(b % a, a, x1, y1);
  x = y1 - (b / a) * x1;
  y = x1;
  return g;
}
int main(){
  int a, b, g, x, y;
  while(scanf("%d %d", &a, &b) != EOF && (a || b)){
    g = egcd(a, b, x, y);
    printf("%d %d %d\n", g, x, y);
  }
  return 0;
}
```

#in_factorial

```
#include <bits/stdc++.h>
#define le 1003
using namespace std;
double n[le];
int main(){
    n[0] = 0.0;
    for(int i = 1; i < le; i++) n[i] = n[i - 1] + log10(i *
1.0);
    int a;
    while(scanf("%d", &a) != EOF) printf("%d\n", (int)(n[a])
+ 1);
    return 0;
}
```

prime_factorization

```
#include <bits/stdc++.h>
#define le 32767
#define ll long long int
using namespace std;
int n[le >> 6];
vector<int> v;
vector<int> p;
#define ck(ve) (n[ve >> 6] & (1 << ((ve >> 1) & 31)))
#define st(ve) (n[ve >> 6] |= (1 << ((ve >> 1) & 31)))
void se(){
    int rt = sqrt(le) + 1, k;
    for(int i = 3; i < rt; i += 2) if(!ck(i)) for(int j = i
* i, k = i << 1; j < le; j += k) st(j);
    v.push_back(2);
    for(int i = 3; i < le; i += 2) if(!ck(i))
v.push_back(i);
}
void fnc(ll a){
    vector<int> pri;
    bool f = false;
    for(int i = 0; i < v.size() && v[i] * v[i] <= a; i++){
        if(a % v[i] == 0){
            int p = 0;
            while(a % v[i] == 0){
                a /= v[i];
                p++;
            }
            pri.push_back(p);
            pri.push_back(v[i]);
        }
    }
    if(a != 1){
        pri.push_back(1);
        pri.push_back(a);
    }
    for(int i = pri.size() - 1; i > 1; printf("%d %d ",
pri[i], pri[i - 1]), i -= 2);
    printf("%d %d\n", pri[1], pri[0]);
    return;
}
int main(){
    //freopen("input.txt", "r", stdin);
    //freopen("output.txt", "w", stdout);
    se();
    string s;
    while(getline(cin, s) && s[0] != '0'){
        int x = 0;
        for(int i = 0; i < s.size(); i++){
            if(s[i] == ' '){
                p.push_back(x);
                x = 0;
            }
            else x = x * 10 + s[i] - '0';
        }
        p.push_back(x);
        ll ans = 1;
        for(int i = 0; i < p.size(); i += 2) ans *=
pow(p[i], p[i + 1]);
        fnc(ans - 1);
        p.clear();
    }
    return 0;
}
```

Adjacency matrix

```
for(int i = 0; i < r; i++) for(int j = 0; j < c; j++)
  scanf("%d", &n[i][j]);
```

Adjacency list

```
scanf("%d %d", &n, &m);
for(int i = 0; i < m; scanf("%d %d", &a, &b),
v[a].push_back(b), v[b].push_back(a), i++);
```

Adjacency list(with cost)

```
for(int i = 0; i < m; i++){
  scanf("%d %d %d", &a, &b, &c);
  v[a].push_back(make_pair(c, b));
  v[b].push_back(make_pair(c, a));
}
```

east:west:north:south

```
int fx[]={+0,+0,+1,-1,-1,+1,-1,+1};
int fy[]={-1,+1,+0,+0,+1,+1,-1,-1};
```

bfs_1D

```
#define le 10004
vector<int> v[le];
bool vis[le];
int dis[le];
void bfs(int a){
  vis[a] = true;
  dis[a] = 0;
  queue<int> q;
  q.push(a);
  while(!q.empty()){
    int p = q.front();
    q.pop();
```

```
    for(int i = 0; i < v[p].size(); i++){
      int ve = v[p][i];
      if(!vis[ve]){
        vis[ve] = true;
        dis[ve] = dis[p] + 1;
        q.push(ve);
      }
    }
  }
}
```

---

bfs_2D

