Contents

1 Basic

1.1 Template

```cpp
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define PB push_back
#define PII pair<int, int>
#define MP make_pair
#define IOS ios_base::sync_with_stdio(false); cin.tie(0)
#define all(x) x.begin(), x.end()
#define REP(x, y, z) for(int x = y; x <= z; x++)
#define maxn

//structure

//declaration

//functions

int main(void)
{
  IOS;


  return 0;
}
```

1.2 .vimrc

```
syntax on
color torte
set nu ts=4 sw=4 ai mouse=a bs=2 ci hls ru nocp
    showmatch ar
hi cursorline cterm=none ctermbg=darkred
filetype plugin indent on

so $VIMRUNTIME/mswin.vim
behave mswin
```

1.3 Basic cmd

```
lpr A.cpp # 列印
lpq # 查詢列印 queue

g++ A.cpp -std=gnu++14 -O2 -Wall -Wshadow -o A
./A
./A < input
./A < input > output
diff output ans
```

1.4 Standard Library

2 Graph

2.1 Dijkstra

```cpp
#include <bits/stdc++.h>
using namespace std;

#define MP make_pair
#define PII pair<int, int>
#define maxn 50000+5

int dis[maxn]; // 預設都是 INF
vector<PII > e[maxn]; // (連到的點， 邊的距離)

void dijk(int cur) // dijk(起點)
{
  int d;
  priority_queue<PII,vector<PII>,greater<PII>> q; // 放(距離, 點編號)，每次會拿距離最小的點出來
  q.push( MP(0, cur) );

  while (!q.empty())
  {
    tie(d, cur) = q.top();
    q.pop();
    if (dis[cur] != 1e9)
        continue; // 如果之前就拜訪過，無視

    dis[cur] = d;

    for (auto i: e[cur])
        if (dis[i.first] == 1e9)
        {
            q.push( MP(d+i.second, i.first) );
        }
    }
}

void init(void)
{
    fill(dis, dis+maxn, 1e9);

    for(int i = 0; i < maxn; i++)
    {
        e[i].clear();
    }
```

```
}
```

2.2 Kruskal 最小生成樹

```
//kruskal algorithm
//minimum spanning tree

#include <bits/stdc++.h>
using namespace std;

#define maxn

struct Edge
{
  int from, to, cost;

  Edge(int _from, int _to, int _cost)
  {
    from = _from;
    to = _to;
    cost = _cost;
  }

  bool operator< (const Edge &r) const
  {
    return cost < r.cost;
  }
};

int parent_arr[maxn];
int n, m, cost;
vector<Edge> edges;

int find(int x)
{
  return parent_arr[x] < 0 ? x : (parent_arr[x] = find(
      parent_arr[x]));
}

void conn(int x,int y)
{
    parent_arr[find(y)]=find(x);
}

void kruskal_algorithm(void)
{
  cost = 0;
  memset(parent_arr, -1, sizeof(parent_arr));
  sort(edges.begin(), edges.end());

  for(int i = 0; i < m; i++)
  {
    Edge tmp = edges[i];

    if(find(tmp.to) == find(tmp.from))
    {
      //不能形成環的邊
      continue;
    }

    else
    {
      cost += tmp.cost;
      conn(tmp.from, tmp.to);

    }
  }
}

int main(void)
{
  while(cin >> n >> m)
  {
    //init
    edges.clear();
    cost = 0;

    for(int i = 0; i < m; i++)
    {
      int a, b, c;
      cin >> a >> b >> c;
      edges.push_back(Edge(a, b, c));
    }

    kruskal_algorithm();

    cout << cost << '\n';
  }
  return 0;
}
```

2.3 Floyd Warshell 任意點最短路

```
for (int k = 1; k <= n; k++)
    for (int i = 1; i <= n; i++)
      for (int j = 1; j <= n; j++)
          if (dis[i][j] > dis[i][k] + dis[k][j])
          {
            // 如果可以以 k 為中繼點，更新 i,j 的最短距離
            dis[i][j] = dis[i][k] + dis[k][j];
      }
```

2.4 Floyd Warshell 有向圖最小環

```
#include <bits/stdc++.h>
using namespace std;

#define maxn 100+5

int n;
int ans;
int dis[maxn][maxn];

int main(void)
{
  while(cin >> n)
  {
    ans = 1e9;
    for(int i = 1; i <= n; i++)
    {
      for(int j = 1; j <= n; j++)
      {
        cin >> dis[i][j];
        if(!dis[i][j])  dis[i][j] = 1e9;
      }
    }

    for (int k = 1; k <= n; k++)
        for (int i = 1; i <= n; i++)
          for (int j = 1; j <= n; j++)
          {
            if (dis[i][j] > dis[i][k] + dis[k][j])
              {
                // 如果可以以 k 為中繼點，更新 i,j 的最短距離
                dis[i][j] = dis[i][k] + dis[k][j];
            }

            if(i == j)
              ans = min(ans, dis[i][j]);
            }

    if(ans == 1e9)
      cout << -1 << '\n';
    else
      cout << ans << '\n';
  }
```

```
  return 0;
}
```

2.5 SPFA

```
#include <bits/stdc++.h>
using namespace std;

#define MP make_pair
#define PII pair<int, int>
#define maxn 500+5

const int INF = 1e9; //比最大可能的距離更大

bool inq[maxn]; // inq[i] 代表 i 在 queue 裡面
int dis[maxn]; // 預設都是 INF
vector<PII> e[maxn]; // (連到的點， 邊的距離)

void spfa(int cur)
{
  queue<int> q;
  dis[cur] = 0;
  q.push(cur);

  while (!q.empty())
  {
      cur = q.front();
      q.pop();
      inq[cur] = false;

      for (auto i: e[cur])
      {
        // 如果點 cur，經過權重 i.S 這條邊，走到 i.F 可以更短，就更新
          if (i.second + dis[cur] < dis[i.first])
          {
            dis[i.first] = dis[cur] + i.second;
            if (!inq[i.first])
            {
              inq[i.first] = true;
              q.push( i.first );
            }
        }
      }
  }
}

void init(void)
{
  fill(dis, dis+maxn, INF);
  for(int i = 0; i < maxn; i++)
  {
    e[i].clear();
  }

  memset(inq, false, sizeof(inq));
}
```

