ACM-ICPC World Final 2009
Team Standard Source Code Library

The Answer to Life, the Universe, and Everything

Fudan University

Yonghui Wu (Coach)
Guodong Feng
Ji Hong
Lei Huang

求解无向图双联通分量

```
void init(){
    memset(ace,-1,sizeof(ace));
    memset(pre,-1,sizeof(pre));
    cnt = top = BCCnum = 0;
}
void dfs(int f,int s){
    int i,b = 0,v;
    ace[s] = pre[s] = ++cnt;
    for (i=begin[s];i<=m && edge[i].first == s;i++){
        v = edge[i].second;
        if (v == f) continue;
        if (pre[v] == -1){
            b++;stk[++top] = make_pair(s,v);
            dfs(s,v);
            if (ace[s] > ace[v]) ace[s] = ace[v];
            else if (ace[v] >= pre[s]){
                while (1){
                    cout<<stk[top].first << " " << stk[top].second <<
endl;
                    if (stk[top].first == s && stk[top].second == v)
{top--;break;}
                    top--;
                }
                cout << endl << endl;BCCnum++;
            }
        }
        else{
            if (pre[s] > pre[v]) stk[++top] = make_pair(s,v);
            if (v != f && ace[s] > pre[v]) ace[s] = pre[v];
        }
    }
    if (s == 1 && top){
        while (top){
            cout << stk[top].first << " " << stk[top].second << endl;
            top--;
        }
        BCCnum++;
    }
}
```

求解2-SAT方案核心代码

```
bool Two_SAT(){
    work_SCC();
    int i,j;
    for (i=1;i<=totalnode;i++) if (block[i] == block[i + totalnode])
return false;
    memset(f,false,sizeof(f));
    for (i=1;i<=totalnode * 2;i++)
        for (j=0;j<g[i].size();j++) if (block[i] != block[g[i][j]])
f[block[i]][block[g[i][j]]] = true;
    for (i=1;i<=totalblock;i++) new_graph[i].clear();
    memset(inner,0,sizeof(inner));
    for (i=1;i<=totalblock;i++){
        crash[i].clear();
        for (j=1;j<=totalblock;j++) if (f[i][j]){
            new_graph[i].push_back(j);
            inner[j]++;
        }
    }
    for (i=1;i<=totalnode;i++){
        crash[block[i]].push_back(block[i + totalnode]);
        crash[block[i + totalnode]].push_back(block[i]);
    }
    while (!Q.empty()) Q.pop();
    memset(used,false,sizeof(used));
    int top = 0,now;
    for (i=1;i<=totalblock;i++) if (inner[i] == 0){
        rank[++top] = i;
        Q.push(i);
        used[i] = true;
    }
    while (!Q.empty()){
        now = Q.front();  Q.pop();
        for (i=0;i<new_graph[now].size();i++) if
(--inner[new_graph[now][i]] == 0){
            rank[++top] = new_graph[now][i];
            Q.push(new_graph[now][i]);
            used[new_graph[now][i]] = true;
        }
    }
```

```
    memset(choose,false,sizeof(choose));
    for (i=top;i>=1;i--)
        if (check(rank[i]))
            choose[rank[i]] = true;
    for (i=1;i<=totalnode;i++) value[i] = choose[block[i]];
    ans[1][1] = P;
    for (i=2;i<=n;i++) if (value[i - 1]) ans[i][1] = 0; else ans[i][1]
= 1;
    for (i=2;i<=m;i++) if (value[i - 1 + n - 1]) ans[1][i] = 0; else
ans[1][i] = 1;
    return true;
}
```