```
#include <bits/stdc++.h>
#define le 1003
using namespace std;
int fx[] = {1, -1, 0, 0}, fy[] = {0, 0, 1, -1}, len[le][le];
bool n[le][le], vis[le][le];
int bfs(int x, int y, int x1, int y1, int r, int c){
  for(int i = 0; i < r; i++) for(int j = 0; j < c; vis[i][j]
= false, len[i][j] = 0, j++);
  vis[y][x] = true;
  len[y][x] = 0;
  queue<pair<int, int> > q;
  q.push(pair<int, int>(x, y));
  while(!q.empty()){
    pair<int, int> p = q.front();
    q.pop();
    for(int i = 0; i < 4; i++){
      int px = p.first + fx[i], py = p.second + fy[i];
      if(px >= 0 && px < c && py >= 0 && py < r &&
vis[py][px] == false && n[py][px] == false){
        vis[py][px] = true;
        len[py][px] = len[p.second][p.first] + 1;
        q.push(pair<int, int>(px, py));
      }
    }
  }
  return len[y1][x1];
}
int main(){
  int r, c, a, b, x, y, x1, y1, w, t;
  while(scanf("%d %d", &r, &c) != EOF && r && c){
    for(int i = 0; i < r; i++) for(int j = 0; j < c; n[i][j]
= false, j++);
    scanf("%d", &t);
    for(int i = 0; i < t; i++){
      scanf("%d %d", &w, &a);
      while(a--){
        scanf("%d", &b);
        n[w][b] = true;
      }
    }
    scanf("%d %d %d %d", &y, &x, &y1, &x1);
    printf("%d\n", bfs(x, y, x1, y1, r, c));
  }
  return 0;
}
```

---

bfs_3D

```
#include <bits/stdc++.h>
#define le 32
using namespace std;
int l, r, c;
char n[le][le][le];
int fx[] = {1, -1, 0, 0, 1, -1};
int fy[] = {0, 0, 1, -1};
struct po{
  int x, y, z;
};
bool vis[le][le][le];
int dis[le][le][le];
void bfs(int z, int y, int x){
  for(int k = 0; k < le; k++) for(int i = 0; i < le; i++)
for(int j = 0; j < le; dis[k][i][j] = 0, vis[k][i][j] =
false, j++);
  dis[z][y][x] = 0;
  vis[z][y][x] = true;
  queue<pair<int, pair<int, int> > > q;
  q.push(make_pair(z, make_pair(y, x)));
  while(!q.empty()){
    pair<int, pair<int, int> > p = q.front();
    q.pop();
    for(int i = 0; i < 6; i++){
      int pz, py, px;
      if(i < 4){
        py = p.second.first + fy[i];
        px = p.second.second + fx[i];
      }
      else{
        py = p.second.first;
        px = p.second.second;
      }
      if(i >= 4) pz = p.first + fx[i];
      else pz = p.first;
      if(pz >= 0 && pz < l && py >= 0 && py < r && px >= 0
&& px < c && n[pz][py][px] != '#' && vis[pz][py][px] ==
false){
        vis[pz][py][px] = true;
        dis[pz][py][px] =
dis[p.first][p.second.first][p.second.second] + 1;
        q.push(make_pair(pz, make_pair(py, px)));
      }
    }
  }
}
int main(){
  //freopen("input.txt", "r", stdin);
  //freopen("output.txt", "w", stdout);
  string s;
  while(scanf("%d %d %d", &l, &r, &c) != EOF && (l || r ||
c)){
    po ss, dd;
    for(int k = 0; k < l; k++){
      for(int i = 0; i < r; i++){
        cin >> s;
        for(int j = 0; j < c; j++){
          if(s[j] == 'S'){
            ss.x = j;
            ss.y = i;
            ss.z = k;
          }
          else if(s[j] == 'E'){
            dd.x = j;
            dd.y = i;
            dd.z = k;
          }
          n[k][i][j] = s[j];
        }
      }
    }
    bfs(ss.z, ss.y, ss.x);
    if(!vis[dd.z][dd.y][dd.x]) printf("Trapped!\n");
    else printf("Escaped in %d minute(s).\n",
dis[dd.z][dd.y][dd.x]);
  }
  return 0;
}
```

---

dfs

```
#include <bits/stdc++.h>
#define le 10004
using namespace std;
vector<int> v[le];
bool vis[le];
void dfs(int a){
  vis[a] = true;
  for(int i = 0; i < v[a].size(); i++){
    if(!vis[v[a][i]]) dfs(v[a][i]);
  }
}
int main(){
  int n, m,  a, b;
  cin >> n >> m;
```

