

Contents

1	Basics.....	1
1.1	random.....	1
1.2	time.....	1
2	flow.....	1
2.1	ISAP.....	1
2.2	MinCostMaxFlow.....	2
2.3	Dinic.....	2
2.4	有源匯有上下界最大流.....	2
2.5	有源匯有上下界最小流.....	3
2.6	無源匯有上下界可行流.....	4
2.7	最大權閉合圖.....	4
2.8	最大密度子圖.....	4
2.9	最小割樹(Gomory-Hu Tree).....	4
2.10	Max Cost Circulation.....	4
3	Math.....	5
3.1	質數與質因數分解(附 moebius 和 phi).....	5
3.2	Miller Rabin(大質數判定).....	5
3.3	pollardRho(找大整數的因數).....	5
3.4	FFT.....	5
3.5	FWT.....	6
3.6	中國剩餘定理(附 extgcd).....	6
3.7	高斯消去法.....	6
3.8	歐拉函數.....	7
3.9	mod 奇質數下的一個平方根.....	7
3.10	mod 奇質數下的 m 次方根.....	7
3.11	Burnside's lemma.....	7
3.12	Lucas's theorem.....	7
3.13	Sum of Two Squares Thm (Legendre).....	7
3.14	Difference of D1-D3 Thm.....	7
4	Geometry.....	7
4.1	幾何們.....	7
4.2	旋轉卡殼(最遠距離對).....	10
4.3	皮克(Pick)定理.....	10
4.4	Minkowski sum.....	10
4.5	三角形的三心.....	10
4.6	Circle Cover.....	10
4.7	minimum enclosing circle.....	11
4.8	minimum enclosing ball.....	11
4.9	矩形重疊面積.....	12
5	Graph.....	12
5.1	HeavyLightDecomp (附 LCA).....	12
5.2	centroid decomposition.....	13
5.3	BCC 割點.....	13
5.4	BCC 橋.....	14
5.5	SCC.....	14
5.6	2-SAT.....	14
5.7	有向最小生成樹(最小樹形圖).....	14
5.8	二分圖匹配(Bipartite Matching).....	15
5.9	二分圖最佳完美匹配(Kuhn Munkres).....	15
5.10	Maximum General graph Matching.....	16
5.11	無向圖最小割(SW min-cut).....	16
5.12	最大團.....	16
5.13	最大團數量.....	17
5.14	Minimum mean cycle.....	17
5.15	Directed Graph Min Cost Cycle.....	18
5.16	Minimum Steiner Tree.....	18
5.17	DominatorTree.....	19
5.18	The first k Shortest Path.....	19
5.19	SPFA.....	21
5.20	DLX.....	21
5.21	混合圖歐拉迴路判定.....	21
5.22	Euler tour.....	21
5.23	Stable Marriage Problem.....	22
6	String.....	22
6.1	KMP.....	22
6.2	Suffix array.....	22
6.3	Trie 與 AC 自動機.....	23
6.4	BWT.....	23
7	Data Structure.....	23
7.1	李超樹.....	23
7.2	KD tree.....	24
7.3	Leftist Heap.....	25

7.4	treap.....	25
8	Others.....	25

1 Basics

1.1 random

```
srand(time(0)); rand()隨機產生數字
random_shuffle(v.begin(), v.end()) //隨機排列
```

1.2 time

```
double START, END; START = clock();
/*---要計算的程式效率區域---*/
END = clock();
cout << (END - START) / CLOCKS_PER_SEC << endl;
```

2 flow

2.1 ISAP

不能慢慢增流!! 要增流請用 Dinic。

```
#define SZ(c) ((int)(c).size())
struct Maxflow{
    typedef int type;
    static const int MAXV = 20010;
    type INF = 1000000; // type 改變這裡也要跟著變
    struct Edge{
        int v, r;
        type c;
        Edge(int _v, type _c, int _r) :
            v(_v), c(_c), r(_r) {}
    };
    int s, t;
    vector<Edge> G[MAXV * 2];
    int iter[MAXV * 2], d[MAXV * 2], gap[MAXV * 2], tot;
    void init(int x){
        tot = x + 2;
        s = x + 1, t = x + 2;
        for (int i = 0; i <= tot; i++){
            G[i].clear();
            iter[i] = d[i] = gap[i] = 0;
        }
    }
    void addEdge(int u, int v, type c){
        G[u].push_back(Edge(v, c, SZ(G[v])));
        G[v].push_back(Edge(u, 0, SZ(G[u]) - 1));
    }
    type dfs(int p, type flow){
        if (p == t)
            return flow;
        for (int &i = iter[p]; i < SZ(G[p]); i++){
            Edge &e = G[p][i];
            if (e.c > 0 && d[p] == d[e.v] + 1){
                type f = dfs(e.v, min(flow, e.c));
                if (f){
                    e.c -= f;
                    G[e.v][e.r].c += f;
                    return f;
                }
            }
        }
        if ((--gap[d[p]]) == 0) d[s] = tot;
        else{
            d[p]++;
            iter[p] = 0;
            ++gap[d[p]];
        }
        return 0;
    }
    type solve(){
        type res = 0;
        gap[0] = tot;
        for (res = 0; d[s] < tot; res += dfs(s, INF));
        return res;
    }
} flow;
```

2.2 MinCostMaxFlow

```

struct MinCostMaxFlow{
    typedef int Tcost;
    static const int MAXV = 20010;
    static const int INFf = 1000000;
    static const Tcost INFc = 1e9;
    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){
        V = n + 2;
        s = n + 1, t = n + 2;
        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;
    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 mnc;
    }
} flow;

```

2.3 Dinic

可以慢慢增流，再叫一次 `flow.solve()` 會輸出增加的流量。

```

struct Dinic{
    static const int MAXV = 10005;
    typedef ll type;
    #define inf 9999999999999999ll
    struct Edge{
        int from, to;
        type cap, flow; int ori;
    };
    int N, s, t;
    vector<Edge> edges;
    vector<int> G[MAXV];
    bool vis[MAXV];
    int d[MAXV];
    int cur[MAXV];
    void init(int _n){
        N = _n + 2; s = _n + 1; t = _n + 2;
        edges.clear();
        for (int i = 0; i <= N; i++) G[i].clear();
    }
    void add_edge(int from, int to, type cap){
        edges.push_back(Edge{from, to, cap, 0, 1});
        edges.push_back(Edge{to, from, 0, 0, 0});
        int m = edges.size();
        G[from].push_back(m - 2);
        G[to].push_back(m - 1);
    }
    bool BFS(){
        memset(vis, 0, sizeof(vis));
        queue<int> q;
        q.push(s);
        d[s] = 0; vis[s] = 1;
        while (!q.empty()){
            int x = q.front();
            q.pop();
            for (int i = 0; i < G[x].size(); i++){
                Edge &e = edges[G[x][i]];
                if (!vis[e.to] && e.cap > e.flow){
                    vis[e.to] = 1;
                    d[e.to] = d[x] + 1;
                    q.push(e.to);
                }
            }
        }
        return vis[t];
    }
    type DFS(int x, type a){
        if (x == t || a == 0) return a;
        type flow = 0, f;
        for (int &i = cur[x]; i < G[x].size(); i++){
            Edge &e = edges[G[x][i]];
            if (d[x] + 1 == d[e.to] && (f = DFS(e.to, min(a,
            e.cap - e.flow))) > 0){
                e.flow += f;
                edges[G[x][i] ^ 1].flow -= f;
                flow += f;
                a -= f;
                if (a == 0) break;
            }
        }
        return flow;
    }
    type solve(){
        type flow = 0;
        while (BFS()){
            memset(cur, 0, sizeof(cur));
            flow += DFS(s, inf);
        }
        return flow;
    }
} flow;

```

2.4 有源匯有上下界最大流

```

// 1<=點數<=202, 1<=邊數<=999
#define MAXN 70005
int sp, tp, cnt, head[210], nxt[MAXN], to[MAXN],
cap[MAXN], dis[1010], st, de, def[210], n;
inline void addedge(int u, int v, int p){
    nxt[++cnt] = head[u], head[u] = cnt, to[cnt] = v,

```

```

cap[cnt] = p;
    nxt[++cnt] = head[v], head[v] = cnt, to[cnt] = u,
cap[cnt] = 0;
}
inline bool bfs(){
    int u, e, v;
    queue<int> que;
    memset(dis, -1, sizeof(dis));
    que.push(sp), dis[sp] = 0;
    while (!que.empty()){
        u = que.front(), que.pop();
        for (int e = head[u]; e; e = nxt[e]){
            if (cap[e] > 0 && dis[v = to[e]] == -1){
                dis[v] = dis[u] + 1, que.push(v);
                if (v == tp)
                    return true;
            }
        }
    }
    return false;
}
inline int dfs(const int &u, const int &flow){
    if (u == tp)
        return flow;
    int res = 0, v, flw;
    for (int e = head[u]; e; e = nxt[e]){
        if (cap[e] > 0 && dis[u] < dis[v = to[e]]){
            flw = dfs(v, min(cap[e], flow - res));
            if (flw == 0)
                dis[v] = -1;
            cap[e] -= flw, cap[e ^ 1] += flw;
            res += flw;
            if (res == flow)
                break;
        }
    }
    return res;
}
inline int dinic(int sp, int tp){
    int ans = 0;
    while (bfs()){
        ans += dfs(sp, 1 << 30);
    }
    return ans;
}
void init(int _n, int _st, int _de){
    n = _n, st = _st, de = _de;
    cnt = 1;
    sp = tp = 0;
    memset(head, -1, sizeof(head));
    memset(def, 0, sizeof(def));
}
void build(int s, int t, int down, int up)
{ //從 s 到 t 的邊，流量限制在區間[down,up]
    addedge(s, t, up - down);
    def[s] += down, def[t] -= down;
}
int solve(){
    int sum = 0;
    sp = n + 1, tp = n + 2;
    for (int i = 1; i <= n; i++){
        if (def[i] > 0)
            sum += def[i], addedge(i, tp, def[i]);
        if (def[i] < 0)
            addedge(sp, i, -def[i]);
    }
    addedge(de, st, 1 << 30);
    if (dinic(sp, tp) == sum){
        head[sp] = 0, head[tp] = 0;
        sp = st;
        tp = de;
        return dinic(sp, tp);
    }
    else return -1; //無可行解
}

```

2.5 有源匯有上下界最小流

n 個點， m 條邊，每條邊 e 有一個流量下界 $lower(e)$ 和流量上界 $upper(e)$ ，給定源點 s 與匯點 t ，求源點到匯點的最小流。

輸入格式

第一行兩個正整數 n, m, s, t 。

之後的 m 行，每行四個整數 $s, t, lower, upper$ 。

輸出格式

如果無解，輸出 -1，否則輸出最小流。

```

const int maxn=50010;
const int maxm=405000;
int n,m,sp,tp,s,t;
int
nxt[maxn],head[maxn],to[maxn],cap[maxn],cnt=0,deg[maxn]
;
int cur[maxn],dis[maxn];
inline void add(int u,int v,int p){
    nxt[cnt]=head[u],to[cnt]=v,cap[cnt]=p,head[u]=cnt++;
    nxt[cnt]=head[v],to[cnt]=u,cap[cnt]=0,head[v]=cnt++;
}
bool bfs(int st,int en){
    memset(dis,-1,sizeof(dis));
    memcpy(cur,head,sizeof(head));
    queue<int> q;
    q.push(st);dis[st]=0;
    while(!q.empty()){
        int u=q.front();q.pop();
        for(int e=head[u];~e;e=nxt[e]){
            int v=to[e];
            if(cap[e]>0&&dis[v]==-1){
                dis[v]=dis[u]+1;
                if(v==en) return true;
                q.push(v);
            }
        }
    }
    return false;
}
inline int dinic(int u,int flow,int ee){
    if(u==ee) return flow;
    int res=0;
    for(int &e=cur[u];~e;e=nxt[e]){
        int v=to[e];
        if(cap[e]>0&&dis[v]>dis[u]){
            int delta=dinic(v,min(flow-res,cap[e]),ee);
            if(delta){
                cap[e]-=delta;cap[e^1]+=delta;
                res+=delta;
                if(res==flow) break;
            }
        }
    }
    return res;
}
int main(){
    memset(head,-1,sizeof(head));
    n=read();m=read();s=read();t=read();
    int i,j,k;
    sp=0;tp=n+1;
    for(i=1;i<=m;++i){
        int u=read(),v=read(),ll=read(),rr=read();
        add(u,v,rr-ll);
        deg[v]+=ll; deg[u]-=ll;
    }
    int sum=0,first;
    add(t,s,inf);
    first=cnt-1;
    for(i=1;i<=n;++i){
        if(deg[i]<0)
            add(i,tp,-deg[i]);
        else if(deg[i]>0)
            add(sp,i,deg[i]),sum+=deg[i];
    }
    int maxflow=0;
    while(bfs(sp,tp))
        maxflow+=dinic(sp,inf,tp);
    if(maxflow==sum){
        maxflow=cap[first];
        for(i=first-1;i<=cnt;++i) cap[i]=0;
        while(bfs(t,s)) maxflow-=dinic(t,inf,s);
        printf("%d\n",maxflow);
    }
}

```

```

    else printf("-1\n");
    return 0;
}

```

2.6 無源匯有上下界可行流

n 個點， m 條邊，每條邊 e 有一個流量下界 $\text{lower}(e)$ 和流量上界 $\text{upper}(e)$ ，求一種可行方案使得在所有點滿足流量平衡條件的前提下，所有邊滿足流量限制。

輸入格式

第一行兩個正整數 n 、 m 。

之後的 m 行，每行四個整數 s 、 t 、 lower 、 upper 。

輸出格式

如果無解，輸出一行 NO。

否則第一行輸出 YES，之後 m 行每行一個整數，表示每條邊的流量。

```

const int maxn=70005;
int
sp,tp,cnt=0,head[205],nxt[maxn],to[maxn],cap[maxn],dis[
1005],low[maxn],def[205],m,n;
inline void add(int u,int v,int p){
    nxt[cnt]=head[u],to[cnt]=v,cap[cnt]=p,head[u]=cnt++;
    nxt[cnt]=head[v],to[cnt]=u,cap[cnt]=0,head[v]=cnt++;
}
inline bool bfs(){
    int u,e,v;
    queue<int> que;
    memset(dis,-1,sizeof(dis));
    que.push(sp),dis[sp]=0;
    while(!que.empty()){
        u=que.front(),que.pop();
        for(int e=head[u];~e;e=nxt[e]){
            if(cap[e]>0&&dis[v=to[e]]==-1){
                dis[v]=dis[u]+1,que.push(v);
                if(v==tp) return true;
            }
        }
    }
    return false;
}
inline int dfs(const int &u,const int &flow){
    if(u==tp) return flow;
    int res=0,v,flw;
    for(int e=head[u];~e;e=nxt[e]){
        if(cap[e]>0&&dis[u]<dis[v=to[e]]){
            flw=dfs(v,min(cap[e],flow-res));
            if(flw==0) dis[v]=-1;
            cap[e]-=flw,cap[e^1]+=flw;
            res+=flw;
            if(res==flow) break;
        }
    }
    return res;
}
inline int dinic(int sp,int tp){
    int ans=0;
    while(bfs()){
        ans+=dfs(sp,1<<30);
    }
    return ans;
}
int main(){
    memset(head,-1,sizeof(head));
    n=read(),m=read();
    int s,t,up,down,sum=0;
    for(int i=1;i<=m;i++){
        s=read(),t=read(),down=read(),up=read();
        add(s,t,up-down);
        low[i]=down,def[s]+=down,def[t]-=down;
    }
    sp=n+1,tp=n+2;
    for(int i=1;i<=n;i++){
        if(def[i]>0) sum+=def[i],add(i,tp,def[i]);
        if(def[i]<0) add(sp,i,-def[i]);
    }
    if(dinic(sp,tp)==sum){
        cout<<"YES"<<endl;
        for(int i=1;i<=m;i++){
            cout<<cap[((i-1)*2)^1]+low[i]<<endl;

