Contents

Basic

/.vimrc

```
set nu ai si cin ts=4 sw=4 sts=4

nmap #2 :! gedit %<.in %<*.in &<CR>
nmap #4 :! date > %<.pt; cat -n % > %<.pt; lpr %<.pt <
    CR>
nmap #9 :! clear ; g++ -std=c++11 -O2 -D AC -o %<.out %
    ; for i in %<*.in; do echo $i; ./%<.out < $i; echo
    ""; done <CR>
nmap #0 :! clear ; g++ -std=c++11 -O2 -D AC -o %<.out %
    ; ./%<.out <CR>
nmap <C-I> :! read -p "CASE:" CASE; gedit %<_$CASE.in <
    CR>
```

default code

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

int main(){
#ifndef AC
  freopen("","r",stdin);
#endif
  ios_base::sync_with_stdio(0);
  cin.tie(0);
}
```

debug list

```
模板要記得 init
priority_queue 要清空
把邊界條件都加入測資
邊界條件 (過程溢位, 題目數據範圍), 會不會爆 long long
是否讀錯題目, 想不到時可以自己讀一次題目
環狀or凸包問題一定要每種都算n次
比較容易有問題的地方換人寫
注意公式有沒有推錯或抄錯
精度誤差 sqrt(大大的東西) + EPS
測試 %lld or %I64d
喇分 random_suffle 隨機演算法
```

Flow

Dinic

```
// 只能拿一次的點都要在點上加上 cap 1
const int INF = 1<<29;
struct Dinic{ //O(VVE)
  static const int MAXV = 5003;
  struct Edge{
    int from, to, cap, flow;
  };

  int n, m, s, t, d[MAXV], cur[MAXV];
  vector<Edge> edges;
  vector<int> G[MAXV];

  void init(int _n=MAXV){
    edges.clear();
    for (int i=0; i<_n; i++)G[i].clear();
  }
```

```
  void AddEdge(int from, int to, int cap){
    edges.push_back( {from,to,cap,0} );
    edges.push_back( {to,from,0,0} );
    m = edges.size();
    G[from].push_back(m-2);
    G[to].push_back(m-1);
  }

  bool dinicBFS(){
    memset(d,-1,sizeof(d));
    queue<int> que;
    que.push(s); d[s]=0;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (int ei:G[u]){
        Edge &e = edges[ei];
        if (d[e.to]<0 && e.cap>e.flow){
          d[e.to]=d[u]+1;
          que.push(e.to);
        }
      }
    }
    return d[t]>=0;
  }

  int dinicDFS(int u, int a){
    if (u==t || a==0)return a;
    int flow=0, f;
    for (int &i=cur[u]; i<(int)G[u].size(); i++){
      Edge &e = edges[ G[u][i] ];
      if (d[u]+1!=d[e.to])continue;
      f = dinicDFS(e.to, min(a, e.cap-e.flow) );
      if (f>0){
        e.flow += f;
        edges[ G[u][i]^1 ].flow -=f;
        flow += f;
        a -= f;
        if (a==0)break;
      }
    }
    return flow;
  }

  int maxflow(int s, int t){
    this->s = s, this->t = t;
    int flow=0, mf;
    while ( dinicBFS() ){
      memset(cur,0,sizeof(cur));
      while ( (mf=dinicDFS(s,INF)) )flow+=mf;
    }
    return flow;
  }
}dinic;

// s=0, t=1;
int fnd(int id ,int out=0){
  // out=0 入點 out=1 出點
  static int spr=1;
  //spr=2 時每個點分成入點,出點
  return id*spr+out+2;
}
```

KM

```
struct KM{
// Maximum Bipartite Weighted Matching (Perfect Match)
  static const int MXN = 650;
  static const int INF = 2147483647; // long long
  int n,match[MXN],vx[MXN],vy[MXN];
  int edge[MXN][MXN],lx[MXN],ly[MXN],slack[MXN];
  // ^^^^ long long
  void init(int _n){
    n = _n;
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        edge[i][j] = 0;
  }
  void add_edge(int x, int y, int w){ // long long
    edge[x][y] = w;
  }
  bool DFS(int x){
    vx[x] = 1;
    for (int y=0; y<n; y++){
      if (vy[y]) continue;
      if (lx[x]+ly[y] > edge[x][y]){
        slack[y] = min(slack[y], lx[x]+ly[y]-edge[x][y
            ]);
      } else {
        vy[y] = 1;
        if (match[y] == -1 || DFS(match[y])){
          match[y] = x;
          return true;
        }
      }
    }
    return false;
  }
  int solve(){
    fill(match,match+n,-1);
    fill(lx,lx+n,-INF);
    fill(ly,ly+n,0);
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        lx[i] = max(lx[i], edge[i][j]);
    for (int i=0; i<n; i++){
      fill(slack,slack+n,INF);
      while (true){
        fill(vx,vx+n,0);
        fill(vy,vy+n,0);
        if ( DFS(i) ) break;
        int d = INF; // long long
        for (int j=0; j<n; j++)
          if (!vy[j]) d = min(d, slack[j]);
        for (int j=0; j<n; j++){
          if (vx[j]) lx[j] -= d;
          if (vy[j]) ly[j] += d;
          else slack[j] -= d;
        }
      }
    }
    int res=0;
    for (int i=0; i<n; i++)
      res += edge[match[i]][i];
    return res;
  }
}graph;
```

KM

```
const int MAX_N = 400 + 10;
const ll INF64 = 0x3f3f3f3f3f3f3f3fLL;
int nl , nr;
int pre[MAX_N];
ll slack[MAX_N];
ll W[MAX_N][MAX_N];
ll lx[MAX_N] , ly[MAX_N];
int mx[MAX_N] , my[MAX_N];
bool vx[MAX_N] , vy[MAX_N];
void augment(int u) {
    if(!u) return;
    augment(mx[pre[u]]);
    mx[pre[u]] = u;
    my[u] = pre[u];
}
inline void match(int x) {
    queue<int> que;
    que.push(x);
    while(1) {
        while(!que.empty()) {
            x = que.front();
```

```
            que.pop();
            vx[x] = 1;
            REP1(y , 1 , nr) {
                if(vy[y]) continue;
                ll t = lx[x] + ly[y] - W[x][y];
                if(t > 0) {
                    if(slack[y] >= t) slack[y] = t ,
                        pre[y] = x;
                    continue;
                }
                pre[y] = x;
                if(!my[y]) {
                    augment(y);
                    return;
                }
                vy[y] = 1;
                que.push(my[y]);
            }
        }
        ll t = INF64;
        REP1(y , 1 , nr) if(!vy[y]) t = min(t , slack[y
            ]);
        REP1(x , 1 , nl) if(vx[x]) lx[x] -= t;
        REP1(y , 1 , nr) {
            if(vy[y]) ly[y] += t;
            else slack[y] -= t;
        }
        REP1(y , 1 , nr) {
            if(vy[y] || slack[y]) continue;
            if(!my[y]) {
                augment(y);
                return;
            }
            vy[y] = 1;
            que.push(my[y]);
        }
    }
}
int main() {
    int m;
    RI(nl , nr , m);
    nr = max(nl , nr);
    while(m--) {
        int x , y;
        ll w;
        RI(x , y , w);
        W[x][y] = w;
        lx[x] = max(lx[x] , w);
    }
    REP1(i , 1 , nl) {
        REP1(x , 1 , nl) vx[x] = 0;
        REP1(y , 1 , nr) vy[y] = 0 , slack[y] = INF64;
        match(i);
    }
    ll ans = 0LL;
    REP1(x , 1 , nl) ans += W[x][mx[x]];
    PL(ans);
    REP1(x , 1 , nl) printf("%d%c",W[x][mx[x]] ? mx[x]
        : 0," \n"[x == nl]);
    return 0;
}
```

min cost max flow

```
// from: https://github.com/bobogei81123/bcw_codebook/
    blob/master/codes/Graph/Flow/CostFlow.cpp
typedef pair<long long, long long> pll;
struct CostFlow {
  static const int MXN = 205;
  static const long long INF = 102938475610293847LL;
  struct Edge {
    int v, r;
    long long f, c;
  };
  int n, s, t, prv[MXN], prvL[MXN], inq[MXN];
  long long dis[MXN], fl, cost;
  vector<Edge> E[MXN];
  void init(int _n, int _s, int _t) {
    n = _n; s = _s; t = _t;
    for (int i=0; i<n; i++) E[i].clear();
    fl = cost = 0;
  }
  void add_edge(int u, int v, long long f, long long c)
       {
    E[u].PB({v, SZ(E[v])  , f,  c});
    E[v].PB({u, SZ(E[u])-1, 0, -c});
  }
  pll flow() {
    while (true) {
      for (int i=0; i<n; i++) {
        dis[i] = INF;
        inq[i] = 0;
      }
      dis[s] = 0;
      queue<int> que;
      que.push(s);
      while (!que.empty()) {
        int u = que.front(); que.pop();
        inq[u] = 0;
        for (int i=0; i<SZ(E[u]); i++) {
          int v = E[u][i].v;
          long long w = E[u][i].c;
          if (E[u][i].f > 0 && dis[v] > dis[u] + w) {
            prv[v] = u; prvL[v] = i;
            dis[v] = dis[u] + w;
            if (!inq[v]) {
              inq[v] = 1;
              que.push(v);
            }
          }
        }
      }
      if (dis[t] == INF) break;
      long long tf = INF;
      for (int v=t, u, l; v!=s; v=u) {
        u=prv[v]; l=prvL[v];
        tf = min(tf, E[u][l].f);
      }
      for (int v=t, u, l; v!=s; v=u) {
        u=prv[v]; l=prvL[v];
        E[u][l].f -= tf;
        E[v][E[u][l].r].f += tf;
      }
      cost += tf * dis[t];
      fl += tf;
    }
    return {fl, cost};
  }
}flow;
```

思想

带下界的网络流问题

问题1：无源汇带上下界可行流
问题描述：有向非简单图G的每条边e都有两个值L[e]与H[e]，现求得一个可行流，使得每条边上的流值都在L[e]和H[e]之间，且对于每个节点v,in_flow(v)=out_flow(v)。
解决方案：重构等价网络流模型G'。对每条在原图中存在的边e，其容量为H[e]-L[e]。对于每个节点v，如果sum(L[in(v)])>sum(L[out(v)])，则从源点引入一条边到v，容量为sum(L[in(v)])-sum(L[out(v)])。如果sum(L[out(v)])>sum(L[in(v)])，则从v连一条边到汇点，容量为sum(L[out(v)])-sum(L[in(v)])。若该图的最大流为满流（s, t都满），则原图存在可行流。
证明：
->：
若G'存在满流，则可根据该满流在原图中得到一个可行流：断开所有与源点、汇点相连的边，并取消这些边上的流，然

后对剩余的每条边e（这些边在原图中也存在），每条边增加流值L[e]。
验证：
1. 平衡性：在G'的满流中，对于非源汇点v，in_flow(v)=out_flow(v),即in_flow(v)-out_flow(v)=0。对G'中的流作调整后,in_flow'(v)-out_flow'(v)=(in_flow(v)+sum(L[in(v)]))-(out_flow(v)+sum(L[out(v)]))-（sum(L[in(v)])-sum(L[out(v)]))=in_flow(v)-out_flow(v)=0。因此调整后的流对每个节点依然平衡。
2. 满足下界：在G'中每条边e有flow(e)>=0(注：flow(e)与flow(u, v)概念上不等价)，因此调整回G后，flow'(e)=flow(e)+L[e]>=L[e]。因此每条边都满足下界。
3. 满足上界：在G'中每条边e有capacity(e)+L[e]=H[e],因此对于G'的满流，有flow(e)+L[e]<=H[e]。因此在G中的每条边流值都不超过上界。
<-：若原图G存在可行流，通过对G中每条边减去L[e]的流，同时使源汇达到满流，可以得到G'的一个可行满流。反向验证同上。
证毕。

问题2：有源汇带上下界可行流
问题描述：存在源点s与汇点t，每条边e有两个值L[e]与H[e]，询问是否存在s->t可行流，使得每条边流值在L[e]和H[e]之间。
解决方案：添加一条边t->s, L[t->s]=0, H[t->s]=inf。若新图存在无源汇可行流，原图有对应解。
证明：对任意网络流，连上t->s即是无源汇可行流。任意无源汇可行流，断开任意一条有流的边，即是一个有源汇可行流。
证毕。

问题3：有源汇带上下界最大流
解决方案：令原图的源点为s，汇点为t。将原图转换为等价的无源汇图，再转换为求无源汇可行流等价的网络流模型G'。令ss为G'的源点，tt为G'的汇点。先求出G'的最大流（若最大流非满流则无解），再断开t->s，基于当前残余网络求出s->t最大流，最后将该网络流转换回等价图G的有源汇最大流。
证明：
->：若G'满流，并进一步求出了s->t的最大流
1. G的等价网络平衡性：由于ss, tt满流，因此在求s->t的最大流时，不存在经过ss或tt的增广路。因此对于任意非s, t节点，求s->t最大流前与s->t最大流后flow(ss, v)+flow(v, tt)始终等于sum(L[in(v)])-sum(L[out(v)])。因此，v在流量等价的G中依然保持流量平衡。
2. G的等价网络流不存在s到t的增广路：对于G中任意边e，其残余边容量c'(e)=H[e]-flow'(e)=H[e]-L[e]-flow(e)=capacity(e)-flow(e)=c(e)。因此与该边在G'中的残余容量相等。因此G中残余网络（与G'残余网络删除ss, tt等价）不存在s到t增广路。
3. G的等价网络满足上下界：证明与问题1相同。
<-：G的任意满足上下界的s-t最大流也可转化成唯一的等价G'无源汇可行流。反向验证同上（只需1,3）。
注：在实际算法中，可以通过不断开t->s直接求一次最大流得到该流值。

问题4：有源汇带上下界最小流
解决方案：令原图的源点为s，汇点为t。将原图转换为等价的无源汇图，再转换为求无源汇可行流的等价网络流模型G'。令ss为G'的源点，tt为G'的汇点。先求出G'的最大流（若最大流非满流则无解），再断开t->s，基于当前残余网络求出t->s最大流，最后将该网络流转换回等价图G的有源汇最小流。
证明：若G'满流，并进一步求出了t->s的最大流，等价图G满足平衡性与上下界，且不存在t->s的增广路（退流路径）。
注：在实际算法中，可以通过断开t->s，求得最大流f后由flow(t->s)-f直接得到答案。(若flow(t->s)-f小于0，说明强制流入t的流都可以流回s，因此此时输出0）。

问题5：无源汇有上下界最小费用可行流
解决方案：构造等价网络流G'，新加入的边费用都为0，该图的最小费用满流为答案。否则无解。
证明：若存在更优解，更优解一定在G'中有一对应的满流，且费用更小（最小费用最大流的残量网络不存在负权环，因此矛盾）。

问题6：有源汇有上下界最小费用可行流/有源汇最小费用最大流
解决方案：构造等价网络流G'，新加入的边费用都为0。等价图存在最小费用满流则有解。若求最小费用最大流，再对s->t求一次最小费用最大流（求之前断开t->s）。
证明：可行流证明略。在等价图中求得最小费用可行流后，残量网络一定不存在负权环，因此可以继续求s->t的最小费用最大流。

问题7：有源汇有上下界最小费用最小流
解决方案：构造等价网络流G'，新加入的边费用都为0，除t->s，该边费用为max(cost(e))*|E|+1。求得最小费用可行流后，该可行流即为最小费用最小流，注意费用值需要减去flow(t->e)*cost(t->e)。该方法也可用来求有源汇带下界最小流。

matching

最大匹配：一个图所有匹配中，所含匹配边数最多的匹配
完美匹配：如果一个图的某个匹配中，所有的顶点都是匹配点，那么它就是一个完美匹配。
最大匹配数：最大匹配的匹配边的数目
最小点覆盖数：选取最少的点，使任意一条边至少有一个端点被选择
最大独立数：选取最多的点，使任意所选两点均不相连
最小路径覆盖数：对于一个 DAG（有向无环图），选取最少条路径，使得每个顶点属于且仅属于一条路径。路径长可以为 0（即单个点）。

定理1：最大匹配数 = 最小点覆盖数（这是 Konig 定理）
定理2：最大匹配数 = 最大独立数
定理3：最小路径覆盖数 = 顶点数 - 最大匹配数

Domination

Independent Set:

無向圖上，選定數點，互不相鄰，稱作「獨立集」。
各點之間不相鄰，換到補圖上面就是，各點之間都有邊。原圖的 Clique ，就是補圖的 Independent Set ；原圖的 Independent Set ，就是補圖的 Clique 。

Maximum Independent Set [NP-complete]
無向圖上，點數最多的Maximum Independent Set。

Maximum Independent Set in Tree [P]
當給定的圖是樹，得利用Greedy Method求解。

Maximum Independent Set in Bipartite Graph [P]
當給定的圖是二分圖，得利用Maximum Cardinality Bipartite Matching求解。

===

Dominating Set
無向圖上，選定數點，其餘點皆與之相鄰，稱作「支配集」。

Minimum Dominating Set [NP-complete]
無向圖上點數最少的Dominating Set。

Minimum Dominating Set in Tree [P]
當給定的圖是樹，得利用DP求解。

Minimum Dominating Set in Bipartite Graph [NP-complete]
當給定的圖是二分圖。

==

independent set
獨立集。選出一些點，互不相鄰。最佳化問題是越多越好。
dominating set
支配集。選出一些點，其餘點皆與之相鄰。最佳化問題是越少越好。

==

Vertex Cover

一張無向圖上，挑選數個點，碰觸到所有邊，這些點就叫做一個「點覆蓋」，可能有許多種。換句話說，每一條邊，都會碰觸到一個以上的選定點。

Minimum Vertex Cover [NP-complete]
一張圖上點數最少的Vertex Cover。

Minimum Vertex Cover in Tree [P]
當給定的圖是樹，得利用Greedy演算法，從樹葉往樹根方向選出節點。

Minimum Vertex Cover in Bipartite Graph [P]
當給定的圖是二分圖，得化作Maximum Cardinality Bipartite Matching解決。

==

Edge Cover

一張無向圖上，挑選數條邊，碰觸到所有點，這些邊就叫做一個「邊覆蓋」，可能有許多種。

Minimum Edge Cover [P]
一張圖上邊數最少的Edge Cover。
得化作Maximum Matching解決。

Minimum Edge Cover in Bipartite Graph [P]
當給定的圖是二分圖，得利用Greedy演算法，優先覆蓋degree最小的點。

Minimum/Maximum Weight Edge Cover [P]
一張圖上權重最小（大）的Edge Cover。
得化作Minimum/Minimum Weight Matching解決。【待補文字】

帶權一般圖匹配