```
  for(int i = 0; i < m; i++){
    scanf("%d %d", &a, &b);
    v[a].push_back(b);
    v[b].push_back(a);
  }
  int co = 0;
  for(int i = 0; i < n; i++){
    if(!vis[i]){
      dfs(i);
      co++;
    }
  }
  cout << co << endl;
  return 0;
}
```

dijkstra_1D

```
#include <bits/stdc++.h>
#define le 102
using namespace std;
vector<pair<int, int> > v[le];
bool vis[le];
int dis[le];
void dijkstra(int a){
  memset(vis, 0, sizeof(vis));
  for(int i = 0; i < le; dis[i] = INT_MAX, i++);
  dis[a] = 0;
  priority_queue<pair<int, int>, vector<pair<int, int> >,
greater<pair<int, int> > > q;
  q.push(make_pair(0, a));
  while(!q.empty()){
    pair<int, int> p = q.top();
    q.pop();
    int no = p.second;
    if(vis[no]) continue;
    vis[no] = true;
    for(int i = 0; i < (int)v[no].size(); i++){
      int e = v[no][i].second, w = v[no][i].first;
      if(dis[e] > dis[no] + w){
        dis[e] = dis[no] + w;
        q.push(make_pair(dis[e], e));
      }
    }
  }
}
int main(){+
  int t, co = 0, n, m, a, b, c;
  for(scanf("%d", &t); t--; ){
    scanf("%d %d", &n, &m);
    for(int i = 0; i < m; i++){
      scanf("%d %d %d", &a, &b, &c);
      v[a].push_back(make_pair(c, b));
      v[b].push_back(make_pair(c, a));
    }
    dijkstra(1);
    if(!vis[n]) printf("Case %d: Impossible\n", ++co);
    else printf("Case %d: %d\n", ++co, dis[n]);
    for(int i = 0; i < le; v[i].clear(), i++);
  }
  return 0;
}
```

dijkstra_2D

```
#include <bits/stdc++.h>
#define le 1003
using namespace std;
int n[le][le], dis[le][le], fx[] = {1, -1, 0, 0}, fy[] = {0,
0, 1, -1};
bool vis[le][le];
int dijkstra(int a, int b, int r, int c){
  for(int i = 0; i < le; i++) for(int j = 0; j < le;
vis[i][j] = false, dis[i][j] = INT_MAX, j++);
  dis[a][b] = n[a][b];
  priority_queue<pair<int, pair<int, int> >,
vector<pair<int, pair<int, int> > >, greater<pair<int,
pair<int, int> > > > q;
  q.push(make_pair(n[a][b], make_pair(a, b)));
  while(!q.empty()){
    pair<int, pair<int, int> > p = q.top();
    q.pop();
    a = p.second.first;
    b = p.second.second;
    int w = p.first;
    for(int i = 0; i < 4; i++){
      int py = a + fy[i], px = b + fx[i];
      if(px >= 0 && px < c && py >= 0 && py < r &&
dis[py][px] > w + n[py][px]){
        dis[py][px] = w + n[py][px];
        q.push(make_pair(dis[py][px], make_pair(py, px)));
      }
    }
  }
  return dis[r - 1][c - 1];
}
int main(){
  int t, r, c;
  for(scanf("%d", &t); t--; ){
    scanf("%d %d", &r, &c);
    for(int i = 0; i < r; i++) for(int j = 0; j < c; j++)
scanf("%d", &n[i][j]);
    printf("%d\n", dijkstra(0, 0, r, c));
  }
  return 0;
}
```

bicoloring_graph