```

```

    }
}
else cout<<"NO"<<endl;
return 0;
}

```

2.7 最大權閉合圖

在一個圖中，我們選取一些點構成集合，記為 V ，且集合中的出邊（即集合中的點的向外連出的弧），所指向的終點（弧頭）也在 V 中，則我們稱 V 為閉合圖。最大權閉合圖即在所有閉合圖中，集合中點的權值之和最大的 V ，我們稱 V 為最大權閉合圖。

算法：

構造一個源點 S ，匯點 T 。我們將 S 與所有權值為正的點連一條容量為其權值的邊，將所有權值為負的點與 T 連一條容量為其權值的絕對值的邊，原來的邊將其容量定為正無窮。

閉合圖最大權 = 正權點數之和 - 最大流

2.8 最大密度子圖

簡單圖裡面找出 n 個點，這 n 個點之間有 m 條邊，讓 m/n 最大。

算法：

假設答案為 k ，則要求解的問題是：選出一個合適的點集 V 和邊集 E ，令 $(|E| - k * |V|)$ 取得最大值。所謂合適是指滿足如下限制：若選擇某條邊，則必選擇其兩端點。

建圖：以原圖的邊作為左側頂點，權值為 1；原圖的點作為右側頂點，權值為 $-k$ （相當於支出 k ）。

若原圖中存在邊 (u,v) ，則新圖中添加兩條邊 $([uv] \rightarrow u)$ ， $([uv] \rightarrow v)$ ，轉換為最大權閉合子圖。

2.9 最小割樹(Gomory-Hu Tree)

用來求兩兩點對之間的最小割。

定義一棵樹 T 為最小割樹，如果對於樹上的所有邊 (s,t) ，樹上去掉 (s,t) 後產生的兩個集合恰好是原圖上 (s,t) 的最小割把原圖分成的兩個集合，且邊 (s,t) 的權值等於原圖上 (s,t) 的最小割。

⇒ 原圖上 u,v 兩點最小割就是最小割樹上 u 到 v 的路徑上權值最小的邊。

構造：在當前點集隨意選取兩個點 u,v ，在原圖上跑出他們之間的最小割，然後就在樹上連一條從 u 到 v ，權值為 $\lambda(u,v)$ 的邊。然後找出 u,v 分屬的兩個點集，對這兩個點集遞迴進行操作。當點集中的點只剩一個的時候停止遞迴時間複雜度 $O(n^3m)$ ，但很難卡滿（跑了 n 次 dinic）。

2.10 Max Cost Circulation

```

struct MaxCostCirc {
    static const int MAXN = 33;
    int n , m;
    struct Edge { int v , w , c , r ; };
    vector<Edge> g[ MAXN ];
    int dis[ MAXN ] , prv[ MAXN ] , prve[ MAXN ];
    bool vis[ MAXN ];
    int ans;
    void init( int _n , int _m ) : n(_n), m(_m) {}
    void adde( int u , int v , int w , int c ) {
        g[ u ].push_back( { v , w , c , SZ( g[ v ] ) } );
        g[ v ].push_back( { u , -w , 0 , SZ( g[ u ] ) -
1 } );
    }
    bool poscyc() {
        fill( dis , dis+n+1 , 0 );
        fill( prv , prv+n+1 , 0 );
        fill( vis , vis+n+1 , 0 );
        int tmp = -1;
        FOR( t , n+1 ) {
            REP( i , 1 , n ) {
                FOR( j , SZ( g[ i ] ) ) {
                    Edge& e = g[ i ][ j ];
                    if( e.c && dis[ e.v ] < dis[ i ]+e.w ) {
                        dis[ e.v ] = dis[ i ]+e.w;
                        prv[ e.v ] = i;
                        prve[ e.v ] = j;
                        if( t == n ) {
                            tmp = i;
                            break;
                        }
                    }
                }
            }
        }
    }
}

```

```

if( tmp == -1 ) return 0;
int cur = tmp;
while( !vis[ cur ] ) {
    vis[ cur ] = 1;
    cur = prv[ cur ];
}
int now = cur , cost = 0 , df = 100000;
do{
    Edge &e = g[ prv[ now ] ][ prve[ now ] ];
    df = min( df , e.c );
    cost += e.w;
    now = prv[ now ];
}while( now != cur );
ans += df*cost; now = cur;
do{
    Edge &e = g[ prv[ now ] ][ prve[ now ] ];
    Edge &re = g[ now ][ e.r ];
    e.c -= df;
    re.c += df;
    now = prv[ now ];
}while( now != cur );
return 1;
}
} circ;

```

3 Math

3.1 質數與質因數分解(附 moebius 和 phi)

```

bool notprime[MAX];
int first[MAX]; //first[n]為n的最小質因數
int p[MAX], u[MAX], phi[MAX];
//存質數,moebius 函數,euler_phi
int top = 0; //質數個數
void build(){
    u[1] = 1; phi[1] = 1;
    for (int i = 2; i < MAX; i++){
        if (!notprime[i]){
            first[i] = i; u[i] = -1; phi[i] = i - 1;
            p[top] = i; top++;
        }
        for (int j = 0; i * p[j] < MAX && j < top;j++){
            first[i * p[j]] = p[j];
            notprime[i * p[j]] = 1;
            if (i % p[j]) {
                u[i * p[j]] = -u[i];
                phi[i * p[j]] = (p[j] - 1) * phi[i];
            }
            else { phi[i*p[j]]=p[j]*phi[i]; break;}
        }
    }
}

```

$$f(n) = \sum_{d|n} g(d) \leftrightarrow g(n) = \sum_{d|n} u\left(\frac{n}{d}\right) f(d)$$

$$u(i) = \begin{cases} 1, & \text{if } n = 1 \\ (-1)^k, & \text{if } n = p_1 * p_2 * \dots * p_k \\ 0, & \text{其它} \end{cases} \quad (\text{這些質數 } p \text{ 兩兩相異})$$

$$\sum_{d|n} \frac{u(d)}{d} = \frac{\phi(n)}{n}$$

3.2 Miller Rabin(大質數判定)

//輸入一个 long long 範圍內的數，是質數返回 true，否則返回 false。定義檢測次數為 TIMES，錯誤率為(1/4)^TIMES

```

#define TIMES 10
long long GetRandom(long long n){
    //cout<<RAND_MAX<<endl;
    ll num = (((unsigned ll)rand()+100000007)*rand())%n;
    return num + 1;
}
long long Mod_Mul(ll a, ll b, ll Mod){
    long long msum = 0;
    while (b){
        if (b & 1)
            msum = (msum + a) % Mod;
        b >>= 1;
        a = (a + a) % Mod;
    }
}

```

```

}
return msum;
}
long long Quk_Mul(ll a, ll b, ll Mod){
    long long qsum = 1;
    while (b) {
        if (b & 1)
            qsum = Mod_Mul(qsum, a, Mod);
        b >>= 1;
        a = Mod_Mul(a, a, Mod);
    }
    return qsum;
}
bool Miller_Rabin(long long n){
    if (n == 2 || n == 3 || n == 5 || n == 7 || n == 11)
        return true;
    if (n == 1 || n % 2 == 0 || n % 3 == 0 || n % 5 == 0 || n % 7 == 0 || n % 11 == 0)
        return false;
    int div2 = 0;
    long long tn = n - 1;
    while (!(tn % 2)){
        div2++;
        tn /= 2;
    }
    for (int tt = 0; tt < TIMES; tt++){
        long long x = GetRandom(n - 1); //隨機得到[1,n-1]
        if (x == 1)
            continue;
        x = Quk_Mul(x, tn, n);
        long long pre = x;
        for (int j = 0; j < div2; j++){
            x = Mod_Mul(x, x, n);
            if (x == 1 && pre != 1 && pre != n - 1)
                return false;
            pre = x;
        }
        if (x != 1)
            return false;
    }
    return true;
}

```

3.3 pollardRho(找大整數的因數)

```

//does not work when n is prime(先用 Miller Rabin 判定)
ll f(ll x, ll mod) { return (Mod_Mul(x, x, mod) + 1) % mod; } //這邊的 Mod_Mul 在 Miller Rabin 大質數判定裡面有
ll pollard_rho(ll n){
    if (!(n & 1))
        return 2;
    while (true){
        ll y = 2, x = rand() % (n - 1) + 1, res = 1;
        for (int sz = 2; res == 1; sz *= 2){
            for (int i = 0; i < sz && res == 1; i++){
                x = f(x, n);
                res = __gcd(abs(x - y), n);
            }
            y = x;
        }
        if (res != 0 && res != n)
            return res;
    }
}

```

3.4 FFT

$$c[k] = \sum_{i+j=k} a[i] * b[j]$$

```

typedef long double db;
#define N 262144 * 4
struct FFT{
    const db pi = acos(-1);
    int len, bitrev[N];
    struct Z{
        db x, y;
        Z(db _x = 0, db _y = 0) : x(_x), y(_y) {}
    }
}

```

```

friend Z operator+(Z a, Z b) { return Z(a.x +
b.x, a.y + b.y); }
friend Z operator-(Z a, Z b) { return Z(a.x -
b.x, a.y - b.y); }
friend Z operator*(Z a, Z b) { return Z(a.x *
b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
} t[N], A[N], B[N], C[N], W[N];

void dft(Z *a, int n, int sig = 1){
    for (int i = 0; i < n; i++){
        if (i < bitrev[i])
            swap(a[i], a[bitrev[i]]);
        for (int i = 2; i <= n; i <= 1){
            int half = i >> 1, times = len / i;
            for (int j = 0; j < half; j++){
                Z w = sig > 0 ? W[times * j] : W[len -
times * j];
                for (int k = j; k < len; k += i){
                    Z u = a[k], v = a[k + half] * w;
                    a[k] = u + v, a[k + half] = u - v;
                }
            }
        }
        if (sig == -1)
            for (int i = 0; i < n; i++)
                a[i].x /= n;
    }
}

void fft(db *c, db *a, db *b, int n, int m)
{ //c=a*b(結果), n為a的長度, m為b的長度
    int lg;
    lg = 0;
    while ((1 << lg) <= (max(n, m) << 1))
        ++lg;
    len = 1 << lg;
    for (int i = 0; i < len; i++)
        bitrev[i] = (bitrev[i >> 1] >> 1) | ((i & 1)
<< (lg - 1));
    for (int i = 0; i <= len; i++)
        W[i] = Z(cos(2 * pi * i / len), sin(2 * pi *
i / len));
    for (int i = 0; i < len; i++)
        A[i] = Z(a[i], 0), B[i] = Z(b[i], 0);
    dft(A, len);
    dft(B, len);
    for (int i = 0; i < len; i++)
        C[i] = A[i] * B[i];
    dft(C, len, -1);
    for (int i = 0; i < len; i++)
        c[i] = C[i].x;
}
};

```

3.5 FWT

$$ans[k] = \sum_{i \oplus j = k} f[i] * g[j]$$

```

struct Fast_Walsh_Hadamard_transform{
    inline void FWT(ll *f, int g, int n){
        int len = 1 << n;
        for (int i = 1; i < len; i <= 1)
            for (int j = 0; j < len; j += i << 1)
                for (int k = j; k < j + i; ++k){
                    ll x = f[k], y = f[k + i];
                    f[k] = x + y, f[k + i] = x - y;
                }
        if (g == -1)
            for (int i = 0; i < len; ++i)
                f[i] >>= n;
    }
    void solve(ll *ans, ll *f, ll *g, int n)
    { // ans=f*g, f和g的長度為(1<n)
        FWT(f, 1, n), FWT(g, 1, n);
        for (int i = 0; i < 1 << n; ++i)
            ans[i] = f[i] * g[i];
        FWT(ans, -1, n);
    }
} fwt;

```

3.6 中國剩餘定理(附 extgcd)

$$(S): \begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

m_1, m_2, \dots, m_n 兩兩互質，則對於任意整數 a_1, a_2, \dots, a_n 都存在 x 滿足上述方程組。

$$x \equiv a_1 t_1 M_1 + a_2 t_2 M_2 + \dots + a_n t_n M_n \pmod{M}$$

其中 $M = m_1 m_2 \dots m_n$, $M_i = M / m_i$ 且 $t_i M_i \equiv 1 \pmod{m_i}$

```

ll exgcd(ll a, ll b, ll &x, ll &y){
    if (b == 0){
        x = 1; y = 0;
        return a;
    }
    ll r = exgcd(b, a % b, x, y);
    ll t = x;
    x = y;
    y = t - a / b * y;
    return r;
}

ll chinese_remainder(int a[], int w[], int n)
{ //w 存放除數, a 存放餘數
    ll M = 1, ans = 0, x, y;
    for (int i = 0; i < n; i++){
        M *= w[i];
        for (int i = 0; i < n; i++){
            ll m = M / w[i];
            exgcd(m, w[i], x, y);
            ans = (ans + x * m * a[i]) % M;
        }
    }
    return (ans % M + M) % M;
}

```

3.7 高斯消去法

```

#define eps 1e-8
void gauss(vector<vector<double>> &A, vector<int> &cols
, vector<int> &rows, vector<int> &ind)
{ //哪些 cols 是係數(等號左邊, 要排在 A 的最前面幾行), 要對哪
些 rows 做高斯消去, ind 為哪些行消完不全零
    int N = min(rows.size(), cols.size());
    for (int i = 0; i < N; i++) {
        int x = i, y = i;
        for (int j = i; j < rows.size(); j++)
            for (int k = i; k < cols.size(); k++)
                if (fabs(A[rows[j]][cols[k]]) >
fabs(A[rows[x]][cols[y]])) x = j, y = k;
        if (fabs(A[rows[x]][cols[y]]) < eps) return;
        swap(rows[i], rows[x]), swap(cols[i], cols[y]);
        ind.emplace_back(rows[i]);
        for (int j = 0; j < rows.size(); j++){
            if (j == i) continue;
            for (int k = i + 1; k < cols.size(); k++)
                A[rows[j]][cols[k]] -=
A[rows[i]][cols[k]] * (A[rows[j]][cols[i]] / A[rows[i]]
[cols[i]]);
            for (int k = cols.size(); k < A[0].size(); k++)
                A[rows[j]][k] -=
A[rows[i]][k] * (A[rows[j]][cols[i]] / A[rows[i]][cols[
i]]);
            A[rows[j]][cols[i]] = 0;
        }
    }
}

vector<double> solve(vector<vector<double>> &A)
{ //n*(n+1)的高斯消去, A 是增廣矩陣
    int n = A.size();
    vector<int> cols, rows, ind;
    for (int i = 0; i < n; i++){
        rows.push_back(i), cols.push_back(i);
        gauss(A, cols, rows, ind);
        if (ind.size() < n)
            return vector<double>(>0); // no or infinite sols
        vector<double> ans(n);
        for (int i = 0; i < n; i++){

```

```

    ans[cols[i]] = A[rows[i]][n] / A[rows[i]][cols[i]];
    return ans;
}

```

3.8 歐拉函數

$a^x \equiv a^{x \% \phi(m) + \phi(m)} \pmod{m}$ 對於 $x > \phi(m)$ 成立。
若 a 和 m 互質則有 $a^{\phi(m)} \equiv 1 \pmod{m}$ 。

