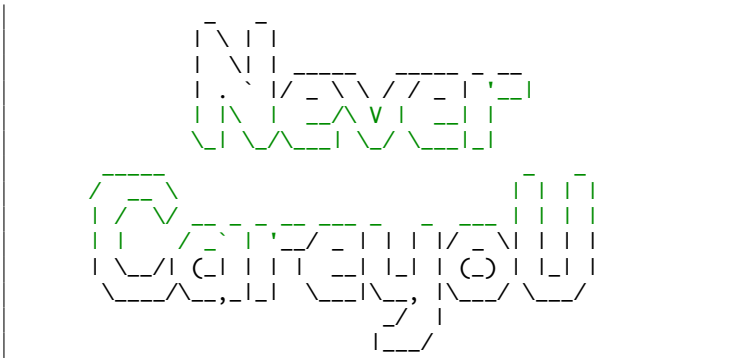
Contents



1 Setup

1.1 設定

```
Ctrl+Alt+T
vim ~/.vimrc

syntax on
set nu
set tabstop=4
set shiftwidth=4
colo evening
set mouse=a
set cin
inoremap ( ()<LEFT>
inoremap [ []<LEFT>
inoremap { {}<LEFT>
inoremap " ""<LEFT>
inoremap ' ''<LEFT>

vim a.cpp
g++ A.cpp -o A
./A
```

1.2 Default

```cpp
#pragma GCC optimize("O2")
#pragma GCC optimize("unroll-loops")//常數優化
#include<bits/stdc++.h>
#define IO cin.tie(0);cout.tie(0);ios_base::
    sync_with_stdio(false)
#define ll long long
#define vt vector
#define pb push_back
#define pll pair<ll,ll>
#define F first
#define S second
#define INF 0x3f3f3f3f //接近9
#define INFF 0x3f3f3f3f3f3f3f3f //接近19
#define PI acos(-1) //等同圓周率
//unsigned ll 0~18,446,744,073,709,551,615
//大數交給python
//解題數>penalty(時間+錯的次數*20)
//每題都要看過
using namespace std;
int main()
{
    IO;
    memset(arr,0,sizeof arr);//放0 -1 0x3f(接近10^9)
    max({a,b,c});//找三個值的最大值
    sort(v.begin(),v.end(),[](int lhs,int rhs){
        return lhs<rhs;
    })//lambda
    for(auto i:vec)  cout<<i<<'\n';
    for(auto &i:vec)    i++;//修改值要加&
    if(x&1) cout<<"odd";
    else    cout<<"even";
    x<<=1;//x*=2
    cerr<<"debug"<<__LINE__<<" line";//在程式碼第幾行
    //只會在cmd輸出 不會再std output 不會wa
    assert(x<=5);//如果條件不成立 會RE
    //可以用來猜測資
    return 0;
}
```

2 Basic

2.1 Binary Search

```cpp
while (l < r)
{
    int mid = l + r >> 1;
    if (q[mid] >= x) r = mid;
    else l = mid + 1;
}
while (l < r)
{
    int mid = l + r + 1 >> 1;
    if (q[mid] <= x) l = mid;
    else r = mid - 1;
}
```

2.2 merge sort

```cpp
ll q[MAXN],tmp[MAXN];
void merge_sort(int q[], int l, int r)
{
    if (l >= r) return;
    int mid = l + r >> 1;
    merge_sort(q, l, mid);//
    merge_sort(q, mid + 1, r);//
    // LL res = merge_sort(q, l, mid) + merge_sort(q,
        mid + 1, r);  //逆序數對
    int k = 0, i = l, j = mid + 1;
    while (i <= mid && j <= r)
        if (q[i] <= q[j])
            tmp[k ++ ] = q[i ++ ];
        else
        {
```

```
            // res += mid - i + 1; //逆序數對
            tmp[k ++ ] = q[j ++ ];
        }
    while (i <= mid)
        tmp[k ++ ] = q[i ++ ];
    while (j <= r)
        tmp[k ++ ] = q[j ++ ];
    for (i = l, j = 0; i <= r; i ++, j ++ )
        q[i] = tmp[j];
    // return res; //逆序數對
}
```

2.3 四舍五入

```
cout << fixed << setprecision(2) << x << "\n";
//四舍五入到小數點後第2位
//fixed 取消科學記號
cout << ceil(x) << "\n";//無條件進位
cout.unsetf(ios::fixed);//關閉固定小數位格式
cout<<x<<endl;
```

2.4 排序

```
priority_queue< int, vector<int>, less<int> > pq;
//降排序
priority_queue< int, vector<int>, greater<int> > pq;
//升排序

less<type>()    //从小到大排序 <
grater<type>()  //从大到小排序 >
less_equal<type>()  //  <=
gtater_equal<type>()//  >=

int numbers[]={20,40,50,10,30};
sort(numbers, numbers+5, greater<int>());
```

2.5 nth element

```
求[first,last]區間第n大小的元素 O(N)
-----------------------------------------------------
int array[] = {5,6,7,8,4,2,1,3,0};
nth_element(array, array+6, array+len);
// 4 0 3 1 2 5 6 7 8
```