```
#include <bits/stdc++.h>
#define le 202
using namespace std;
vector<int> v[le];
bool vis[le];
int dis[le];
bool bfs(int a){
  for(int i = 0; i < le; vis[i] = false, dis[i] = -1, i++);
  vis[a] = true;
  dis[a] = 0;
  queue<int> q;
  q.push(a);
  while(!q.empty()){
    int p = q.front();
    q.pop();
    for(int i = 0; i < v[p].size(); i++){
      int e = v[p][i], w = (dis[p] + 1) % 2;
      if(vis[e] && dis[e] != w) return false;
      else if(vis[e] == false){
        vis[e] = true;
        dis[e] = w;
        q.push(e);
      }
    }
  }
  return true;
}
int main(){
  //freopen("input.txt", "r", stdin);
  //freopen("output.txt", "w", stdout);
  int n, m, a, b;
  while(scanf("%d", &n) != EOF && n){
    scanf("%d", &m);
    for(int i = 0; i < m; scanf("%d %d", &a, &b),
v[a].push_back(b), v[b].push_back(a), i++);
    printf("%s\n", bfs(0) ? "BICOLORABLE." : "NOT
BICOLORABLE.");
    for(int i = 0; i < le; v[i].clear(), i++);
  }
  return 0;
}
```

MST(prim's algo)

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

```
#define le 55
using namespace std;
vector<pair<int, int> > v[le];
bool vis[le];
int mst(int a){
  memset(vis, 0, sizeof(vis));
  int sum = 0;
  priority_queue<pair<int, int>, vector<pair<int, int> >,
greater<pair<int, int> > > q;
  q.push(make_pair(0, a));
  while(!q.empty()){
    pair<int, int> p = q.top();
    q.pop();
    int n = p.second;
    if(vis[n]) continue;
    vis[n] = true;
    sum += p.first;
    for(int i = 0; i < v[n].size(); i++){
      int e = v[n][i].second;
      if(!vis[e]) q.push(v[n][i]);
    }
  }
  return sum;
}
int main(){
  //freopen("input.txt", "r", stdin);
  //freopen("output.txt", "w", stdout);
  int t, co = 0, n, a;
  for(scanf("%d", &t); t--; ){
    scanf("%d", &n);
    int sum = 0;
    for(int i = 0; i < n; i++){
      for(int j = 0; j < n; j++){
        scanf("%d", &a);
        if(a){
          v[i].push_back(make_pair(a, j));
          v[j].push_back(make_pair(a, i));
          sum += a;
        }
      }
    }
    int ans = mst(0);
    bool f = true;
    for(int i = 0; i < n; i++){
      if(!vis[i]){
        f = false;
        break;
      }
    }
    if(f) printf("Case %d: %d\n", ++co, sum - ans);
    else printf("Case %d: -1\n", ++co);
    for(int i = 0; i < le; v[i].clear(), i++);
  }
  return 0;
}
```

---

mst(kruskal algo)

```
#include <bits/stdc++.h>
#define le 500005
using namespace std;
int p[1003];
struct edge{
  int x, y;
  double cost;
};
bool comp(edge a, edge b){
  return (a.cost - b.cost < 1e-9);
}
double dis(double a, double b, double x, double y){
  a = (a - x) * (a - x);
  b = (b - y) * (b - y);
  return sqrt(a + b);
}
int fnc(int a){
  if(p[a] == a) return a;
  p[a] = fnc(p[a]);
  return p[a];
}
int main(){
  int t, co = 0, n;
  double a, b, r;
  for(scanf("%d", &t); t--; ){
    vector<pair<double, double> > h;
    scanf("%d %lf", &n, &r);
    for(int i = 0; i < n; i++){
      p[i] = i;
      scanf("%lf %lf", &a, &b);
      h.push_back(make_pair(a, b));
    }
    edge arr[le];
    int l = 0;
    for(int i = 0; i < n - 1; i++){
      for(int j = i + 1; j < n; j++){
        double d = dis(h[i].first, h[i].second, h[j].first,
h[j].second);
        arr[l].x = i;
        arr[l].y = j;
        arr[l].cost = d;
        l++;
      }
    }
    sort(arr, arr + l, comp);
    int st = n;
    double bus = 0, tr = 0;
    for(int i = 0; i < l; i++){
      int ii = fnc(arr[i].x);
      int ll = fnc(arr[i].y);
      if(ii != ll){
        if(arr[i].cost - r <= 1e-9){
          st--;
          bus += arr[i].cost;
        }
        else tr += arr[i].cost;
        p[ll] = ii;
      }
    }
    printf("Case #%d: %d %.0lf %.0lf\n", ++co, st, floor(bus
+ 0.5) ,floor(tr + 0.5));
  }
  return 0;
}
```

---

second_best_MST