```
struct Graph {
  // Minimum General Weighted Matching (Perfect Match)
  static const int MXN = 105;
  int n, edge[MXN][MXN];
  int match[MXN],dis[MXN],onstk[MXN];
  vector<int> stk;
  void init(int _n) {
    n = _n;
    for( int i = 0 ; i < n ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        edge[ i ][ j ] = 0;
  }
  void add_edge(int u, int v, int w)
  { edge[u][v] = edge[v][u] = w; }
  bool SPFA(int u){
    if (onstk[u]) return true;
    stk.PB(u);
    onstk[u] = 1;
    for (int v=0; v<n; v++){
      if (u != v && match[u] != v && !onstk[v]){
        int m = match[v];
        if (dis[m] > dis[u] - edge[v][m] + edge[u][v]){
          dis[m] = dis[u] - edge[v][m] + edge[u][v];
          onstk[v] = 1;
          stk.PB(v);
          if (SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }
  int solve() {
    // find a match
    for (int i=0; i<n; i+=2){
      match[i] = i+1;
      match[i+1] = i;
    }
    while (true){
      int found = 0;
      for( int i = 0 ; i < n ; i ++ )
        onstk[ i ] = dis[ i ] = 0;
      for (int i=0; i<n; i++){
        stk.clear();
        if (!onstk[i] && SPFA(i)){
          found = 1;
          while (SZ(stk)>=2){
            int u = stk.back(); stk.pop_back();
            int v = stk.back(); stk.pop_back();
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if (!found) break;
    }
    int ret = 0;
    for (int i=0; i<n; i++)
      ret += edge[i][match[i]];
    ret /= 2;
    return ret;
  }
}graph;
```

Geometry

2D Point Template

```
typedef double T;
struct Point {
  T x,y;
  Point (T _x=0, T _y=0):x(_x),y(_y){}

  bool operator < (const Point &b)const{
    //return tie(x,y) < tie(b.x,b.y);
    //return atan2(y,x) < atan2(b.y,b.x);
    assert(0 && "choose compare");
  }
  bool operator == (const Point &b)const{
    //return tie(x,y) == tie(b.x,b.y);
    //return atan2(y,x) == atan2(b.y,b.x);
    assert(0 && "choose compare");
  }
  Point operator + (const Point &b)const{
    return Point(x+b.x,y+b.y);
  }
  Point operator - (const Point &b)const{
    return Point(x-b.x,y-b.y);
  }
  T operator * (const Point &b)const{
    return x*b.x + y*b.y;
```

```
  }
  T operator % (const Point &b)const{
    return x*b.y - y*b.x;
  }
  Point operator * (const T &d)const{
    return Point(d*x,d*y);
  }
  double abs2() { return x*x+y*y; }
  double abs() { return sqrt( abs2() ); }
};
typedef Point pdd;
double abs2(pdd a){
  return a.abs2();
}
```

Intersection of two circle

```
typedef Point pdd;
typedef ld double;
vector<pdd> interCircle(pdd o1, double r1, pdd o2,
    double r2) {
  ld d2 = (o1 - o2).abs2();
  ld d = sqrt(d2);
  if (d < fabs(r1-r2)) return {};
  if (d > r1+r2) return {};
  pdd u = 0.5*(o1+o2) + ((r2*r2-r1*r1)/(2.0*d2))*(o1-o2
      );
  double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d) *
      (-r1+r2+d));
  pdd v = A / (2.0*d2) * pdd(o1.S-o2.S, -o1.F+o2.F);
  return {u+v, u-v};
}
```

Convex Hull

```
#include "2Dpoint.cpp"

// retunr H, 第一個點會在 H 出現兩次
void ConvexHull(vector<Point> &P, vector<Point> &H){
    int n = P.size(), m=0;
    sort(P.begin(),P.end());
    H.clear();

    for (int i=0; i<n; i++){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }

    for (int i=n-2; i>=0; i--){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }
}
```

外心 Circumcentre

```
#include "2Dpoint.cpp"

pdd circumcentre(pdd &p0, pdd &p1, pdd &p2){
  pdd a = p1-p0;
  pdd b = p2-p0;
  double c1 = a.abs2()*0.5;
  double c2 = b.abs2()*0.5;
  double d = a % b;
  double x = p0.x + ( c1*b.y - c2*a.y ) / d;
  double y = p0.y + ( c2*a.x - c1*b.x ) / d;
  return pdd(x,y);
}
```

Smallest Covering Circle

```
#include "circumcentre.cpp"
pair<pdd,double> SmallestCircle(int n, pdd _p[]){
  static const int MAXN = 1000006;
  static pdd p[MAXN];
  memcpy(p,_p,sizeof(pdd)*n);
  random_shuffle(p,p+n);

  double r2=0;
  pdd cen;
  for (int i=0; i<n; i++){
    if ( (cen-p[i]).abs2() <=r2)continue;
    cen = p[i], r2=0;
    for (int j=0; j<i; j++){
      if ( (cen-p[j]).abs2()<=r2 )continue;
      cen = (p[i]+p[j])*0.5;
      r2 = (cen-p[i]).abs2();
      for (int k=0; k<j; k++){
        if ( (cen-p[k]).abs2()<=r2 )continue;
        cen = circumcentre(p[i],p[j],p[k]);
        r2 = (cen-p[k]).abs2();
      }
    }
  }

  return {cen,r2};
}
// auto res = SmallestCircle(,);
```

半平面交

```
// from BCW

const double EPS = 1e-9;

pdd interPnt(Line l1, Line l2, bool &res){
    pdd p1, p2, q1, q2;
    tie(p1, p2) = l1;
    tie(q1, q2) = l2;
  double f1 = cross(p2, q1, p1);
    double f2 = -cross(p2, q2, p1);
  double f = (f1 + f2);

    if(fabs(f) < EPS) {
        res = false;
        return {0, 0};
    }

    res = true;
  return (f2 / f) * q1 + (f1 / f) * q2;
}

bool isin(Line l0, Line l1, Line l2) {
    // Check inter(l1, l2) in l0
    bool res;
    pdd p = interPnt(l1, l2, res);
    return cross(l0.S, p, l0.F) > EPS;
}

/* If no solution, check: 1. ret.size() < 3
 * Or more precisely, 2. interPnt(ret[0], ret[1])
 * in all the lines. (use (l.S - l.F).cross(p - l.F) >
     0
 */
vector<Line> halfPlaneInter(vector<Line> lines) {
    int sz = lines.size();
    vector<double> ata(sz), ord(sz);
    for (int i=0; i<sz; i++) {
        ord[i] = i;
        pdd d = lines[i].S - lines[i].F;
        ata[i] = atan2(d.y, d.x);
    }
    sort(ALL(ord), [&](int i, int j) {
        if (abs(ata[i] - ata[j]) < EPS) {
```

```
            return cross(lines[i].S, lines[j].S, lines[
                i].F) < 0;
        }
        return ata[i] < ata[j];
    });
    vector<Line> fin;
    for (int i=0; i<sz; i++) {
        if (!i or fabs(ata[ord[i]] - ata[ord[i-1]]) >
            EPS) {
            fin.PB(lines[ord[i]]);
        }
    }

    deque<Line> dq;
    for (int i=0; i<SZ(fin); i++) {
        while(SZ(dq) >= 2 and
              not isin(fin[i], dq[SZ(dq)-2], dq[SZ(dq)
                  -1])) {
            dq.pop_back();
        }
        while(SZ(dq) >= 2 and
              not isin(fin[i], dq[0], dq[1])) {
            dq.pop_front();
        }
        dq.push_back(fin[i]);
    }

    while (SZ(dq) >= 3 and
           not isin(dq[0], dq[SZ(dq)-2], dq[SZ(dq)-1]))
                {
        dq.pop_back();
    }

    while (SZ(dq) >= 3 and
           not isin(dq[SZ(dq)-1], dq[0], dq[1])) {
        dq.pop_front();
    }
    vector<Line> res(ALL(dq));
    return res;
}
```

圓交