3 flow

3.1 dinic

```
1. O(V^2*E) 點<1000 -> dp or flow
//flow在二分圖O(E*sqrt(V)) V,E<=2e5
#define pb push_back
#define SZ(x) ((int)x.size())
const ll MAXN = 點的數量;
-----------------------------------------------------
2. code
flow.init(點數,源點,匯點);
flow.add_edge(from,to,最大流量);
flow.flow(); //回傳int型態 該圖最大流量
-----------------------------------------------------
struct Dinic{
  struct Edge{ int v,f,re; };
  int n,s,t,level[MAXN];
  vector<Edge> E[MAXN];
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void add_edge(int u, int v, int f){
    E[u].pb({v,f,SZ(E[v])});
    E[v].pb({u,0,SZ(E[u])-1});
  }
  bool BFS(){
    for (int i=0; i<n; i++) level[i] = -1;
    queue<int> que;
    que.push(s);
    level[s] = 0;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (auto it : E[u]){
        if (it.f > 0 && level[it.v] == -1){
          level[it.v] = level[u]+1;
          que.push(it.v);
    } } }
    return level[t] != -1;
  }
  int DFS(int u, int nf){
    if (u == t) return nf;
    int res = 0;
    for (auto &it : E[u]){
      if (it.f > 0 && level[it.v] == level[u]+1){
        int tf = DFS(it.v, min(nf,it.f));
        res += tf; nf -= tf; it.f -= tf;
        E[it.v][it.re].f += tf;
        if (nf == 0) return res;
    } }
    if (!res) level[u] = -1;
    return res;
  }
  int flow(int res=0){
    while ( BFS() )
      res += DFS(s,2147483647);
    return res;
} }flow;
```

3.2 MinCostFlow

```
1. 加上#define
#define pb push_back
#define SZ(x) ((int)x.size())
typedef int Tcost;
const int MAXV = 20010;
const int INFf = 1000000;
const Tcost INFc  = 1e9;
-----------------------------------------------------
2. code
flow.init(點數,源點,終點);
flow.addEdge(from,to,最大流量,每單位流量花費);
pair<int,int> res = flow.solve();
//最大流量 及 在最大流量情形下的最小花費
-----------------------------------------------------
struct MinCostMaxFlow{
  struct Edge{
    int v, cap;
    Tcost w;
    int rev;
    Edge(){}
    Edge(int t2, int t3, Tcost t4, int t5)
    : v(t2), cap(t3), w(t4), rev(t5) {}
  };
  int V, s, t;
  vector<Edge> g[MAXV];
  void init(int n, int _s, int _t){
    V = n; s = _s; t = _t;
    for(int i = 0; i <= V; i++) g[i].clear();
  }
  void addEdge(int a, int b, int cap, Tcost w){
    g[a].push_back(Edge(b, cap, w, (int)g[b].size()));
    g[b].push_back(Edge(a, 0, -w, (int)g[a].size()-1));
  }
  Tcost d[MAXV];
  int id[MAXV], mom[MAXV];
  bool inqu[MAXV];
  queue<int> q;
  pair<int,Tcost> solve(){
    int mxf = 0; Tcost mnc = 0;
    while(1){
      fill(d, d+1+V, INFc);
      fill(inqu, inqu+1+V, 0);
      fill(mom, mom+1+V, -1);
      mom[s] = s;
      d[s] = 0;
      q.push(s); inqu[s] = 1;
      while(q.size()){
        int u = q.front(); q.pop();
        inqu[u] = 0;
        for(int i = 0; i < (int) g[u].size(); i++){
          Edge &e = g[u][i];
          int v = e.v;
          if(e.cap > 0 && d[v] > d[u]+e.w){
            d[v] = d[u]+e.w;
            mom[v] = u;
            id[v] = i;
            if(!inqu[v]) q.push(v), inqu[v] = 1;
      } } }
```

```
      if(mom[t] == -1) break ;
      int df = INFf;
      for(int u = t; u != s; u = mom[u])
        df = min(df, g[mom[u]][id[u]].cap);
      for(int u = t; u != s; u = mom[u]){
        Edge &e = g[mom[u]][id[u]];
        e.cap              -= df;
        g[e.v][e.rev].cap += df;
      }
      mxf += df;
      mnc += df*d[t];
    }
    return {mxf,mnc};
} }flow;
```

3.3 Kuhn Munkres 最大完美二分匹配

```
1. KM 二分圖最大權匹配 O(N^3) //點的數量
// 最小權完美二分圖匹配 每個邊權*-1 跑KM 再把return值
    *-1
// 只要看出題目是KM 直接建邊帶模板就會過
-----------------------------------------------------
2. code
graph.init(n); //左邊x有幾個點
graph.addEdge(x,y,w); //左邊x->右邊y 權重w
graph.solve(); //回傳完美匹配下最大總權重
-----------------------------------------------------
struct KM{ // max weight, for min negate the weights
  int n, mx[MXN], my[MXN], pa[MXN];
  ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
  bool vx[MXN], vy[MXN];
  void init(int _n) { // 1-based
    n = _n;
    for(int i=1; i<=n; i++) fill(g[i], g[i]+n+1, 0);
  }
  void addEdge(int x, int y, ll w) {g[x][y] = w;}
  void augment(int y) {
    for(int x, z; y; y = z)
      x=pa[y], z=mx[x], my[y]=x, mx[x]=y;
  }
  void bfs(int st) {
    for(int i=1; i<=n; ++i) sy[i]=INF, vx[i]=vy[i]=0;
    queue<int> q; q.push(st);
    for(;;) {
      while(q.size()) {
        int x=q.front(); q.pop(); vx[x]=1;
        for(int y=1; y<=n; ++y) if(!vy[y]){
          ll t = lx[x]+ly[y]-g[x][y];
          if(t==0){
            pa[y]=x;
            if(!my[y]){augment(y);return;}
            vy[y]=1, q.push(my[y]);
          }else if(sy[y]>t) pa[y]=x,sy[y]=t;
      } }
      ll cut = INF;
      for(int y=1; y<=n; ++y)
        if(!vy[y]&&cut>sy[y]) cut=sy[y];
      for(int j=1; j<=n; ++j){
        if(vx[j]) lx[j] -= cut;
        if(vy[j]) ly[j] += cut;
        else sy[j] -= cut;
      }
      for(int y=1; y<=n; ++y) if(!vy[y]&&sy[y]==0){
        if(!my[y]){augment(y);return;}
        vy[y]=1, q.push(my[y]);
  } } }
  ll solve(){
    fill(mx, mx+n+1, 0); fill(my, my+n+1, 0);
    fill(ly, ly+n+1, 0); fill(lx, lx+n+1, -INF);
    for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y)
      lx[x] = max(lx[x], g[x][y]);
    for(int x=1; x<=n; ++x) bfs(x);
    ll ans = 0;
    for(int y=1; y<=n; ++y) ans += g[my[y]][y];
    return ans;
} }graph;
```