2.6 SPFA 找負環

```
#include <bits/stdc++.h>
using namespace std;

#define MP make_pair
#define PII pair<int, int>
#define maxn 500+5

const int INF = 1e9; //比最大可能的距離更大

bool inq[maxn]; // inq[i] 代表 i 在 queue 裡面
int dis[maxn]; // 預設都是 INF
int updateCount[maxn];
int vis[maxn];
vector<PII> e[maxn]; // (連到的點， 邊的距離)
int n, m;

void spfa(int cur)
{
  queue<int> q;
  dis[cur] = 0;
  q.push(cur);

  while (!q.empty())
  {
      cur = q.front();
      q.pop();
      inq[cur] = false;

      for (auto i: e[cur])
      {
        // 如果點 cur，經過權重 i.S 這條邊，走到 i.F 可以更短，就更新
          if (i.second + dis[cur] < dis[i.first])
          {
            dis[i.first] = dis[cur] + i.second;
            if (!inq[i.first])
            {
              // updateCount 紀錄一個點被放到 queue 幾次
              updateCount[i.first]++;
              if(updateCount[i.first] > n)
              {
                continue;
              }
                inq[i.first] = true;
                q.push( i.first );
            }
        }
      }
  }
}

void init(void)
{
  fill(dis, dis+maxn, INF);
  for(int i = 0; i < maxn; i++)
  {
    e[i].clear();
  }
  memset(updateCount, 0, sizeof(updateCount));
  memset(inq, false, sizeof(inq));
}

bool dfs(int cur)
{
    vis[cur]=true;
    if(cur==n)return true;

    for(int i = 0; i < e[cur].size(); i++)
        if(!vis[e[cur][i].first])
            if(dfs(e[cur][i].first))
                return true;
    return false;
}

bool check()
{
    memset(vis,false, sizeof(vis));
    for(int i = 1; i <= n; i++)
        if(updateCount[i]>n && dfs(i))
            return true;
    return false;
}

int main(void)
{
  int x, y, z;
  while(cin >> n >> m)
  {
```

```
    init();

    for(int i = 0; i < m; i++)
    {
      cin >> x >> y >> z;
      e[x].push_back(MP(y, z));
    }

    spfa(1);

    if(dis[n]!=INF && !check())
      cout << dis[n] << '\n';
    else
      cout << "There a negative cycle or no path\n";
  }
  return 0;
}
```

2.7 拓樸排序

```
#include <bits/stdc++.h>
#define maxn 50005
using namespace std;
struct edge
{
    int t,next;
}   in[maxn*4];
//n vertex has n*4 maximum edges

int n,m,e,first[maxn],s[maxn],top;
//  first   紀錄是否有固定順序
//  s 紀錄順序

bool fail,ins[maxn],vis[maxn];
//  vis 是否訪問
//  ins 在做dfs的當下 那點是否被訪問過

void add(int x,int y)
{
    in[e].t=y;
    in[e].next=first[x];
    first[x]=e++;
}
void dfs(int cur)
{
    ins[cur]=vis[cur]=true;
    for(int i=first[cur]; ~i; i=in[i].next)
    {
        if(!vis[in[i].t])
            dfs(in[i].t);
        else if(ins[in[i].t])
            fail=true;
    }
    ins[cur]=false;
    s[top++]=cur;
}
int main(void)
{
    int x,y;
    while(cin >> n >> m)
    {
        //init
        e = 0;
        top = 0;
        fail = false;
        memset(first, -1, sizeof(first));
        memset(ins, false, sizeof(ins));
        memset(vis, false, sizeof(vis));

        for(int i = 1; i <= m; i++)
        {
            scanf("%d %d",&x,&y);
            add(x,y);
        }

        for(int i = 1; i <= n; i++)
            if(!vis[i])
                dfs(i);

        if(fail)
            puts("-1");
        else
            for(int i = top-1; i >= 0; i--)
                printf("%d\n",s[i]);
    }
    return 0;
}
```

2.8 LCA 樹的最短路/找共同祖先

```
#include <bits/stdc++.h>
using namespace std;

#define maxn
#define LG //LG = log2n
#define PB push_back
#define MP make_pair

int f[LG][maxn];
int dep[maxn]; // dep[i] 是點 i 的深度，root 深度是 0，下一層的深度是 1...
int depw[maxn];
int n, m;

// if no weight
// int e[maxn];

//if the edge with weight
vector< pair<int, int> > e[maxn];

void dfs(int cur,int fa) { // 多帶一個父節點的參數，是在樹上 dfs 常見的技巧，可以省去平常 dfs 需要的 vis 陣列
    f[0][cur] = fa;
    for (auto i: e[cur]) if (i != fa) {
    dep[i] = dep[cur]+1;
    dfs(i, cur);
    }
}

int lca(int x,int y) {
    // 跟 swap(x,y) 是一樣的意思
    if (dep[x] < dep[y]) return lca(y,x);
    // 這裡開始 dep[x] >= dep[y] 一定成立

    for (int i=LG-1; i>=0; i--)
        if (dep[x]-(1<<i) >= dep[y]) // 先想辦法把 x,y 調到同深度
            x = f[i][x];
        if (x==y) return x; // 如果發現同深度時，是同一個點就回傳找到 LCA 了

    // 否則盡量想辦法往上走，只要 x,y 同時往上走 2^i 步還不是相同的點，就 greedy 走
    for (int i=LG-1; i>=0; i--)
        if (f[i][x] != f[i][y])
        {
            x = f[i][x];
            y = f[i][y];
        }
  assert(f[0][x] == f[0][y]); // 走完以後，會發現 x,y 停在 lca 的正下方一個點
  return f[0][x];
}
void make_lca() {
  dep[1] = depw[1] = 0;
  dfs(1, 1); // 拿 1 當 root，且 1 的父節點是 1
  for (int i=1; i<LG; i++)
    for (int j=1; j<=n; j++)
```

```
      f[i][j] = f[i-1][f[i-1][j]]; // j 往上走 2^(i-1) 再往上走 2^(i-1) = 往上走 2^i 步
}

int main(void)
{
    while(cin >> n >> m)
    {
        //init
        for(int i = 0; i < maxn; i++)
            e[i].clear();

        int x, y, z;

        for(int i = 0; i < n-1; i++)
        {
            // if no weight
            // cin >> x >> y;
            // e[x].PB(y);
            // e[y].PB(x);

            cin >> x >> y >> z;
            e[x].PB(MP(y, z));
            e[y].PB(MP(x, z));
        }

        //make LCA
        make_lca();

        for(int i = 0; i < m; i++)
        {
            cin >> x >> y;
            cout << dep[x]+dep[y]-2*dep[lca(x, y)] << '\n';
        }
    }
    return 0;
}
```