```
typedef Point pdd;
typedef ld double;
vector<pdd> interCircle(pdd o1, double r1, pdd o2,
    double r2) {
  ld d2 = (o1 - o2).abs2();
  ld d = sqrt(d2);
  if (d < fabs(r1-r2)) return {};
  if (d > r1+r2) return {};
  pdd u = 0.5*(o1+o2) + ((r2*r2-r1*r1)/(2.0*d2))*(o1-o2
      );
  double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d) *
      (-r1+r2+d));
  pdd v = A / (2.0*d2) * pdd(o1.S-o2.S, -o1.F+o2.F);
  return {u+v, u-v};
}
```

線段交

```
// from PEC

const double EPS = 1e-9;

pdd interPnt(pdd p1, pdd p2, pdd q1, pdd q2, bool &res)
    {
  double f1 = cross(p2, q1, p1);
  double f2 = -cross(p2, q2, p1);
  double f = (f1 + f2);

  if(fabs(f) < EPS) {
    res = false;
    return {};
    }

  res = true;
  return (f2 / f) * q1 + (f1 / f) * q2;
}
```

Mathmatics

LinearPrime

```
const int MAXP = 100; //max prime
vector<int> P;  // primes
void build_prime(){
  static bitset<MAXP> ok;
  int np=0;
  for (int i=2; i<MAXP; i++){
    if (ok[i]==0)P.push_back(i), np++;
    for (int j=0; j<np && i*P[j]<MAXP; j++){
      ok[ i*P[j] ] = 1;
      if ( i%P[j]==0 )break;
    }
  }
}
```

BigInt

```
struct Bigint{
  static const int LEN = 60;
  static const int BIGMOD = 10000;
  int s;
  int vl, v[LEN];
  //  vector<int> v;
  Bigint() : s(1) { vl = 0; }
  Bigint(long long a) {
    s = 1; vl = 0;
    if (a < 0) { s = -1; a = -a; }
    while (a) {
      push_back(a % BIGMOD);
      a /= BIGMOD;
    }
  }
  Bigint(string str) {
    s = 1; vl = 0;
    int stPos = 0, num = 0;
    if (!str.empty() && str[0] == '-') {
      stPos = 1;
      s = -1;
    }
    for (int i=SZ(str)-1, q=1; i>=stPos; i--) {
      num += (str[i] - '0') * q;
      if ((q *= 10) >= BIGMOD) {
        push_back(num);
        num = 0; q = 1;
      }
    }
    if (num) push_back(num);
  }
  int len() const { return vl; /* return SZ(v); */ }
  bool empty() const { return len() == 0; }
  void push_back(int x) { v[vl++] = x; /* v.PB(x); */ }
  void pop_back() { vl--; /* v.pop_back(); */ }
  int back() const { return v[vl-1]; /* return v.back()
      ; */ }
  void n() { while (!empty() && !back()) pop_back(); }
  void resize(int nl) {
    vl = nl; fill(v, v+vl, 0);
    //    v.resize(nl); // fill(ALL(v), 0);
  }
  void print() const {
    if (empty()) { putchar('0'); return; }
    if (s == -1) putchar('-');
```

```
    printf("%d", back());
    for (int i=len()-2; i>=0; i--) printf("%.4d",v[i]);
  }
  friend std::ostream& operator << (std::ostream& out,
      const Bigint &a) {
    if (a.empty()) { out << "0"; return out; }
    if (a.s == -1) out << "-";
    out << a.back();
    for (int i=a.len()-2; i>=0; i--) {
      char str[10];
      snprintf(str, 5, "%.4d", a.v[i]);
      out << str;
    }
    return out;
  }
  int cp3(const Bigint &b)const {
    if (s != b.s) return s > b.s ? 1 : -1;
    if (s == -1) return -(-*this).cp3(-b);
    if (len() != b.len()) return len()>b.len()?1:-1;
    for (int i=len()-1; i>=0; i--)
      if (v[i]!=b.v[i]) return v[i]>b.v[i]?1:-1;
    return 0;
  }
  bool operator < (const Bigint &b)const{ return cp3(b)
      ==-1; }
  bool operator <= (const Bigint &b)const{ return cp3(b
      )<=0; }
  bool operator >= (const Bigint &b)const{ return cp3(b
      )>=0; }
  bool operator == (const Bigint &b)const{ return cp3(b
      )==0; }
  bool operator != (const Bigint &b)const{ return cp3(b
      )!=0; }
  bool operator > (const Bigint &b)const{ return cp3(b)
      ==1; }
  Bigint operator - () const {
    Bigint r = (*this);
    r.s = -r.s;
    return r;
  }
  Bigint operator + (const Bigint &b) const {
    if (s == -1) return -(-(*this)+(-b));
    if (b.s == -1) return (*this)-(-b);
    Bigint r;
    int nl = max(len(), b.len());
    r.resize(nl + 1);
    for (int i=0; i<nl; i++) {
      if (i < len()) r.v[i] += v[i];
      if (i < b.len()) r.v[i] += b.v[i];
      if(r.v[i] >= BIGMOD) {
        r.v[i+1] += r.v[i] / BIGMOD;
        r.v[i] %= BIGMOD;
      }
    }
    r.n();
    return r;
  }
  Bigint operator - (const Bigint &b) const {
    if (s == -1) return -(-(*this)-(-b));
    if (b.s == -1) return (*this)+(-b);
    if ((*this) < b) return -(b-(*this));
    Bigint r;
    r.resize(len());
    for (int i=0; i<len(); i++) {
      r.v[i] += v[i];
      if (i < b.len()) r.v[i] -= b.v[i];
      if (r.v[i] < 0) {
        r.v[i] += BIGMOD;
        r.v[i+1]--;
      }
    }
    r.n();
    return r;
  }
  Bigint operator * (const Bigint &b) {
    Bigint r;
    r.resize(len() + b.len() + 1);
    r.s = s * b.s;
    for (int i=0; i<len(); i++) {
      for (int j=0; j<b.len(); j++) {
        r.v[i+j] += v[i] * b.v[j];
        if(r.v[i+j] >= BIGMOD) {
          r.v[i+j+1] += r.v[i+j] / BIGMOD;
          r.v[i+j] %= BIGMOD;
        }
      }
    }
    r.n();
    return r;
  }
  Bigint operator / (const Bigint &b) {
    Bigint r;
    r.resize(max(1, len()-b.len()+1));
    int oriS = s;
    Bigint b2 = b; // b2 = abs(b)
    s = b2.s = r.s = 1;
    for (int i=r.len()-1; i>=0; i--) {
      int d=0, u=BIGMOD-1;
      while(d<u) {
        int m = (d+u+1)>>1;
        r.v[i] = m;
        if((r*b2) > (*this)) u = m-1;
        else d = m;
      }
      r.v[i] = d;
    }
    s = oriS;
    r.s = s * b.s;
    r.n();
    return r;
  }
  Bigint operator % (const Bigint &b) {
    return (*this)-(*this)/b*b;
  }
};
```

Random

```
inline int ran(){
  static int x = 20167122;
  return x = (x * 0xdefaced + 1) & INT_MAX;
}
```

Theorem

```
/*
Lucas's Theorem:
  For non-negative integer n,m and prime P,
  C(m,n) mod P = C(m/M,n/M) * C(m%M,n%M) mod P
  = mult_i ( C(m_i,n_i) )
  where m_i is the i-th digit of m in base P.
------------------------------------------------------
Pick's Theorem
  A = i + b/2 - 1
------------------------------------------------------
Kirchhoff's theorem
  A_{ii} = deg(i), A_{ij} = (i,j) \in E ? -1 : 0
  Deleting any one row, one column, and cal the det(A)
*/
```

Miller Rabin

```
typedef long long LL;

inline LL bin_mul(LL a, LL n,const LL& MOD){
  LL re=0;
  while (n>0){
    if (n&1) re += a;
```

```
    a += a; if (a>=MOD) a-=MOD;
    n>>=1;
  }
  return re%MOD;
}

inline LL bin_pow(LL a, LL n,const LL& MOD){
  LL re=1;
  while (n>0){
    if (n&1) re = bin_mul(re,a,MOD);
    a = bin_mul(a,a,MOD);
    n>>=1;
  }
  return re;
}

bool is_prime(LL n){
  //static LL sprp[3] = { 2LL, 7LL, 61LL};
  static LL sprp[7] = { 2LL, 325LL, 9375LL,
    28178LL, 450775LL, 9780504LL,
    1795265022LL };
  if (n==1 || (n&1)==0 ) return n==2;
  int u=n-1, t=0;
  while ( (u&1)==0 ) u>>=1, t++;
  for (int i=0; i<3; i++){
    LL x = bin_pow( sprp[i]%n, u, n);
    if (x==0 || x==1 || x==n-1)continue;

    for (int j=1; j<t; j++){
      x=x*x%n;
      if (x==1 || x==n-1)break;
    }
    if (x==n-1)continue;
    return 0;
  }
  return 1;
}
```

ax+by=gcd(a,b)

```
typedef pair<int, int> pii;
pii extgcd(int a, int b){
  if(b == 0) return make_pair(1, 0);
  else{
    int p = a / b;
    pii q = extgcd(b, a % b);
    return make_pair(q.second, q.first - q.second * p);
  }
}
```

FFT

```
const double pi = atan(1.0)*4;
struct Complex {
    double x,y;
    Complex(double _x=0,double _y=0)
        :x(_x),y(_y) {}
    Complex operator + (Complex &tt) { return Complex(x
        +tt.x,y+tt.y); }
    Complex operator - (Complex &tt) { return Complex(x
        -tt.x,y-tt.y); }
    Complex operator * (Complex &tt) { return Complex(x
        *tt.x-y*tt.y,x*tt.y+y*tt.x); }
};
void fft(Complex *a, int n, int rev) {
    // n是大于等于相乘的两个数组长度的2的幂次
    // 从0开始表示长度，对a进行操作
    // rev==1进行DFT，==-1进行IDFT
    for (int i = 1,j = 0; i < n; ++ i) {
        for (int k = n>>1; k > (j^=k); k >>= 1);
        if (i<j) std::swap(a[i],a[j]);
    }
    for (int m = 2; m <= n; m <<= 1) {
        Complex wm(cos(2*pi*rev/m),sin(2*pi*rev/m));
        for (int i = 0; i < n; i += m) {
            Complex w(1.0,0.0);
            for (int j = i; j < i+m/2; ++ j) {
                Complex t = w*a[j+m/2];
                a[j+m/2] = a[j] - t;
                a[j] = a[j] + t;
                w = w * wm;
            }
        }
    }
    if (rev==-1) {
        for (int i = 0; i < n; ++ i) a[i].x /= n,a[i].y
             /= n;
    }
}
```

FWHT

```
// FWHT template

const int MAXN = 1<<20;

void FWHT(int a[], int l=0, int r=MAXN-1){
  if (l==r)return;

  int mid = (l+r)>>1+1, n = r-l+1;
  FWHT(a,l,mid-1);
  FWHT(a,mid,r);

  for (int i=0; i<(n>>1); i++){
    int a1=a[l+i], a2=a[mid+i];
    a[l+i] = a1+a2;
    a[mid+i] = a1-a2;
  }
}
```

Hash

```
typedef long long LL;
LL X=7122;
LL P1=712271227;
LL P2=179433857;
LL P3=179434999;

struct HASH{
    LL a, b, c;
    HASH(LL a=0, LL b=0, LL c=0):a(a),b(b),c(c){ }
    HASH operator + (HASH B){
        return HASH((a+B.a)%P1,(b+B.b)%P2,(c+B.c)%P3);
    }
  HASH operator + (LL B){
    return (*this)+HASH(B,B,B);
  }
  HASH operator * (LL B){
    return HASH(a*B%P1,a*B%P2,a*B%P3);
  }
    bool operator < (const HASH &B)const{
        if (a!=B.a)return a<B.a;
        if (b!=B.b)return b<B.b;
        return c<B.c;
    }
    void up(){ (*this) = (*this)*X; }
};

int main(){
}
```

GaussElimination

```
// by bcw_codebook

const int MAXN = 300;
const double EPS = 1e-8;

int n;
double A[MAXN][MAXN];

void Gauss() {
  for(int i = 0; i < n; i++) {
    bool ok = 0;
    for(int j = i; j < n; j++) {
      if(fabs(A[j][i]) > EPS) {
        swap(A[j], A[i]);
        ok = 1;
        break;
      }
    }
    if(!ok) continue;

    double fs = A[i][i];
    for(int j = i+1; j < n; j++) {
      double r = A[j][i] / fs;
      for(int k = i; k < n; k++) {
        A[j][k] -= A[i][k] * r;
      }
    }
  }
}
```

Inverse

```
int inverse[100000];
void invTable(int b, int p) {
  inverse[1] = 1;
  for( int i = 2; i <= b; i++ ) {
    inverse[i] = (long long)inverse[p%i] * (p-p/i) % p;
  }
}

int inv(int b, int p) {
  return b == 1 ? 1 : ((long long)inv(p % b, p) * (p-p/
      b) % p);
}
```

IterSet