4 Math

4.1 Binary exponentiation

```
O(logn)的時間計算a^b的方法(a^b%m)
-----------------------------------------------------
long long binpow(long long a,long long b){//long long m
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;//res = res * a % m;
        a = a * a;//a = a * a % m;
        b >>= 1;
    }
    return res;
}
```

4.2 Euclidean gcd

```
O(log min(a,b)) 找最大公因數
-----------------------------------------------------
int gcd (int a, int b) {
    return b ? gcd (b, a % b) : a;
}
```

4.3 lcm 最小公倍數

```
找最小公倍數
-----------------------------------------------------
int lcm (int a, int b) {
    return a / gcd(a, b) * b;
}
```

4.4 Miller Rabin

```
O(logN)判斷n是否是質數(回傳bool)
-----------------------------------------------------
1. 看n的範圍在哪, 把數字放進magic陣列
2. 改s的值(magic的陣列大小)
3. 會用到快速乘的模板(mul)(x*x%n)
4. 會用到快速冪模板(mypow)(a^u%n)
-----------------------------------------------------
// n < 4,759,123,141        3 :  2, 7, 61
// n < 1,122,004,669,633    4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383    6 :  pirmes <= 13
// n < 2^64  (ll 範圍)      7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n-2] if
// you want to use magic.
// 判斷質數
// 放x -> true or false (if x range 在 long long)
// O(logN)
-----------------------------------------------------
LL magic[]={}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n); //a^u%n
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n); //x*x%n
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size)
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

4.5 快速乘

```
O(1)算x*y%mod
-----------------------------------------------------
LL mul(LL x,LL y,LL mod){
  LL ret=x*y-(LL)((long double)x/mod*y)*mod;
  // LL ret=x*y-(LL)((long double)x*y/mod+0.5)*mod;
  return ret<0?ret+mod:ret;
}
```

5 Geometry

5.1 ConvexHull

```
凸包 O(NlogN)
-----------------------------------------------------
vt<Point> arr; //存入所有點
arr=convex_hull(arr); //順時鐘 凸包點
-----------------------------------------------------
struct Point{
  double x, y;
  Point operator+(const Point& _p){
    return {(x+_p.x), (y+_p.y)};
  }
  Point operator-(const Point& _p){
    return {(x-_p.x), (y-_p.y)};
  }
  double operator^(const Point& _p){
    return {_p.x*y-x*_p.y};
  }
  bool operator<(const Point& _p){
    if(x==_p.x) return y<_p.y;
    return x<_p.x;
  }
};
double cross(Point o, Point a, Point b){
  return (a-o) ^ (b-o);
}
vector<Point> convex_hull(vector<Point> pt){
  sort(pt.begin(), pt.end());
  int top = 0;
  vector<Point> stk(2*pt.size());
  for(int i = 0; i<(int)pt.size();i++){
    while(top>=2&&cross(stk[top-2], stk[top-1], pt[i])
        <=0)
        top--;
    stk[top++] = pt[i];
  }
  for(int i=pt.size()-2,t=top+1;i>=0;i--){
    while(top>=t&&cross(stk[top-2],stk[top-1],pt[i])
        <=0)top--;
    stk[top++]=pt[i];
  }
  stk.resize(top-1);
  return stk;
}
```

6 Graph

6.1 Strongly Connected Component

```
1. define
#define PB push_back
#define FZ(x) memset(x,0,sizeof(x))
const ll MXN = 1e5+10;
-----------------------------------------------------
2. code
// 有向環 連通塊
scc.init(n); //給點數
scc.addEdge(u,v); //u->v
scc.solve();
if(scc.bln[u]==scc.bln[v]) 表示u跟v是同一群
-----------------------------------------------------
struct Scc{
  int n, nScc, vst[MXN], bln[MXN];
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n){
    n = _n;
    for (int i=0; i<MXN; i++)
      E[i].clear(), rE[i].clear();
  }
  void addEdge(int u, int v){
    E[u].PB(v); rE[v].PB(u);
  }
  void DFS(int u){
    vst[u]=1;
    for (auto v : E[u]) if (!vst[v]) DFS(v);
    vec.PB(u);
  }
  void rDFS(int u){
    vst[u] = 1; bln[u] = nScc;
    for (auto v : rE[u]) if (!vst[v]) rDFS(v);
  }
  void solve(){
    nScc = 0;
    vec.clear();
    FZ(vst);
    for (int i=0; i<n; i++)
      if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    FZ(vst);
    for (auto v : vec)
      if (!vst[v]){
        rDFS(v); nScc++;
      }
  }
}scc;
```

6.2 BCC based on vertex