2.9 Hungarian 二分圖匹配

```
#include <bits/stdc++.h>
using namespace std;

#define MP make_pair
#define PB push_back
#define maxn 50005

struct edge
{
    int t,next;
}   in[maxn*8];

int n,m,e,first[maxn];
bool vis[maxn],fail,side[maxn];

void add(int x,int y)
{
    in[e].t=y;
    in[e].next=first[x];
    first[x]=e++;
}

void dfs(int cur,bool tf)
{
    vis[cur]=true;
    side[cur]=tf;
    for(int i = first[cur]; ~i; i = in[i].next)
    {
        if(fail)
            return;
        if(vis[in[i].t] && side[in[i].t]==tf)
            fail=true;
        else if(!vis[in[i].t])
            dfs(in[i].t, !tf);
    }
}

int main(void)
{
    int x,y;
    while(cin >> n >> m)
    {
        e = 0;
        fail = false;
        memset(first, -1, sizeof(first));
        memset(vis, false, sizeof(vis));

        for(int i = 0;i < m; i++)
        {
            cin >> x >> y;

            add(x,y);
            add(y,x);
        }

        for(int i = 1; i <= n; i++)
            if(!vis[i])
                dfs(i,false);

        if(fail)
            puts("no");
        else
            puts("yes");
    }
    return 0;
}
```

2.10 樹上兩點最遠距離

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define PB push_back
#define PII pair<int, int>
#define MP make_pair
#define IOS ios_base::sync_with_stdio(false); cin.tie(0)
#define all(x) x.begin(), x.end()
#define REP(x, y, z) for(int x = y; x <= z; x++)
#define maxn 100000+5

//structure

//declaration
int n;
vector<PII> e[maxn];
PII first;

//functions
void dfs(int ver, int fa, int dep)
{
    if(dep > first.second)
    {
        first.first = ver;
        first.second = dep;
    }

    for(auto i : e[ver])
    {
        if(i.first != fa)
        {
            dfs(i.first, ver, dep+i.second);
        }
    }

    return;
}

int main(void)
```

```
{
    IOS;

    while(cin >> n)
    {
        //init
        REP(i, 1, n)
        {
            e[i].clear();
        }

        first.second = 0;


        int x, y, z;

        REP(i, 1, n-1)
        {
            cin >> x >> y >> z;

            e[x].PB(MP(y, z));
            e[y].PB(MP(x, z));
        }

        dfs(1, 1, 0);

        first.second = 0;

        dfs(first.first, first.first, 0);

        printf("%.1f\n", (float)(first.second)/2);
    }

    return 0;
}
```

2.11 Max Flow

```
struct Edge{
    int t,r,opp,next; // t=邊的終點，r=剩餘流量, opp=反向邊編號, next=下一條邊編號(鍊表)
    Edge () {}
    Edge (int a,int b,int c,int d) { t=a; r=b; opp=c; next=d; }
}in[M*25];
int e,st,ed; // 當前邊標號, 源點, 匯點
int first[M],arc[M],dis[M]; // arc: bfs完以後，每個點找剩餘流找到哪條邊的標記
void add(int x,int y,int z) { // 加一條 x->y， 流量 z 的邊
    in[e] = Edge(y,z,e+1,first[x]);
    first[x] = e++;
    in[e] = Edge(x,0,e-1,first[y]);
    first[y] = e++;
}
void init() {
    e = 0;
    MSET(first, -1);
    MSET(dis, 0);
}
bool bfs() {
    int cur;
    queue<int> q;
    REP(i,0,n) dis[i] = 0;
    REP(i,0,n) arc[i] = first[i];
    dis[ed] = 1;
    q.push(ed);

    while (!q.empty()) {
        cur = q.front();
        q.pop();
        for (int i=first[cur]; ~i; i=in[i].next)
            if (in[in[i].opp].r && !dis[in[i].t]) {
                dis[in[i].t] = dis[cur] + 1;
                q.push(in[i].t);
            }
    }
    return dis[st] > 0;
}
int dfs(int cur,int flow) {
    if (cur==ed) return flow;
    int re=0, tmp;
    for (int i=arc[cur]; ~i; i=arc[cur]=in[i].next)
        if (dis[in[i].t]==dis[cur]-1 && in[i].r>0) { // 如果距離編號剛好小1，且有剩餘流量
            tmp = dfs(in[i].t, min(in[i].r, flow));
            re += tmp;
            flow -= tmp;
            in[i].r -= tmp;
            in[in[i].opp].r += tmp;
            if (!flow) return re;
        }
    return re;
}
int maxflow() {
    int res = 0;
    while(bfs()) res += dfs(st, 2000000000);
    return res;
}
```

2.12 Max Flow Minimun Cut

```
const int INF = 1e9;
struct edge{int t,r,cost,next;};
int dis[M],pre[M],rec[M];
bool inq[M];

bool spfa()
{
  int cur;
  MSET(inq,false);
  REP(i,0,n)dis[i]=INF;
  dis[st]=0;
  q.push(st);

  while(!q.empty())
  {
    cur=q.front();
    q.pop();
    inq[cur]=false;
    for(int i=first[cur];~i;i=in[i].next)
    if(in[i].r>0 && dis[cur]+in[i].cost<dis[in[i].t])
    {
      dis[in[i].t]=dis[cur]+in[i].cost;
      pre[in[i].t]=cur;
      rec[in[i].t]=i;
      if(!inq[in[i].t])
      {
        q.push(in[i].t);
        inq[in[i].t]=true;
      }
    }
  }
  if(dis[ed]==INF)return false;
  return true;
}

int costflow()
{
  int delta,mincost=0,maxflow=0;
  while(spfa())
  {
    delta=INF;
    for(int i=ed;i!=st;i=pre[i])
      if(in[rec[i]].r<delta)
        delta=in[rec[i]].r;
    for(int i=ed;i!=st;i=pre[i])
    {
      in[rec[i]].f+=delta;
      in[in[rec[i]].opp].f-=delta;
```

```
      in[rec[i]].r-=delta;
      in[in[rec[i]].opp].r+=delta;
    }
    mincost+=dis[ed]*delta;
    maxflow+=delta;
  }
  return mincost;
}
```

2.13 尤拉路徑

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define PB push_back
#define PII pair<int, int>
#define MP make_pair
#define IOS ios_base::sync_with_stdio(false); cin.tie(0)
#define all(x) x.begin(), x.end()
#define REP(x, y, z) for(int x = y; x <= z; x++)
#define maxn 50000+5
#define maxm 200000+5

//structure

//declaration
int n, m;
int st;
vector<PII> adj[maxn];
vector<bool> edges;
int chk[maxn];
//functions


void dfs(int v)
{
    for(auto i : adj[v])
    {
        if(edges[i.first] == true)
        {
            edges[i.first] = false;
            dfs(i.second);
            cout << i.second << ' ' << v << '\n';
        }
    }
}

int main(void)
{
    //IOS;

    while(cin >> n >> m)
    {

        //init
        REP(i, 1, n)
            adj[i].clear();
        edges.clear();
        memset(chk, 0, sizeof(chk));

        int x, y;

        REP(i, 1, m)
        {
            cin >> x >> y;
            edges.PB(true);
            adj[x].PB(MP(i-1, y));
            adj[y].PB(MP(i-1, x));
            chk[x]++;
            chk[y]++;
        }

        //find 奇數點
        st = 1;
        REP(i, 1, n)
        {
            if(chk[i]%2 == 1)
            {
                st = i;
                break;
            }
        }

        dfs(st);
    }

    return 0;
}
```