```
// get all subset in set S

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


}
```

SG

```
Sprague-Grundy

1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0：後手(P)必勝
不為 0：先手(N)必勝

int mex(set S) {
  // find the min number >= 0 that not in the S
  // e.g. S = {0, 1, 3, 4} mex(S) = 2
}

state = []
int SG(A) {
  if (A not in state) {
    S = sub_states(A)
    if( len(S) > 1 ) state[A] = reduce(operator.xor, [
        SG(B) for B in S])
    else state[A] = mex(set(SG(B) for B in next_states(
        A)))
  }
  return state[A]
}
```

Graph

Dijkstra

```
typedef struct Edge{
    int v; long long len;
    bool operator > (const Edge &b)const { return len>b
        .len; }
} State;

const long long INF = 1LL<<60;

void Dijkstra(int n, vector<Edge> G[], long long d[],
    int s, int t=-1){
    static priority_queue<State, vector<State>, greater
        <State> > pq;
    while ( pq.size() )pq.pop();
    for (int i=1; i<=n; i++)d[i]=INF;
    d[s]=0; pq.push( (State){s,d[s]} );
    while ( pq.size() ){
        auto x = pq.top(); pq.pop();
        int u = x.v;
        if (d[u]<x.len)continue;
        if (u==t)return;
        for (auto &e:G[u]){
            if (d[e.v] > d[u]+e.len){
                d[e.v] = d[u]+e.len;
                pq.push( (State) {e.v,d[e.v]} );
            }
        }
    }
}
```

Euler Circuit

```
//CF 723E
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 300;

struct EDGE{
    int u ,v ;
    int type;
};

int n, m, deg[MAXN];
vector <EDGE> edges;
vector<int> G[MAXN];
bool vis[MAXN*MAXN];
bool alive[MAXN][MAXN];
bool visN[MAXN];
vector<int> ans;

void add_edge(int u, int v, int type=0){
    edges.push_back( EDGE{u,v,type} );
    edges.push_back( EDGE{v,u,type} );
```

```
    G[u].push_back( edges.size()-2 );
    G[v].push_back( edges.size()-1 );
    deg[u]++, deg[v]++;
    alive[u][v]=alive[v][u]|=type^1;
}

void input(){
    memset(visN,0,sizeof(visN));
    memset(vis,0,sizeof(vis));
    memset(alive,0,sizeof(alive));
    memset(deg,0,sizeof(deg));
    edges.clear();
    ans.clear();
    for (int i=0; i<MAXN; i++)G[i].clear();

    scanf("%d%d",&n ,&m);
    for (int i=0, u, v; i<m; i++){
        scanf("%d%d", &u, &v);
        add_edge(u,v);
    }
}

void add_Graph(){
    vector<int> tmp;
    for (int i=1; i<=n; i++)if (deg[i]%2==1){
        tmp.push_back(i);
    }
    printf("%d\n",n-tmp.size());
    for (int i=0; i<tmp.size(); i+=2){
        add_edge(tmp[i],tmp[i+1],1);
    }
}

void dfs(int u){
    visN[u]=1;
    for (int i=0; i<G[u].size(); i++)if (!vis[ G[u][i
        ]>>1 ]){
        EDGE &e = edges[ G[u][i] ];
        int v = e.v;
        vis[ G[u][i]>>1 ]=1;
        dfs(v);
    }
    ans.push_back(u);
}

int main(){
    int T; scanf("%d",&T);
    while (T--){
        input();
        add_Graph();
        for (int i=1; i<=n; i++)if (!visN[i]){
            dfs(i);
            for (int j=0 ;j<ans.size()-1; j++){
                int u = ans[j], v=ans[j+1];
                if (alive[u][v]){
                    alive[u][v]=alive[v][u]=0;
                    printf("%d %d\n",u ,v);
                }
            }
            ans.clear();
        }
    }
}
```

一般圖匹配

```
#define MAXN 505
vector<int>g[MAXN];//用vector存圖
int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],vis[MAXN];
int t,n;
inline int lca(int u,int v){//找花的花托
  for(++t;;swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;//這種方法可以不用清空vis陣列
    u=st[pa[match[u]]];
  }
}
#define qpush(u) q.push(u),S[u]=0
inline void flower(int u,int v,int l,queue<int> &q){
  while(st[u]!=l){
    pa[u]=v;//所有未匹配邊的pa都是雙向的
    if(S[v=match[u]]==1)qpush(v);//所有奇點變偶點
    st[u]=st[v]=l,u=pa[v];
  }
}
inline bool bfs(int u){
  for(int i=1;i<=n;++i)st[i]=i;//st[i]表示第i個點的集合
  memset(S+1,-1,sizeof(int)*n);//-1:沒走過 0:偶點 1:奇
      點
  queue<int>q;qpush(u);
  while(q.size()){
    u=q.front(),q.pop();
    for(size_t i=0;i<g[u].size();++i){
      int v=g[u][i];
      if(S[v]==-1){
        pa[v]=u,S[v]=1;
        if(!match[v]){//有增廣路直接擴充
          for(int lst;u;v=lst,u=pa[v])
            lst=match[u],match[u]=v,match[v]=u;
          return 1;
        }
        qpush(match[v]);
      }else if(!S[v]&&st[v]!=st[u]){
        int l=lca(st[v],st[u]);//遇到花，做花的處理
        flower(v,u,l,q),flower(u,v,l,q);
      }
    }
  }
  return 0;
}
inline int blossom(){
  memset(pa+1,0,sizeof(int)*n);
  memset(match+1,0,sizeof(int)*n);
  int ans=0;
  for(int i=1;i<=n;++i)
    if(!match[i]&&bfs(i))++ans;
  return ans;
}

int main(){
  int T, m; cin >> T;

  while ( cin >> n >> m  ){
    for (int i=1; i<=n; i++) g[i].clear();
    for (int i=1, u, v; i<=m; i++){
      cin >> u >> v;
      g[u].push_back(v);
      g[v].push_back(u);
    }
    cout << blossom() << endl;
  }
}
```

Hungarian

```
vector<int> G[MAXN];
int n;
int match[MAXN]; // Matching Result
int visit[MAXN];

bool dfs(int u) {
    for ( auto v:G[u] ) {
        if (!visit[v]) {
            visit[v] = true;
            if (match[v] == -1 || dfs(match[v])) {
                match[v] = u;
                match[u] = v;
                return true;
```

```
            }
        }
    }
    return false;
}

int hungarian() {
    int res = 0;
    memset(match, -1, sizeof(match));
    for (int i = 0; i < n; i++) {
        if (match[i] == -1) {
            memset(visit, 0, sizeof(visit));
            if (dfs(i)) res += 1;
        }
    }
    return res;
}
```

Strongly Connected Component(SCC)

```
#define MXN 100005
#define PB push_back
#define FZ(s) memset(s,0,sizeof(s))

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 add_edge(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++;
    }
  }
}
};
```

LCA

```
//lv紀錄深度
//father[多少冪次][誰]
//已經建好每個人的父親是誰 (father[0][i]已經建好)
//已經建好深度 (lv[i]已經建好)
void makePP(){
  for(int i = 1; i < 20; i++){
    for(int j = 2; j <= n; j++){
      father[i][j]=father[i-1][ father[i-1][j] ];
    }
  }
}
int find(int a, int b){
  if(lv[a] < lv[b]) swap(a,b);
  int need = lv[a] - lv[b];
  for(int i = 0; need!=0; i++){
    if(need&1) a=father[i][a];
    need >>= 1;
  }
  for(int i = 19 ;i >= 0 ;i--){
    if(father[i][a] != father[i][b]){
      a=father[i][a];
      b=father[i][b];
    }
  }
  return a!=b?father[0][a] : a;
}
```

Maximum Clique

```
const int MAXN = 105;
int best;
int m ,n;
int num[MAXN];
// int x[MAXN];
int path[MAXN];
int g[MAXN][MAXN];

bool dfs( int *adj, int total, int cnt ){
    int i, j, k;
    int t[MAXN];
    if( total == 0 ){
        if( best < cnt ){
            // for( i = 0; i < cnt; i++) path[i] = x[i
                ];
            best = cnt; return true;
        }
        return false;
    }
    for( i = 0; i < total; i++){
        if( cnt+(total-i) <= best ) return false;
        if( cnt+num[adj[i]] <= best ) return false;
        // x[cnt] = adj[i];
        for( k = 0, j = i+1; j < total; j++ )
            if( g[ adj[i] ][ adj[j] ] )
                t[ k++ ] = adj[j];
                if( dfs( t, k, cnt+1 ) ) return true;
    } return false;
}
int MaximumClique(){
    int i, j, k;
    int adj[MAXN];
    if( n <= 0 ) return 0;
    best = 0;
    for( i = n-1; i >= 0; i-- ){
        // x[0] = i;
        for( k = 0, j = i+1; j < n; j++ )
            if( g[i][j] ) adj[k++] = j;
        dfs( adj, k, 1 );
        num[i] = best;
    }
    return best;
}
```

Tarjan

ccd ..

```
// 0 base
struct TarjanSCC{
```

```
  static const int MAXN = 1000006;
  int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
  vector<int> G[MAXN];
  stack<int> stk;
  bool ins[MAXN];

  void tarjan(int u){
    dfn[u] = low[u] = ++count;
    stk.push(u);
    ins[u] = true;

    for(auto v:G[u]){
      if(!dfn[v]){
        tarjan(v);
        low[u] = min(low[u], low[v]);
      }else if(ins[v]){
        low[u] = min(low[u], dfn[v]);
      }
    }

    if(dfn[u] == low[u]){
      int v;
      do {
      v = stk.top();
      stk.pop();
      scc[v] = scn;
      ins[v] = false;
      } while(v != u);
      scn++;
    }
  }

  void getSCC(){
    memset(dfn,0,sizeof(dfn));
    memset(low,0,sizeof(low));
    memset(ins,0,sizeof(ins));
    memset(scc,0,sizeof(scc));
    count = scn = 0;
    for(int i = 0 ; i < n ; i++ ){
      if(!dfn[i]) tarjan(i);
    }
  }

}SCC;
```

2-SAT

```
const int MAXN = 2020;

struct TwoSAT{
    static const int MAXv = 2*MAXN;
    vector<int> GO[MAXv],BK[MAXv],stk;
    bool vis[MAXv];
    int SC[MAXv];

    void imply(int u,int v){ // u imply v
        GO[u].push_back(v);
        BK[v].push_back(u);
    }
    int dfs(int u,vector<int>*G,int sc){
        vis[u]=1, SC[u]=sc;
        for (int v:G[u])if (!vis[v])
            dfs(v,G,sc);
        if (G==GO)stk.push_back(u);
    }
    int scc(int n=MAXv){
        memset(vis,0,sizeof(vis));
        for (int i=0; i<n; i++)if (!vis[i])
            dfs(i,GO,-1);
        memset(vis,0,sizeof(vis));
        int sc=0;
        while (!stk.empty()){
            if (!vis[stk.back()])
                dfs(stk.back(),BK,sc++);
            stk.pop_back();
        }
    }
}SAT;

int main(){
    SAT.scc(2*n);
    bool ok=1;
    for (int i=0; i<n; i++){
        if (SAT.SC[2*i]==SAT.SC[2*i+1])ok=0;
    }
    if (ok){
        for (int i=0; i<n; i++){
            if (SAT.SC[2*i]>SAT.SC[2*i+1]){
                cout << i << endl;
            }
        }
    }
    else puts("NO");
}
```

曼哈頓 MST

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

const int MAXN = 100005;
const int OFFSET = 2000; // y-x may < 0, offset it, if
    y-x too large, please write a unique function
const int INF = 0xFFFFFFF;
int n;
int x[MAXN], y[MAXN], p[MAXN];

typedef pair<int, int> pii;
pii bit[MAXN]; // [ val, pos ]

struct P {
    int x, y, id;
    bool operator<(const P&b ) const {
        if ( x == b.x ) return y > b.y;
        else return x > b.x;
    }
};
vector<P> op;

struct E {
    int x, y, cost;
    bool operator<(const E&b ) const {
        return cost < b.cost;
    }
};
vector<E> edges;

int find(int x) {
    return p[x] == x ? x : p[x] = find(p[x]);
}

void update(int i, int v, int p) {
    while ( i ) {
        if ( bit[i].first > v ) bit[i] = {v, p};
        i -= i & (-i);
    }
}

pii query(int i) {
    pii res = {INF, INF};
    while ( i < MAXN ) {
        if ( bit[i].first < res.first ) res = {bit[i].
            first, bit[i].second};
        i += i & (-i);
    }
    return res;
}

void input() {
    cin >> n;
    for ( int i = 0 ; i < n ; i++ ) cin >> x[i] >> y[i
        ], op.push_back((P) {x[i], y[i], i});
```

```
}

void mst() {
    for ( int i = 0 ; i < MAXN ; i++ ) p[i] = i;
    int res = 0;
    sort(edges.begin(), edges.end());
    for ( auto e : edges ) {
        int x = find(e.x), y = find(e.y);
        if ( x != y ) {
            p[x] = y;
            res += e.cost;
        }
    }
    cout << res << endl;
}

void construct() {
    sort(op.begin(), op.end());
    for ( int i = 0 ; i < n ; i++ ) {
        pii q = query(op[i].y - op[i].x + OFFSET);
        update(op[i].y - op[i].x + OFFSET, op[i].x + op
            [i].y, op[i].id);
        if ( q.first == INF ) continue;
        edges.push_back((E) {op[i].id, q.second, abs(x[
            op[i].id]-x[q.second]) + abs(y[op[i].id]-y[
            q.second]) });
    }
}

void solve() {

    // [45 ~ 90 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF,
        INF};
    construct();

    // [0 ~ 45 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF,
        INF};
    for ( int i = 0 ; i < n ; i++ ) swap(op[i].x, op[i
        ].y);
    construct();
    for ( int i = 0 ; i < n ; i++ ) swap(op[i].x, op[i
        ].y);

    // [-90 ~ -45 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF,
        INF};
    for ( int i = 0 ; i < n ; i++ ) op[i].y *= -1;
    construct();

    // [-45 ~ 0 deg]
    for ( int i = 0 ; i < MAXN ; i++ ) bit[i] = {INF,
        INF};
    for ( int i = 0 ; i < n ; i++ ) swap(op[i].x, op[i
        ].y);
    construct();

    // mst
    mst();

}

int main () {
    input();
    solve();
    return 0;
}
```