```
#include <bits/stdc++.h>
#define le 102
using namespace std;
int p[le];
int ans;
struct edge{
  int u, v, w;
};
bool comp(edge a, edge b){
  return a.w < b.w;
}
int fnc(int a){
  if(p[a] == a) return a;
  p[a] = fnc(p[a]);
  return p[a];
}
vector<edge> ve;
vector<edge> v;
int mst(int n){
  sort(v.begin(), v.end(), comp);
  int sum = 0, co = 0;
  for(int i = 0; i < (int)v.size(); i++){
    int a = fnc(v[i].u);
    int b = fnc(v[i].v);
    if(a != b){
      p[b] = a;
      sum += v[i].w;
      co++;
      ve.push_back(v[i]);
      if(co == n - 1) break;
    }
```

```
  }
  return sum;
}
int mst1(int n){
  int sum = INT_MAX;
  for(int j = 0; j < (int)ve.size(); j++){
    for(int i = 1; i < n + 1; p[i] = i, i++);
    int cost = 0, co = 0;
    for(int i = 0; i < (int)v.size(); i++){
      if(v[i].v == ve[j].v && v[i].u == ve[j].u && v[i].w ==
ve[j].w) continue;
      int a = fnc(v[i].u);
      int b = fnc(v[i].v);
      if(a != b){
        p[b] = a;
        co++;
        cost += v[i].w;
        if(co == n - 1) break;
      }
    }
    if(co < n - 1) continue;
    sum = min(sum, cost);
  }
  return sum;
}
int main(){
  //freopen("input.txt", "r", stdin);
  //freopen("output.txt", "w", stdout);
  int t, n, m, a, b, c;
  for(scanf("%d", &t); t--; ){
    ve.clear();
    scanf("%d %d", &n, &m);
    for(int i = 1; i < n + 1; p[i] = i, i++);
    edge ve;
    for(int i = 0; i < m; scanf("%d %d %d", &a, &b, &c),
ve.u = a, ve.v = b, ve.w = c, v.push_back(ve), i++);
    ans = mst(n);
    printf("%d %d\n", ans, mst1(n));
    v.clear();
  }
  return 0;
}
```

---

top_sort_dfs

```
#include <bits/stdc++.h>
#define le 11
using namespace std;
vector<int> v[le];
vector<int> ans;
bool vis[le];
void dfs(int a){
    vis[a] = true;
    for(int i = 0; i < v[a].size(); i++){
        int e = v[a][i];
        if(!vis[e]) dfs(e);
    }
    ans.push_back(a);
}
int main(){
    freopen("input.txt", "r", stdin);
    int n, m, a, b;
    scanf("%d %d", &n, &m);
    for(int i = 0; i < m; i++){
        scanf("%d %d", &a, &b);
        v[a].push_back(b);
    }
    for(int i = 1; i < n + 1; i++){
        if(!vis[i]) dfs(i);
    }
    for(int i = ans.size() - 1; i > 0; i--) printf("%d ",
ans[i]);
    printf("%d\n", ans[0]);
    return 0;
}
```

---

top_sort_bfs

```
#include <bits/stdc++.h>
#define le 12
using namespace std;
int in[le];
vector<int> v[le];
vector<int> ans;
void bfs(int n){
  priority_queue<int, vector<int>, greater<int> > q;
  for(int i = 1; i < n + 1; i++) if(in[i] == 0) q.push(i);
  while(!q.empty()){
    int p = q.top();
    q.pop();
    ans.push_back(p);
    for(int i = 0; i < v[p].size(); i++){
      int e = v[p][i];
      in[e]--;
      if(in[e] == 0) q.push(e);
    }
  }
}
int main(){
  //freopen("input.txt", "r", stdin);
  int n, m, a, b;
  scanf("%d %d", &n, &m);
  for(int i = 0; i < m; i++){
    scanf("%d %d", &a, &b);
    in[b]++;
    v[a].push_back(b);
  }
  bfs(n);
  for(int i = 0; i < ans.size() - 1; i++) printf("%d ",
ans[i]);
  printf("%d\n", ans[ans.size() - 1]);
  return 0;
}
```

---

recursionallPossibleSubSet