3 Math

3.1 尤拉函數線上

```
#include<bits/stdc++.h>
using namespace std;
#define maxn 46340 //bcz sqrt(2^31-1)~=46340.95 and 46341 not prime
bool prime[maxn];
void prime_table(){
  memset(prime,true,sizeof(prime));
  prime[0]=prime[1]=false;
  for (int i = 2; i < maxn; ++i)
    if (prime[i])
      for (int j = i*i; j < maxn; j+=i)
        prime[j]=false;
    }

int eularphi(int n)
{
  if (n==0) return n;
  int ans=n;
  for (int i = 2; i < maxn; ++i)
  {
    if(prime[i] && n%i==0){
      ans=ans/i*(i-1);
      while(n%i==0&&n)
        n/=i;
    }
  }
  if (n!=1){
    ans=ans/n*(n-1);
  }
  return ans;
}

int main()
{
  prime_table();
  int in;
  while(~scanf("%d",&in)){
    printf("%d\n", eularphi(in));
  }
  return 0;
}
```

3.2 尤拉函數建表

```
// all numbers smaller than or equal to n.
#include<iostream>
using namespace std;
#define maxn 250000
// Computes and prints totien of all numbers
// smaller than or equal to n.
void eularphi_table(int n)
{
```

```
  // Create and initialize an array to store
  // phi or totient values
  long long phi[n+1];
  for (int i=1; i<=n; i++)
    phi[i] = i; // indicates not evaluated yet
          // and initializes for product
          // formula.

  // Compute other Phi values
  for (int p=2; p<=n; p++)
  {
    // If phi[p] is not computed already,
    // then number p is prime
    if (phi[p] == p)
    {
      // Phi of a prime number p is
      // always equal to p-1.
      phi[p] = p-1;

      // Update phi values of all
      // multiples of p
      for (int i = 2*p; i<=n; i += p)
      {
      // Add contribution of p to its
      // multiple i by multiplying with
      // (1 - 1/p)
      phi[i] = (phi[i]/p) * (p-1);
      }
    }
  }

  // Print precomputed phi values
  for (int i=1; i<=n; i++)
  cout << "Totient of " << i << " is "<< phi[i] << '\n'
      ;
}

int main()
{
  freopen("o.out","w",stdout); //for test
  int n = maxn;
  eularphi_table(n);
  return 0;
}
```

3.3 Fibonacci 線上

```
#include <iostream>
#include <cstring>
using namespace std;
#define LL long long
//注意，f0=1,f1=1,f2=2...
const LL mod=1e9+7; // 避免數值過大造成 overflow，因此將所有數值都 mod 10^9+7

struct Matrix  {
  LL a[2][2];
  void all_0() // 清空矩陣
  {
    memset(a, 0, sizeof(a));
  }
  void I() // 讓矩陣變成單位方陣
  {
    a[0][0]=1; a[0][1]=0;
    a[1][0]=0; a[1][1]=1;
  }
  void X() // 讓矩陣變成文章中的矩陣 A
  {
    a[0][0]=1; a[0][1]=1;
    a[1][0]=1; a[1][1]=0;
  }
};

Matrix operator*(const Matrix &a, const Matrix &b) // 矩陣乘法
{
  Matrix ret;
  ret.all_0();
  for (LL i=0; i<2; i++)  {
    for (LL j=0; j<2; j++)  {
      for (LL k=0; k<2; k++)  {
        ret.a[i][j]+=a.a[i][k]*b.a[k][j];
        ret.a[i][j]%=mod;
      }
    }
  }
  return ret;
}

Matrix power(Matrix a, LL n) // 快速冪
{
  Matrix ret;
  ret.I();
  if (n==0)  return ret;
  ret.X();
  if (n==1)  return ret;
  ret=power(a, n/2);
  ret=ret*ret;
  if (n%2==1)  ret=ret*a;
  return ret;
}

LL query(LL n)
{
  Matrix tmp;
  tmp.X();
  tmp=power(tmp, n);
  LL ret=tmp.a[1][0]+tmp.a[1][1]; // 因為初始的矩陣 X[0] 的兩個元的值都是 1，所以矩陣相乘的結果相當於把矩陣 A 下面的兩個元加起來
  ret%=mod;
  return ret;
}

int main()
{
  LL n;
  while (cin >> n)  {
    cout << query(n) << endl;
  }
  return 0;
}
```

3.4 質數

```
#define maxn 46340
//bcz sqrt(2^31-1)~=46340.95 and 46341 46340 not prime
bool prime[maxn];
void prime_table(){
  memset(prime,true,sizeof(prime));
  prime[0]=prime[1]=false;
  for (int i = 2; i < maxn; ++i)
    if (prime[i])
      for (int j = i*i; j < maxn; j+=i)
        prime[j]=false;
    }
```

3.5 質數 linear

```
#include <bits/stdc++.h>
using namespace std;
#define N 1000000
long long int not_prime[N];
vector<long long int> prime;
void prime_sieve(){
    memset(not_prime,0,sizeof(not_prime));
    not_prime[1]=1;
    for(long long int i=2;i<N;i++){
```

```
        if(!not_prime[i]){
            prime.push_back(i);
        }
        for(long long int j=0;j<prime.size()&&i*prime[j]<N;j++){
            not_prime[i*prime[j]]=1;
            if(i%prime[j]==0)
                break;
        }
    }
}
```

3.6 模逆元

```
#include <stdio.h>
//ax == 1 (mod m),求x
struct gcdstruct {
    int d;
    int x;
    int y;
};

gcdstruct EXTENDED_EUCLID(int a, int b) {
    gcdstruct aa, bb;
    if (b == 0) {
        aa.d = a;
        aa.x = 1;
        aa.y = 0;
        return aa;
    } else {
        bb = EXTENDED_EUCLID(b, a % b);
        aa.d = bb.d;
        aa.x = bb.y;
        aa.y = bb.x - bb.y * (a / b);
    }
    return aa;
}


 int inverse(int a, int m) {
     int x;
     gcdstruct aa;
     aa = EXTENDED_EUCLID(a, m);
     return aa.x;
 }
 int main() {
     int a, m;
     scanf("%d %d",&a,&m);

     printf("%d\n",inverse(a, m));
     getchar();
     return 0;
 }
```