最小平均環

```
// from BCW

/* minimum mean cycle */
const int MAXE = 1805;
const int MAXN = 35;
const double inf = 1029384756;
const double eps = 1e-6;
struct Edge {
  int v,u;
  double c;
};
int n,m,prv[MAXN][MAXN], prve[MAXN][MAXN], vst[MAXN];
Edge e[MAXE];
vector<int> edgeID, cycle, rho;
double d[MAXN][MAXN];
inline void bellman_ford() {
  for(int i=0; i<n; i++) d[0][i]=0;
  for(int i=0; i<n; i++) {
    fill(d[i+1], d[i+1]+n, inf);
    for(int j=0; j<m; j++) {
      int v = e[j].v, u = e[j].u;
      if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
        d[i+1][u] = d[i][v]+e[j].c;
        prv[i+1][u] = v;
        prve[i+1][u] = j;
      }
    }
  }
}
double karp_mmc() {
  // returns inf if no cycle, mmc otherwise
  double mmc=inf;
  int st = -1;
  bellman_ford();
  for(int i=0; i<n; i++) {
    double avg=-inf;
    for(int k=0; k<n; k++) {
      if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i])
          /(n-k));
      else avg=max(avg,inf);
    }
    if (avg < mmc) tie(mmc, st) = tie(avg, i);
  }
  for(int i=0; i<n; i++) vst[i] = 0;
  edgeID.clear(); cycle.clear(); rho.clear();
  for (int i=n; !vst[st]; st=prv[i--][st]) {
    vst[st]++;
    edgeID.PB(prve[i][st]);
    rho.PB(st);
  }
  while (vst[st] != 2) {
    int v = rho.back(); rho.pop_back();
    cycle.PB(v);
    vst[v]++;
  }
  reverse(ALL(edgeID));
  edgeID.resize(SZ(cycle));
  return mmc;
}
```

BCC

```
// from BCW

struct BccEdge {
  static const int MXN = 100005;
  struct Edge { int v,eid; };
  int n,m,step,par[MXN],dfn[MXN],low[MXN];
  vector<Edge> E[MXN];
  DisjointSet djs;
  void init(int _n) {
    n = _n; m = 0;
    for (int i=0; i<n; i++) E[i].clear();
    djs.init(n);
  }
  void add_edge(int u, int v) {
    E[u].PB({v, m});
    E[v].PB({u, m});
    m++;
  }
  void DFS(int u, int f, int f_eid) {
```

```
    par[u] = f;
    dfn[u] = low[u] = step++;
    for (auto it:E[u]) {
      if (it.eid == f_eid) continue;
      int v = it.v;
      if (dfn[v] == -1) {
        DFS(v, u, it.eid);
        low[u] = min(low[u], low[v]);
      } else {
        low[u] = min(low[u], dfn[v]);
      }
    }
  }
  void solve() {
    step = 0;
    memset(dfn, -1, sizeof(int)*n);
    for (int i=0; i<n; i++) {
      if (dfn[i] == -1) DFS(i, i, -1);
    }
    djs.init(n);
    for (int i=0; i<n; i++) {
      if (low[i] < dfn[i]) djs.uni(i, par[i]);
    }
  }
}graph;
```

DominatorTree

```
// PEC VER

// idom[n] is the unique node that strictly dominates n
    but does
// not strictly dominate any other node that strictly
    dominates n.
// idom[n] = 0 if n is entry or the entry cannot reach
    n.
struct DominatorTree{
  static const int MAXN = 200010;
  int n,s;
  vector<int> g[MAXN],pred[MAXN];
  vector<int> cov[MAXN];
  int dfn[MAXN],nfd[MAXN],ts;
  int par[MAXN];
  int sdom[MAXN],idom[MAXN];
  int mom[MAXN],mn[MAXN];

  inline bool cmp(int u,int v) { return dfn[u] < dfn[v
      ]; }

  int eval(int u) {
    if(mom[u] == u) return u;
    int res = eval(mom[u]);
    if(cmp(sdom[mn[mom[u]]],sdom[mn[u]]))
      mn[u] = mn[mom[u]];
    return mom[u] = res;
  }

  void init(int _n, int _s) {
    n = _n;
    s = _s;
    REP1(i,1,n) {
      g[i].clear();
      pred[i].clear();
      idom[i] = 0;
    }
  }
  void add_edge(int u, int v) {
    g[u].push_back(v);
    pred[v].push_back(u);
  }
  void DFS(int u) {
    ts++;
    dfn[u] = ts;
    nfd[ts] = u;
    for(int v:g[u]) if(dfn[v] == 0) {
      par[v] = u;
      DFS(v);
    }
  }
  void build() {
    ts = 0;
    REP1(i,1,n) {
      dfn[i] = nfd[i] = 0;
      cov[i].clear();
      mom[i] = mn[i] = sdom[i] = i;
    }
    DFS(s);
    for (int i=ts; i>=2; i--) {
      int u = nfd[i];
      if(u == 0) continue ;
      for(int v:pred[u]) if(dfn[v]) {
        eval(v);
        if(cmp(sdom[mn[v]],sdom[u])) sdom[u] = sdom[mn[
            v]];
      }
      cov[sdom[u]].push_back(u);
      mom[u] = par[u];
      for(int w:cov[par[u]]) {
        eval(w);
        if(cmp(sdom[mn[w]],par[u])) idom[w] = mn[w];
        else idom[w] = par[u];
      }
      cov[par[u]].clear();
    }
    REP1(i,2,ts) {
      int u = nfd[i];
      if(u == 0) continue ;
      if(idom[u] != sdom[u]) idom[u] = idom[idom[u]];
    }
  }
}dom;
```

General Weighted Graph Max Matching

```
#define N 110
#define inf 0x3f3f3f3f
int G[ N ][ N ] , ID[ N ];
int match[ N ] , stk[ N ];
int vis[ N ] , dis[ N ];
int n , m , k , top;
bool SPFA( int u ){
  stk[ top ++ ] = u;
  if( vis[ u ] ) return true;
  vis[ u ] = true;
  for( int i = 1 ; i <= k ; i ++ ){
    if( i != u && i != match[ u ] && !vis[ i ] ){
      int v = match[ i ];
      if( dis[ v ] < dis[ u ] + G[ u ][ i ] - G[ i ][ v
          ] ){
        dis[ v ] = dis[ u ] + G[ u ][ i ] - G[ i ][ v
            ];
        if( SPFA( v ) ) return true;
      }
    }
  }
  top --; vis[ u ] = false;
  return false;
}
int MaxWeightMatch() {
  for( int i = 1 ; i <= k ; i ++ ) ID[ i ] = i;
  for( int i = 1 ; i <= k ; i += 2 ) match[ i ] = i + 1
      , match[ i + 1 ] = i;
  for( int times = 0 , flag ; times < 3 ; ){
    memset( dis , 0 , sizeof( dis ) );
    memset( vis , 0 , sizeof( vis ) );
    top = 0; flag = 0;
    for( int i = 1 ; i <= k ; i ++ ){
      if( SPFA( ID[ i ] ) ){
        flag = 1;
        int t = match[ stk[ top - 1 ] ] , j = top - 2;
        while( stk[ j ] != stk[ top - 1 ] ){
          match[ t ] = stk[ j ];
```

```
          swap( t , match[ stk[ j ] ] );
          j --;
        }
        match[ t ] = stk[ j ]; match[ stk[ j ] ] = t;
        break;
      }
    }
    if( !flag ) times ++;
    if( !flag ) random_shuffle( ID + 1 , ID + k + 1 );
  }
  int ret = 0;
  for( int i = 1 ; i <= k ; i ++ )
    if( i < match[ i ] ) ret += G[ i ][ match[ i ] ];
  return ret;
}
int main(){
  int T; scanf("%d", &T);
  for ( int cs = 1 ; cs <= T ; cs ++ ){
    scanf( "%d%d%d" , &n , &m , &k );
    memset( G , 0x3f , sizeof( G ) );
    for( int i = 1 ; i <= n ; i ++ ) G[ i ][ i ] = 0;
    for( int i = 0 ; i < m ; i ++ ){
      int u, v, w;
            scanf( "%d%d%d" , &u , &v , &w );
      G[ u ][ v ] = G[ v ][ u ] = w;
    }
    if( k & 1 ){ puts( "Impossible" ); continue; }
    for( int tk = 1; tk <= n ; tk ++ )
      for( int i = 1 ; i <= n ; i ++ )
        for( int j = 1 ; j <= n ; j ++ )
          G[ i ][ j ] = min( G[ i ][ j ] , G[ i ][ tk ]
               + G[ tk ][ j ] );
    for( int i = 1 ; i <= k ; i ++ ){
      for( int j = 1 ; j <= k ; j ++ )
        G[ i ][ j ] = -G[ i ][ j ];
      G[ i ][ i ] = -inf;
    }
    printf( "%d\n" , -MaxWeightMatch() );
  }
}
```

Data Structure

Sparse Table

```
const int MAXN = 200005;
const int lgN = 20;

struct SP{ //sparse table
  int Sp[MAXN][lgN];
  function<int(int,int)> opt;
  void build(int n, int *a){ // 0 base
    for (int i=0 ;i<n; i++) Sp[i][0]=a[i];

    for (int h=1; h<lgN; h++){
      int len = 1<<(h-1), i=0;
      for (; i+len<n; i++)
        Sp[i][h] = opt( Sp[i][h-1] , Sp[i+len][h-1] );
      for (; i<n; i++)
        Sp[i][h] = Sp[i][h-1];
    }
  }
  int query(int l, int r){
    int h = __lg(r-l+1);
    int len = 1<<h;
    return opt( Sp[l][h] , Sp[r-len+1][h] );
  }
};
```

Treap

```
#include<bits/stdc++.h>
using namespace std;
template<class T,unsigned seed>class treap{
  public:
    struct node{
      T data;
      int size;
      node *l,*r;
      node(T d){
        size=1;
        data=d;
        l=r=NULL;
      }
      inline void up(){
        size=1;
        if(l)size+=l->size;
        if(r)size+=r->size;
      }
      inline void down(){
      }
    }*root;
    inline int size(node *p){return p?p->size:0;}
    inline bool ran(node *a,node *b){
      static unsigned x=seed;
      x=0xdefaced*x+1;
      unsigned all=size(a)+size(b);
      return (x%all+all)%all<size(a);
    }
    void clear(node *&p){
      if(p)clear(p->l),clear(p->r),delete p,p=NULL;
    }
    ~treap(){clear(root);}
    void split(node *o,node *&a,node *&b,int k){
      if(!k)a=NULL,b=o;
      else if(size(o)==k)a=o,b=NULL;
      else{
        o->down();
        if(k<=size(o->l)){
          b=o;
          split(o->l,a,b->l,k);
          b->up();
        }else{
          a=o;
          split(o->r,a->r,b,k-size(o->l)-1);
          a->up();
        }
      }
    }
    void merge(node *&o,node *a,node *b){
      if(!a||!b)o=a?a:b;
      else{
        if(ran(a,b)){
          a->down();
          o=a;
          merge(o->r,a->r,b);
        }else{
          b->down();
          o=b;
          merge(o->l,a,b->l);
        }
        o->up();
      }
    }
    void build(node *&p,int l,int r,T *s){
      if(l>r)return;
      int mid=(l+r)>>1;
      p=new node(s[mid]);
      build(p->l,l,mid-1,s);
      build(p->r,mid+1,r,s);
      p->up();
    }
    inline int rank(T data){
      node *p=root;
      int cnt=0;
      while(p){
        if(data<=p->data)p=p->l;
        else cnt+=size(p->l)+1,p=p->r;
      }
```

```
      return cnt;
    }
    inline void insert(node *&p,T data,int k){
      node *a,*b,*now;
      split(p,a,b,k);
      now=new node(data);
      merge(a,a,now);
      merge(p,a,b);
    }
};
treap<int ,20141223>bst;
int n,m,a,b;
int main(){
  //當成二分查找樹用
  while(~scanf("%d",&a))bst.insert(bst.root,a,bst.rank(
      a));
  while(~scanf("%d",&a))printf("%d\n",bst.rank(a));
  bst.clear(bst.root);
  return 0;
}
```