```
1. define
#define REP(x,y) for(ll x=0;x<y;x++)
#define PB push_back
const ll MXN = 1e5+10;
-----------------------------------------------------
2. code
// 無向圖 連通分量
graph.init(n); //n個點
graph.addEdge(a,b); //建邊
vt<vt<ll>> res=graph.solve(); //return 連通分量+橋
//連通分量/橋(只有兩個元素)/關節點(出現在兩個vt)
-----------------------------------------------------
struct BccVertex {
  int n,nScc,step,dfn[MXN],low[MXN];
  vector<int> E[MXN],sccv[MXN];
  int top,stk[MXN];
  void init(int _n) {
    n = _n; nScc = step = 0;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void addEdge(int u, int v)
  { E[u].PB(v); E[v].PB(u); }
  void DFS(int u, int f) {
    dfn[u] = low[u] = step++;
    stk[top++] = u;
    for (auto v:E[u]) {
      if (v == f) continue;
      if (dfn[v] == -1) {
        DFS(v,u);
        low[u] = min(low[u], low[v]);
        if (low[v] >= dfn[u]) {
          int z;
          sccv[nScc].clear();
          do {
            z = stk[--top];
            sccv[nScc].PB(z);
          } while (z != v);
          sccv[nScc++].PB(u);
        }
      }else
        low[u] = min(low[u],dfn[v]);
  } }
  vector<vector<int>> solve() {
    vector<vector<int>> res;
    for (int i=0; i<n; i++)
      dfn[i] = low[i] = -1;
    for (int i=0; i<n; i++)
      if (dfn[i] == -1) {
        top = 0;
        DFS(i,i);
      }
    REP(i,nScc) res.PB(sccv[i]);
    return res;
  }
}graph;
```

6.3 graph

```
稠密圖(m>=n^2) 用鄰接矩陣
稀疏圖(m<n^2) 用鄰接表(vt)

最短路:
1.單源最短路
    (1.) 所有權邊都是正數 -> Dijkstra O(mlogn)
```

```
    (2.) 有負權邊 -> Bellman-Ford O(nm)
    (3.) 有負權邊, 但🈚有負環 -> SPFA O(m)~O(nm)
2. 多源匯最短路
    (1.) 所有權邊都是正數 -> Floyd O(n^3)
```

6.4 bfs

```
vector<vector<int>> adj;  // adjacency list
    representation
int n; // number of nodes
int s; // source vertex

queue<int> q;
vector<bool> used(n);
vector<int> d(n), p(n);

q.push(s);
used[s] = true;
p[s] = -1;
while (!q.empty()) {
    int v = q.front();
    q.pop();
    for (int u : adj[v]) {
        if (!used[u]) {
            used[u] = true;
            q.push(u);
            d[u] = d[v] + 1;
            p[u] = v;
        }
    }
}
```

6.5 dfs

```
vector<vector<int>> adj; // graph represented as an
    adjacency list
int n; // number of vertices

vector<bool> visited;

void dfs(int v) {
    visited[v] = true;
    for (int u : adj[v]) {
        if (!visited[u])
            dfs(u);
    }
}
```

6.6 dijkstra(單源最短路徑)(堆優化O(mlogn))

```
#define INF 0x3FFFFFFF
typedef pair<int,int>PII;
const int MAXN = 100010;
vector<PII> G[MAXN];
void add_edge(int u,int v,int d){
    G[u].push_back(make_pair(v,d));
}
void init(int n){
    for(int i=0;i<n;i++)
        G[i].clear();
}
int vis[MAXN];
int dis[MAXN];
void dijkstra(int s,int n){
    for(int i=0;i<n;i++)vis[i] = 0;
    for(int i=0;i<n;i++)dis[i] = (i == s ? 0 : INF);
    priority_queue<PII,vector<PII>,greater<PII> >q;
  q.push(make_pair(dis[s],s));
    while(!q.empty()){
        PII p = q.top();
        int x = p.second;
        q.pop();
        if(vis[x])continue;
        vis[x] = 1;
        for(int i=0;i<G[x].size();i++){
            int y = G[x][i].first;
            int d = G[x][i].second;
            if(!vis[y]&&dis[x] + d < dis[y]){
                dis[y] = dis[x] + d;
                q.push(make_pair(dis[y],y));
            }
        }
    }
}
int main()
{
    cin>>n>>m;
    for(ll i=0,a,b,w;i<m;i++)
        cin>>a>>b>>w;add(a,b,w);
    dijkstra(1,n);
    for(ll i=1;i<=n;i++)    cout<<dis[i]<<" ";
    return 0;
}
```

6.7 bellman ford

```
#include<bits/stdc++.h>
using namespace std;
const int N=100010;
int dist[N],backup[N];
int k,n,m;
struct edge{
    int a;int b;int w;
}edge[N];
int bellman_ford()
{
    memset(dist,0x3f,sizeof dist);
    dist[1]=0;
    for(int i=1;i<=k;i++)
    {
        memcpy(backup,dist,sizeof dist);
        for(int j=1;j<=m;j++)
        {
            int a=edge[j].a,b=edge[j].b,w=edge[j].w;
            dist[b]=min(dist[b],backup[a]+w);
        }
    }
    return dist[n];
}
int main()
{
    cin>>n>>m>>k;
    for(int i=1;i<=m;i++)
    {
        int a,b,c;
        cin>>a>>b>>c;
        edge[i].a=a,edge[i].b=b,edge[i].w=c;
    }
    int t=bellman_ford();
    if(t>=0x3f3f3f3f/2)puts("impossible");
    else cout<<t<<endl;
}
```

6.8 SPFA 求最短路徑