3.7 擴展歐基里德

```
#include <bits/stdc++.h>
#define ll long long
//ax+by==gcd(a,b)
//ax+by==1
//ax+by==c
ll gcdEx(ll a, ll b, ll &x, ll &y)
{
    if(b==0)
    {
        x = 1,y = 0;
        return a;
    }
    else
    {
        ll r = gcdEx(b, a%b, x, y); /* r = GCD(a, b) = GCD(b, a%b) */
        ll t = x;
        x = y;
        y = t - a/b * y;
        return r;
    }
}
void extendex(ll m,ll n ,ll &x, ll &y ){
  //printf("%d %d %d %d\n",m,n,x,y );
  x%=n;
  y%=m;
}
int main(int argc, char const *argv[])
{
    ll m,m_ans,n,n_ans,c;
    scanf("%lld %lld %lld",&m,&n,&c);
    m_ans=m;
    n_ans=n;
    gcdEx(m,n,m_ans,n_ans);
    m_ans*=c;
    n_ans*=c;
    extendex(m,n,m_ans,n_ans);
    printf("%lld %lld\n",m_ans,n_ans );
    return 0;
}
```

3.8 中國剩餘定理-韓信點兵

```
#include <iostream>

using namespace std;
//参数可为负数的扩展欧几里德定理
void exOJLD(int a, int b, int & x, int & y) {
    //根据欧几里德定理
    if (b == 0) { //任意数与0的最大公约数为其本身。
        x = 1;
        y = 0;
    } else {
        int x1, y1;
        exOJLD(b, a % b, x1, y1);
        if (a * b < 0) { //异号取反
            x = -y1;
            y = a / b * y1 - x1;
        } else { //同号
            x = y1;
            y = x1 - a / b * y1;
        }
    }
}
//剩余定理
int calSYDL(int a[], int m[], int k) {
    int N[k]; //这个可以删除
    int mm = 1; //最小公倍数
    int result = 0;
    for (int i = 0; i < k; i++) {
        mm *= m[i];
    }
    for (int j = 0; j < k; j++) {
        int L, J;
        exOJLD(mm / m[j], -m[j], L, J);
        N[j] = m[j] * J + 1; //1
        N[j] = mm / m[j] * L; //2 1和2这两个值应该是相等的。
        result += N[j] * a[j];
    }
    return (result % mm + mm) % mm; //落在(0, mm)之间，这么写是为了防止result初始为负数，本例中不可能为负可以直接 写成：return result%mm;即可。
}

int main() {
    int a[3] = {2,3,2}; //a[i]=n%m[i]
    int m[3] = {3,5,7};
    cout << calSYDL(a, m, 3) << endl;
    return 0;
}
```

4 Dynamic Programming

4.1 LIS

```
if(lis.size()==0||tmp>lis.back())
    lis.push_back(tmp);
else
    *lower_bound(lis.begin(),lis.end(),tmp)=tmp;
```

4.2 LCS

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

using namespace std;

int arr[1500][1500];

void LCS(string str1, string str2){

  memset(arr, 0, sizeof(arr));
  for (int i = 1; i <= str1.size(); i++){

    for (int j = 1; j <= str2.size(); j++){

      if (str1[i - 1] == str2[j - 1]){

        arr[i][j] = arr[i - 1][j - 1] + 1;
      }
      else{

        arr[i][j] = max(arr[i - 1][j], arr[i][j - 1]);
      }
    }
  }
}
```

4.3 Minimum Coin Change

```
#include <bits/stdc++.h>
using namespace std;

#define maxn 100000
#define maxm 100000

int coins[maxn];
int table[maxm];
int n, target;

int minCoins(int n, int tar)
{
    table[0] = 0;

    for (int i = 1; i <= tar; i++)
        table[i] = 1e9;

    // Compute minimum coins required for all
    // values from 1 to V
    for (int i = 1; i <= tar; i++)
    {
        // Go through all coins smaller than i
        for (int j=  0; j < n; j++)
            if (coins[j] <= i)
            {
                int sub_res = table[i-coins[j]];
                if (sub_res != 1e9 && sub_res + 1 <
                    table[i])
                    table[i] = sub_res + 1;
            }
    }
}

int main(void)
{
    while(cin >> n >> target)
    {
        for(int i = 0; i < n; i++)
        {
            cin >> coins[i];
        }

        minCoins(n, target);
    }
}
```

4.4 Coin Change

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

using namespace std;

int coin[] = {1, 5, 10, 25, 50};
int arr[100000];

void build(){

  memset(arr, 0, sizeof(arr));
  arr[0] = 1;
  for (int i = 0; i < 5; i++){

    for (int j = 1; j < 10000; j++){

      if (j >= coin[i]){

        arr[j] = arr[j] + arr[j - coin[i]];
      }
    }
  }
}
```

4.5 2D Maximum SubArray

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

using namespace std;

#define size 4

int arr[size][size];

int maxSubArr(){

  int b[size];
  int MAX = -11111111;

  for(int i = 0 ; i < size; i++){

    memset(b, 0, sizeof(b));
    for(int j = i ; j < size ; j++){

      int s = 0;
      for(int k = 0 ; k < size ; k++){

        b[k] += arr[j][k];
        s += b[k];
        if(s <= 0)
          s = b[k];
        if(s > MAX)
          MAX = s;
      }
    }
  }
  return MAX;
}