2D Range Tree

```
// remember sort x !!!!!
typedef int T;
const int LGN = 20;
const int MAXN = 100005;

struct Point{
    T x, y;
    friend bool operator < (Point a, Point b){
        return tie(a.x,a.y) < tie(b.x,b.y);
    }
};
struct TREE{
    Point pt;
    int toleft;
}tree[LGN][MAXN];
struct SEG{
    T mx, Mx;
    int sz;
    TREE *st;
}seg[MAXN*4];

vector<Point> P;

void build(int l, int r, int o, int deep){
    seg[o].mx = P[l].x;
    seg[o].Mx = P[r].x;
    seg[o].sz = r-l+1;;

    if(l == r){
        tree[deep][r].pt = P[r];
        tree[deep][r].toleft = 0;
        seg[o].st = &tree[deep][r];
        return;
    }
    int mid = (l+r)>>1;
    build(l,mid,o+o,deep+1);
    build(mid+1,r,o+o+1,deep+1);

    TREE *ptr = &tree[deep][l];
    TREE *pl = &tree[deep+1][l], *nl = &tree[deep+1][
        mid+1];
    TREE *pr = &tree[deep+1][mid+1], *nr = &tree[deep
        +1][r+1];

    int cnt = 0;
    while(pl != nl && pr != nr) {
        *(ptr) = pl->pt.y <= pr->pt.y ? cnt++, *(pl++):
             *(pr++);
        ptr -> toleft = cnt; ptr++;
    }
    while(pl != nl) *(ptr) = *(pl++), ptr -> toleft =
        ++cnt, ptr++;
    while(pr != nr) *(ptr) = *(pr++), ptr -> toleft =
        cnt, ptr++;

}
int main(){
    int n; cin >> n;
    for(int i = 0 ;i < n; i++){
        T x,y; cin >> x >> y;
        P.push_back((Point){x,y});
    }
    sort(P.begin(),P.end());
    build(0,n-1,1,0);
}
```

ext heap

```
#include <bits/extc++.h>
typedef __gnu_pbds::priority_queue<int> heap_t;
heap_t a,b;

int main() {
  a.clear();
  b.clear();
  a.push(1);
  a.push(3);
  b.push(2);
  b.push(4);
  assert(a.top() == 3);
  assert(b.top() == 4);
  // merge two heap
  a.join(b);
  assert(a.top() == 4);
  assert(b.empty());

  return 0;
}
```

KD tree

```
// from BCW

const int MXN = 100005;

struct KDTree {
  struct Node {
    int x,y,x1,y1,x2,y2;
    int id,f;
    Node *L, *R;
  }tree[MXN];
  int n;
  Node *root;

  long long dis2(int x1, int y1, int x2, int y2) {
    long long dx = x1-x2;
    long long dy = y1-y2;
    return dx*dx+dy*dy;
  }
  static bool cmpx(Node& a, Node& b){ return a.x<b.x; }
  static bool cmpy(Node& a, Node& b){ return a.y<b.y; }
  void init(vector<pair<int,int>> ip) {
    n = ip.size();
    for (int i=0; i<n; i++) {
      tree[i].id = i;
      tree[i].x = ip[i].first;
      tree[i].y = ip[i].second;
    }
    root = build_tree(0, n-1, 0);
  }
  Node* build_tree(int L, int R, int dep) {
    if (L>R) return nullptr;
    int M = (L+R)/2;
    tree[M].f = dep%2;
    nth_element(tree+L, tree+M, tree+R+1, tree[M].f ?
        cmpy : cmpx);
```

```
    tree[M].x1 = tree[M].x2 = tree[M].x;
    tree[M].y1 = tree[M].y2 = tree[M].y;

    tree[M].L = build_tree(L, M-1, dep+1);
    if (tree[M].L) {
      tree[M].x1 = min(tree[M].x1, tree[M].L->x1);
      tree[M].x2 = max(tree[M].x2, tree[M].L->x2);
      tree[M].y1 = min(tree[M].y1, tree[M].L->y1);
      tree[M].y2 = max(tree[M].y2, tree[M].L->y2);
    }

    tree[M].R = build_tree(M+1, R, dep+1);
    if (tree[M].R) {
      tree[M].x1 = min(tree[M].x1, tree[M].R->x1);
      tree[M].x2 = max(tree[M].x2, tree[M].R->x2);
      tree[M].y1 = min(tree[M].y1, tree[M].R->y1);
      tree[M].y2 = max(tree[M].y2, tree[M].R->y2);
    }

    return tree+M;
  }
  int touch(Node* r, int x, int y, long long d2){
    long long dis = sqrt(d2)+1;
    if (x<r->x1-dis || x>r->x2+dis || y<r->y1-dis || y>
        r->y2+dis)
      return 0;
    return 1;
  }
  void nearest(Node* r, int x, int y, int &mID, long
      long &md2) {
    if (!r || !touch(r, x, y, md2)) return;
    long long d2 = dis2(r->x, r->y, x, y);
    if (d2 < md2 || (d2 == md2 && mID < r->id)) {
      mID = r->id;
      md2 = d2;
    }
    // search order depends on split dim
    if ((r->f == 0 && x < r->x) ||
        (r->f == 1 && y < r->y)) {
      nearest(r->L, x, y, mID, md2);
      nearest(r->R, x, y, mID, md2);
    } else {
      nearest(r->R, x, y, mID, md2);
      nearest(r->L, x, y, mID, md2);
    }
  }
  int query(int x, int y) {
    int id = 1029384756;
    long long d2 = 102938475612345678LL;
    nearest(root, x, y, id, d2);
    return id;
  }
}tree;
```

Link Cut tree

```
// from bcw codebook

const int MXN = 100005;
const int MEM = 100005;

struct Splay {
  static Splay nil, mem[MEM], *pmem;
  Splay *ch[2], *f;
  int val, rev, size;
  Splay () : val(-1), rev(0), size(0) {
    f = ch[0] = ch[1] = &nil;
  }
  Splay (int _val) : val(_val), rev(0), size(1) {
    f = ch[0] = ch[1] = &nil;
  }
  bool isr() {
    return f->ch[0] != this && f->ch[1] != this;
  }
  int dir() {
    return f->ch[0] == this ? 0 : 1;
  }
  void setCh(Splay *c, int d) {
    ch[d] = c;
    if (c != &nil) c->f = this;
    pull();
  }
  void push() {
    if (rev) {
      swap(ch[0], ch[1]);
      if (ch[0] != &nil) ch[0]->rev ^= 1;
      if (ch[1] != &nil) ch[1]->rev ^= 1;
      rev=0;
    }
  }
  void pull() {
    size = ch[0]->size + ch[1]->size + 1;
    if (ch[0] != &nil) ch[0]->f = this;
    if (ch[1] != &nil) ch[1]->f = this;
  }
} Splay::nil, Splay::mem[MEM], *Splay::pmem = Splay::
    mem;
Splay *nil = &Splay::nil;

void rotate(Splay *x) {
  Splay *p = x->f;
  int d = x->dir();
  if (!p->isr()) p->f->setCh(x, p->dir());
  else x->f = p->f;
  p->setCh(x->ch[!d], d);
  x->setCh(p, !d);
  p->pull(); x->pull();
}

vector<Splay*> splayVec;
void splay(Splay *x) {
  splayVec.clear();
  for (Splay *q=x;; q=q->f) {
    splayVec.push_back(q);
    if (q->isr()) break;
  }
  reverse(begin(splayVec), end(splayVec));
  for (auto it : splayVec) it->push();
  while (!x->isr()) {
    if (x->f->isr()) rotate(x);
    else if (x->dir()==x->f->dir()) rotate(x->f),rotate
        (x);
    else rotate(x),rotate(x);
  }
}

Splay* access(Splay *x) {
  Splay *q = nil;
  for (;x!=nil;x=x->f) {
    splay(x);
    x->setCh(q, 1);
    q = x;
  }
  return q;
}
void evert(Splay *x) {
  access(x);
  splay(x);
  x->rev ^= 1;
  x->push(); x->pull();
}
void link(Splay *x, Splay *y) {
//  evert(x);
  access(x);
  splay(x);
  evert(y);
  x->setCh(y, 1);
}
void cut(Splay *x, Splay *y) {
//  evert(x);
  access(y);
  splay(y);
  y->push();
```

```
  y->ch[0] = y->ch[0]->f = nil;
}

int N, Q;
Splay *vt[MXN];

int ask(Splay *x, Splay *y) {
  access(x);
  access(y);
  splay(x);
  int res = x->f->val;
  if (res == -1) res=x->val;
  return res;
}
int main(int argc, char** argv) {
  scanf("%d%d", &N, &Q);
  for (int i=1; i<=N; i++)
    vt[i] = new (Splay::pmem++) Splay(i);
  while (Q--) {
    char cmd[105];
    int u, v;
    scanf("%s", cmd);
    if (cmd[1] == 'i') {
      scanf("%d%d", &u, &v);
      link(vt[v], vt[u]);
    } else if (cmd[0] == 'c') {
      scanf("%d", &v);
      cut(vt[1], vt[v]);
    } else {
      scanf("%d%d", &u, &v);
      int res=ask(vt[u], vt[v]);
      printf("%d\n", res);
    }
  }

  return 0;
}
```

String

KMP

```
template<typename T>
void build_KMP(int n, T *s, int *f){ // 1 base
  f[0]=-1, f[1]=0;
  for (int i=2; i<=n; i++){
    int w = f[i-1];
    while (w>=0 && s[w+1]!=s[i])w = f[w];
    f[i]=w+1;
  }
}

template<typename T>
int KMP(int n, T *a, int m, T *b){
  build_KMP(m,b,f);
  int ans=0;

  for (int i=1, w=0; i<=n; i++){
    while ( w>=0 && b[w+1]!=a[i] )w = f[w];
    w++;
    if (w==m){
      ans++;
      w=f[w];
    }
  }
  return ans;
}
```

AC 自動機

```
// remember make_fail() !!!
// notice MLE

const int sigma = 62;
const int MAXC = 200005;

inline int idx(char c){
    if ('A'<= c && c <= 'Z')return c-'A';
    if ('a'<= c && c <= 'z')return c-'a' + 26;
    if ('0'<= c && c <= '9')return c-'0' + 52;
}

struct ACautomaton{
    struct Node{
        Node *next[sigma], *fail;
        int cnt; // dp
        Node(){
            memset(next,0,sizeof(next));
            fail=0;
            cnt=0;
        }
    } buf[MAXC], *bufp, *ori, *root;

    void init(){
        bufp = buf;
        ori = new (bufp++) Node();
        root = new (bufp++) Node();
    }

    void insert(int n, char *s){
        Node *ptr = root;
        for (int i=0; s[i]; i++){
            int c = idx(s[i]);
            if (ptr->next[c]==NULL)
                ptr->next[c] = new (bufp++) Node();
            ptr = ptr->next[c];
        }
        ptr->cnt=1;
    }

    Node* trans(Node *o, int c){
        while (o->next[c]==NULL) o = o->fail;
        return o->next[c];
    }

    void make_fail(){
        static queue<Node*> que;

        for (int i=0; i<sigma; i++)
            ori->next[i] = root;
        root->fail = ori;

        que.push(root);
        while ( que.size() ){
            Node *u = que.front(); que.pop();
            for (int i=0; i<sigma; i++){
                if (u->next[i]==NULL)continue;
                u->next[i]->fail = trans(u->fail,i);
                que.push(u->next[i]);
            }
            u->cnt += u->fail->cnt;
        }
    }
} ac;
```

Z-value

```
z[0] = 0;
for ( int bst = 0, i = 1; i < len ; i++ ) {
  if ( z[bst] + bst <= i ) z[i] = 0;
  else z[i] = min(z[i - bst], z[bst] + bst - i);
  while ( str[i + z[i]] == str[z[i]] ) z[i]++;
  if ( i + z[i] > bst + z[bst] ) bst = i;
}

// 回文版
```

```
void Zpal(const char *s, int len, int *z) {
    // Only odd palindrome len is considered
    // z[i] means that the longest odd palindrom
        centered at
    // i is [i-z[i] .. i+z[i]]
    z[0] = 0;
    for (int b=0, i=1; i<len; i++) {
        if (z[b] + b >= i) z[i] = min(z[2*b-i], b+z[b]-
            i);
        else z[i] = 0;
        while (i+z[i]+1 < len and i-z[i]-1 >= 0 and
               s[i+z[i]+1] == s[i-z[i]-1]) z[i] ++;
        if (z[i] + i > z[b] + b) b = i;
    }
}
```