```
// spfa判斷最短路
//给定一个 n 个点 m 条边的有向图，图中可能存在重边和自
    环， 边权可能为负数。
// 请你求出 1 号点到 n 号点的最短距离，如果无法从 1 号
    点走到 n 号点，则输出 impossible。
// 数据保证不存在负权回路。
// 3 3
// 1 2 5
// 2 3 -3
// 1 3 4
// =
// 2 //表示 1 号点到 n 号点的最短距离。
#include<iostream>
#include<queue>
#include<cstring>
using namespace std;

const int N=1e5+10;

#define fi first
#define se second

typedef pair<int,int> PII;//到源点的距离，下标号

int h[N],e[N],w[N],ne[N],idx=0;
int dist[N];//各点到源点的距离
bool st[N];
int n,m;
```

```
void add(int a,int b,int c){
    e[idx]=b;w[idx]=c;ne[idx]=h[a];h[a]=idx++;
}

int spfa(){
    queue<PII> q;
    memset(dist,0x3f,sizeof dist);
    dist[1]=0;
    q.push({0,1});
    st[1]=true;
    while(q.size()){
        PII p=q.front();
        q.pop();
        int t=p.se;
        st[t]=false;//从队列中取出来之后该节点st被标记为false,代表之后该节点如果发生更新可再次入队
        for(int i=h[t];i!=-1;i=ne[i]){
            int j=e[i];
            if(dist[j]>dist[t]+w[i]){
                dist[j]=dist[t]+w[i];
                if(!st[j]){//当前已经加入队列的结点，无需再次加入队列，即便发生了更新也只用更新数值即可，重复添加降低效率
                    st[j]=true;
                    q.push({dist[j],j});
                }
            }
        }
    }
    if(dist[n]==0x3f3f3f3f) return -1;
    else return dist[n];
}

int main(){
    scanf("%d%d",&n,&m);
    memset(h,-1,sizeof h);
    while(m--){
        int a,b,c;
        scanf("%d%d%d",&a,&b,&c);
        add(a,b,c);
    }
    int res=spfa();
    if(res==-1) puts("impossible");
    else printf("%d",res);

    return 0;
}
```

6.9 SPFA 判斷負環

```
// 给定一个 n 个点 m 条边的有向图，图中可能存在重边和自环，边权可能为负数。
// 请你判断图中是否存在负权回路。
// 3 3
// 1 2 -1
// 2 3 4
// 3 1 -4
// =
// Yes
#include <cstring>
#include <iostream>
#include <queue>
using namespace std;
const int N = 2e3 + 10, M = 1e4 + 10;
int n, m;
int head[N], e[M], ne[M], w[M], idx;
bool st[N];
int dist[N];
int cnt[N]; //cnt[x] 表示 当前从1-x的最短路的边数
void add(int a, int b, int c)
{
    e[idx] = b;
    ne[idx] = head[a];
    w[idx] = c;
    head[a] = idx++;
}
bool spfa(){
    // 这里不需要初始化dist数组为 正无穷/初始化的原因是，如果存在负环，那么dist不管初始化为多少，都会被更新
    queue<int> q;
    //不仅仅是1了，因为点1可能到不了有负环的点，因此把所有点都加入队列
    for(int i=1;i<=n;i++){
        q.push(i);
        st[i]=true;
    }
    while(q.size()){
        int t = q.front();
        q.pop();
        st[t]=false;
        for(int i = head[t];i!=-1; i=ne[i]){
            int j = e[i];
            if(dist[j]>dist[t]+w[i]){
                dist[j] = dist[t]+w[i];
                cnt[j] = cnt[t]+1;
                if(cnt[j]>=n){
                    return true;
                }
                if(!st[j]){
                    q.push(j);
                    st[j]=true;
                }
            }
        }
    }
    return false;
}
int main()
{
    cin >> n >> m;
    memset(head, -1, sizeof head);
    for (int i = 0; i < m; i++) {
        int a, b, c;
        cin >> a >> b >> c;
        add(a, b, c);
    }
    if (spfa()) {
        cout << "Yes" << endl;
    }
    else {
        cout << "No" << endl;
    }
    return 0;
}
```

6.10 floyd

```
#include <iostream>
using namespace std;

const int N = 210, M = 2e+10, INF = 1e9;

int n, m, k, x, y, z;
int d[N][N];

void floyd() {
    for(int k = 1; k <= n; k++)
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
                d[i][j] = min(d[i][j], d[i][k] + d[k][j
                    ]);
}

int main() {
    cin >> n >> m >> k;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            if(i == j) d[i][j] = 0;
            else d[i][j] = INF;
    while(m--) {
        cin >> x >> y >> z;
        d[x][y] = min(d[x][y], z);
        //注意保存最小的边
    }
    floyd();
    while(k--) {
        cin >> x >> y;
        if(d[x][y] > INF/2) puts("impossible");
        //由于有负权边存在所以约大过INF/2也很合理
        else cout << d[x][y] << endl;
    }
    return 0;
```

```
}
```

6.11 無向圖中字典序最小歐拉路徑

```
//給一個無向圖，點的編號最多F500，邊數最多F1024，首先輸入一個m代表邊的數量，然後讓輸出字典序最小的歐拉路徑(字典序最小—經過點的編號字典序最小)。題目保證至少有一個歐拉路徑。
int n = 500, m,ans[1100], cnt,d[N],g[N][N];
void dfs(int u)
{
    //因F最後的歐拉路徑的序列是ans數組逆序，
    //節點u只有在遍歷完所有邊之後最後才會加到ans數組F面，所以逆序過來就是最小的字典序
    for (int i = 1; i <= n; i ++ )
        if (g[u][i])//F邊優化
            g[u][i] --, g[i][u] -- ; dfs(i);
    ans[ ++ cnt] = u;
}
int main()
{
    cin >> m;
    while (m -- )
    {
        int a, b;
        cin >> a >> b;
        g[a][b] ++, g[b][a] ++ ;
        d[a] ++, d[b] ++ ;
    }
    int start = 1;
    while (!d[start]) ++start; // 較小編號作F起點
    //數據保證有解一定存在歐拉F路,那F讓第一條度數F奇數的點作F起點
    for (int i = 1; i <= 500; ++i) {
        if (d[i] % 2) { // 奇數點作F起點
            start = i;
            break;
        }
    }
    dfs(start);
    for (int i = cnt; i; i -- ) printf("%d\n", ans[i]);
    return 0;
}
```

6.12 topological sort