3.9 mod 奇質數下的一個平方根

```

void calcH(int &t, int &h, const int p){ //p 為奇質數
    int tmp = p - 1;
    for (t = 0; (tmp & 1) == 0; tmp /= 2) t++;
    h = tmp;
}
long long mul(ll a, ll b, ll Mod) //見 3.2
long long mypow(ll a, ll b, ll Mod) //a 的 b 次方快速幂
// solve equation x^2 mod p = a, p 為奇質數
bool solve(int a, int p, int &x, int &y){
    if (p == 2){
        x = y = 1;
        return true;
    }
    int p2 = p / 2, tmp = mypow(a, p2, p);
    if (tmp == p - 1)
        return false;
    if ((p + 1) % 4 == 0){
        x = mypow(a, (p + 1) / 4, p);
        y = p - x;
        return true;
    }
    else{
        int t, h, b, pb;
        calcH(t, h, p);
        if (t >= 2){
            do{
                b = rand() % (p - 2) + 2;
            } while (mypow(b, p / 2, p) != p - 1);
            pb = mypow(b, h, p);
        }
        int s = mypow(a, h / 2, p);
        for (int step = 2; step <= t; step++){
            int ss = (((ll)(s * s) % p) * a) % p;
            for (int i = 0; i < t - step; i++)
                ss = mul(ss, ss, p);
            if (ss + 1 == p) s = (s * pb) % p;
            pb = ((ll)pb * pb) % p;
        }
        x = ((ll)s * a) % p;
        y = p - x;
    }
    return true;
}

```

3.10 mod 奇質數下的 m 次方根

```

// Finds the primitive root modulo p
int generator(int p){
    vector<int> fact;
    int phi = p - 1, n = phi;
    for (int i = 2; i * i <= n; ++i){
        if (n % i == 0){
            fact.push_back(i);
            while (n % i == 0)
                n /= i;
        }
    }
    if (n > 1) fact.push_back(n);
    for (int res = 2; res <= p; ++res){
        bool ok = true;
        for (int factor : fact){
            if (powmod(res, phi / factor, p) == 1){
                ok = false;
                break;
            }
        }
        if (ok) return res;
    }
}

```

```

    return -1;
}
//finds all numbers x such that x^k=a (mod n)
vector<int> solve(int n, int k, int a){
    vector<int> ans;
    if (a == 0){
        ans.push_back(0);
        return ans;
    }
    int g = generator(n);
    // Baby-step giant-step discrete logarithm algorithm
    int sq = (int)sqrt(n + .0) + 1;
    vector<pair<int, int>> dec(sq);
    for (int i = 1; i <= sq; ++i)
        dec[i - 1] = {powmod(g, i * sq * k % (n - 1), n), i};
    sort(dec.begin(), dec.end());
    int any_ans = -1;
    for (int i = 0; i < sq; ++i){
        int my = powmod(g, i * k % (n - 1), n) * a % n;
        auto it = lower_bound(dec.begin(), dec.end(), make_
pair(my, 011));
        if (it != dec.end() && it->first == my){
            any_ans = it->second * sq - i;
            break;
        }
    }
    if (any_ans == -1) return ans;
    // Print all possible answers
    int delta = (n - 1) / gcd(k, n - 1);
    for (int cur = any_ans % delta; cur < n - 1; cur += delta)
        ans.push_back(powmod(g, cur, n));
    sort(ans.begin(), ans.end());
    return ans;
}

```

3.11 Burnside's lemma

對於一個置換 f ，若一個染色方案 s 經過置換後不變(ex.轉?度是一樣的)，稱 s 為 f 的不動點。將 f 的不動點數目記為 $C(f)$ ，則可以證明等價類數目為所有 $C(f)$ 的平均值。

3.12 Lucas's theorem

Lucas' Theorem:
For non-negative integer n, m and prime P ,
 $C(m, n) \bmod P = C(m/M, n/M) * C(m \% M, n \% M) \bmod P$
 $= \text{mult_i} (C(m_i, n_i))$
where m_i is the i -th digit of m in base P .

3.13 Sum of Two Squares Thm (Legendre)

For a given positive integer N , let
 $D1 = (\# \text{ of } d \setminus \text{in } \setminus N \text{ dividing } N \text{ that } d \equiv 1 \pmod{4})$
 $D3 = (\# \text{ of } d \setminus \text{in } \setminus N \text{ dividing } N \text{ that } d \equiv 3 \pmod{4})$
then N can be written as a sum of two squares in exactly $R(N) = 4(D1 - D3)$ ways.

3.14 Difference of D1-D3 Thm

let $N = 2^t * [p1^{e1} * \dots * pr^{er}] * [q1^{f1} * \dots * qs^{fs}]$
 $\quad \quad \quad \leftarrow \text{mod } 4 = 1 \text{ prime} \rightarrow \quad \quad \quad \leftarrow \text{mod } 4 = 3 \text{ prime} \rightarrow$
 then $D1 - D3 = (e1+1)(e2+1)\dots(er+1)$ if f_i all even
 $\quad \quad \quad 0$ if any f_i is odd

4 Geometry

4.1 幾何們

```

#define X first
#define Y second
#define pi acos(-1.0)
#define eps 1e-8
typedef double type;
typedef pair<type, type> P;
int dcmp(double x){
    if (fabs(x) < eps)
        return 0;
    return x < 0 ? -1 : 1;
}

```

```

}
struct Line{P p, v;};
//atan2 的範圍是-pi~pi
bool operator<(Line l1, Line l2) { return atan2(l1.v.Y,
l1.v.X) < atan2(l2.v.Y, l2.v.X); }
bool equal(type x, type y) { return fabs(x - y) <
eps; }
bool less(type x, type y) { return x < y - eps; }
bool greater(type x, type y) { return x > y + eps; }
P operator+(P p1, P p2) { return P(p1.X + p2.X, p1.Y +
p2.Y); }
P operator-(P p1, P p2) { return P(p1.X - p2.X, p1.Y -
p2.Y); }
type operator*(P p1, P p2) { return p1.X * p2.X + p1.Y
* p2.Y; }
P operator*(double t, P p) { return P(t * p.X, t *
p.Y); }
P operator/(P p, double t) { return P(p.X / t, p.Y / t)
; }
type operator^(P p1, P p2) { return (p1.X * p2.Y - p1.Y
* p2.X); }
double len(P p) { return sqrt(1.0*p.X*p.X+p.Y*p.Y); }
double angle(P p1, P p2){ //p1 轉到 p2 範圍是 0~2*pi
if ((p1 ^ p2) < 0) return 2 * pi -
acos((double)(p1 * p2) / len(p1) / len(p2));
return acos((double)(p1 * p2) / len(p1) / len(p2));
}
bool on(P a, P p1, P p2) { return ((dcmp((p1 - a) * (p2
- a)) <= 0) && dcmp((p1 - a) ^ (p2 - a)) == 0); }
bool in(P a, P p1, P p2) { return dcmp((p1 ^ a) * (p2 ^
a)) < 0; }
bool cross(P p1, P p2, P p3, P p4)
{ //p1-p2 線段和 p3-p4 線段是否相交
if (on(p3, p1, p2) || on(p4, p1, p2) || on(p1, p3,
p4) || on(p2, p3, p4))
return 1;
if (in(p2 - p1, p3 - p1, p4 - p1) && in(p4 - p3, p1
- p3, p2 - p3))
return 1;
return 0;
}
double torad(double deg) { return pi * deg / 180.0; }
P rotate(P p, double rad) { return P(p.X * cos(rad) -
p.Y * sin(rad),
p.X * sin(rad) + p.Y
* cos(rad)); }
double dist(P p, Line l) { return fabs((p - l.p) ^ l.v)
/ len(l.v); }
P LineIntersect(Line l1, Line l2){//兩直線平行時不能叫
double t = 1.0 * ((l2.p - l1.p) ^ l2.v) / (l1.v ^
l2.v);
return l1.p + t * l1.v;
}
bool SegLineIntersect(P p1, P p2, Line l)
{ //線段 p1-p2 和直線 l 有沒有相交
Line l1;
l1.p = p1, l1.v = p2 - p1;
if (dcmp(l1.v ^ (l.p - p1)) == 0 || dcmp(l1.v ^ (l.p
- p2)) == 0)
return 1;
return in(l.v, p1 - l.p, p2 - l.p);
}
type area2(vector<P> ps) { //兩倍多邊形面積
type res = 0;
for (int i = 0; i < ps.size(); i++)
res += (ps[i] ^ ps[(i + 1) % ps.size()]);
if (res < 0)
res = -res;
return res;
}
bool inPolygon(P p, vector<P> poly){
int wn = 0;
int n = poly.size();
for (int i = 0; i < n; i++){
if (on(p, poly[i], poly[(i + 1) % n]))
return -1; //在邊界
int k = dcmp((poly[(i + 1) % n] - poly[i]) ^ (p

```

```

- poly[i]));
int d1 = dcmp(poly[i].Y - p.Y);
int d2 = dcmp(poly[(i + 1) % n].Y - p.Y);
if (k > 0 && d1 <= 0 && d2 > 0)
wn++;
if (k < 0 && d2 <= 0 && d1 > 0)
wn--;
}
if (wn != 0)
return 1; //內部
return 0; //外部
}
vector<P> ConvexHull(vector<P> ps){
int nn = ps.size();
sort(ps.begin(), ps.end());
vector<P> res;
int k = 0;
for (int i = 0; i < nn; i++){
while (k > 1 && dcmp((ps[i] - res[k - 2]) ^
(res[k - 1] - res[k - 2])) >= 0) {
res.pop_back();
k--;
}
res.push_back(ps[i]);
k++;
}
int t = k;
for (int i = nn - 2; i >= 0; i--){
while (k > t && dcmp((ps[i] - res[k - 2]) ^
(res[k - 1] - res[k - 2])) >= 0){
res.pop_back();
k--;
}
res.push_back(ps[i]);
k++;
}
if (nn > 1)
res.pop_back();
return res;
};
struct Half_Plane_Intersection
{ //半平面交(所有直線左側的交集)
const static int MAXN = 100005;
int n;
Line L[MAXN], s[MAXN];
vector<P> a; //結果存在這，是一個凸包
void init() { n = 0; }
void add_Line(Line l) { L[n++] = l; }
bool OnLeft(Line l, P p) { return dcmp(l.v ^ (p -
l.p)) >= 0; }
int solve(){
a.clear();
sort(L, L + n); //sort
int first, last;
P *p = new P[n];
Line *q = new Line[n];
q[first = last = 0] = L[0];
for (int i = 1; i < n; i++){
while (first < last && !OnLeft(L[i], p[last
- 1]))
last--;
while (first < last && !OnLeft(L[i],
p[first]))
first++;
q[++last] = L[i];
if (dcmp(q[last].v ^ q[last - 1].v) == 0){
last--;
if (OnLeft(q[last], L[i].p))
q[last] = L[i];
}
if (first < last)
p[last - 1] = LineIntersect(q[last - 1],
q[last]);
}
while (first < last && !OnLeft(q[first], p[last
- 1]))
last--;
if (last - first <= 1)
return 0;
}

```



```

p[last] = LineIntersect(q[last], q[first]);

for (int i = first; i <= last; i++)
    a.push_back(p[i]);
return a.size();
}
} hpi;
struct Circle{
    P c;
    type r;
    P point(double a) { return P(c.X + cos(a) * r, c.Y + sin(a) * r); }
};
int LineCircleIntersect(Line L, Circle C, vector<P> &sol){ //返回交點個數，sol 存交點們
    type a = L.v.X, b = L.p.X - C.c.X, c = L.v.Y, d = L.p.Y - C.c.Y;
    type e = a * a + c * c, f = 2 * (a * b + c * d), g = b * b + d * d - C.r * C.r;
    type delta = f * f - 4 * e * g;
    if (dcmp(delta) < 0)
        return 0;
    if (dcmp(delta) == 0)
    {
        sol.push_back(L.p - (f / (2 * e)) * L.v);
        return 1;
    }
    double t1 = (-f - sqrt(delta)) / (2 * e);
    sol.push_back(L.p + t1 * L.v);
    double t2 = (-f + sqrt(delta)) / (2 * e);
    sol.push_back(L.p + t2 * L.v);
    return 2;
}
int CircleIntersect(Circle C1, Circle C2, vector<P> &sol){
    double d = len(C1.c - C2.c);
    if (dcmp(d) == 0){
        if (dcmp(C1.r - C2.r) == 0)
            return -1; //兩圓重合
        return 0;
    }
    if (dcmp(C1.r + C2.r - d) < 0)
        return 0;
    if (dcmp(fabs(C1.r - C2.r) - d) > 0)
        return 0;
    double a = atan2(C2.c.Y - C1.c.Y, C2.c.X - C1.c.X);
    double da = acos((C1.r * C1.r + d * d - C2.r * C2.r) / (2 * C1.r * d)); //最好判一下括號裡面是否在[-1,1]
    P p1 = make_pair(C1.c.X + cos(a - da) * C1.r, C1.c.Y + sin(a - da) * C1.r);
    P p2 = make_pair(C1.c.X + cos(a + da) * C1.r, C1.c.Y + sin(a + da) * C1.r);
    sol.push_back(p1);
    if (p1 == p2)
        return 1;
    sol.push_back(p2);
    return 2;
}
int PointCircleTangents(P p, Circle C, vector<P> &sol)
{ //返回切線條數，sol 存切線向量們
    P u = C.c - p;
    double dist = len(u);
    if (dist < C.r)
        return 0;
    if (dcmp(dist - C.r) == 0)
    {
        sol.push_back(rotate(u, pi / 2));
        return 1;
    }
    double ang = asin(C.r / dist);
    sol.push_back(rotate(u, -ang));
    sol.push_back(rotate(u, ang));
    return 2;
}
double Circle_Segment_Intersect_area(P A, P B, Circle C)
{ //圓心和線段兩端點圍成的三角形與圓的交集面積
    P CA = C.c - A, CB = C.c - B;

```

```

    double da = len(CA), db = len(CB);
    da = dcmp(da - C.r);
    db = dcmp(db - C.r);

    if (da <= 0 && db <= 0)
        return fabs((CA ^ CB)) * 0.5;

    vector<P> sol;
    int num = LineCircleIntersect(Line{A, B - A}, C, sol);
    double cnt = C.r * C.r;
    P q;

    if (da <= 0 && db > 0){
        q = on(sol[0], A, B) ? sol[0] : sol[1];
        double area = fabs((CA ^ (C.c - q))) * 0.5;
        double ang = acos((CB * (C.c - q)) / len(CB) / len(C.c - q));
        return area + cnt * ang * 0.5;
    }
    if (db <= 0 && da > 0){
        q = on(sol[0], A, B) ? sol[0] : sol[1];
        double area = fabs((CB ^ (C.c - q))) * 0.5;
        double ang = acos((CA * (C.c - q)) / len(CA) / len(C.c - q));
        return area + cnt * ang * 0.5;
    }
    if (num == 2){
        double big_area = cnt * acos((CA * CB) / len(CA) / len(CB)) * 0.5;
        double small_area = cnt * acos(((C.c - sol[0]) * (C.c - sol[1])) / len(C.c - sol[0]) / len(C.c - sol[1])) * 0.5;
        double delta_area = fabs((C.c - sol[0]) ^ (C.c - sol[1])) * 0.5;
        if (!on(sol[0], A, B))
            return big_area;
        return big_area + delta_area - small_area;
    }
    return cnt * acos((CA * CB) / len(CA) / len(CB)) * 0.5;
}

double Circle_Polygon_Intersect_area(vector<P> ps, Circle C)
{ //多邊形與圓的交集面積
    double res = 0;
    int sz = ps.size();
    for (int i = 0; i < sz; i++) {
        double tmp = Circle_Segment_Intersect_area(ps[i], ps[(i + 1) % sz], C);
        if (((ps[i] - C.c) ^ (ps[(i + 1) % sz] - C.c)) < 0)
            tmp = -tmp;
        res += tmp;
    }
    if (res < 0)
        res = -res;
    return res;
}

int CircleTangents(Circle A, Circle B, vector<P> &a, vector<P> &b)
{ //返回切線條數，-1 表示無窮條切線。a[i]和b[i]分別是第 i 條切線在圓 A 與 B 上的交點
    int cnt = 0;
    if (A.r < B.r) {
        swap(A, B);
        swap(a, b);
    }
    type d2 = (A.c.X - B.c.X) * (A.c.X - B.c.X) + (A.c.Y - B.c.Y) * (A.c.Y - B.c.Y);
    type rdif = A.r - B.r;
    type rsum = A.r + B.r;
    if (dcmp(d2 - rdif * rdif) < 0)
        return 0; //內含

    double base = atan2(B.c.Y - A.c.Y, B.c.X - A.c.X);
    if (dcmp(d2) == 0 && dcmp(A.r - B.r) == 0)