int main(){
```

```
35
36    #ifdef DBG
37    freopen("1.in", "r", stdin);
38    freopen("2.out", "w", stdout);
39      #endif
40
41    for(int i = 0 ; i < size ; i++)
42      for(int j = 0 ; j < size ; j++)
43        cin >> arr[i][j];
44
45    maxSubArr();
46
47    return 0;
48  }
```

5 Data Structure

5.1 Big Number

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  #define ll long long
6
7  const int size = 1000;
8  const int carrySys = 10;
9
10
11 struct BigNum{
12
13   int len;
14   int bgNum[size];
15   bool sign;
16
17   void reset(){
18
19     len = 1;
20     memset(bgNum, 0, sizeof(bgNum));
21   }
22
23
24   BigNum add(const BigNum lhs, const BigNum rhs){
25
26     BigNum sum;
27     sum.reset();
28
29     int l = std::max(rhs.len, lhs.len);
30
31     for (int i = 0; i < l; i++)
32     {
33
34       sum.bgNum[i] += lhs.bgNum[i] + rhs.bgNum[i];
35       if (sum.bgNum[i] >= carrySys)
36       {
37
38         sum.bgNum[i + 1]++;
39         sum.bgNum[i] -= carrySys;
40       }
41     }
42     if (sum.bgNum[l])
43       l++;
44     sum.len = l;
45
46     if (!lhs.sign && !rhs.sign)
47       sum.sign = false;
48     else
49       sum.sign = true;
50
51     return sum;
52   }
53
54   BigNum sub(const BigNum lhs, const BigNum rhs, bool s
       ){
55
56     BigNum ans;
57     ans.reset();
58
59     int l = max(rhs.len, lhs.len);
60     int tmp[size];
61     memset(tmp, 0, sizeof(tmp));
62     copy(lhs.bgNum, lhs.bgNum + lhs.len, tmp);
63     for (int i = 0; i < l; i++)
64     {
65
66       if (tmp[i] < rhs.bgNum[i] && i != l - 1)
67       {
68
69         tmp[i + 1] -= 1;
70         tmp[i] += carrySys;
71       }
72       ans.bgNum[i] = tmp[i] - rhs.bgNum[i];
73     }
74
75     if (ans.bgNum[l - 1] < 0)
76     {
77
78       ans.bgNum[l - 1] = abs(ans.bgNum[l - 1]);
79       ans.sign = false;
80     }
81     else
82       ans.sign = true;
83
84     ans.len = l;
85
86     while (ans.len > 1 && !ans.bgNum[ans.len - 1])
87     {
88
89       ans.len--;
90     }
91
92     ans.sign = s;
93
94     return ans;
95   }
96   void intToBigNum(ll x){
97
98     if(x < 0){
99       sign = false;
100      x *= -1;
101    }
102    else
103      sign = true;
104
105    reset();
106    if(x == 0)
107      return;
108
109    len = 0;
110    while(x){
111
112      bgNum[len++] = x % 10;
113      x /= 10;
114    }
115  }
116  void strToBigNum(char x[]){
117
118    reset();
119    len = strlen(x);
120    int l = 0;
121    int a = -1;
122    if(x[0] == '-'){
123
124      sign = false;
125      a++;
126    }
127    else{
128
129      sign = true;
130    }
131
```

```
    for(int i = len-1 ; i > a ; i--){

      bgNum[l++] = x[i] - '0';
    }
    if(!sign){

      len--;
    }
  }

  void strToBigNum(string x){

    reset();

    if(x[0] == '-')
      sign = false;
    else
      sign = true;

    reverse(x.begin(), x.end());
    len = x.size();
    if(!sign)
      len--;

    for(int i = 0 ; i < len ; i++){

      bgNum[i] = x[i]-'0';
    }
  }

  BigNum operator+(const BigNum &rhs){

    BigNum a = *this;
    BigNum b = rhs;

    if(sign && rhs.sign)
      return add(*this, rhs);
    else if(!sign && rhs.sign){

      a.sign = true;
      return (a > b ? sub(a, b, false) : sub(b, a, true
          ));
    }
    else if(sign && !rhs.sign){

      b.sign = true;
      return (a > b ? sub(a, b, true) : sub(b, a, false
          ));
    }
    else//!sign && !rhs.sign
      return add(*this, rhs);

  }

  BigNum operator-(const BigNum &rhs){

    BigNum a = *this;
    BigNum b = rhs;

    if(sign && rhs.sign)
      return ((*this >= rhs) ? sub(*this, rhs, true) :
          sub(rhs, *this, false));
    else if(!sign && rhs.sign){

      b.sign = false;
      return add(a, b);
    }
    else if(sign && !rhs.sign){

      b.sign = true;
      return add(a, b);
    }
    else{

      a.sign = true;
      b.sign = true;
      if(a > b){

        return sub(a, b, false);
      }
      else{

        return sub(b, a, true);
      }
    }
  }

  // BigNum operator * (const BigNum &rhs){

  //  // cout<< "mul" << endl;
  //  BigNum ans;
  //  ans.reset();
  //  for(int i = 0 ; i < len ; i++){

  //    for(int j = 0 ; j < rhs.len ; j++){


  //      int l = i + j;
  //      ans.bgNum[l] += bgNum[i] * rhs.bgNum[j];
  //      while(ans.bgNum[l] >= carrySys){

  //        ans.bgNum[l+1] += ans.bgNum[l] / carrySys;
  //        ans.bgNum[l] = ans.bgNum[l] % carrySys;
  //      }
  //    }
  //  }

  //  ans.len = len + rhs.len;
  //  if(!ans.bgNum[ans.len-1]){

  //    ans.len--;
  //  }
  //  return ans;
  // }

  friend bool operator < (const BigNum &lhs, const
      BigNum &rhs){

    // cout << lhs.len << rhs.len << endl;
    if(lhs.sign < rhs.sign)
      return true;
    else if(lhs.sign > rhs.sign)
      return false;
    else{

      if(lhs.len < rhs.len)
        return true;
      else if(lhs.len == rhs.len){

        for(int i = 0 ; i < lhs.len ; i++){

          if(lhs.bgNum[i] < rhs.bgNum[i])
            return true;
        }
        return false;
      }
      else
        return false;
    }
  }

  friend bool operator > (const BigNum &lhs, const
      BigNum &rhs){

    if(lhs.sign > rhs.sign)
      return true;
    else if(lhs.sign < rhs.sign)
      return false;
    else{

      if (lhs.len > rhs.len)
        return true;
      else if (lhs.len == rhs.len){
```

```
        for (int i = 0; i < lhs.len; i++){

          if (lhs.bgNum[i] > rhs.bgNum[i])
            return true;
        }
        return false;
      }
      else
        return false;
    }

  }

  friend bool operator >= (const BigNum &lhs, const
      BigNum &rhs){

    return !(lhs < rhs);
  }

  friend bool operator <= (const BigNum &lhs, const
      BigNum &rhs){

    return !(lhs > rhs);
  }
  BigNum operator = (const BigNum &rhs){

    len = rhs.len;
    copy(rhs.bgNum, rhs.bgNum+rhs.len, bgNum);
    sign = rhs.sign;
  }

  friend ostream& operator<<(ostream &out, const BigNum
       &num){

    if(!num.sign){

      cout << "-";
    }

    out << num.bgNum[num.len-1];
    for(int i = num.len-2 ; i >= 0 ; i--)
      out << num.bgNum[i];
    return out;
  }

  BigNum(){ reset(); }
  BigNum(int x){  reset(); intToBigNum(x);  }
  BigNum(ll x){ reset(); intToBigNum(x); }
  BigNum(string x){

    reset();
    strToBigNum(x);
  }
  BigNum(char x[]){

    reset();
    strToBigNum(x);
  }

};

int main(){

  #ifdef DBG
  freopen("1.in", "r", stdin);
  freopen("2.out", "w", stdout);
  #endif // DEBUG


  // char a[] = "12345";
  // char b[] = "-2345";

  // BigNum x = a;
  // BigNum y = b;

  // cout << x << " " << y << endl;

  string a, b;

  while(cin >> a >> b){

    BigNum x, y;
    // cout << "aaa: ";
    x = a;
    y = b;

    BigNum z = x - y;
    cout << z << endl;
    // cout << z.len << endl;
  }


  return 0;
}
```