```
#include <iostream>
#include <vector>
#include <cstdio>
#include <queue>
#include <cstring>
#include <algorithm>
using namespace std;
int in[10010];
vector<int> v[10010];
int main()
{
    int n,m;
    while(~scanf("%d %d",&n,&m)&&n&&m)
    {
        memset(in,0,sizeof in);//清空入度
        for(int i=0;i<=n;i++) v[i].clear();
        for(int i=0;i<m;i++)
        {
            int x,y;
            cin >>x>>y;//比如x贏了y 或者y是x的兒子 ;那F就讓x指向y;
            v[x].push_back(y);
            in[y]++;//y的入度加1
        }
        priority_queue<int,vector<int>,greater<int> >q;
        //優先隊列，設置從小到大排序，小的在隊列下F
        for(int i=0;i<n;i++)
        {
            if(in[i]==0)
                q.push(i);//把入度F0的節點壓入隊列
        }
        while(!q.empty())
        {
            int xx=q.top();
            q.pop();
            n--;//每次去掉一個節點
            for(int i=0;i<v[xx].size();i++)
            {
                int yy=v[xx][i];
                in[yy]--;
                if(!in[yy])
                    q.push(yy);//如果去掉上一個節點之後下一個節點的入度變F0，則壓入隊列中
            }
        }
        if(n) cout <<"NO"<<endl;//如果有環的話節點數不會F0
        else cout <<"YES"<<endl;
    }
    return 0;
}
```

7 String

7.1 KMP

```
在字串s找是否存在qs字串 KMP(s,qs) return bool
-----------------------------------------------------
const ll MAXN = 1e5+10;
string s,qs;
ll failure[MAXN];
-----------------------------------------------------
bool KMP(string& t, string& p)
{
    if (p.size() > t.size()) return false;
    for (ll i=1, j=failure[0]=-1; i<p.size(); ++i)
    {
        while (j >= 0 && p[j+1] != p[i])
            j = failure[j];
        if (p[j+1] == p[i]) j++;
        failure[i] = j;
    }
    bool flag=false;
    for (ll i=0, j=-1; i<t.size(); ++i)
    {
        while (j >= 0 && p[j+1] != t[i])
            j = failure[j];
        if (p[j+1] == t[i]) j++;
        if (j == p.size()-1)
        {
            flag=true;
            j = failure[j];
        }
    }
    return flag;
}
```

7.2 Z Value

```
找字串每個位置的lcp
//abcaaeab
//80011020
-----------------------------------------------------
int z[MAXN];
void Z_value(const string& s) { //z[i] = lcp(s[1...],s[i...])
  int i, j, left, right, len = s.size();
  left=right=0; z[0]=len;
  for(i=1;i<len;i++) {
    j=max(min(z[i-left],right-i),0);
    for(;i+j<len&&s[i+j]==s[j];j++);
    z[i]=j;
    if(i+z[i]>right) {
      right=i+z[i];
      left=i;
}   }   }
```

8 Data Structure

8.1 DSU

```
void init()
{
    for(ll i=1;i<=n;i++) p[i]=i;
}
ll find(ll x)
```

```
{
    if(p[x]!=x) p[x]=find(p[x]);
    return p[x];
}
void merge(ll a,ll b)
{
    if(find(a)!=find(b))
    {
        p[find(a)]=find(b);
    }
}
```

8.2 trie

```
//在給定的 N 個整數 A1, A2……AN 中選出兩個進行 xor運算, 得到的結果最大是多少?
//第一行輸入一個整數 N。
// 第二行輸入 N 個整數 A1~AN。
#include<bits/stdc++.h>
#define IO cin.tie(0);ios_base::sync_with_stdio(false)
#define ll long long
using namespace std;
const ll MAXN = 1e5+10;
ll n,ans,idx;
ll a[MAXN],son[31*MAXN][2];
void insert(int x)
{
    int p=0;
    for(ll i=30;i>=0;i--)
    {
        int u=x>>i&1; // x第i位的二進制數
        if(!son[p][u])  son[p][u]=++idx; // 如果F路, 就建新的路
        p=son[p][u]; // p指向idx所指的下標
    }
}
ll query(ll x) //返回第i元素前與二進制a[i]相比最多位數不同的數
{
    int p=0;
    int res=0;
    for(ll i=30;i>=0;i--)
    {
        ll u=x>>i&1;
        if(son[p][!u])
        {
            p=son[p][!u];
            res=res*2+!u;
        }
        else
        {
            p=son[p][u];
            res=res*2+u;
        }
    }
    return res;
}
int main()
{
    IO;
    cin>>n;
    for(ll i=0;i<n;i++)
    {
        cin>>a[i];
        insert(a[i]);//建樹
        int t=query(a[i]);//返回第i元素前與二進制a[i]相比最多位數不同的數
        ans=max(ans,a[i]^t);
    }
    cout<<ans<<'\n';
    return 0;
}
```

8.3 Segment tree