```

```

    return -1; //無限多條切線
if (dcmp(d2 - rdiff * rdiff) == 0)
{ //內切, 1 條切線
    a.push_back(A.point(base));
    b.push_back(B.point(base));
    cnt++;
    return 1;
}
//有外共切線
double ang = acos((A.r - B.r) / sqrt(d2));
a.push_back(A.point(base + ang));
b.push_back(B.point(base + ang));
cnt++;
a.push_back(A.point(base - ang));
b.push_back(B.point(base - ang));
cnt++;
if (d2 == rsum * rsum)
{ //一條公切線
    a.push_back(A.point(base));
    b.push_back(B.point(pi + base));
    cnt++;
}
else if (d2 > rsum * rsum)
{ //兩條內公切線
    double ang = acos(rsum / sqrt(d2));
    a.push_back(A.point(base + ang));
    b.push_back(B.point(pi + base + ang));
    cnt++;
    a.push_back(A.point(base - ang));
    b.push_back(B.point(pi + base - ang));
    cnt++;
}
return cnt;
}
}

```

4.2 旋轉卡殼(最遠距點對)

```

pt = ConvexHull(pt), n = pt.size();
double ans = 0;
int j = 0;
for (int i = 0; i < n; i++){
    while (1){
        double ang=angle(pt[(i+1)%n]-pt[i],pt[(j+1)%n]-pt[j]);
        if (ang < pi) j = (j + 1) % n;
        else break;
    }
    ans = max(ans, len(pt[j] - pt[i]));
    ans = max(ans, len(pt[j] - pt[(i + 1) % n]));
}

```

4.3 皮克(Pick)定理

給定頂點座標均是整點(或正方形格子點)的簡單多邊形, 面積 A 和內部格點數目 i 、邊上格點數目 b 的關係: $A = i + \frac{b}{2} - 1$ 。

4.4 Minkowski sum

```

vector<P> minkowski(vector<P> p, vector<P> q){
    int n = p.size(), m = q.size();
    P c = P(0, 0);
    for (int i = 0; i < m; i++){
        c = c + q[i];
    }
    c = (1.0 / m) * c;
    for (int i = 0; i < m; i++){
        q[i] = q[i] - c;
    }
    int cur = -1;
    for (int i = 0; i < m; i++){
        if ((q[i] ^ (p[0] - p[n - 1])) > -eps)
            if (cur == -1 || (q[i] ^ (p[0] - p[n - 1])) > (q[cur] ^ (p[0] - p[n - 1])))
                cur = i;
    }
    vector<P> h;
    p.push_back(p[0]);
    for (int i = 0; i < n; i++){
        while (true){
            h.push_back(p[i] + q[cur]);
            int nxt = (cur + 1 == m ? 0 : cur + 1);

```

```

        if ((q[cur] ^ (p[i + 1] - p[i])) < -eps)
            cur = nxt;
        else if ((q[nxt] ^ (p[i + 1] - p[i])) > (q[cur] ^ (p[i + 1] - p[i])))
            cur = nxt;
        else
            break;
    }
    for (auto &&i : h) i = i + c;
    return ConvexHull(h);
}

```

4.5 三角形的三心

```

P inCenter( P &a, P &b, P &c) { // 內心
    double a = len(B-C), b = len(C-A), c = len(A-B);
    return (A * a + B * b + C * c) / (a + b + c);
}
P circumCenter( P &a, P &b, P &c) { // 外心
    P bb = b - a, cc = c - a;
    double db=bb.X*bb.X+bb.Y*bb.Y, dc=cc.X*cc.X+cc.Y*cc.Y, d=2*(bb ^ cc);
    return a-P(bb.Y*dc-cc.Y*db, cc.X*db-bb.X*dc) / d;
}
P orthoCenter( P &a, P &b, P &c) { // 垂心
    P ba = b - a, ca = c - a, bc = b - c;
    double Y = ba.Y * ca.Y * bc.Y,
        A = ca.X * ba.Y - ba.X * ca.Y,
        x0= (Y+ca.X*ba.Y*b.X-ba.X*ca.Y*c.X) / A,
        y0= -ba.X * (x0 - c.X) / ba.Y + ca.Y;
    return P(x0, y0);
}

```

4.6 Circle Cover

```

#define N 1021
struct CircleCover{
    int C;
    Circle c[N];
    bool g[N][N], overlap[N][N];
    // Area[i] : area covered by at least i circles
    double Area[N];
    void init(int _C) { C = _C; }
    bool CCinter(Circle &a, Circle &b, P &p1, P &p2){
        P o1 = a.c, o2 = b.c;
        double r1 = a.r, r2 = b.r;
        if (len(o1 - o2) > r1 + r2)
            return false;
        if (len(o1 - o2) < max(r1, r2) - min(r1, r2))
            return true;
        double d2 = (o1 - o2) * (o1 - o2);
        double d = sqrt(d2);
        if (d > r1 + r2)
            return false;
        P u = 0.5 * (o1 + o2) + ((r2 * r2 - r1 * r1) / (2 * d2)) * (o1 - o2);
        double A = sqrt((r1 + r2 + d) * (r1 - r2 + d) * (r1 + r2 - d) * (-r1 + r2 + d));
        P v = A * P(o1.Y - o2.Y, -o1.X + o2.X) / (2 * d2);
        p1 = u + v;
        p2 = u - v;
        return true;
    }
    struct Teve{
        P p;
        double ang;
        int add;
        Teve() {}
        Teve(P _a, double _b, int _c) : p(_a), ang(_b), add(_c) {}
        bool operator<(const Teve &a) const{
            return ang < a.ang;
        }
    } eve[N * 2];
    // strict: x = 0, otherwise x = -1
    bool disjunct(Circle &a, Circle &b, int x){
        return dcmp(len(a.c - b.c) - a.r - b.r) > x;
    }
    bool contain(Circle &a, Circle &b, int x){

```

```

    return dcmp(a.r - b.r - len(a.c - b.c)) > x;
}
bool contain(int i, int j) {
    /* c[j] is non-strictly in c[i]. */
    return (dcmp(c[i].r - c[j].r) > 0 ||
            (dcmp(c[i].r - c[j].r) == 0 && i < j)) &&
            contain(c[i], c[j], -1);
}
void solve(){
    for (int i = 0; i <= C + 1; i++)
        Area[i] = 0;
    for (int i = 0; i < C; i++)
        for (int j = 0; j < C; j++)
            overlap[i][j] = contain(i, j);
    for (int i = 0; i < C; i++)
        for (int j = 0; j < C; j++)
            g[i][j] = !(overlap[i][j] || overlap[j][i] ||
                        disjunct(c[i], c[j], -1));
    for (int i = 0; i < C; i++) {
        int E = 0, cnt = 1;
        for (int j = 0; j < C; j++)
            if (j != i && overlap[j][i])
                cnt++;
        for (int j = 0; j < C; j++)
            if (i != j && g[i][j]) {
                P aa, bb;
                CCinter(c[i], c[j], aa, bb);
                double A = atan2(aa.Y - c[i].c.Y, aa.X - c[i].c.X);
                double B = atan2(bb.Y - c[i].c.Y, bb.X - c[i].c.X);
                eve[E++] = Teve(bb, B, 1);
                eve[E++] = Teve(aa, A, -1);
                if (B > A)
                    cnt++;
            }
        if (E == 0)
            Area[cnt] += pi * c[i].r * c[i].r;
        else{
            sort(eve, eve + E);
            eve[E] = eve[0];
            for (int j = 0; j < E; j++){
                cnt += eve[j].add;
                Area[cnt] += (eve[j].p ^ eve[j + 1].p) * .5;
                double theta = eve[j + 1].ang - eve[j].ang;
                if (theta < 0)
                    theta += 2. * pi;
                Area[cnt] +=
                    (theta - sin(theta)) * c[i].r * c[i].r * .5;
            }
        }
    }
}
};

```

4.7 minimum enclosing circle

```

struct Mec{
    // return pair of center and r
    type norm2(P x) { return x.X * x.X + x.Y * x.Y; }
    static const int N = 101010;
    int n;
    P p[N], cen;
    double r2;
    void init(int _n, P _p[]) {
        n = _n;
        memcpy(p, _p, sizeof(P) * n);
    }
    double sqr(double a) { return a * a; }
    P center(P p0, P p1, P p2) {
        P a = p1 - p0;
        P b = p2 - p0;
        double c1 = norm2(a) * 0.5;
        double c2 = norm2(b) * 0.5;
        double d = a ^ b;
        double x = p0.X + (c1 * b.Y - c2 * a.Y) / d;
        double y = p0.Y + (a.X * c2 - b.X * c1) / d;
        return P(x, y);
    }
}

```

```

pair<P, double> solve() {
    random_shuffle(p, p + n);
    r2 = 0;
    for (int i = 0; i < n; i++){
        if (norm2(cen - p[i]) <= r2)
            continue;
        cen = p[i];
        r2 = 0;
        for (int j = 0; j < i; j++){
            if (norm2(cen - p[j]) <= r2)
                continue;
            cen = P((p[i].X + p[j].X) / 2, (p[i].Y + p[j].Y) / 2);
            r2 = norm2(cen - p[j]);
            for (int k = 0; k < j; k++){
                if (norm2(cen - p[k]) <= r2)
                    continue;
                cen = center(p[i], p[j], p[k]);
                r2 = norm2(cen - p[k]);
            }
        }
    }
    return {cen, sqrt(r2)};
}
} mec;

```

4.8 minimum enclosing ball

```

struct Pt{ type x, y, z;};
Pt operator+(Pt p1, Pt p2) { return Pt{p1.x + p2.x, p1.y + p2.y, p1.z + p2.z}; }
Pt operator-(Pt p1, Pt p2) { return Pt{p1.x - p2.x, p1.y - p2.y, p1.z - p2.z}; }
type operator*(Pt p1, Pt p2) { return p1.x * p2.x + p1.y * p2.y + p1.z * p2.z; }
Pt operator*(Pt p, type t) { return Pt{p.x * t, p.y * t, p.z * t}; }
Pt operator/(Pt p, type t) { return Pt{p.x / t, p.y / t, p.z / t}; }
type norm2(Pt p) { return p.x * p.x + p.y * p.y + p.z * p.z; }
type norm(Pt p) { return sqrt(p.x * p.x + p.y * p.y + p.z * p.z); }
struct min_enclosing_ball{
    static const int N = 202020;
    int n, nouter;
    Pt pt[N], outer[4], res;
    double radius, tmp;
    void ball() {
        Pt q[3];
        double m[3][3], sol[3], L[3], det;
        int i, j;
        res.x = res.y = res.z = radius = 0;
        switch (nouter) {
            case 1:
                res = outer[0];
                break;
            case 2:
                res = (outer[0] + outer[1]) / 2;
                radius = norm2(res - outer[0]);
                break;
            case 3:
                for (i = 0; i < 2; ++i)
                    q[i] = outer[i + 1] - outer[0];
                for (i = 0; i < 2; ++i)
                    for (j = 0; j < 2; ++j)
                        m[i][j] = (q[i] * q[j]) * 2;
                for (i = 0; i < 2; ++i)
                    sol[i] = (q[i] * q[i]);
                if (fabs(det = m[0][0] * m[1][1] - m[0][1] * m[1][0]) < eps)
                    return;
                L[0] = (sol[0] * m[1][1] - sol[1] * m[0][1]) / det;
                L[1] = (sol[1] * m[0][0] - sol[0] * m[1][0]) / det;
                res = outer[0] + q[0] * L[0] + q[1] * L[1];
                radius = norm2(res - outer[0]);
                break;
        }
    }
}

```

```

case 4:
    for (i = 0; i < 3; ++i)
        q[i] = outer[i + 1] - outer[0], sol[i] = (q[i]
* q[i]);
    for (i = 0; i < 3; ++i)
        for (j = 0; j < 3; ++j)
            m[i][j] = (q[i] * q[j]) * 2;
    det = m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1]
[2] * m[2][0] + m[0][2] * m[2][1] * m[1][0] - m[0][2]
* m[1][1] * m[2][0] - m[0][1] * m[1][0] * m[2][2] - m[0]
[0] * m[1][2] * m[2][1];
    if (fabs(det) < eps)
        return;
    for (j = 0; j < 3; ++j) {
        for (i = 0; i < 3; ++i)
            m[i][j] = sol[i];
        L[j] = (m[0][0] * m[1][1] * m[2][2] + m[0][1] *
m[1][2] * m[2][0] + m[0][2] * m[2][1] * m[1][0] - m[0]
[2] * m[1][1] * m[2][0] - m[0][1] * m[1][0] * m[2][2] -
m[0][0] * m[1][2] * m[2][1]) / det;
        for (i = 0; i < 3; ++i)
            m[i][j] = (q[i] * q[j]) * 2;
    }
    res = outer[0];
    for (i = 0; i < 3; ++i)
        res = res + q[i] * L[i];
    radius = norm2(res - outer[0]);
}
void minball(int n){
    ball();
    if (nouter < 4)
        for (int i = 0; i < n; i++)
            if (norm2(res - pt[i]) - radius > eps){
                outer[nouter++] = pt[i];
                minball(i);
                --nouter;
            }
            if (i > 0) {
                Pt Tt = pt[i];
                memmove(&pt[1], &pt[0], sizeof(Pt) * i);
                pt[0] = Tt;
            }
}
void solve() { // n points in pt
    random_shuffle(pt, pt + n);
    radius = -1;
    for (int i = 0; i < n; i++)
        if (norm2(res - pt[i]) - radius > eps)
            nouter = 1, outer[0] = pt[i], minball(i);
    printf("%.9f\n", sqrt(radius));
}
} B;

```

4.9 矩形重疊面積

給你很多平面上的矩形，請求出它們覆蓋的總表面積。

有 $n \leq 100,000$ 個矩形。

接下來有 n 列， $L, R, D, U (0 \leq L < R \leq 1,000,000$;

$0 \leq D < U \leq 1,000,000)$ 代表矩形的左、右、下、上四個邊界座標。

const int maxn=1000000+10;

```

struct P{
    int x,d,u,val;
    bool operator < (const P &rhs) const { return
x<rhs.x; }
}a[200000+10];

int ST[5*maxn],tag[5*maxn];

void modify(int l,int r,int L,int R,int id,int val){
    if(l==L && r==R) { tag[id]+=val; return; }
    int mid=(L+R)/2;
    if(r<=mid) modify(l,r,L,mid,2*id,val);
    else if(l>mid) modify(l,r,mid+1,R,2*id+1,val);
    else
        modify(l,mid,L,mid,2*id,val),
        modify(mid+1,r,mid+1,R,2*id+1,val);
    ST[id] = (tag[2*id] ? mid-L+1 : ST[2*id]) +

```

```

(tag[2*id+1] ? R-mid : ST[2*id+1]);
}
main(){
    int n; scanf("%d",&n);
    for(int i=0;i<n;i++){
        int x1,y1,x2,y2;
        scanf("%d%d%d%d",&x1,&x2,&y1,&y2);
        a[2*i]=(P){x1,y1,y2-1,1};
        a[2*i+1]=(P){x2,y1,y2-1,-1};
    }
    sort(a,a+2*n);
    int x=0, val=0;
    ll ans=0ll;
    for(int i=0;i<2*n;i++){
        ans+= (ll) (a[i].x-x)*val;
        modify(a[i].d,a[i].u,0,maxn-1,1,a[i].val);
        x=a[i].x;
        val=ST[1];
    }
    printf("%lld\n",ans);
}