```
#include <bits/stdc++.h>
#define le 10002
using namespace std;
int len, n[le];
int main(){
  scanf("%d", &len);
  for(int i = 0; i < len; scanf("%d", &n[i]), i++);
  int ct = pow(2, len);
  for(int i = 0; i < ct; i++){
    for(int j = 0; j < len; j++){
      if((i & (1 << j)) > 0) printf("%d ", n[j]);
    }
    printf("\n");
  }
  return 0;
}
```

LCS && path

```
#include <bits/stdc++.h>
#define le 3003
using namespace std;
string a, b, s, ss;
bool vis[le][le];
int dis[le][le];
int fnc(int i, int j){
  if(i == a.size() || j == b.size()) return 0;
  if(vis[i][j]) return dis[i][j];
  int ans = 0;
  if(a[i] == b[j]) ans = 1 + fnc(i + 1, j + 1);
  else ans = max(fnc(i + 1, j), fnc(i, j + 1));
  dis[i][j] = ans;
  vis[i][j] = true;
  return dis[i][j];
}
void pri(int i, int j){
  if(i == a.size() || j == b.size()){
    if(ss == "") ss = s;
    else if(ss > s) ss = s;
    return;
```

```
  }
  if(a[i] == b[j]){
    s += a[i];
    pri(i + 1, j + 1);
    s.erase(s.size() - 1);
  }
  else{
    if(dis[i + 1][j] > dis[i][j + 1]) pri(i + 1, j);
    else pri(i, j + 1);
  }
}
int main(){
  //freopen("input.txt", "r", stdin);
  //freopen("output.txt", "w", stdout);
  int t, co = 0;
  for(scanf("%d", &t); t--; ){
    for(int i = 0; i < le; i++) for(int j = 0; j < le;
vis[i][j] = false, dis[i][j] = 0, j++);
    cin >> a >> b;
    int x = fnc(0, 0);
    printf("Case %d: ", ++co);
    if(x > 0){
      ss = s = "";
      pri(0, 0);
      cout << ss << endl;
    }
    else cout << "-1\n";
  }
  return 0;
}
```

Mohammad samiul islam

```
#define MAXD 2

double cosineRule3Side ( double a, double b, double c ) {
    double res = (SQ(a)+SQ(b)-SQ(c)) / (2*a*b);
    if ( res < -1 ) res = -1; if ( res > 1 ) res = 1;
    return acos ( res );
}

struct myVec {
    int d; //Dimension
    double val[MAXD];//Contains value of each component

    myVec add ( myVec b ) {
        myVec res; FOR(i,0,d) res.val[i] = val[i] +
b.val[i];return res;
    }
    myVec sub ( myVec b ) {
        myVec res; FOR(i,0,d) res.val[i] = val[i] -
b.val[i];return res;
    }
    myVec mul ( double t ) {
        myVec res; FOR(i,0,d)res.val[i] = val[i] * t;return
res;
    }
    myVec div ( double t ) {
        myVec res; FOR(i,0,d) res.val[i] = val[i] / t;return
res;
    }
    bool operator == ( myVec b ) {
        FOR(i,0,d) if ( fabs ( val[i] - b.val[i] ) > eps )
return false; return true;
    }
    myVec perp2D() {
        myVec res = (*this);
        swap ( res.val[0], res.val[1] );
        res.val[0] *= -1;
        return res;
    }
    double dot ( myVec v ) { //Finds *this (dot) v
        double res = 0; for ( int i = 0; i < d; i++ ) res +=
val[i] * v.val[i];
        return res;
    }
    double length () { //Finds length of current vector
        return sqrt ( this->dot( *this ) );
    }
    myVec unitVec () {
        return (*this).div ( length() ); // v / ||v||
    }
    double angleBetween ( myVec b ) { //Angle between two
vectors
        double res = dot( b ) / ( length() * b.length() );
        if ( res > 1 ) res = 1; if ( res < -1 ) res = -1;
        return acos (res);
    }
    double polarAngle2D() { //Angle from x-axis
        double res = atan2 ( val[1], val[0] );
        if ( res + eps < 0 ) res += 2 * pi;
        return res;
    }
    double cross2D ( myVec v ) { //Cross the two values.
Only for 2D. Z compo 0.
        return val[0]*v.val[1] - val[1]*v.val[0];
    }
};