```
struct seg_tree{
  ll a[MXN],val[MXN*4],tag[MXN*4],NO_TAG=0;
  void push(int i,int l,int r){
    if(tag[i]!=NO_TAG){
      val[i]+=tag[i]; // update by tag
      if(l!=r){
        tag[cl(i)]+=tag[i]; // push
        tag[cr(i)]+=tag[i]; // push
      }
      tag[i]=NO_TAG;
  } }
  void pull(int i,int l,int r){
    int mid=(l+r)>>1;
    push(cl(i),l,mid);push(cr(i),mid+1,r);
    val[i]=max(val[cl(i)],val[cr(i)]); // pull
  }
  void build(int i,int l,int r){
    if(l==r){
      val[i]=a[l]; // set value
      return;
    }
    int mid=(l+r)>>1;
    build(cl(i),l,mid);build(cr(i),mid+1,r);
    pull(i,l,r);
  }
  void update(int i,int l,int r,int ql,int qr,int v){
    push(i,l,r);
    if(ql<=l&&r<=qr){
      tag[i]+=v; // update tag
      return;
    }
    int mid=(l+r)>>1;
    if(ql<=mid) update(cl(i),l,mid,ql,qr,v);
    if(qr>mid) update(cr(i),mid+1,r,ql,qr,v);
    pull(i,l,r);
  }
  ll query(int i,int l,int r,int ql,int qr){
    push(i,l,r);
    if(ql<=l&&r<=qr)
      return val[i]; // update answer
      ll mid=(l+r)>>1,ret=0;
    if(ql<=mid) ret=max(ret,query(cl(i),l,mid,ql,qr));
    if(qr>mid) ret=max(ret,query(cr(i),mid+1,r,ql,qr));
    return ret;
} }tree;
```

9 Others

9.1 dp 背包問題

```
//有 N 件物品和一個容量F V 的背包。第 i 件物品的費用是 c[i], 價
// 值是 w[i]。求解將哪些物品裝入背包可使價值總和最大。
#include<iostream>
using namespace std;
const int N = 1010;
int v[N], w[N], dp[N];//dp[N][N]
int main(){
    int n, m;
    cin >> n >> m;
    for(int i = 1; i <= n; i++){
        cin >> v[i] >> w[i];
    }
    for(int i = 1; i <= n; i++){
        // for(int j = 0; j <= m; j++){
        //     if(j < v[i])
        //         dp[i][j] = dp[i-1][j];
        //     else
        //         dp[i][j]=max(dp[i-1][j], dp[i-1][j-v[i]]+w[i]);
        // }
        for(int j = m; j >=v[i]; j--){
            dp[j] = max(dp[j], dp[j-v[i]]+w[i]);
        }
    }
    cout<<dp[m]<<endl;;
    return 0;
}
```

9.2 dp 完全背包問題

```
// 有 N 種物品和一個容量F V 的背包, 每種物品都有無限件可用。 第 i 種物
// 品的費用是 c[i], 價值是 w[i]。 求解將哪些物品裝入背包可使這些物品的費用總
// 和不超過背包容量, 且價值總和最大。
#include <stdio.h>
#include <stdlib.h>
#define INF 50000000
```

```
typedef struct coin {
  int price, weight;
} coin;
void dynamicPackage(coin *coins, int n, int v)
{
  if (v < 0) {
    printf("This is impossible.\n");
    return;
  }
  int i, j, *dp;
  // 动态分配内存
  dp = (int *)malloc(sizeof(int) * (v + 1));
  // 初始化
  dp[0] = 0;
  for (i = 1; i <= v; i ++) dp[i] = INF;
  // 完全背包问题
  for (i = 1; i <= n; i ++) {
    for (j = coins[i].weight; j <= v; j ++) {
      dp[j] = (dp[j] < dp[j - coins[i].weight] + coins[
          i].price) ? dp[j] : dp[j - coins[i].weight] +
           coins[i].price;
    }
  }
  if (dp[v] >= INF)
    printf("This is impossible.\n");
  else
    printf("The minimum amount of money in the piggy-
        bank is %d.\n", dp[v]);
  // 清理内存
  free(dp);
  dp = NULL;
}
int main(void)
{
  int t, e, f, n, i;
  coin *coins;
  scanf("%d", &t);
  while (t --) {
    scanf("%d %d", &e, &f);
    scanf("%d", &n);
    // 接收货币
    coins = (coin *)malloc(sizeof(coin) * (n + 1));
    if (coins == NULL)  exit(-1);
    for (i = 1; i <= n; i ++) {
      scanf("%d %d", &coins[i].price, &coins[i].weight)
          ;
    }
    // 完全背包
    dynamicPackage(coins, n, f - e);
    free(coins);
    coins = NULL;
  }
  return 0;
}
```

9.3 dp 多重背包問題