```

5 Graph

5.1 HeavyLightDecomp (附 LCA)

```

#define REP(i, s, e) for (int i = (s); i <= (e); i++)
#define REPD(i, s, e) for (int i = (s); i >= (e); i--)
#define PII pair<int, int>
const int MAXN = 100010;
const int LOG = 19;
struct HLD{
    int n;
    vector<int> g[MAXN];
    int sz[MAXN], dep[MAXN];
    int ts, tid[MAXN], tdi[MAXN], tl[MAXN], tr[MAXN];
    // ts : timestamp, useless after yutruli
    // tid[ u ] : pos. of node u in the seq.
    // tdi[i] : node at pos i of the seq.
    // tl,tr[u]:subtree interval in the seq. of node u
    int prt[MAXN][LOG], head[MAXN];
    // head[ u ] : head of the chain contains u
    void dfsz(int u, int p){
        dep[u] = dep[p] + 1;
        prt[u][0] = p;
        sz[u] = 1;
        head[u] = u;
        for (int &v : g[u])
            if (v != p){
                dep[v] = dep[u] + 1;
                dfsz(v, u);
                sz[u] += sz[v];
            }
    }
    void dfs1(int u){
        ts++;
        tid[u] = tl[u] = tr[u] = ts;
        tdi[tid[u]] = u;
        sort(g[u].begin(), g[u].end(), [&](int a, int b)
{ return sz[a] > sz[b]; });
        bool flag = 1;
        for (int &v : g[u])
            if (v != prt[u][0]){
                if (flag)
                    head[v] = head[u], flag = 0;
                dfs1(v);
                tr[u] = tr[v];
            }
    }
    inline int lca(int a, int b){
        if (dep[a] > dep[b])
            swap(a, b);
        int diff = dep[b] - dep[a];
        REPD(k, LOG - 1, 0)
            if (diff & (1 << k)){
                b = prt[b][k];
            }
        if (a == b)
            return a;
    }
}

```

```

    REPD(k, LOG - 1, 0)
    if (prt[a][k] != prt[b][k]){
        a = prt[a][k];
        b = prt[b][k];
    }
    return prt[a][0];
}
void init(int _n){
    n = _n;
    REP(i, 1, n)
        g[i].clear();
}
void addEdge(int u, int v){
    g[u].push_back(v);
    g[v].push_back(u);
}
void yutruli(){
    dfssz(1, 0);
    ts = 0;
    dfshl(1);
    REP(k, 1, LOG - 1)
        REP(i, 1, n)
            prt[i][k] = prt[prt[i][k - 1]][k - 1];
}
vector<PII> getPath(int u, int v){
    vector<PII> res;
    while (tid[u] < tid[head[v]]){
        res.push_back(PII(tid[head[v]], tid[v]));
        v = prt[head[v]][0];
    }
    res.push_back(PII(tid[u], tid[v]));
    reverse(res.begin(), res.end());
    return res;
}
/* res : list of intervals from u to v
 * u must be ancestor of v
 * usage :
 * vector< PII >& path = tree.getPath( u , v )
 * for( PII tp : path ) {
 *     int l , r; tie( l , r ) = tp;
 *     upd( l , r );
 *     uu = tree.tdi[ l ] , vv = tree.tdi[ r ];
 *     uu ~> vv is a heavy path on tree
 * }
 */
}
} tree;

```

5.2 centroid decomposition

從 u 到 v 的最短路徑，必會通過重心樹上的 $\text{lca}(u, v)$

```

struct Centroid_Decomposition{
    typedef long long type;
    int subSize[100005];
    bool used[100005];
    vector<pair<int, type>> tree[100005];
    int cd_father[100005], dep[100005]; //cd_father[i]:i
    在重心樹上的父親，dep[i]:i 在重心樹上的深度
    type dis[20][100005];
    //dis[d][v]:v 到重心樹上深度為 d 的祖先的距離
    int idx[100005];
    //idx[i]:i 是 cd_father[i] 在重心樹上的第幾號兒子

    void addEdge(int u, int v, type w) {
        tree[u].push_back(make_pair(v, w));
        tree[v].push_back(make_pair(u, w));
    }
    int dfs(int u, int p) {
        subSize[u] = 1;
        for (pair<int, type> v : tree[u])
            if (v.first != p && !used[v.first])
                subSize[u] += dfs(v.first, u);
        return subSize[u];
    }
    int get_centroid(int u, int p, int n) {
        for (pair<int, type> v : tree[u])
            if (v.first != p && subSize[v.first] > n / 2 && !
                used[v.first])
                return get_centroid(v.first, u, n);
        return u;
    }
}

```

```

}
void get_distance(int u, int p, int depp, type dist){
    dis[depp][u] = dist;
    for (pair<int, type> v : tree[u])
        if (v.first != p && !used[v.first])
            get_distance(v.first, u, depp, dist+v.second);
}
int centroid_decomposition(int u, int p, int depp, in
t id) { //一開始叫(1,-1,0,0)
    int n = dfs(u, p);
    int centroid = get_centroid(u, p, n);
    dep[centroid] = depp, cd_father[centroid] = p, idx[
centroid] = id;
    get_distance(centroid, p, depp, 0);
    used[centroid] = 1;
    int cur = 0;
    for (pair<int, type> v : tree[centroid])
        if (v.first != p && !used[v.first])
            centroid_decomposition(v.first, centroid, depp
+ 1, cur++);
    return centroid;
}
} cd;

```

5.3 BCC 割點

```

struct BCC{
    struct edge{ int u, v; };
    int dfs_clock;
    int bcc_cnt; //Number of bcc
    vector<int> bcc[maxn]; //1~bcc_cnt
    int pre[maxn], iscut[maxn], bccno[maxn];

    vector<int> v[maxn];
    vector<edge> S;

    void init(int _n){
        for (int i = 0; i <= _n; i++)
            v[i].clear();
        S.clear();
    }
    void add_edge(int x, int y){
        v[x].push_back(y);
        v[y].push_back(x);
        S.push_back(edge{x, y});
    }
    int dfs_bcc(int u, int fa){
        int lowu = pre[u] = ++dfs_clock;
        int child = 0;
        for (int i = 0; i < v[u].size(); i++){
            int x = v[u][i];
            if (!pre[x]){
                child++;
                S.push_back(edge{u, x});
                int lowx = dfs_bcc(x, u);
                lowu = min(lowu, lowx);
                if (lowx >= pre[u]){
                    bcc_cnt++;
                    iscut[u] = 1;
                    while (1){
                        edge now = S.back();
                        S.pop_back();
                        if (bccno[now.u] != bcc_cnt) {
                            bccno[now.u] = bcc_cnt;
                            bcc[bcc_cnt].push_back(now.u);
                        }
                        if (bccno[now.v] != bcc_cnt){
                            bccno[now.v] = bcc_cnt;
                            bcc[bcc_cnt].push_back(now.v);
                        }
                        if (now.u == u && now.v == x)
                            break;
                    }
                }
            }
            else if (pre[x] < pre[u] && x != fa) {
                S.push_back(edge{u, x});
                lowu = min(lowu, pre[x]);
            }
        }
    }
}

```

```

    }
    if (fa < 0 && child == 1)
        iscut[u] = 0;
    return lowu;
}
void solve(int nn) {
    memset(pre, 0, sizeof(pre));
    memset(iscut, 0, sizeof(iscut));
    memset(bccno, 0, sizeof(bccno));
    dfs_clock = bcc_cnt = 0;
    for (int i = 0; i < nn; i++)
        bcc[i].clear();
    for (int i = 0; i < nn; i++) {
//Note that you may want to change the range of index.
        if (!pre[i])
            dfs_bcc(i, -1);
    }
}
} graph;

```

5.4 BCC 橋

```

struct BCC{
    int n, m;
    vector<int> v[maxn];
    int dfs_clock;
    int bcc_cnt; //Number of bcc
    vector<int> bcc[maxn]; //1~bcc_cnt
    map<int, bool> bridge[maxn];
    // Using bridge[i][j] to record the edge connects
    point i and point j.
    // complexity O(log)
    int pre[maxn], bccno[maxn];
    bool book[maxn];
    void init(int _n){
        n = _n;
        for (int i = 1; i <= n; i++){
            v[i].clear();
        }
    }
    void add_edge(int x, int y){
        v[x].push_back(y);
        v[y].push_back(x);
    }
    int dfs_bcc(int u, int fa){
        int lowu = pre[u] = ++dfs_clock;
        for (int i = 0; i < v[u].size(); i++){
            int x = v[u][i];
            if (!pre[x]){
                int lowx = dfs_bcc(x, u);
                lowu = min(lowu, lowx);
                if (lowx > pre[u]) {
                    bridge[u][x] = 1;
                    bridge[x][u] = 1;
                }
            }
            else if (pre[x] < pre[u] && x != fa)
                lowu = min(lowu, pre[x]);
        }
        return lowu;
    }
    void dfs_getbcc(int now){
        book[now] = 1;
        bccno[now] = bcc_cnt;
        bcc[bcc_cnt].push_back(now);
        for (int i = 0; i < v[now].size(); i++){
            if (!book[v[now][i]]
                && !bridge[now][v[now][i]])
                dfs_getbcc(v[now][i]);
        }
    }
    void find_bcc(int nn){
        memset(pre, 0, sizeof(pre));
        memset(bccno, 0, sizeof(bccno));
        dfs_clock = bcc_cnt = 0;
        for (int i = 1; i <= nn; i++)
            bridge[i].clear();
        for (int i = 1; i <= nn; i++)
            bcc[i].clear();
        for (int i = 1; i <= nn; i++)

```

```

        { //Note that you may want to change the node
        range.
            if (!pre[i])
                dfs_bcc(i, -1);
        }
        memset(book, 0, sizeof(book));
        for (int i = 1; i <= nn; i++){
            if (!book[i]) {
                bcc_cnt++;
                dfs_getbcc(i);
            }
        }
    } graph;

```

5.5 SCC

```

int n, m;
vector<int> v[maxn], rv[maxn]; //都要連!!
int scc_cnt; //Number of scc
int used[maxn], sccno[maxn];
vector<int> vs;

void dfs1(int now){
    used[now] = 1;
    for (int i = 0; i < v[now].size(); i++) {
        if (!used[v[now][i]])
            dfs1(v[now][i]);
    }
    vs.push_back(now);
}

void dfs2(int now){
    used[now] = 1;
    sccno[now] = scc_cnt;
    for (int i = 0; i < rv[now].size(); i++){
        if (!used[rv[now][i]])
            dfs2(rv[now][i]);
    }
}

void find_scc(int nn){
    memset(sccno, 0, sizeof(sccno));
    scc_cnt = 0;
    memset(used, 0, sizeof(used));
    for (int i = 1; i <= nn; i++){
//Note that you may want to change the node range.
        if (!used[i]) dfs1(i);
    }
    memset(used, 0, sizeof(used));
    for (int i = vs.size() - 1; i >= 0; i--){
        if (!used[vs[i]]){
            scc_cnt++;
            dfs2(vs[i]);
        }
    }
    vs.clear();
}

```

5.6 2-SAT

i 表示第 i 個敘述為真，i+n 表示第 i 個敘述為假
 sccno[i]==sccno[i+n]相等=>炸掉
 sccno[i]>sccno[i+n] true

5.7 有向最小生成樹(最小樹形圖)

```

struct MDST{
#define MAXN 1010
#define MAXM 1000010
#define INF INT_MAX
    struct Edge{ int from, to, cost; };
    int n, m;
    Edge edge[MAXM];
    int pre[MAXN]; //存儲父節點
    int vis[MAXN]; //標記作用
    int id[MAXN]; //id[i]記錄節點 i 所在環的編號
    int in[MAXN]; //in[i]記錄 i 入邊中最小的權值
    void init(int _n){
        n = _n;
        m = 0;
    }

```

```

}
void addEdge(int u, int v, int c) { edge[m++] =
Edge{u, v, c}; }
int zhuliu(int root) { //root 根 n 點數 m 邊數
int res = 0, u, v;
while (1){
for (int i = 0; i < n; i++){
in[i] = INF; //初始化
for (int i = 0; i < m; i++){
Edge E = edge[i];
if (E.from != E.to && E.cost < in[E.to]){
pre[E.to] = E.from; //記錄前驅
in[E.to] = E.cost; //更新
}
}
}
for (int i = 0; i < n; i++){
if (i != root && in[i] == INF)
return -1; //有其他孤立點 則不存在最小樹狀圖
//找有向環
int tn = 0; //記錄當前查找中 環的總數
memset(id, -1, sizeof(id));
memset(vis, -1, sizeof(vis));
in[root] = 0; //根
for (int i = 0; i < n; i++){
res += in[i]; //累加
v = i;
//找圖中的有向環 三種情況會終止 while 迴圈
//1,直到出現帶有同樣標記的點說明成環
//2,節點已經屬於其他環
//3,遍歷到根
while (vis[v] != i && id[v] == -1 && v != root)
{
vis[v] = i; //標記
v = pre[v]; //一直向上找
} //因為找到某節點屬於其他環 或者 遍歷到根 說明當前
沒有找到有向環
if (v != root && id[v] == -1) { //必須上述查找已
經找到有向環
for (int u = pre[v]; u != v; u = pre[u])
id[u] = tn; //記錄節點所屬的 環編號
id[v] = tn++; //記錄節點所屬的 環編號 環編號累加
}
}
if (tn == 0)
break; //不存在有向環
//可能存在獨立點
for (int i = 0; i < n; i++){
if (id[i] == -1)
id[i] = tn++; //環數累加
//對有向環縮點 和 SCC 縮點很像吧
for (int i = 0; i < m; i++) {
v = edge[i].to;
edge[i].from = id[edge[i].from];
edge[i].to = id[edge[i].to];
//<u, v>有向邊
//兩點不在同一個環 u 到 v 的距離為 邊權 cost - in[v]
if (edge[i].from != edge[i].to)
edge[i].cost -= in[v]; //更新邊權值 繼續下一條邊
的判定
}
n = tn; //以環總數為下次操作的點數 繼續執行上述操作 直
到沒有環
root = id[root];
}
return res;
}
} graph;

```

5.8 二分圖匹配(Bipartite Matching)

```

/*
最大匹配+最小邊涵蓋=最大獨立集合+最小點涵蓋=V(general)
最大匹配=最小點涵蓋(二分圖)
DAG 最短路徑覆蓋=點數-最大匹配
*/

```

```

#define MAX_V 1005 //max(|U|,|V|)
struct Bipartite_Matching {
int V;
vector<int> G[MAX_V]; //V -> U
vector<int> rG[MAX_V]; //V -> U 可註解掉
int match_u[MAX_V]; //match[U]=V
int match_v[MAX_V]; //match[V]=U 可註解掉
bool used[MAX_V]; //used[V] are used for dfs

bool r[MAX_V], c[MAX_V]; //最小點覆蓋用, 所求點 i 為
r[i]=0 或者 c[i]=1
void INIT(int x){
V = x;
for (int i = 0; i < MAX_V; i++){
G[i].clear();
rG[i].clear(); //可註解掉
}
}
void add_edge(int x, int y){
G[x].push_back(y);
rG[y].push_back(x); //可註解掉
}
bool dfs(int now){
used[now] = 1;
r[now] = 1; //最小點覆蓋
for (int i = 0; i < G[now].size(); i++){
int x = G[now][i], w = match_u[x];
c[x] = 1; //最小點覆蓋
if (w == -1 || (!used[w] && dfs(w))){
match_u[x] = now;
match_v[now] = x; //可註解掉
return 1;
}
}
}
return 0;
}
int bipartite_matching(){
int res = 0;
memset(match_u, -1, sizeof(match_u));
memset(match_v, -1, sizeof(match_v)); //可註解掉
for (int i = 0; i < V; i++){
if (match_v[i] == -1) { //可註解掉
memset(used, 0, sizeof(used));
if (dfs(i))
res++;
}
}
return res;
}
void min_point_cover() {
for (int i = 0; i < V; i++)
r[i] = c[i] = 0;
for (int i = 0; i < V; i++) {
memset(used, 0, sizeof(used));
if (match_v[i] == -1)
dfs(i);
}
}
}
} BM;