Suffix Array

```
const int MAX = 1020304;
int ct[MAX], he[MAX], rk[MAX];
int sa[MAX], tsa[MAX], tp[MAX][2];
void suffix_array(char *ip){
  int len = strlen(ip);
  int alp = 256;
  memset(ct, 0, sizeof(ct));
  for(int i=0;i<len;i++) ct[ip[i]+1]++;
  for(int i=1;i<alp;i++) ct[i]+=ct[i-1];
  for(int i=0;i<len;i++) rk[i]=ct[ip[i]];
  for(int i=1;i<len;i*=2){
    for(int j=0;j<len;j++){
      if(j+i>=len) tp[j][1]=0;
      else tp[j][1]=rk[j+i]+1;
      tp[j][0]=rk[j];
    }
    memset(ct, 0, sizeof(ct));
    for(int j=0;j<len;j++) ct[tp[j][1]+1]++;
    for(int j=1;j<len+2;j++) ct[j]+=ct[j-1];
    for(int j=0;j<len;j++) tsa[ct[tp[j][1]]++]=j;
    memset(ct, 0, sizeof(ct));
    for(int j=0;j<len;j++) ct[tp[j][0]+1]++;
    for(int j=1;j<len+1;j++) ct[j]+=ct[j-1];
    for(int j=0;j<len;j++)
      sa[ct[tp[tsa[j]][0]]++]=tsa[j];
    rk[sa[0]]=0;
    for(int j=1;j<len;j++){
      if( tp[sa[j]][0] == tp[sa[j-1]][0] &&
        tp[sa[j]][1] == tp[sa[j-1]][1] )
        rk[sa[j]] = rk[sa[j-1]];
      else
        rk[sa[j]] = j;
    }
  }
  for(int i=0,h=0;i<len;i++){
    if(rk[i]==0) h=0;
    else{
      int j=sa[rk[i]-1];
      h=max(0,h-1);
      for(;ip[i+h]==ip[j+h];h++);
    }
    he[rk[i]]=h;
  }
}
```

Suffix Automaton

```
// par : fail link
// val : a topological order ( useful for DP )
// go[x] : automata edge ( x is integer in [0,26) )

struct SAM{
  struct State{
    int par, go[26], val;
    State () : par(0), val(0){ FZ(go); }
    State (int _val) : par(0), val(_val){ FZ(go); }
  };
  vector<State> vec;
  int root, tail;

  void init(int arr[], int len){
    vec.resize(2);
    vec[0] = vec[1] = State(0);
    root = tail = 1;
    for (int i=0; i<len; i++)
      extend(arr[i]);
  }
  void extend(int w){
    int p = tail, np = vec.size();
    vec.PB(State(vec[p].val+1));
    for ( ; p && vec[p].go[w]==0; p=vec[p].par)
      vec[p].go[w] = np;
    if (p == 0){
      vec[np].par = root;
    } else {
      if (vec[vec[p].go[w]].val == vec[p].val+1){
        vec[np].par = vec[p].go[w];
      } else {
        int q = vec[p].go[w], r = vec.size();
        vec.PB(vec[q]);
        vec[r].val = vec[p].val+1;
        vec[q].par = vec[np].par = r;
        for ( ; p && vec[p].go[w] == q; p=vec[p].par)
          vec[p].go[w] = r;
      }
    }
    tail = np;
  }
};
```

迴文字動機

```
// remember init()      !!!
// remember make_fail() !!!
// insert s need 1 base !!!
// notice MLE
const int sigma = 62;
const int MAXC = 1000006;
inline int idx(char c){
    if ('a'<= c && c <= 'z')return c-'a';
    if ('A'<= c && c <= 'Z')return c-'A'+26;
    if ('0'<= c && c <= '9')return c-'0'+52;
}
struct PalindromicTree{
    struct Node{
        Node *next[sigma], *fail;
        int len, cnt; // for dp
        Node(){
            memset(next,0,sizeof(next));
            fail=0;
            len = cnt = 0;
        }
    } buf[MAXC], *bufp, *even, *odd;

    void init(){
        bufp = buf;
        even = new (bufp++) Node();
        odd  = new (bufp++) Node();
        even->fail = odd;
        odd->len = -1;
    }

    void insert(char *s){
        Node* ptr = even;
        for (int i=1; s[i]; i++){
            ptr = extend(ptr,s+i);
        }
    }

    Node* extend(Node *o, char *ptr){
        int c = idx(*ptr);
```

```
        while ( *ptr != *(ptr-1-o->len) )o=o->fail;
        Node *&np = o->next[c];
        if (!np){
            np = new (bufp++) Node();
            np->len = o->len+2;
            Node *f = o->fail;
            if (f){
                while ( *ptr != *(ptr-1-f->len) )f=f->
                    fail;
                np->fail = f->next[c];
            }
            else {
                np->fail = even;
            }
            np->cnt = np->fail->cnt;
        }
        np->cnt++;
        return np;
    }
} PAM;
```

smallest rotation

```
string mcp(string s){
  int n = s.length();
  s += s;
  int i=0, j=1;
  while (i<n && j<n){
    int k = 0;
    while (k < n && s[i+k] == s[j+k]) k++;
    if (s[i+k] <= s[j+k]) j += k+1;
    else i += k+1;
    if (i == j) j++;
  }
  int ans = i < n ? i : j;
  return s.substr(ans, n);
}
Contact GitHub API Training Shop Blog About
```

Dark Code

Search

Others

數位統計

```
int dfs(int pos, int state1, int state2 ....., bool
    limit, bool zero) {
    if ( pos == -1 ) return 是否符合條件;
    int &ret = dp[pos][state1][state2][....];
    if ( ret != -1 && !limit ) return ret;
    int ans = 0;
    int upper = limit ? digit[pos] : 9;
    for ( int i = 0 ; i <= upper ; i++ ) {
        ans += dfs(pos - 1, new_state1, new_state2,
            limit & ( i == upper), ( i == 0) && zero);
    }
    if ( !limit ) ret = ans;
    return ans;
}

int solve(int n) {
    int it = 0;
    for ( ; n ; n /= 10 ) digit[it++] = n % 10;
    return dfs(it - 1, 0, 0, 1, 1);
}
```

Stable Marriage

```
// normal stable marriage problem
// input:
//3
//Albert Laura Nancy Marcy
//Brad Marcy Nancy Laura
//Chuck Laura Marcy Nancy
//Laura Chuck Albert Brad
//Marcy Albert Chuck Brad
//Nancy Brad Albert Chuck

#include<bits/stdc++.h>
using namespace std;
const int MAXN = 505;

int n;
int favor[MAXN][MAXN]; // favor[boy_id][rank] = girl_id
    ;
int order[MAXN][MAXN]; // order[girl_id][boy_id] = rank
    ;
int current[MAXN]; // current[boy_id] = rank; boy_id
    will pursue current[boy_id] girl.
int girl_current[MAXN]; // girl[girl_id] = boy_id;

void initialize() {
  for ( int i = 0 ; i < n ; i++ ) {
    current[i] = 0;
    girl_current[i] = n;
    order[i][n] = n;
  }
}

map<string, int> male, female;
string bname[MAXN], gname[MAXN];
int fit = 0;

void stable_marriage() {

  queue<int> que;
  for ( int i = 0 ; i < n ; i++ ) que.push(i);
  while ( !que.empty() ) {
    int boy_id = que.front();
    que.pop();

    int girl_id = favor[boy_id][current[boy_id]];
    current[boy_id] ++;

    if ( order[girl_id][boy_id] < order[girl_id][
        girl_current[girl_id]] ) {
      if ( girl_current[girl_id] < n ) que.push(
          girl_current[girl_id]); // if not the first
          time
      girl_current[girl_id] = boy_id;
    } else {
      que.push(boy_id);
    }
  }

}

int main() {
  cin >> n;

  for ( int i = 0 ; i < n;  i++ ) {
    string p, t;
    cin >> p;
    male[p] = i;
    bname[i] = p;
    for ( int j = 0 ; j < n ; j++ ) {
      cin >> t;
      if ( !female.count(t) ) {
        gname[fit] = t;
        female[t] = fit++;
      }
      favor[i][j] = female[t];
    }
```

```
  }

  for ( int i = 0 ; i < n ; i++ ) {
    string p, t;
    cin >> p;
    for ( int j = 0 ; j < n ; j++ ) {
      cin >> t;
      order[female[p]][male[t]] = j;
    }
  }

  initialize();
  stable_marriage();

  for ( int i = 0 ; i < n ; i++ ) {
    cout << bname[i] << " " << gname[favor[i][current[i
        ] - 1]] << endl;
  }

}
```

STL

```
// algorithm
random_shuffle(a,a+n);
next_permutation(a,a+n); // need sort
nth_element (a, a+k, a+n); // kth
*min_element(a,a+n);
*unique(a,a+n); // need sort
stable_sort(a,a+n); // merge sort

// bitset (s[0] is right most)
operator[] //
count() // count number of 1
set() // all to 1
set(k) //   s[k] to 1
set(k,0) // s[k] to 0
flip()  // all flip
flip(k) // s[k] flip
to_ulong()
to_string()

// unique vector
sort(a.begin(),a.end())
a.earse( unique(a.begin(),a.end()), a.end() )
```

1D/1D dp 優化

```
#include<bits/stdc++.h>
#include<cmath>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
#define IOS ios_base::sync_with_stdio(0); cin.tie(0);
#define clean(n,val) memset((n),(val),sizeof(n))
#define MP make_pair
#define PB push_back
#define ll long long
#define debug(x) x
typedef pair<int, int> PI;
const int INF = 0xFFFFFFF;
const int MOD = 1e9;
const int MAXN = 100005;


int t, n, L;
int p;
char s[MAXN][35];
ll sum[MAXN] = {0};
long double dp[MAXN] = {0};
int prevd[MAXN] = {0};

long double pw(long double a, int n) {
    if ( n == 1 ) return a;
    long double b = pw(a, n/2);
    if ( n & 1 ) return b*b*a;
    else return b*b;
}

long double f(int i, int j) {
//    cout << (sum[i] - sum[j]+i-j-1-L) << endl;
    return pw(abs(sum[i] - sum[j]+i-j-1-L), p) + dp[j];
}

struct INV {
    int L, R, pos;
};

INV stk[MAXN*10];
int top = 1, bot = 1;

void update(int i) {
    while ( top > bot && i < stk[top].L && f(stk[top].L
        , i) < f(stk[top].L, stk[top].pos) ) {
        stk[top - 1].R = stk[top].R;
        top--;
    }

    int lo = stk[top].L, hi = stk[top].R, mid, pos =
        stk[top].pos;
    //if ( i >= lo ) lo = i + 1;
    while ( lo != hi ) {
        mid = lo + (hi - lo) / 2;
        if ( f(mid, i) < f(mid, pos) ) hi = mid;
        else lo = mid + 1;
    }

    if ( hi < stk[top].R ) {
        stk[top + 1] = (INV) { hi, stk[top].R, i };
        stk[top++].R = hi;
    }
}

int main() {
    #ifdef LOCAL
        freopen("input.txt", "r", stdin);
        //freopen("output.txt", "w", stdout);
    #endif // LOCAL

    cin >> t;
    while ( t-- ) {
        cin >> n >> L >> p;
        dp[0] = sum[0] = 0;
        for ( int i = 1 ; i <= n ; i++ ) {
            cin >> s[i];
            sum[i] = sum[i-1] + strlen(s[i]);
            dp[i] = numeric_limits<long double>::max();
        }

        stk[top] = (INV) {1, n + 1, 0};
        for ( int i = 1 ; i <= n ; i++ ) {
            if ( i >= stk[bot].R ) bot++;
            dp[i] = f(i, stk[bot].pos);
            update(i);
//            cout << (ll) f(i, stk[bot].pos) << endl;
        }

        if ( dp[n] > 1e18 ) {
            cout << "Too hard to arrange" << endl;
        } else {
            vector<PI> as;
            cout << (ll)dp[n] << endl;
        }
    }
```

```
    return 0;
}
```

python 小抄

```
#!/usr/bin/env python3

# 帕斯卡三角形
n = 10
dp = [ [1 for j in range(n)] for i in range(n) ]
for i in range(1,n):
    for j in range(1,n):
        dp[i][j] = dp[i][j-1] + dp[i-1][j]

for i in range(n):
    print( ' '.join( '{:5d}'.format(x) for x in dp[i] )
         )

# EOF
while True:
    try:
        n, m = map(int, input().split())
    except:
        break
    print( min(n,m), max(n,m) )

# input a sequence of number
a = [ int(x) for x in input().split() ]
a.sort()
print( ''.join( str(x)+' ' for x in a ) )

# LCS
ncase = int( input() )
for _ in range(ncase):
    n, m = [int(x) for x in input().split()]
    a, b = "$"+input(), "$"+input()

    dp = [ [int(0) for j in range(m+1)] for i in range(
        n+1) ]

    for i in range(1,n+1):
        for j in range(1,m+1):
            dp[i][j] = max(dp[i-1][j],dp[i][j-1])
            if a[i]==b[j]:
                dp[i][j] = max(dp[i][j],dp[i-1][j-1]+1)

    for i in range(1,n+1):
        print(dp[i][1:])

    print('a={:s}, b={:s}, |LCS(a,b)|={:d}'.format(a
        [1:],b[1:],dp[n][m]))

# Basic operator
a, b = 10, 20
a/b  # 0.5
a//b # 0
a%b  # 10
a**b # 10^20

# if, else if, else
if a==0:
    print('zero')
elif a>0:
    print('postive')
else:
    print('negative')

# stack         # C++
stack = [3,4,5]
stack.append(6) # push()
stack.pop()     # pop()
stack[-1]       # top()
len(stack)      # size() O(1)

# queue         # C++
from collections import deque
queue = deque([3,4,5])
queue.append(6) # push()
queue.popleft() # pop()
queue[0]        # front()
len(queue)      # size() O(1)
```

Bitwise Operation

```
inline int pop_count(int x){
  x = (0x55555555&x) + (0x55555555&(x>>1));
  x = (0x33333333&x) + (0x33333333&(x>>2));
  x = (0x0F0F0F0F&x) + (0x0F0F0F0F&(x>>4));
  x = (0x00FF00FF&x) + (0x00FF00FF&(x>>8));
  x = (0x0000FFFF&x) + (0x0000FFFF&(x>>16));
  return x;
}
```