```
// 有N种物品和一个容量为V的背包。第i种物品最多有n[i]
    件，每件费用是c[i]，价值是w[i]。求解将哪些物品装入
    背包可使这些物品的费用总和不超过背包容量，且价值总
    和最大
// dp[i][v] = max{dp[i - 1][v - k * c[i]] + w[i] | 0 <=
    k <= n[i]}
#include <stdio.h>
#include <stdlib.h>
typedef struct rice {
    int price, weight, num;
} rice;
void dynamic(rice *rices, int m, int n)
{
    int i, j, cur, k, **dp;
    // 动态申请二维数组
    dp = (int **)malloc(sizeof(int *) * (m + 1));
    for (i = 0; i <= m; i ++)
        dp[i] = (int *)malloc(sizeof(int) * (n + 1));
    // 初始化
    for (i = 0; i <= m; i ++)
        for (j = 0; j <= n; j ++)
            dp[i][j] = 0;
    // 动态规划
    for (i = 1; i <= m; i ++) {
        for (j = 1; j <= n; j ++) {
            for (k = 0; k <= rices[i].num; k ++) {
                if (j - k * rices[i].price >= 0) {
                    cur = dp[i - 1][j - k * rices[i].
                        price] + k * rices[i].weight;
                    dp[i][j] = dp[i][j] > cur ? dp[i][j
                        ] : cur;
                } else {
                    break;
                }
            }
        }
    }
    printf("%d\n", dp[m][n]);
    for (i = 0; i <= m; i ++)
        free(dp[i]);
}
int main(void)
{
    int i, c, n, m;
    rice rices[2010];
    scanf("%d", &c);
    while (c --) {
        scanf("%d %d", &n, &m);
        // 接收数据
        for (i = 1; i <= m; i ++) {
            scanf("%d %d %d", &rices[i].price, &rices[i
                ].weight, &rices[i].num);
        }
        // 多重背包问题
        dynamic(rices, m, n);
    }
    return 0;
}
```

9.4 LIS 最長上升子序列

```
// 5
// 4 2 3 1 5
// =
// 3 (2,3,5)
#include <bits/stdc++.h>
#define maxn 1005
#define INF 99999999
using namespace std;
int n,a[maxn];
int dp[maxn];    //dp[i]:長度爲i+1的上升子序列中末尾元
    素的最小值（不存在的話就是INF）
int main()
{
  int i,j;
  scanf("%d",&n);
  for(i=0;i<n;i++)
    scanf("%d",&a[i]);
  fill(dp,dp+n,INF);   //初始化dp數組爲INF
  for(i=0;i<n;i++)     //找到更新dp[i]的位置🄵用a[i]更
      新之
    {
      *lower_bound(dp,dp+n,a[i])=a[i];//找到>=a[i]的
          第一個元素，🄵用a[i]替🄵;
      /* for(j=0;j<n;j++)    //觀察dp數組的填充過程，
          dp裏面保存着最長不下降子序列
          printf("%d ",dp[j]);
        printf("\n");  */
    }
  printf("%d\n",lower_bound(dp,dp+n,INF)-dp);  //第一個
      INF出現的位置即爲LIS長度
  return 0;
}
```

9.5 LCS 最長公共子序列

```
// 第1行輸出上述兩個最長公共子序列的長度。
// 第2行輸出所有可能出現的最長公共子序列個數，答案可能
    很大，只要將答案對100,000,000求余即可。
// ABCBDAB.
// BACBBD.
// =
// 4
// 7
#include<bits/stdc++.h>
#include<string>
using namespace std;
```

```
const int mod=1e8;
char a[5010],b[5010];
int f[5010][5010],g[5010][5010];
int len1,len2;
int main(){
    scanf("%s%s",a+1,b+1);
    len1=strlen(a+1);
    len2=strlen(b+1);
    len1--,len2--;
    for(int i=0;i<=len1;++i)
        for(int j=0;j<=len2;++j)
            g[i][j]=1;
    for (int i = 1 ;i <=len1; i++)
        for( int j = 1; j <=len2; j++) {
            if (a[i]==b[j]){
                f[i][j]=f[i-1][j-1] + 1;
                g[i][j]=g[i-1][j-1];
                if(f[i][j]==f[i-1][j]) g[i][j]=(g[i][j
                    ]+g[i-1][j])%mod;
                if(f[i][j]==f[i][j-1]) g[i][j]=(g[i][j
                    ]+g[i][j-1])%mod;
            }
            else{
                f[i][j] = max(f[i - 1][j], f[i][j - 1])
                    ;
                g[i][j]=0;
                if(f[i][j]==f[i-1][j]) g[i][j]=(g[i][j
                    ]+g[i-1][j])%mod;
                if(f[i][j]==f[i][j-1]) g[i][j]=(g[i][j
                    ]+g[i][j-1])%mod;
                if(f[i][j]==f[i-1][j-1]) g[i][j]=(g[i][[
                    j]-g[i-1][j-1]+mod)%mod;
            }
        }
    cout<<f[len1][len2]<<"\n"<<(g[len1][len2]+mod)%mod
        <<"\n";
    return 0;
}
```

9.6 LCIS 最長公共上升子序列

```
// 給出有 n 個元素的數組 a[] , m 個元素的數組 b[] ,求出
    它們的最長上升公共子序列的長度.
// F[i][j]表示以a串的前i個整數與b串的前j個整數且以b[j]
    爲結尾構成的LCIS的長度。
// O(n^2)
// 4
// 2 2 1 3
// 2 1 2 3
// =
// 2
#include <iostream>
using namespace std;
const int N = 3010;
int n;
int a[N], b[N];
int f[N][N];
int main()
{
    //input
    cin >> n;
    for (int i = 1; i <= n; ++ i) cin >> a[i];
    for (int i = 1; i <= n; ++ i) cin >> b[i];
    //dp
    for (int i = 1; i <= n; ++ i)
    {
        int maxv = 1;
        for (int j = 1; j <= n; ++ j)
        {
            f[i][j] = f[i - 1][j];
            if (b[j] == a[i]) f[i][j] = max(f[i][j],
                maxv);
            if (b[j] < a[i]) maxv = max(maxv, f[i - 1][
                j] + 1);
        }
    }
    //find result
    int res = 0;
    for (int i = 0; i <= n; ++ i) res = max(res, f[n][i
        ]);
    cout << res << endl;
    return 0;
}
```