```

5.9 二分圖最佳完美匹配(Kuhn Munkres)

```

struct KM{
static const int MXN = 1005;
#define INF 2147483647 // LL
int n, match[MXN], vx[MXN], vy[MXN]; //match[y][x]
int edge[MXN][MXN], lx[MXN], ly[MXN], slack[MXN];
// ^^^^ LL
// construct lx[] and ly[] satisfies
lx[x]+ly[y]>=edge[x][y], and minimize the sum of lx[]
and ly[]
// if lx[x]+ly[y]==edge[x][y], match x and y.
// maximum weight equals to the sum of lx and ly
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 addEdge(int x, int y, int w) { // LL
    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])
        //如果是 double 要改成 lx[x]+ly[y]>edge[x][y]+eps
            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() { //LL
    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; //LL
    for (int i = 0; i < n; i++)
        res += edge[match[i]][i];
    return res;
}
} graph;

```

5.10 Maximum General graph Matching

```

const int N = 514, E = (2e5) * 2;
struct Graph{
    int to[E], bro[E], head[N], e;
    int lnk[N], vis[N], stp, n;
    void init(int _n){
        stp = 0;
        e = 1;
        n = _n;
        for (int i = 1; i <= n; i++)
            lnk[i]=vis[i]=bro[i]=head[i]=to[i]=0;
    }
    void add_edge(int u, int v){
        to[e] = v, bro[e] = head[u], head[u] = e++;
        to[e] = u, bro[e] = head[v], head[v] = e++;
    }
    bool dfs(int x){
        vis[x] = stp;
        for (int i = head[x]; i; i = bro[i]) {
            int v = to[i];
            if (!lnk[v]) {
                lnk[x] = v, lnk[v] = x;
                return true;
            }
            else if (vis[lnk[v]] < stp) {

```

```

                int w = lnk[v];
                lnk[x] = v, lnk[v] = x, lnk[w] = 0;
                if (dfs(w))
                {
                    return true;
                }
                lnk[w] = v, lnk[v] = w, lnk[x] = 0;
            }
        }
        return false;
    }
    int solve(){
        int ans = 0;
        for (int i = 1; i <= n; i++)
            if (!lnk[i]){
                stp++;
                ans += dfs(i);
            }
        return ans;
    }
} graph;

```

5.11 無向圖最小割(SW min-cut)

```

// global min cut struct SW(無向圖)
struct SW_min_cut{ // O(V^3)
    static const int MXN = 514;
    int n, vst[MXN], del[MXN];
    int edge[MXN][MXN], wei[MXN];
    void init(int _n){
        n = _n;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                edge[i][j] = 0;
        for (int i = 0; i < n; i++)
            del[i] = 0;
    }
    void addEdge(int u, int v, int w){
        edge[u][v] += w;
        edge[v][u] += w;
    }
    void search(int &s, int &t){
        memset(vst, 0, sizeof(vst));
        memset(wei, 0, sizeof(wei));
        s = t = -1;
        while (true){
            int mx = -1, cur = 0;
            for (int i = 0; i < n; i++)
                if (!del[i] && !vst[i] && mx < wei[i])
                    cur = i, mx = wei[i];
            if (mx == -1)
                break;
            vst[cur] = 1;
            s = t;
            t = cur;
            for (int i = 0; i < n; i++)
                if (!vst[i] && !del[i])
                    wei[i] += edge[cur][i];
        }
    }
    int solve(){
        int res = 2147483647;
        for (int i = 0, x, y; i < n - 1; i++){
            search(x, y);
            res = min(res, wei[y]);
            del[y] = 1;
            for (int j = 0; j < n; j++)
                edge[x][j] = (edge[j][x] += edge[y][j]);
        }
        return res;
    }
} graph;

```

5.12 最大團

```

#define N 111
struct MaxClique { // 0-base
    typedef bitset<N> Int;
    Int linkto[N], v[N];

```



```

int n;
void init(int _n){
    n = _n;
    for (int i = 0; i < n; i++){
        linkto[i].reset();
        v[i].reset();
    }
}
void addEdge(int a, int b) { v[a][b] = v[b][a] = 1; }
int popcount(const Int &val) { return val.count(); }
int lowbit(const Int &val) { return val._Find_first(); }
int ans, stk[N];
int id[N], di[N], deg[N];
Int cans;
void maxclique(int elem_num, Int candi){
    if (elem_num > ans){
        ans = elem_num;
        cans.reset();
        for (int i = 0; i < elem_num; i++){
            cans[id[stk[i]]] = 1;
        }
    }
    int potential = elem_num + popcount(candi);
    if (potential <= ans) return;
    int pivot = lowbit(candi);
    Int smaller_candi = candi & (~linkto[pivot]);
    while (smaller_candi.count() && potential > ans) {
        int next = lowbit(smaller_candi);
        candi[next] = !candi[next];
        smaller_candi[next] = !smaller_candi[next];
        potential--;
        if (next == pivot || (smaller_candi & linkto[next]).count()){
            stk[elem_num] = next;
            maxclique(elem_num + 1, candi & linkto[next]);
        }
    }
}
int solve(){
    for (int i = 0; i < n; i++) {
        id[i] = i;
        deg[i] = v[i].count();
    }
    sort(id, id + n, [&](int id1, int id2) { return deg[id1] > deg[id2]; });
    for (int i = 0; i < n; i++)
        di[id[i]] = i;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (v[i][j])
                linkto[di[i]][di[j]] = 1;
    Int cand;
    cand.reset();
    for (int i = 0; i < n; i++)
        cand[i] = 1;
    ans = 1;
    cans.reset();
    cans[0] = 1;
    maxclique(0, cand);
    return ans;
}
} solver;

```

5.13 最大團數量

```

// bool g[][] : adjacent array indexed from 1 to n
void dfs(int sz){
    int i, j, k, t, cnt, best = 0;
    if(ne[sz]==ce[sz]){ if (ce[sz]==0) ++ans; return; }
    for(t=0, i=1; i<=ne[sz]; ++i){
        for (cnt=0, j=ne[sz]+1; j<=ce[sz]; ++j)
            if (!g[1st[sz][i]][1st[sz][j]]) ++cnt;
        if (t==0 || cnt<best) t=i, best=cnt;
    } if (t && best<=0) return;
}

```

```

for (k=ne[sz]+1; k<=ce[sz]; ++k) {
    if (t>0){ for (i=k; i<=ce[sz]; ++i)
        if (!g[1st[sz][t]][1st[sz][i]]) break;
        swap(1st[sz][k], 1st[sz][i]);
    } i=1st[sz][k]; ne[sz+1]=ce[sz+1]=0;
    for (j=1; j<k; ++j)if (g[i][1st[sz][j]])
        1st[sz+1][++ne[sz+1]]=1st[sz][j];
    for (ce[sz+1]=ne[sz+1], j=k+1; j<=ce[sz]; ++j)
        if (g[i][1st[sz][j]]) 1st[sz+1][++ce[sz+1]]=1st[sz][j];
    dfs(sz+1); ++ne[sz]; --best;
    for (j=k+1, cnt=0; j<=ce[sz]; ++j) if (!g[i][1st[sz][j]]) ++cnt;
    if (t==0 || cnt<best) t=k, best=cnt;
    if (t && best<=0) break;
}
}
void work(){
    ne[0]=0; ce[0]=0;
    for(int i=1; i<=n; ++i) 1st[0][++ce[0]]=i;
    ans=0; dfs(0);
}

```

5.14 Minimum mean cycle

也可以二分搜答案並用 SPFA 找負環(如果|V|太大存不下)。

```

/* minimum mean cycle O(VE) */
struct MMC{
#define SZ(c) ((int)(c).size())
#define E 101010
#define V 1021
#define inf 1e9 //可能不夠大
#define eps 1e-6
    struct Edge{
        int v, u;
        double c;
    };
    int n, m, prv[V][V], prve[V][V], vst[V];
    Edge e[E];
    vector<int> edgeID, cycle, rho;
    double d[V][V];
    void init(int _n){
        n = _n;
        m = 0;
    }
    // WARNING: TYPE matters
    void addEdge(int vi, int ui, double ci){
        e[m++] = {vi, ui, ci};
    }
    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 solve(){
        // 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);
        }
    }
}

```

```

memset(vst, 0, sizeof(vst));
edgeID.clear();
cycle.clear();
rho.clear();
for (int i = n; !vst[st]; st = prv[i--][st]){
    vst[st]++;
    edgeID.push_back(prve[i][st]);
    rho.push_back(st);
}
while (vst[st] != 2){
    int v = rho.back();
    rho.pop_back();
    cycle.push_back(v);
    vst[v]++;
}
reverse(edgeID.begin(), edgeID.end());
edgeID.resize(SZ(cycle));
return mmc;
}
} mmc;

```

5.15 Directed Graph Min Cost Cycle

如果點數邊數夠小就直接 Floyd 後看哪個 $e[i][i]$ 最小。

```

// works in O(N M)
#define INF 1000000000000000000ll
#define N 5010 //通常別開這麼大，會MLE
#define M 200010
struct edge{
    int to; ll w;
    edge(int a = 0, ll b = 0) : to(a), w(b) {}
};
struct node{
    ll d; int u, next;
    node(ll a=0, int b=0, int c=0):d(a),u(b),next(c) {}
} b[M];
struct DirectedGraphMinCycle
{
    vector<edge> g[N], grev[N];
    ll dp[N][N], p[N], d[N], mu;
    bool inq[N];
    int n, bn, bsz, hd[N];
    void b_insert(ll d, int u){
        int i = d / mu;
        if (i >= bn)
            return;
        b[++bsz] = node(d, u, hd[i]);
        hd[i] = bsz;
    }
    void init(int _n){
        n = _n;
        for (int i = 1; i <= n; i++){
            g[i].clear();
        }
    }
    void addEdge(int ai, int bi, ll ci){
        g[ai].push_back(edge(bi, ci));
    }
    ll solve(){
        fill(dp[0], dp[0] + n + 1, 0);
        for (int i = 1; i <= n; i++){
            fill(dp[i] + 1, dp[i] + n + 1, INF);
            for (int j = 1; j <= n; j++){
                if (dp[i - 1][j] < INF){
                    for (int k = 0; k < (int)g[j].size(); k++){
                        dp[i][g[j][k].to] = min(dp[i][g[j][k].to],
                                                dp[i - 1][j] + g[j][k].w);
                    }
                }
            }
        }
        mu = INF;
        ll bunbo = 1;
        for (int i = 1; i <= n; i++){
            if (dp[n][i] < INF){
                ll a = -INF, b = 1;
                for (int j = 0; j <= n - 1; j++){
                    if (dp[j][i] < INF){
                        if (a * (n - j) < b * (dp[n][i] - dp[j][i])){
                            a = dp[n][i] - dp[j][i];
                            b = n - j;
                        }
                    }
                }
            }
        }
    }
}

```

```

}
if (mu * b > bunbo * a)
    mu = a, bunbo = b;
}
if (mu < 0) return -1; // negative cycle
if (mu == INF) return INF; // no cycle
if (mu == 0) return 0;
for (int i = 1; i <= n; i++){
    for (int j = 0; j < (int)g[i].size(); j++){
        g[i][j].w *= bunbo;
    }
    memset(p, 0, sizeof(p));
    queue<int> q;
    for (int i = 1; i <= n; i++){
        q.push(i);
        inq[i] = true;
    }
    while (!q.empty()){
        int i = q.front();
        q.pop();
        inq[i] = false;
        for (int j = 0; j < (int)g[i].size(); j++){
            if (p[g[i][j].to] > p[i] + g[i][j].w - mu){
                p[g[i][j].to] = p[i] + g[i][j].w - mu;
                if (!inq[g[i][j].to]){
                    q.push(g[i][j].to);
                    inq[g[i][j].to] = true;
                }
            }
        }
    }
}
for (int i = 1; i <= n; i++){
    grev[i].clear();
}
for (int i = 1; i <= n; i++){
    for (int j = 0; j < (int)g[i].size(); j++){
        g[i][j].w += p[i] - p[g[i][j].to];
        grev[g[i][j].to].push_back(edge(i, g[i][j].w));
    }
}
ll mldc = n * mu;
for (int i = 1; i <= n; i++){
    bn = mldc / mu, bsz = 0;
    memset(hd, 0, sizeof(hd));
    fill(d + i + 1, d + n + 1, INF);
    b_insert(d[i] = 0, i);
    for (int j = 0; j <= bn - 1; j++){
        for (int k = hd[j]; k; k = b[k].next){
            int u = b[k].u;
            ll du = b[k].d;
            if (du > d[u])
                continue;
            for (int l = 0; l < (int)g[u].size(); l++){
                if (g[u][l].to > i) {
                    if (d[g[u][l].to] > du + g[u][l].w) {
                        d[g[u][l].to] = du + g[u][l].w;
                        b_insert(d[g[u][l].to], g[u][l].to);
                    }
                }
            }
        }
    }
    for (int j = 0; j < (int)grev[i].size(); j++){
        if (grev[i][j].to > i)
            mldc = min(mldc, d[grev[i][j].to] + grev[i][j].w);
    }
}
return mldc / bunbo;
}
} graph;

```

5.16 Minimum Steiner Tree

在無向圖上找一棵子樹，可以把 P 中的點連通起來，且邊權總和最小。
令 $dp[S][i]$ 表示以點 i 為根，以 $S \subseteq P$ 為 terminal set 構造出來的斯坦納樹，這樣我們最後的答案就會是 $dp[P][u \in P]$ 。

$dp[S][i] = \min(dp[T][j] + dp[S-T][j] + dis(i, j) : j \in V, T \subseteq S)$
 $dis(i, j)$ 表示 $i \sim j$ 的最短路徑

這其實還可以優化，令 $H[j] = \min(dp[T][j] + dp[S-T][j] : T \subseteq S)$

則 $dp[S][i] = \min(H[j] + dis(i, j) : j \in V \setminus \{i\})$

$H[i]$ 是可以被預先算出來的。

// $O(V^3 T + V^2 2^T)$

```

struct SteinerTree{
#define V 33

```

```

#define T 8
#define INF 1023456789
int n, dst[V][V], dp[1 << T][V], tdst[V];
void init(int _n){
    n = _n;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            dst[i][j] = INF;
            dst[i][i] = 0;
        }
    }
    void add_edge(int ui, int vi, int wi){
        dst[ui][vi] = min(dst[ui][vi], wi);
        dst[vi][ui] = min(dst[vi][ui], wi);
    }
    void shortest_path(){
        for (int k = 0; k < n; k++){
            for (int i = 0; i < n; i++){
                for (int j = 0; j < n; j++){
                    dst[i][j] = min(dst[i][j], dst[i][k] + dst[k][j]);
                }
            }
        }
        int solve(const vector<int> &ter){
            shortest_path();
            int t = (int)ter.size();
            for (int i = 0; i < (1 << t); i++){
                for (int j = 0; j < n; j++){
                    dp[i][j] = INF;
                }
                for (int i = 0; i < n; i++){
                    dp[0][i] = 0;
                }
                for (int msk = 1; msk < (1 << t); msk++){
                    if (msk == (msk & (-msk))){
                        int who = __lg(msk);
                        for (int i = 0; i < n; i++){
                            dp[msk][i] = dst[ter[who]][i];
                            continue;
                        }
                        for (int i = 0; i < n; i++){
                            for (int submsk = (msk - 1) & msk; submsk; submsk = (submsk - 1) & msk){
                                dp[msk][i] = min(dp[msk][i], dp[submsk][i] + dp[msk ^ submsk][i]);
                            }
                            for (int i = 0; i < n; i++){
                                tdst[i] = INF;
                                for (int j = 0; j < n; j++){
                                    tdst[i] = min(tdst[i], dp[msk][j] + dst[j][i]);
                                }
                            }
                            for (int i = 0; i < n; i++){
                                dp[msk][i] = tdst[i];
                            }
                        }
                    }
                }
                int ans = INF;
                for (int i = 0; i < n; i++){
                    ans = min(ans, dp[(1 << t) - 1][i]);
                }
                return ans;
            }
        } solver;
    }