矩陣數定理

新的方法介绍
下面我们介绍一种新的方法——Matrix-Tree定理(Kirchhoff矩阵-树定理)。

Matrix-Tree定理是解决生成树计数问题最有力的武器之一。它首先于1847年被Kirchhoff证明。在介绍定理之前，我们首先明确几个概念：
1、G的度数矩阵D[G]是一个n*n的矩阵，并且满足：当i≠j时，dij=0；当i=j时，dij等于vi的度数。
2、G的邻接矩阵A[G]也是一个n*n的矩阵， 并且满足：如果vi、vj之间有边直接相连，则aij=1，否则为0。
我们定义G的Kirchhoff矩阵(也称为拉普拉斯算子)C[G]为C[G]=D[G]-A[G]，
则Matrix-Tree定理可以描述为：G的所有不同的生成树的个数等于其Kirchhoff矩阵C[G]任何一个n-1阶主子式的行列式的绝对值。
所谓n-1阶主子式，就是对于r(1≤r≤n)，将C[G]的第r行、第r列同时去掉后得到的新矩阵，用Cr[G]表示。

生成树计数
算法步骤：
1、 构建拉普拉斯矩阵
     Matrix[i][j] =
degree(i) , i==j
          -1，i-j有边
           0，其他情况
2、 去掉第r行，第r列 (r任意)
3、 计算矩阵的行列式

```
/* ******************************************
MYID    : Chen Fan
LANG    : G++
PROG    : Count_Spaning_Tree_From_Kuangbin
****************************************** */
#include <stdio.h>
#include <string.h>
#include <algorithm>
#include <iostream>
#include <math.h>
using namespace std;
const double eps = 1e-8;
const int MAXN = 110;
int sgn(double x)
{
    if(fabs(x) < eps)return 0;
    if(x < 0)return -1;
    else return 1;
}
double b[MAXN][MAXN];
```

```
double det(double a[][MAXN],int n)
{
    int i, j, k, sign = 0;
    double ret = 1;
    for(i = 0;i < n;i++)
    for(j = 0;j < n;j++) b[i][j] = a[i][j];
    for(i = 0;i < n;i++)
    {
        if(sgn(b[i][i]) == 0)
        {
            for(j = i + 1; j < n;j++)
            if(sgn(b[j][i]) != 0) break;
            if(j == n)return 0;
            for(k = i;k < n;k++) swap(b[i][k],b[j][k]);
            sign++;
        }
        ret *= b[i][i];
        for(k = i + 1;k < n;k++) b[i][k]/=b[i][i];
        for(j = i+1;j < n;j++)
        for(k = i+1;k < n;k++) b[j][k] -= b[j][i]*b[i][
            k];
    }
    if(sign & 1)ret = -ret;
    return ret;
}
double a[MAXN][MAXN];
int g[MAXN][MAXN];
int main()
{
    int T;
    int n,m;
    int u,v;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d%d",&n,&m);
        memset(g,0,sizeof(g));
        while(m--)
        {
            scanf("%d%d",&u,&v);
            u--;v--;
            g[u][v] = g[v][u] = 1;
        }
        memset(a,0,sizeof(a));
        for(int i = 0;i < n;i++)
        for(int j = 0;j < n;j++)
        if(i != j && g[i][j])
        {
            a[i][i]++;
            a[i][j] = -1;
        }
        double ans = det(a,n-1);
        printf("%.0lf\n",ans);
    }
    return 0;
}
```

CYK

```
// 2016 NCPC from sunmoon

// 轉換

#define MAXN 55
struct CNF{
  int s,x,y;//s->xy | s->x, if y==-1
  int cost;
  CNF(){}
  CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
};
int state;//規則數量
map<char,int> rule;//每個字元對應到的規則，小寫字母為終
    端字符
vector<CNF> cnf;
inline void init(){
  state=0;
  rule.clear();
  cnf.clear();
}
inline void add_to_cnf(char s,const string &p,int cost)
    {
  if(rule.find(s)==rule.end())rule[s]=state++;
  for(auto c:p)if(rule.find(c)==rule.end())rule[c]=
      state++;
  if(p.size()==1){
    cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
  }else{
    int left=rule[s];
    int sz=p.size();
    for(int i=0;i<sz-2;++i){
      cnf.push_back(CNF(left,rule[p[i]],state,0));
      left=state++;
    }
    cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],
        cost));
  }
}

// 計算

vector<long long> dp[MAXN][MAXN];
vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會
    有無限小的情形
inline void relax(int l,int r,const CNF &c,long long
    cost,bool neg_c=0){
  if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[
      l][r][c.s])){
    if(neg_c||neg_INF[l][r][c.x]){
      dp[l][r][c.s]=0;
      neg_INF[l][r][c.s]=true;
    }else dp[l][r][c.s]=cost;
  }
}
inline void bellman(int l,int r,int n){
  for(int k=1;k<=state;++k)
    for(auto c:cnf)
      if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n)
          ;
}
inline void cyk(const vector<int> &tok){
  for(int i=0;i<(int)tok.size();++i){
    for(int j=0;j<(int)tok.size();++j){
      dp[i][j]=vector<long long>(state+1,INT_MAX);
      neg_INF[i][j]=vector<bool>(state+1,false);
    }
    dp[i][i][tok[i]]=0;
    bellman(i,i,tok.size());
  }
  for(int r=1;r<(int)tok.size();++r){
    for(int l=r-1;l>=0;--l){
      for(int k=l;k<r;++k)
        for(auto c:cnf)
          if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][
              c.y]+c.cost);
      bellman(l,r,tok.size());
    }
  }
}
```

DP 優化

單調性

第二種優化的方法是使用轉移方程本身的特性，以減少需考慮
    的子狀態數目。在講這種優化
前我們先定義兩種 DP 的問題：
1D/1D DP[j] = min(0≤i<j) { DP[i] + w(i, j) }; DP[0] = k
2D/1D DP[i][j] = min(i<k≤j) { DP[i][k - 1] + DP[k][j] }
    + w(i, j); DP[i][i] = 0
定義 DP 的類型後，我們來定義單調性。

1. 凹四邊形不等式 (concave Monge condition)
若 A 要滿足凹四邊形不等式，則對任意 a < b, c < d 要有 A[a][c] + A[b][d] ≥ A[a][d] + A[b][c]
2. 凸四邊形不等式 (convex Monge condition)
若 A 要滿足凸四邊形不等式，則對任意 a < b, c < d 要有 A[a][c] + A[b][d] ≤ A[a][d] + A[b][c]
3. 凹完全單調性 (concave totally monotone)
若 A 要滿足凹完全單調性，則對任意 a < b, c < d 要有 A[a][c] ≤ A[b][c] => A[a][d] ≤ A[b][d]
4. 凸完全單調性 (convex totally monotone)
若 A 要滿足凸完全單調性，則對任意 a < b, c < d 要有 A[a][c] ≥ A[b][c] => A[a][d] ≥ A[b][d]
其中由 1 可推得 3，由 2 可推得 4。而要判斷一個函數 w(i, j)是否符合凹四邊形不等式，等價於要檢驗 w(i, j) + w(i + 1, j + 1) ≥ w(i, j + 1) + w(i + 1, j) 是否成立。同理，若我們要判斷函式 w(i, j)是否符合凸四邊形不等式，也只需檢驗 w(i, j) + (i + 1, j + 1) ≤ w(i, j + 1) + w(i + 1, j)。

凹性 1D/1D 優化

若定義 F[i][j] = DP[i] + w(i, j)，則若 w(i, j)符合凹四邊形不等式，可推知 F 必定也符合。此時 F[i][j]即具有凹完全單調性。而根據定義，DP[j]會是 min0≤i<jF[i][j]，所以我們用一資料結構來維護每條尚未計算之行的當前最小值在哪。且若 DP[j]已完成，由凹四邊形不等式，F[j + 1] ~ F[n]的當前最佳解位置必遞減。

右圖是剛得出 DP[j]而未將 DP[j]加入資料結構前的狀態，一但我們成功將第 j 列加入後，我們即可直接將第 j+1 行的當前最佳解視為 DP[j + 1]之值。(因為此時 F[0][j + 1] ~ F[j][j + 1]都被考慮到了)所以 DP 的瓶頸是在我們如何維護此結構。

一般而言對 1D/1D 凹的問題我們使用 Stack 來維護當前最佳解，Stack 中的每個元素都包含三個值(L, R, p)，代表在 L ~ R 行得當前最佳解是 p。當我們要加入新的一行 j 時，有兩種 case：

1. F[j][j + 1] ≥ F[Stack.top().p][j + 1]
這代表在第 j+1 行時最佳解的位置就已經比 j 還要小了，故此行根本無需加入。

2. F[j][j + 1] < F[Stack.top().p][j + 1]
此時代表列 j 是有必要加入的，於是我們從 Stack 的頂端開始考慮。若以(nL, nR, np)代表當前Stack 頂端元素的(nL, nR, np)則若 F[j][nR] < F[np][nR]就將 Stack 頂端的元素丟棄，如此反覆直到 Stack 已經空了或是 F[j][nR] ≥ F[np][nR]。當發生第一種狀況時直接 push 入(j + 1, n, j)即可，然而若是第二種狀況則需在(nL, nR)之間二分搜，找到第一次發生列 np 比列 j 好的地方。設此點為 m，則需將 Stack 頂端之元素修正為(m, nR, np)後再 push 入(j + 1, m - 1, j)。最後只需在取值後判斷 Stack頂端之元素是否過期(nR < j + 1)，若是即將之 pop 出便可。根據以上方法我們可得一算法在 O(nlgn)內求出 DP[n]。

凸性 1D/1D 優化
凸性的 1D/1D 優化與凹性十分相似。在凸四邊形不等式的影響下，當前最佳解的位置是呈現遞增，所以在凸性 1D/1D 問題中我們改用 Deque 來維護當前最佳位置。要插入新的一列時，我們就從右邊開始刪除元素，最後一樣二分搜出確切位置。而要取值時則是從左側取出，過期的東西也是從左側拿出。

這個作法的複雜度與凹性 1D/1D 一模一樣是 O(nlgn)。不過不管是凹性還是凸性，如果能利用 w(i, j)函數的特性而在 O(1)的時間內計算出 m 點所在，複雜度便可降至 O(n)。

對於凹性的 2D/1D 問題可考慮枚舉一維後每次視為 1D/1D 的問題。定義：
DP_i[j] = min0≤k<j{ DP_i[k] + u_i(k, j) } (1 ≤ j ≤ n - i)

u_i(k, j) = DP_(i+k+1)[j - k - 1] + w(i, i + j)

若可證明 u_i(k, j)符合凹四邊形不等式，則可以凹性 1D/1D 之法解 DP_i。總複雜度為(n^2lgn)。
不過要注意的是因為 i 值較小的 u 會用到 i 值較大的 DP，故枚舉 i 值時須從大枚舉至小。

凸性 2D/1D 優化

如果 w(i, j)符合凸單調性，且有 w(i, i + 2) ≥ max(w(i, i + 1), w(i + 1, i + 2))，則 DP[i][j]也會符合凸四邊形不等式。
相較於凹性 2D/1D，凸性 2D/1D 能作的優化更多，他有一個重要的定理：
令 K[i, j]為使 DP[i][j]達到最小值的 k 值，則有 K[i][j - 1] ≤ K[i][j] ≤ K[i + 1][j]。

因為以上定理，我們 DP 時，可以從 j - i 較小的狀態開始 DP 起。當我們在作 DP[i][j]時我們只需枚舉 K[i][j - 1] ~ K[i + 1][j]，故對於所有 j - i = c 的狀態，將他們都求出所需枚舉的狀態數只有：
K[2][1 + c] - K[1][c] + 1 + K[3][2 + c] - K[2][1 + c] + 1 + K[4][3 + c] - K[3][2 + c] + 1 ….
= K[n + 1 - c][n] - K[1][c] + n - c = O(n)。
而 j - i 總共只有 O(n)種，故求出整個 DP 表格便只需要 O(n^2)，比起直接枚舉快了一維。切記 2D/1D 的凸單調性優化算是這四種中在競賽最常見的，甚至在 TOI 模考中也出過，所以就算其他三種單調優化都不會，也得弄清楚這種。

Mo’s algorithm

```
int l = 0, r = 0, nowAns = 0, BLOCK_SIZE, n, m;
int ans[];
struct QUE{
    int l, r, id;
    friend bool operator < (QUE a, QUE b){
        if(a.l / BLOCK_SIZE != b.l / BLOCK_SIZE)
            return a.l / BLOCK_SIZE < b.l / BLOCK_SIZE;
        return a.r < b.r;
    }
}querys[];

inline void move(int pos, int sign) {
    // update nowAns
}

void solve() {
    BLOCK_SIZE = int(ceil(pow(n, 0.5)));
    sort(querys, querys + m);
    for (int i = 0; i < m; ++i) {
        const QUE &q = querys[i];
        while (l > q.l) move(--l, 1);
        while (r < q.r) move(r++, 1);
        while (l < q.l) move(l++, -1);
        while (r > q.r) move(--r, -1);
        ans[q.id] = nowAns;
    }
}
```

Persistence