struct myLine {
    myVec a, b; //a is displacement, b is direction.
    //Builds a line from two points
    myLine lineFromPoints ( myVec x, myVec y ) {
        myLine m;m.a = x; m.b = y.sub ( x );
        return m;
    }
    //Finds point on line, given t.
    myVec atPos ( double t ) {
        return a.add ( b.mul ( t ) ); // a + tb;
    }
    double lineToPointDistance ( myVec p, double t ) {
        p = p.sub ( a ); //Take it to origin
        t = b.dot ( p ) / ( b.length() * b.length() );
//point of intersection
        myVec x = b.mul ( t ); //tb
        return ( p.sub(x).length() ); //xp length()
    }
    double segmentToPointDistance ( myVec p, double &t ) {
        p = p.sub ( a ); //Take it to origin
        t = b.dot ( p ) / ( b.length() * b.length() );
        if ( t + eps < 0 || t > 1 + eps ) { //Not on segment
            return min ( p.length(), p.sub(b).length() );
        }
        myVec x = b.mul ( t ); //tb
        return ( p.sub(x).length() ); //xp length()
    }
     bool overlapParallel ( myLine l ) {
            double p, q, r, s;
            if ( b.val[0] == 0 ) {
                p = a.val[1]; q = atPos(1).val[1]; r =
l.a.val[1]; s = l.atPos ( 1 ).val[1];
                if ( min ( r, s ) > max ( p, q ) ) return
false;
                if ( max ( r, s ) < min ( p, q ) ) return
false;
                return true;
            }
            else {
                p = a.val[0]; q = atPos(1).val[0]; r =
l.a.val[0]; s = l.atPos ( 1 ).val[0];
                if ( min ( r, s ) > max ( p, q ) ) return
false;
                if ( max ( r, s ) < min ( p, q ) ) return
false;
                return true;
            }
    }
    char lineAndLineIntersection2D ( myLine l, double &t,
double &s ) {
        if ( b.cross2D ( l.b) == 0 ) {
            if ( l.a.sub(a).cross2D(l.b) == 0 ) {
                if ( overlapParallel ( l ) ) return 'o';
//overlaps
                else return 'p'; //parallel
            }
                else return 'd'; //disjoint and parallel
```

```
        }
        myVec w = a.sub ( l.a );
        myVec p = l.b.perp2D(), z = b.perp2D();
        t = -(w.dot(p))/p.dot(b); //for current line
        s = w.dot(z)/z.dot(l.b); //for line l
        return 'i';
    }
    double lineAndLineDistance2D ( myLine l ) {
        double t, s; //First check if the intersect
        char r = lineAndLineIntersection2D ( l, t, s );
        if ( r == 'i' ) return 0; //Intersects. 0 distance.
        //Parallel Lines
        return lineToPointDistance ( l.a, t );
    }
    double lineAndSegmentDistance2D ( myLine l ) {
        double t, s;
        char r = lineAndLineIntersection2D ( l, t, s );
        if ( r == 'i' && s + eps > 0 && s < 1 + eps ) {
                return 0; //Valid intersection
        }
        double res = lineToPointDistance ( l.a, t );
        res = min ( res, lineToPointDistance ( l.a.add(l.b),
t ) );
        return res;
    }
    double segmentAndSegmentDistance2D ( myLine l ) {
        double t, s;
        char r = lineAndLineIntersection2D ( l, t, s );
        if ( r =='i' && t+eps > 0 && t < 1 + eps && s + eps
> 0 && s < 1 + eps ) {
                return 0; //Valid intersection
        }
        double res = segmentToPointDistance ( l.a, t );
        res = min ( res, segmentToPointDistance (
l.a.add(l.b), t ) );
        res = min ( res, l.segmentToPointDistance ( a, t )
);
        res = min ( res, l.segmentToPointDistance ( a.add (
b ), t ) );
        return res;
    }
    myLine reflect ( myVec p, myVec norm ) {
        myVec ap = p.sub ( a ); //Starting to Point of
Reflection
        norm = norm.unitVec();

        double d = fabs ( ap.dot ( norm ) );

        myVec m = p.add ( norm.mul ( d ) );
        myVec h = m.sub ( a ).mul ( 2 );
        m = a.add ( h );

        myLine ray = ray.lineFromPoints ( p, m );
        return ray;
    }
};

struct myCir {
    myVec a;
    double r;
    myVec atPos ( double t ) {
        myVec res;
        res.val[0] = a.val[0] + r * cos ( t );
        res.val[1] = a.val[1] + r * sin ( t );
        return res;
    }
    char circleAndLineIntersection2D ( myLine l, double &t1,
double &t2 ) {
        double t3;
        double d = l.lineToPointDistance ( a, t3 );
        if ( d > r + eps ) return 'd';
        if ( fabs ( d - r ) <= eps ) return 't';

        myVec m = l.atPos ( t3 );
        myVec am = m.sub ( a );