5.2 Disjoin Set

```
#define SIZE 10000

int arr[SIZE];

void init(int n) // give a initial length
{
  for(int i=0; i<n; i++)
    arr[i] = -1;
}

int find(int x)
{ // find the father point
  return arr[x] < 0 ? x : (arr[x] = find(arr[x])); //
      update every child to the other father
}

void Union(int x, int y)
{
  x = find(x);
  y = find(y);

  if(x == y)
    return;

  if(arr[x] <= arr[y])
  {
    arr[x] += arr[y];
    arr[y] = x;
  }
  else
  {
    arr[y] += arr[x];
    arr[x] = y;
  }
}
```

5.3 Segment Tree

```
#define SIZE 100000

int st[SIZE];
int st_val[SIZE];

void st_build(int *st, int *st_val, int now, int ls,
    int rs)
{
  if(ls == rs)
    st[now] = st_val[ls];
  else
  {
    st_build(st, st_val, now*2, ls, (ls+rs)/2);
    st_build(st, st_val, now*2+1, (ls+rs)/2+1, rs);
```

```
    st[now] = max(st[now*2], st[now*2+1]);
  }
}

// ls and rs are query range, begin and end is whole st[] range
int query(int now, int ls, int rs, int begin, int end)
{
  int mid = (begin+end)/2;
  int ret = 0;

  if(ls <= begin && rs >= end)
    return st[now];

  // it is find max now (modify here)
  if(ls <= mid)
    ret = max(ret, query(now*2, ls, rs, begin, mid));

  if(rs > mid)
    ret = max(ret, query(now*2+1, ls, rs, mid+1, end));

  return ret;
}
```

6 Others

6.1 Roman to Int

```
unordered_map<char, int> value{{'I', 1}, {'V', 5}, {'X', 10}, {'L', 50}, {'C', 100}, {'D', 500}, {'M', 1000}};

int romanToInt(string s){

  if(s.empty())
    return 0;

  int maxDigit = -1;
  int ans = 0;
  for(int i = s.size()-1 ; i >= 0 ; i--){

    const int current = value[s[i]];
    if(current >= maxDigit){

      ans += value[s[i]];
      maxDigit = current;
    }
    else{

      ans -= value[s[i]];
    }
  }
  return ans;
}
```

6.2 misc

```
STL func 用法

priority_queue<int> p;
priority_queue<int> //大到小，預設
priority_queue<int, vector<int>, greater<int> > //小到大

map:

查找不重複資料數量:用map實作，直接讀map.size()
map.count(key)用來判斷key有無對應val，有1無0

vector:

vector大小相關
  預開: v.reserve(N);
    vector<int> bar (5,0); 五個0
  myvector.capacity()//看目前最大容量

insert:
  v.insert (v.begin()+i,av.begin(),av.end()); //把整個av插入v[i]左邊
   it = myvector.begin();
   it = myvector.insert ( it , 200 ); //插入完後指向插完的左邊
    v.insert (v.begin(),2,300);  //插兩個300 在v.begin()左邊


list:(隨機存取慢，增減資料快)
  insert:與vector相同

  sort: l.sort()  (預設小到大)，可自放cmp
    bool mycomparison (double first, double second)
    { return ( int(first)<int(second) ); }
  unique: l.unique()  合併重複項，可自定義(cmp)
    bool same_integral_part (double first, double second)
    { return ( int(first)==int(second) ); }

  merge:  first.merge(second);    //second會清空，會排序，可自定義排序方式


string：
  string str="We think in generalities, but we live in details.";
    string str2 = str.substr (3,5);     // "think"
    size_t pos = str.find("live");      // position of "live" in str
    string str3 = str.substr (pos);     // get from "live" to the end
    cout << str2 << ' ' << str3 << '\n';
    'think live in details.'


math function 用法：
#define PI 3.14159265
atan2 (y,x) * 180 / PI; //給x,y座標回傳角度
cos ( x * PI / 180.0 ); //cos(x)==?
log();
log10();
```

6.3 Recent update

```
//uva 1648
//設往上x次，則往下n-x次(n次移動)
//level=xu-(n-x)d大等0
//所以(u+d)大等nd min(x)=nd/(u+d)
//否則x=nd/(u+d)+1
//整除時多上少下一次 答案u+d
#include <bits/stdc++.h>
using namespace std;
int main(int argc, char const *argv[])
{

  int n,m,t1,t2;
  while(~scanf("%d %d",&n,&m)){
    std::vector<int> up;
    std::vector<int> down;
    for (int i = 0; i < m; ++i)
    {
      scanf("%d %d",&t1,&t2);
      up.push_back(t1);
      down.push_back(t2);
    }
    int minn=(1<<31-1);
    for (int i = 0; i < up.size(); ++i)
```

```
    {
      if ((n*down[i])%(up[i]+down[i])==0)
      {
        minn=min(minn,up[i]+down[i]);
      }
      else{
        int tmp =(n*down[i])/(up[i]+down[i])+1;
        minn=min(minn,tmp*up[i]-(n-tmp)*down[i]);
      }
    }
    printf("%d\n",minn );
  }
  return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define maxn 1000000000000000
#define MP make_pair
#define PLILI pair<long long int,long long int>
ll m;
ll compose(ll n, ll k){
  ll x=1;
    for(int i=1;i<=k;i++)
    {
        if(x/i>m/(n-i+1)) return m+1; //直接視為超大數(超越m)
        x*=n-i+1;           //促使bin_serach往右走
        x/=i;              //怕OF 原本形式為x*(n-i+1)>m
    }                  //L13,14玄學，一定能除盡
  return x;
}

ll bio(ll m,ll k){
  ll l = 2*k;
  ll r = m;
  ll mid = (l+r)/2;

  while(l<=r){
    // printf("call compose where l=%d mid=%d r=%d k==%d\n",l,mid,r,k);
    ll tmp =compose(mid,k);
    if (tmp<m)
    {
      l=mid+1;
      mid=(l+r)/2;
    }
    else if (tmp>m){
      r=mid-1;
      mid=(l+r)/2;
    }
    else{
      return mid;
    }
  }
  return -1;

}

int main(int argc, char const *argv[])
{
  int t;
  scanf("%d",&t);
  while(t--){
    scanf("%lld",&m);
    vector<PLILI> v;
    for (ll i = 1; ; ++i)
    {
      if (compose(i*2,i)>m)
        break;
      ll tmp = bio(m,i);

      if (tmp==-1)
        continue;
      v.push_back(MP(tmp,i));
      if ((tmp-i)!=i)
      {
        v.push_back(MP(tmp,tmp-i));
      }

    }
    sort(v.begin(), v.end());
    printf("%d\n",v.size() );
    for (int i = 0; i < v.size(); ++i)
    {
      if(i==0)
        printf("(%lld,%lld)",v[i].first,v[i].second);
      else
        printf(" (%lld,%lld)",v[i]. first,v[i].second);
    }
    printf("\n");

  }
  return 0;
}
```