```

5.17 DominatorTree

對於有向圖 G (可能有環), 其中起點 r 可以到達所有點, 當 u 是所有到達 v 的路徑的必經點時, 稱 u 支配 v 。可以構建支配樹, 其中每個點被所有它的祖先支配, 又支配它子樹中的結點。

```

const int MAXN = 100010;
#define REP(i, s, e) for (int i = (s); i <= (e); i++)
#define REPD(i, s, e) for (int i = (s); i >= (e); i--)
struct DominatorTree{
    int n, m, s; //點數 n, 邊數 m, 起點為 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];
    //支配樹上 i 的 parent 為 idom[i], 若無 parent 就會是 0
    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 _m, int _s){
    ts = 0;
    n = _n; m = _m; s = _s;
    REP(i, 1, n)
        g[i].clear(),
        pred[i].clear();
}
void addEdge(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(){
    REP(i, 1, n){
        dfn[i] = nfd[i] = 0;
        cov[i].clear();
        mom[i] = mn[i] = sdom[i] = i;
    }
    dfs(s);
    REPD(i, n, 2){
        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();
    }
    REP(i, 2, n){
        int u = nfd[i];
        if (u == 0)
            continue;
        if (idom[u] != sdom[u])
            idom[u] = idom[idom[u]];
    }
}
} domT;

```

5.18 The first k Shortest Path

```

// time:  $O(|E| \lg |E| + |V| \lg |V| + K)$ 
// memory:  $O(|E| \lg |E| + |V|)$ 
struct KSP{
    // 1-base
#define LL long long
#define N 1005
#define INF INT_MAX
    struct nd{
        int u, v, d;
        nd(int ui = 0, int vi = 0, int di = INF){
            u = ui;
            v = vi;
            d = di;
        }
    }

```

```

};
struct heap {
    nd *edge;
    int dep;
    heap *chd[4];
};
static int cmp(heap *a, heap *b) { return a->edge-
>d > b->edge->d; }
struct node{
    int v;
    LL d;
    heap *H;
    nd *E;
    node() {}
    node(LL _d, int _v, nd *_E){
        d = _d; v = _v; E = _E;
    }
    node(heap *_H, LL _d){
        H = _H; d = _d;
    }
    friend bool operator<(node a, node b) { return
a.d > b.d; }
};
int n, k, s, t, dst[N];
nd *nxt[N];
vector<nd *> g[N], rg[N];
heap *nullNd, *head[N];
void init(int _n, int _k, int _s, int _t){
    n = _n; k = _k; s = _s; t = _t;
    for (int i = 1; i <= n; i++){
        g[i].clear();
        rg[i].clear();
        nxt[i] = NULL;
        head[i] = NULL;
        dst[i] = -1;
    }
}
void addEdge(int ui, int vi, int di){
    nd *e = new nd(ui, vi, di);
    g[ui].push_back(e);
    rg[vi].push_back(e);
}
queue<int> dfsQ;
void dijkstra(){
    while (dfsQ.size())
        dfsQ.pop();
    priority_queue<node> Q;
    Q.push(node(0, t, NULL));
    while (!Q.empty()){
        node p = Q.top();
        Q.pop();
        if (dst[p.v] != -1)
            continue;
        dst[p.v] = p.d;
        nxt[p.v] = p.E;
        dfsQ.push(p.v);
        for (auto e : rg[p.v])
            Q.push(node(p.d + e->d, e->u, e));
    }
}
heap *merge(heap *curNd, heap *newNd){
    if (curNd == nullNd)
        return newNd;
    heap *root = new heap;
    memcpy(root, curNd, sizeof(heap));
    if (newNd->edge->d < curNd->edge->d){
        root->edge = newNd->edge;
        root->chd[2] = newNd->chd[2];
        root->chd[3] = newNd->chd[3];
        newNd->edge = curNd->edge;
        newNd->chd[2] = curNd->chd[2];
        newNd->chd[3] = curNd->chd[3];
    }
    if (root->chd[0]->dep < root->chd[1]->dep)
        root->chd[0] = merge(root->chd[0], newNd);
    else
        root->chd[1] = merge(root->chd[1], newNd);
    root->dep = max(root->chd[0]->dep, root->chd[1]-
>dep) + 1;
}

```

```

return root;
}
vector<heap *> V;
void build(){
    nullNd = new heap;
    nullNd->dep = 0;
    nullNd->edge = new nd;
    fill(nullNd->chd, nullNd->chd + 4, nullNd);
    while (not dfsQ.empty()){
        int u = dfsQ.front();
        dfsQ.pop();
        if (!nxt[u]) head[u] = nullNd;
        else head[u] = head[nxt[u]->v];
        V.clear();
        for (auto &&e : g[u]){
            int v = e->v;
            if (dst[v] == -1) continue;
            e->d += dst[v] - dst[u];
            if (nxt[u] != e) {
                heap *p = new heap;
                fill(p->chd, p->chd + 4, nullNd);
                p->dep = 1;
                p->edge = e;
                V.push_back(p);
            }
        }
        if (V.empty())
            continue;
        make_heap(V.begin(), V.end(), cmp);
#define L(X) ((X << 1) + 1)
#define R(X) ((X << 1) + 2)
        for (size_t i = 0; i < V.size(); i++){
            if (L(i) < V.size())
                V[i]->chd[2] = V[L(i)];
            else
                V[i]->chd[2] = nullNd;
            if (R(i) < V.size())
                V[i]->chd[3] = V[R(i)];
            else
                V[i]->chd[3] = nullNd;
        }
        head[u] = merge(head[u], V.front());
    }
}
vector<LL> ans; //答案存在這，前 k 短路徑的長度
void first_K(){
    ans.clear();
    priority_queue<node> Q;
    if (dst[s] == -1) return;
    ans.push_back(dst[s]);
    if (head[s] != nullNd)
        Q.push(node(head[s], dst[s]+head[s]->edge->d));
    for (int _ = 1; _ < k and not Q.empty(); _++){
        node p = Q.top(), q;
        Q.pop();
        ans.push_back(p.d);
        if (head[p.H->edge->v] != nullNd){
            q.H = head[p.H->edge->v];
            q.d = p.d + q.H->edge->d;
            Q.push(q);
        }
        for (int i = 0; i < 4; i++)
            if (p.H->chd[i] != nullNd){
                q.H = p.H->chd[i];
                q.d = p.d - p.H->edge->d + p.H-
>chd[i]->edge->d;
                Q.push(q);
            }
    }
}
void solve(){
    dijkstra();
    build();
    first_K();
}
} solver;

```

5.19 SPFA

判有向圖有沒有負環，可以設一個超級源點，從那個點 spfa

```
procedure Shortest-Path-Faster-Algorithm(G, s)
  for each vertex v ≠ s in V(G)
    d(v) := ∞
  d(s) := 0
  offer s into Q
  cnt[s] = 0 //cnt 記錄更新到目前用了幾條邊
  while Q is not empty
    u := poll Q
    for each edge (u, v) in E(G)
      if d(u) + w(u, v) < d(v) then
        cnt[v] = cnt[u] + 1;
        //如果 cnt[v] > n 表示有負環
        d(v) := d(u) + w(u, v)
        if v is not in Q then
          offer v into Q
```

判有向圖：設超級源點連到每個點，開始 dfs 某個點設 inque=1，dfs 完設 inque=0。如果 dfs 到某個 inque=1 的點表示有圈。

5.20 DLX

```
const int maxnode = 100100; //最多多少個‘1’
const int MaxM = 1010;
const int MaxN = 1010;

struct DLX{
  int n, m, SIZE;
  int U[maxnode], D[maxnode], R[maxnode], L[maxnode],
  Row[maxnode], Col[maxnode];
  int H[MaxN], S[MaxM];
  int ansd, ans[MaxN];
  void init(int _n, int _m) {
    n = _n, m = _m;
    for (int i = 0; i <= m; i++) {
      S[i] = 0, U[i] = D[i] = i;
      L[i] = i - 1, R[i] = i + 1;
    }
    R[m] = 0, L[0] = m, SIZE = m;
    for (int i = 1; i <= n; i++)
      H[i] = -1;
    ansd = INT_MAX;
  }
  void Link(int r, int c) {
    ++S[Col[++SIZE] = c], Row[SIZE] = r;
    D[SIZE] = D[c], U[D[c]] = SIZE;
    U[SIZE] = c, D[c] = SIZE;
    if (H[r] < 0)
      H[r] = L[SIZE] = R[SIZE] = SIZE;
    else {
      R[SIZE] = R[H[r]], L[R[H[r]]] = SIZE;
      L[SIZE] = H[r], R[H[r]] = SIZE;
    }
  }
  void exact_Remove(int c) {
    L[R[c]] = L[c], R[L[c]] = R[c];
    for (int i = D[c]; i != c; i = D[i])
      for (int j = R[i]; j != i; j = R[j]) {
        U[D[j]] = U[j], D[U[j]] = D[j];
        --S[Col[j]];
      }
  }
  void repeat_remove(int c){
    for (int i = D[c]; i != c; i = D[i])
      L[R[i]] = L[i], R[L[i]] = R[i];
  }
  void repeat_resume(int c){
    for (int i = U[c]; i != c; i = U[i])
      L[R[i]] = R[L[i]] = i;
  }
  int f() { //估價函數
    bool vv[MaxM];
    int ret = 0, c, i, j;
    for (c = R[0]; c != 0; c = R[c])
      vv[c] = 1;
    for (c = R[0]; c != 0; c = R[c])
      if (vv[c]) {
```

```
      ++ret, vv[c] = 0;
      for (i = D[c]; i != c; i = D[i])
        for (j = R[i]; j != i; j = R[j])
          vv[Col[j]] = 0;
    }
    return ret;
}
void repeat_dance(int d) {
  if (d + f() >= ansd) return;
  if (R[0] == 0) {
    if (d < ansd) ansd = d;
    return;
  }
  int c = R[0], i, j;
  for (i = R[0]; i; i = R[i])
    if (S[i] < S[c])
      c = i;
  for (i = D[c]; i != c; i = D[i])
  {
    repeat_remove(i);
    for (j = R[i]; j != i; j = R[j])
      repeat_remove(j);
    repeat_dance(d + 1);
    for (j = L[i]; j != i; j = L[j])
      repeat_resume(j);
    repeat_resume(i);
  }
}
void exact_resume(int c) {
  for (int i = U[c]; i != c; i = U[i])
    for (int j = L[i]; j != i; j = L[j])
      ++S[Col[U[D[j]] = D[U[j]] = j]];
  L[R[c]] = R[L[c]] = c;
}
bool exact_Dance(int d) {
  if (R[0] == 0){
    ansd = d;
    return true;
  }
  int c = R[0];
  for (int i = R[0]; i != 0; i = R[i])
    if (S[i] < S[c])
      c = i;
  exact_Remove(c);
  for (int i = D[c]; i != c; i = D[i]) {
    ans[d] = Row[i];
    for (int j = R[i]; j != i; j = R[j])
      exact_Remove(Col[j]);
    if (exact_Dance(d + 1))
      return true;
    for (int j = L[i]; j != i; j = L[j])
      exact_resume(Col[j]);
  }
  exact_resume(c);
  return false;
}
};
```

5.21 混合圖歐拉迴路判定

對所有的無向邊隨便定向，之後再進行調整。

統計每個點的出入度，如果有某個點出入度之差為奇數，則不存在歐拉回路。把每個點的出入度之差除以 2，得 x 。則對每個頂點改變與之相連的 x 條邊的方向就可以使得該點出入度相等。現在問題就變成了改變哪些邊的方向能讓每個點出入度相等了，構造網路流模型。

有向邊不能改變方向，所以不添加有向邊。對於在開始的時候任意定向的無向邊，按所定的方向加邊，容量為 1。

對於剛才提到的 x ，如果 x 大於 0，則建一條 s （源點）到當前點容量為 x 的邊，如果 x 小於 0，建一條從當前點到 t （匯點）容量為 $|x|$ 的邊。

這時與原點相連的都是缺少入度的點，與匯點相連的都是缺少出度的點。建圖完成了，求解最大流，如果能滿流分配，則存在歐拉回路。查看流量分配，所有流量非 0 的邊就是要改變方向的邊。

5.22 Euler tour

//求歐拉回路或歐拉路，鄰接陣形式，複雜度 $O(n^2)$

//返回路徑長度，path 返回路徑（有向圖是得到的是反向路徑）

```
//傳入圖的大小 n 和鄰接陣 mat，不相交鄰點邊權 0
//可以有自環與重邊，分為無向圖和有向圖
#define MAXN 100
void find_path(int n, int mat[][MAXN], int now, int &step, int *path){
    int i;
    for (i = n - 1; i >= 0; i--){
        while (mat[now][i]){
            mat[now][i]--; //無向圖加上 mat[i][now]--;
            find_path(n, mat, i, step, path);
        }
        path[step++] = now;
    }
}
int euclid_path(int n, int mat[][MAXN], int start, int *path){
    int ret = 0;
    find_path(n, mat, start, ret, path);
    return ret;
}
```

5.23 Stable Marriage Problem

```
// gp_boy[i][j]為第 i 個男的第 j 個喜歡的女的的編號
// gp_girl[i][j]為第 i 個女的對第 j 個男的好感度(越有好感數字越大)
// 答案：第 i 個男的和第 boy[i]個女的結婚，girl[boy[i]]=i
int n, gp_boy[505][505], gp_girl[505][505], boy[505], girl[505], rankl[505];
void Gale_Shapley(){
    memset(boy, 0, sizeof(boy));
    memset(girl, 0, sizeof(girl));
    for (int i = 1; i <= n; i++) rankl[i] = 1;
    while (1){
        int flag = 0;
        for (int i = 1; i <= n; i++){
            if (!boy[i]){
                int g = gp_boy[i][rankl[i]++];
                if (!girl[g]) boy[i] = g, girl[g] = i;
                else if (gp_girl[g][i] > gp_girl[g][girl[g]])
                    boy[girl[g]] = 0, girl[g] = i, boy[i] = g;
                flag = 1;
            }
        }
        if (!flag) break;
    }
}
```

6 String

6.1 KMP

```
int fail[maxn]; //Failure function
void getfail(char *P, int *fail){
    int mm = strlen(P);
    fail[0] = 0;
    fail[1] = 0;
    for (int i = 1; i < mm; i++){
        int j = fail[i];
        while (j && P[i] != P[j])
            j = fail[j];
        fail[i + 1] = (P[i] == P[j]) ? j + 1 : 0;
    }
}
void find(char *T, char *P, int *fail){ //T 裡面找 P
    int nn = strlen(T), mm = strlen(P);
    getfail(P, fail);
    int j = 0;
    for (int i = 0; i < nn; i++){
        while (j && T[i] != P[j])
            j = fail[j];
        if (T[i] == P[j])
            j++;
        if (j == mm)
            { //do something }
    }
}
// string a,b;
// a.find(b, pos)回傳 a 在 pos 後第一次出現 b 的位置，找不到
```