        //Need to handle when line passes through center
        double x = am.polarAngle2D();
        double temp = d / r; if ( temp > 1 ) temp = 1; if (
temp < -1 ) temp = -1;
        double theta = pi / 2 - asin ( temp ); //Using sin
law find internal angle.

        t1 = x + theta;
        t2 = x - theta;
        return 'i';
    }
    char sphereAndLineIntersect ( myLine l, double &t1,
double &t2 ) {
        double tp = 0;
        double d = l.lineToPointDistance ( a, tp );
        if ( d > r + eps ) return 'd';
        if ( fabs ( d - r ) < eps ) {
            t1 = tp;
            return 't';
        }
        double chord = sqrt ( r * r - d * d );
        t1 = tp - chord / l.b.length();
        t2 = tp + chord / l.b.length();
        return 'i';
    }
     char circleAndCircleIntersection2D ( myCir c2, double
&t1, double &t2 ) {
        myVec d = c2.a.sub ( a );
        if ( d.length() > r + c2.r + eps ) return 'd';
//Case 1
        if ( d.length() + c2.r + eps < r ) return 'd';
//Case 2
        if ( a == c2.a && fabs ( r - c2.r ) <= eps ) {
            if ( r == 0 ) {
                t1 = 0;
                return 't'; //Case 7
            }
            return 's'; //Case 6
        }
        if ( fabs ( d.length() - r - c2.r ) <= eps ||
             fabs ( d.length() + c2.r - r ) <= eps ) {
            t1 = d.polarAngle2D();
            return 't'; //Case 3 and 4
        }
        double theta = cosineRule3Side ( r, d.length(), c2.r
);
        double m = d.polarAngle2D ();
        t1 = m - theta;
        t2 = m + theta;
        return 'i'; //Case 5
    }
    int circleToCircleTangentLine (myCir c2,myLine
&l1,myLine &l2,myLine &l3,myLine &l4){
    //First circle must be smaller or equal to second circle
        if (r>c2.r + eps ) return
c2.circleToCircleTangentLine ( *this, l1, l2, l3, l4 );

        myVec oo = c2.a.sub ( a );
        double d = oo.length();

        if ( fabs ( d ) < eps && fabs ( r - c2.r ) < eps )
//Infinite tangents
            return -1;
        if ( d + r + eps < c2.r ) //No tangents
            return 0;

          double base = oo.polarAngle2D();

        if ( fabs ( d + r - c2.r ) < eps ) { //Contains
Circle
            l1 = l1.lineFromPoints ( atPos ( base + pi ),
atPos ( base + pi ) );
            return 1;
        }

        double ang = pi - acos ( (c2.r - r ) / d );
        l1 = l1.lineFromPoints ( atPos ( base + ang ),
c2.atPos ( base + ang ) );
```

```
        l2 = l2.lineFromPoints ( atPos ( base - ang ),
c2.atPos ( base - ang ) );

        if ( d + eps < r + c2.r ) return 2; //Circle
intersects

        if ( fabs ( d - r - c2.r ) < eps ) { //Circle
tangent
            l3 = l3.lineFromPoints ( atPos ( base ), atPos (
base ) );
            return 3;
        }

        //Disjoint Circle
        ang = acos ( ( c2.r + r ) / d );
        l3 = l3.lineFromPoints ( atPos ( base + ang ),
c2.atPos ( base + ang + pi ) );
        l4 = l4.lineFromPoints ( atPos ( base - ang ),
c2.atPos ( base - ang + pi ) );

        return 4;
    }
};


bool collinear ( myVec a, myVec b, myVec c ) {
    myVec ab = b.sub(a), ac = c.sub(a);
    double d = fabs ( ab.dot(ac) );
    if ( fabs ( d - ab.length() * ac.length() ) <= eps )
return true;
    return false;
}
```