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

using namespace std;

#define PROBLEM "1650"
#define mod 1000000007

string str;
long long dp[1005][1005];
long long sum[1005][1005];
int len = 0;

void build(){

  len = str.size()+1 ;

  dp[1][1]  = 1;
  sum[1][1] = 1;

  for(int i = 2 ; i <= len ; i++){

    for(int j = 1 ; j <= i ; j++){

      if(str[i-2] == 'D' || str[i-2] == '?'){

        dp[i][j] = (dp[i][j] + sum[i-1][i-1] - sum[i-1][j-1]) % mod;
        // for(int k = j ; k < i ; k++){

        //  dp[i][j] += dp[i-1][k];
        //  dp[i][j] %= mod;
        // }
      }
      if(str[i-2] == 'I' || str[i-2] == '?'){

        dp[i][j] = (dp[i][j] + sum[i-1][j-1]) % mod;
      }
      sum[i][j] = (dp[i][j] + sum[i][j-1]) % mod;
    }
  }
}

int main(){

  #ifdef DBG
  freopen("UVA" PROBLEM ".in", "r", stdin);
  freopen("UVA" PROBLEM ".out", "w", stdout);
  #endif

  while(cin >> str){

    memset(dp, 0, sizeof(dp));
    memset(sum, 0, sizeof(sum));
    build();
    cout << (sum[len][len]+mod)%mod << endl;
  }

  return 0;
```

```
}


//dijk第二長路徑
#include <bits/stdc++.h>
using namespace std;
#define maxn 105
#define MP make_pair
#define PII pair<int,int>
int dis[maxn][maxn][2];
vector<PII> e[maxn];

void dijk(int cur){

  int d ,st = cur;
  priority_queue<PII,vector<PII>,greater<PII> > q;

  q.push( MP(0,cur) );

  while(!q.empty()){
    tie(d,cur) = q.top();
    q.pop();
    if (dis[st][cur][0]==1e9){
      dis[st][cur][0]=d;
    }
    else{
      if (d<dis[st][cur][0])
      {
        dis[st][cur][1]=dis[st][cur][0];
        dis[st][cur][0]=d;
      }
      else if (dis[st][cur][0]<d && d<dis[st][cur][1])
      {
        dis[st][cur][1]=d;
      }
      else{
        continue;
      }
    }

    for (auto i : e[cur])
    {
      q.push(MP(d+i.second,i.first));
    }
  }
}

void init(void){
  for (int i = 0; i < maxn; ++i)
  {
    for (int j = 0; j < maxn; ++j)
    {
      dis[i][j][0]=1e9;
      dis[i][j][1]=1e9;
    }
  }

  for (int i = 0; i < maxn; ++i)
  {
    e[i].clear();
  }
}

int main(int argc, char const *argv[])
{
  int dots,roads,tes=1;
  int x,y,d;
  while(~scanf("%d %d",&dots,&roads)){
    printf("Set #%d\n",tes++ );
    init();
    for (int i = 0; i < roads; ++i)
    {
      scanf("%d %d %d",&x,&y,&d);
      e[x].push_back(MP(y,d));
      e[y].push_back(MP(x,d));
    }

    for (int i = 0; i < dots; ++i)
    {
      dijk(i);
    }

    // for (int i = 0; i < dots; ++i)
    // {
    //  printf("%d %d == %d\n",0,i,dis[0][i] );
    // }
    int ques,q;
    scanf("%d",&ques);
    for (int i = 0; i < ques; ++i)
    {
      scanf("%d %d",&x,&y);

      if (dis[x][y][1] == 1e9)
      {
        printf("?\n");
      }
      else{
        printf("%d\n",dis[x][y][1] );
      }
    }

  }
  return 0;
}
```

```
#include<cstdio>
#include<cmath>
using namespace std;
const double g=9.81;
//k 拉力常數 l 繩長 s 高度 m 重量
int main(){
    double k,l,s,m;
    while(~scanf("%lf%lf%lf%lf",&k,&l,&s,&m)&&(k||l||s
        ||m)){
        double dtl=(m*g+sqrt(m*m*g*g+2.0*k*l*m*g))/k;
        if(s>l+dtl) puts("Stuck in the air.");
        else{
            double v=2.0*g*s;
            if(s>l) v-=k*(s-l)*(s-l)/m;
            if(v<=100.0) puts("James Bond survives.");
            else puts("Killed by the impact.");
        }
    }
    return 0;
}
```

```
// h_prime 4n+1是質數 求範圍內兩質相乘之合數總和
#include <bits/stdc++.h>
using namespace std;
#define maxn 1000001
bool vis[maxn+5];
bool h_semi[maxn+5];
int prefix_sum[maxn+5];
long long int pri[maxn+5],pn=0;
void h_table(){
  memset(vis,false,sizeof(vis));
  memset(h_semi,false,sizeof(h_semi));
  for (int i = 5; i <= maxn; i += 4){
    if (vis[i]) continue;
    pri[pn++] = i;
    for (int j = i; j <= maxn; j += i){
      vis[j] = true;
    }
  }

  for (int i = 0; i < pn; ++i)
  {
    if (pri[i]*pri[i]>maxn)
      break;
    for (int j = i; j <pn ; ++j)
    {
      if (pri[i]*pri[j]>maxn)
        break;
      h_semi[pri[i]*pri[j]]=true;
    }
```

```
  }
  for (int i = 1; i <=maxn ; ++i)
    prefix_sum[i]=prefix_sum[i-1]+(h_semi[i]?1:0);
}


int main(int argc, char const *argv[])
{
  h_table();
  int in;

  while(~scanf("%d",&in)&&in){
    printf("%d %d\n",in,prefix_sum[in]);
  }
  return 0;
}
```

```
// N_Queen(0,num);
int Queen[37000][14];
int tmp[14];
int cnt;
bool row[14], L[27], R[27];

void N_Queen(int k, int Num){
  if(k == Num){
    for(int j = 0; j < Num; j++)
      Queen[cnt][j] = tmp[j];
    cnt++;
    return;
  }
  for(int i = 0; i < Num; i++) {
    int right= k+i, left= k-i+Num-1;
    if(!row[i] && !L[left] && !R[right]) {
      row[i] = L[left] = R[right] = true;
      tmp[k]=i;
      N_Queen(k+1, Num);
      row[i] = L[left] = R[right] = false;
    }
  }
}
```