回傳 a.npos。

6.2 Suffix array

```
struct SuffixArray{
    string s;
    int n;
    vector<int> sa, pos, lcp, tmp, cnt; //lcp 就是 height
    vector<vector<int>> sparse;
    SuffixArray(string t) : s(t) {
        n = s.size();
        sa.assign(n, 0); pos.assign(n, 0);
        tmp.assign(n, 0); cnt.assign(max(n, 256), 0);
        lcp.assign(n - 1, 0); sparse.clear();
        BuildSA(); BuildLCP();
    }
    void CountingSort(int gap){
        fill(cnt.begin(), cnt.end(), 0);
        for (int i = 0; i < n; ++i){
            if (i + gap >= n){ ++cnt[0]; continue; }
            ++cnt[pos[i + gap] + 1];
        }
        int sum = 0;
        for (int i = 0; i < (int)cnt.size(); ++i){
            int cur = cnt[i];
            cnt[i] = sum;
            sum += cur;
        }
        for (int i = 0; i < n; ++i) {
            int cur = sa[i];
            if (cur + gap >= n) tmp[cnt[0]++] = cur;
            else tmp[cnt[pos[cur + gap] + 1]++] = cur;
        }
        for (int i = 0; i < n; ++i) sa[i] = tmp[i];
    }
    void BuildSA() {
        for (int i = 0; i < n; ++i)
            sa[i] = i, pos[i] = s[i];
        for (int gap = 1; gap <= 1) {
            auto cmp = [&](int a, int b) {
                if (pos[a] - pos[b])
                    return pos[a] < pos[b];
                a += gap;
                b += gap;
                return (a < n && b < n) ? pos[a] < pos[b] : a > b;
            };
            // sort(sa.begin(), sa.end(), cmp);
            CountingSort(gap);
            CountingSort(0);
            tmp[0] = 0;
            for (int i = 1; i < n; ++i)
                tmp[i] = tmp[i - 1] + cmp(sa[i - 1], sa[i]);
            for (int i = 0; i < n; ++i)
                pos[sa[i]] = tmp[i];
            if (tmp[n - 1] == n - 1) break;
        }
    }
    void BuildLCP() {
        for (int i = 0, k = 0; i < n; ++i)
            if (pos[i] - n + 1) {
                for (int j = sa[pos[i] + 1]; s[j + k] == s[i + k]; ++k);
                lcp[pos[i]] = k;
                if (k) k--;
            }
        sparse.push_back(lcp);
        for (int i = 0; ++i) {
            int len = n - (1 << (i + 1));
            if (len <= 0) break;
            sparse.push_back(vector<int>(len));
            for (int j = 0; j < len; ++j) {
                int left = sparse[i][j], right = sparse[i][j + (1 << i)];
                sparse[i + 1][j] = min(left, right);
            }
        }
    }
    int GetLCP(int l, int r)
    { // rank(就是 pos, 0-based)為 [l, r] 中間 height 的最小值
        if (l >= r)
            return n;
    }
```

```

int len = r - 1;
int lg = 31 - __builtin_clz(len);
return min(sparse[lg][l], sparse[lg][r-(1<<lg)]);
}
int solve_LCP(int a, int b)
{ // 字串為 0-based, a, b 為原本位置
  int l = min(pos[a], pos[b]);
  int r = max(pos[a], pos[b]);
  return min(min(n - a, n - b), GetLCP(l, r));
}
pair<int, int> FindOccurs(int p, int len)
{ // p=pos[原本位置] // rank p 的長度為 len 的前綴在 rank
  為[ret.first, ret.second]的前綴有出現
  pair<int, int> ret = {p, p};
  int lo = 0, hi = p;
  while (lo < hi) {
    int mid = (lo + hi) >> 1;
    if (GetLCP(mid, p) < len) lo = mid + 1;
    else hi = mid;
  }
  ret.first = lo;
  lo = p, hi = n - 1;
  while (lo < hi) {
    int mid = (lo + hi + 1) >> 1;
    if (GetLCP(p, mid) < len) hi = mid - 1;
    else lo = mid;
  }
  ret.second = hi;
  return ret; //sa[ret.first]~sa[ret.second]
}
};

```

6.3 Trie 與 AC 自動機

```

struct Trie{
  int ch[maxnode][sigma_size];
  //Total number of nodes / total number of characters
  int val[maxnode];
  int sz;
  int fail[maxnode]; //Failure function
  int last[maxnode]; //Suffix link
  void init(){
    sz = 1;
    memset(ch[0], 0, sizeof(ch[0]));
  }
  int idx(char c) { return c - 'a'; } //The number
  representing the character c may need to be changed
  void insert(char *s, int vv){
    int u = 0, nn = strlen(s);
    for (int i = 0; i < nn; i++){
      int c = idx(s[i]);
      if (!ch[u][c]){
        memset(ch[sz], 0, sizeof(ch[sz]));
        val[sz] = 0;
        ch[u][c] = sz++;
      }
      u = ch[u][c];
    }
    val[u] = vv;
  }
  void getfail(){
    queue<int> q;
    fail[0] = 0;
    for (int c = 0; c < sigma_size; c++){
      int u = ch[0][c];
      if (u){
        fail[u] = 0;
        q.push(u);
        last[u] = 0;
      }
    }
    while (!q.empty()){
      int r = q.front();
      q.pop();
      for (int c = 0; c < sigma_size; c++) {
        int u = ch[r][c];
        if (!u){
          ch[r][c] = ch[fail[r]][c];
          continue;
        }
      }
    }
  }
};

```

```

}
q.push(u);
int vv = fail[r];
while (vv & !ch[vv][c])
  vv = fail[vv];
fail[u] = ch[vv][c];
last[u] = val[fail[u]]? fail[u]:last[fail[u]];
//走到結點 u 可能代表找到很多種以 u 為結尾的字串，沿著 last[u]
這種邊走可以找出所有這種字串。
}
}
}
void print(int j){
  if (j){
    //do something
    print(last[j]);
  }
}
void find(char *T){
  int nn = strlen(T);
  int j = 0;
  for (int i = 0; i < nn; i++){
    int c = idx(T[i]);
    while (j && !ch[j][c])
      j = fail[j];
    j = ch[j][c];
    if (val[j])
      print(j);
    else if (last[j])
      print(last[j]);
  }
}
} ac;

```

6.4 BWT

```

struct BurrowsWheeler{
#define SIGMA 26
#define BASE 'a'
  vector<int> v[SIGMA];
  void BWT(char *ori, char *res){
    // make ori -> ori + ori
    // then build suffix array
  }
  void iBWT(char *ori, char *res){
    for (int i = 0; i < SIGMA; i++){
      v[i].clear();
      int len = strlen(ori);
      for (int i = 0; i < len; i++){
        v[ori[i] - BASE].push_back(i);
      }
      vector<int> a;
      for (int i = 0, ptr = 0; i < SIGMA; i++){
        for (auto j : v[i]){
          a.push_back(j);
          ori[ptr++] = BASE + i;
        }
      }
      for (int i = 0, ptr = 0; i < len; i++){
        res[i] = ori[a[ptr]];
        ptr = a[ptr];
      }
      res[len] = 0;
    }
  }
} bwt;

```

7 Data Structure

7.1 李超樹

```

struct LiChao_min{
  struct line{
    ll m, c;
    line(ll _m = 0, ll _c = 0){
      m = _m;
      c = _c;
    }
    ll eval(ll x) { return m * x + c; }
  };
  struct node{
    node *l, *r;
  };
};

```

```

line f;
node(line v){
    f = v;
    l = r = NULL;
}
};
typedef node *pnode;
pnode root;
int sz;
#define mid ((l + r) >> 1)
void insert(line &v, int l, int r, pnode &nd){
    if (!nd){
        nd = new node(v);
        return;
    }
    ll trl = nd->f.eval(l), trr = nd->f.eval(r);
    ll vl = v.eval(l), vr = v.eval(r);
    if (trl <= vl && trr <= vr)
        return;
    if (trl > vl && trr > vr){
        nd->f = v;
        return;
    }
    if (trl > vl)
        swap(nd->f, v);
    if (nd->f.eval(mid) < v.eval(mid))
        insert(v, mid + 1, r, nd->r);
    else
        swap(nd->f, v), insert(v, l, mid, nd->l);
}
ll query(int x, int l, int r, pnode &nd){
    if (!nd) return LLONG_MAX;
    if (l == r) return nd->f.eval(x);
    if (mid >= x)
        return min(nd->f.eval(x), query(x, l, mid, nd->l));
    return min(nd->f.eval(x), query(x, mid+1, r, nd->r));
}
/* -sz <= query_x <= sz */
void init(int _sz){
    sz = _sz + 1;
    root = NULL;
}
void add_line(ll m, ll c) {
    line v(m, c);
    insert(v, -sz, sz, root);
}
ll query(ll x) { return query(x, -sz, sz, root); }
};

```

7.2 KD tree

有一個 $N \times N$ 的棋盤，每個格子內有一個整數，初始時的時候全部為 0，現在需要維護兩種操作：

- 1 x y A $1 \leq x, y \leq N$ ，A 是正整數。將格子 x, y 裡的數字加上 A
- 2 x1 y1 x2 y2 $1 \leq x1 \leq x2 \leq N, 1 \leq y1 \leq y2 \leq N$ 。輸出 $x1, y1, x2, y2$ 這個矩形內的數字和

- 3 無 終止程式 <https://oi-wiki.org/ds/kdt/>

```

const int maxn = 200010;
int n, op, x1, xr, y1, yr, lstans;
struct node{
    int x, y, v;
} s[maxn];
bool cmp1(int a, int b) { return s[a].x < s[b].x; }
bool cmp2(int a, int b) { return s[a].y < s[b].y; }
double a = 0.725;
int rt, cur, d[maxn], lc[maxn], rc[maxn], L[maxn], R[maxn], D[maxn], U[maxn],
    siz[maxn], sum[maxn];
int g[maxn], t;
void print(int x){
    if (!x)
        return;
    print(lc[x]);
    g[++t] = x;
    print(rc[x]);
}
void maintain(int x){

```

```

    siz[x] = siz[lc[x]] + siz[rc[x]] + 1;
    sum[x] = sum[lc[x]] + sum[rc[x]] + s[x].v;
    L[x] = R[x] = s[x].x;
    D[x] = U[x] = s[x].y;
    if (lc[x])
        L[x] = min(L[x], L[lc[x]]), R[x] = max(R[x], R[lc[x]]),
        D[x] = min(D[x], D[lc[x]]), U[x] = max(U[x], U[lc[x]]);
    if (rc[x])
        L[x] = min(L[x], L[rc[x]]), R[x] = max(R[x], R[rc[x]]),
        D[x] = min(D[x], D[rc[x]]), U[x] = max(U[x], U[rc[x]]);
}
int build(int l, int r){
    if (l > r) return 0;
    int mid = (l + r) >> 1;
    double av1 = 0, av2 = 0, va1 = 0, va2 = 0;
    for (int i = l; i <= r; i++)
        av1 += s[g[i]].x, av2 += s[g[i]].y;
    av1 /= (r - l + 1);
    av2 /= (r - l + 1);
    for (int i = l; i <= r; i++)
        va1 += (av1 - s[g[i]].x) * (av1 - s[g[i]].x),
        va2 += (av2 - s[g[i]].y) * (av2 - s[g[i]].y);
    if (va1 > va2)
        nth_element(g+l, g+mid, g+r+1, cmp1), d[g[mid]] = 1;
    else
        nth_element(g+l, g+mid, g+r+1, cmp2), d[g[mid]] = 2;
    lc[g[mid]] = build(l, mid - 1);
    rc[g[mid]] = build(mid + 1, r);
    maintain(g[mid]);
    return g[mid];
}
void rebuild(int &x){
    t = 0;
    print(x);
    x = build(1, t);
}
bool bad(int x) { return a * siz[x] <= (double)max(siz[lc[x]], siz[rc[x]]); }
void insert(int &x, int v){
    if (!x){
        x = v;
        maintain(x);
        return;
    }
    if (d[x] == 1){
        if (s[v].x <= s[x].x) insert(lc[x], v);
        else insert(rc[x], v);
    }
    else{
        if (s[v].y <= s[x].y) insert(lc[x], v);
        else insert(rc[x], v);
    }
    maintain(x);
    if (bad(x))
        rebuild(x);
}
int query(int x){
    if (!x || xr < L[x] || x1 > R[x] || yr < D[x] || y1 > U[x])
        return 0;
    if (x1 <= L[x] && R[x] <= xr && y1 <= D[x] && U[x] <= yr)
        return sum[x];
    int ret = 0;
    if (x1 <= s[x].x && s[x].x <= xr && y1 <= s[x].y && s[x].y <= yr)
        ret += s[x].v;
    return query(lc[x]) + query(rc[x]) + ret;
}
int main(){
    scanf("%d", &n);
    while (~scanf("%d", &op)){
        if (op == 1){
            cur++;
            scanf("%d%d%d", &s[cur].x, &s[cur].y, &s[cur].v);
            insert(rt, cur);
        }
        if (op == 2) {
            scanf("%d%d%d%d", &x1, &y1, &xr, &yr);
            printf("%d\n", lstans = query(rt));
        }
    }
}

```



```

    if (op == 3) return 0;
}
}

```

7.3 Leftist Heap

```

typedef int type;
struct Node{
    type key;
    int dist; int lc, rc;
};
vector<Node> vv;
struct Leftist_Heap{
    int root;
    Leftist_Heap() { root = -1; }
    type top(){
        assert(root >= 0);
        return vv[root].key;
    }
    int merge(int a, int b){
        if (a==-1) return b; if (b==-1) return a;
        if (vv[b].key < vv[a].key) //小根堆是<, 否則>
            swap(a, b);
        vv[a].rc = merge(vv[a].rc, b);
        if (vv[a].lc == -1 || ((vv[a].rc == -
1) && (vv[vv[a].rc].dist > vv[vv[a].lc].dist)))
            swap(vv[a].lc, vv[a].rc);
        if (vv[a].rc == -1) vv[a].dist = 0;
        else vv[a].dist = vv[vv[a].rc].dist + 1;
        return a;
    }
    void push(type ins){
        Node x;
        x.dist = 0, x.key = ins, x.lc = x.rc = -1;
        vv.push_back(x);
        root = merge(root, vv.size() - 1);
    }
    void pop(){
        assert(root != -1);
        root = merge(vv[root].lc, vv[root].rc);
    }
};

```

7.4 treap

```

struct Treap{
    int sz, val, pri, tag;
    Treap *l, *r;
    Treap(int _val){
        val = _val;
        sz = 1;
        pri = rand();
        l = r = NULL;
        tag = 0;
    }
};
void push(Treap *a){
    if (a->tag){
        Treap *swp = a->l;
        a->l = a->r;
        a->r = swp;
        int swp2;
        if (a->l)
            a->l->tag ^= 1;
        if (a->r)
            a->r->tag ^= 1;
        a->tag = 0;
    }
}
int Size(Treap *a) { return a ? a->sz : 0; }
void pull(Treap *a){
    a->sz = Size(a->l) + Size(a->r) + 1;
}
Treap *merge(Treap *a, Treap *b){
    //a的val全小於b的val
    if (!a || !b)
        return a ? a : b;
    if (a->pri > b->pri){
        push(a);

```

```

        a->r = merge(a->r, b);
        pull(a);
        return a;
    }
    else{
        push(b);
        b->l = merge(a, b->l);
        pull(b);
        return b;
    }
}
void split(Treap *t, int k, Treap *&a, Treap *&b){
    if (!t){
        a = b = NULL;
        return;
    }
    push(t);
    if (Size(t->l) + 1 <= k){
        a = t;
        split(t->r, k - Size(t->l) - 1, a->r, b);
        pull(a);
    }
    else{
        b = t;
        split(t->l, k, a, b->l);
        pull(b);
    }
}

```

8 Others

1. Staircase Nim: 第 $1 \sim n$ 個階梯上面各有一些石頭，兩個人輪流進行操作。每次操作可以從某個階梯移動一些石頭到它前面一個階梯上(特別的，第 1 個階梯移到第 0 個)，最後石頭全部移動到第 0 個階梯。這個問題只要對第奇數個階梯做 Nim 即可。
 2. priority_queue<Node, vector<Node>, cmp> pq;
- ```

struct cmp{
 bool operator()(Node a, Node b){
 if (a.x == b.x) return a.y > b.y;
 return a.x > b.x;
 }
};
3. return day of week on y year m month d day
int zeller(int y, int m, int d) {
 if (m<=2) y--, m+=12; int c=y/100; y%=100;
 int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
 if (w<0) w+=7; return(w);
}

```