以下代码用以求解最小树形图

```
int n,g[maxn][maxn],used[maxn],pass[maxn],eg[maxn],more,queue[maxn];
void combine(int id, int &sum){
    int tot=0,from,i,j,k;
    for(;id!=0&&!pass[id];id=eg[id]) {queue[tot++]=id; pass[id]=1;}
    for(from=0; from<tot&&queue[from]!=id; from++);
    if(from==tot)return; more=1;
    for(i=from;i<tot;i++){
        sum+=g[eg[queue[i]]][queue[i]];
        if(i!=from){ used[queue[i]]=1;
            for(j=0;j<=n;j++) if(!used[j])
                if(g[queue[i]][j]<g[id][j])
g[id][j]=g[queue[i]][j];
        }
    }
    for(i=1;i<=n;i++) if(!used[i]&&i!=id) {
        for(j=from;j<tot;j++) { k=queue[j];
        if(g[i][id]>g[i][k]-g[eg[k]][k])
g[i][id]=g[i][k]-g[eg[k]][k];
        }
    }
}
int msdt(int root) {
    int i,j,k,sum=0;
    memset(used,0,sizeof(used));
    for(more=1;more;){ more=0;
        memset(eg,0,sizeof(eg));
        for(i=1;i<=n;i++) if(!used[i]&&i!=root) {
            for(j=1,k=0;j<=n;j++) if(!used[j]&&i!=j)
                if(k==0||g[j][i]<g[k][i]) k=j;
            eg[i]=k;
        } memset(pass,0,sizeof(pass));
        for(i=1;i<=n;i++) if(!used[i]&&!pass[i]&&i!=root)
combine(i,sum);
    }
    for(i=1;i<=n;i++) if(!used[i]&&i!=root) sum+=g[eg[i]][i];
    return sum;
}
```

以下代码实现KM算法

```
bool Find(int s){
    _x[s] = true;
    int i,t;
    for (i=0;i<n;i++) if (!_y[i] && x[s] + y[i] == g[s][i]){
        t = lt[i]; _y[i] = true; lt[i] = s;
        if (t == -1 || Find(t)) return true;
        lt[i] = t;
    }return false;
}
int km(){
    memset(lt,-1,sizeof(lt));
    memset(y,0,sizeof(y));
    for (i=0;i<n;i++)for(j=0;j<n;j++)if (x[i]<g[i][j])x[i]=、g[i][j];
    for (k=0;k<n;k++){
        while (true){
            memset(_x,false,sizeof(_x));
            memset(_y,false,sizeof(_y));
            if (Find(k)) break;
            d = MAX;
            for (i=0;i<n;i++)if(_x[i])for(j=0;j<n;j++)if(!_y[j] &&
x[i]+y[j] != g[i][j] && d > x[i]+y[j]-g[i][j]) d = x[i]+y[j]-g[i][j];
            if (d == MAX) break;
            for (i=0;i<n;i++) if (_x[i]) x[i] -= d;
            for (i=0;i<n;i++) if (_y[i]) y[i] += d;
        }
    }
    int ans = 0;for (i=0;i<n;i++) ans += g[lt[i]][i];
    return ans;
}
```

以下代码用以求解任意图匹配

```
bool g[ran][ran],inq[ran],inpath[ran],inblo[ran];
int n,Match[ran],start,finish,newbase,father[ran],base[ran];
queue<int> q;
int lca(int u,int v){
    memset(inpath,0,sizeof(inpath));
    while(1){
        u=base[u];
        inpath[u]=1;
        if(u==start)break;
        u=father[Match[u]];
    }
    while(1){
        v=base[v];
        if(inpath[v])break;
        v=father[Match[v]];
    } return v;
}
void reset_trace(int u){
    while(base[u]!=newbase){
        int v=Match[u];
        inblo[base[u]]=inblo[base[v]]=1;
        u=father[v];
        if(base[u]!=newbase)father[u]=v;
    }
}
void blossom_contract(int u,int v){
    newbase=lca(u,v);
    memset(inblo,0,sizeof(inblo));
    reset_trace(u);reset_trace(v);
    if(base[u]!=newbase)father[u]=v;
    if(base[v]!=newbase)father[v]=u;
    for(int i=1; i<=n; i++)if(inblo[base[i]]){
        base[i]=newbase;
        if(!inq[i])q.push(i);
    }
}
void find_augmenting_path(){
    memset(inq,0,sizeof(inq));memset(father,0,sizeof(father));
    for(int i=1; i<=n; i++)base[i]=i;
    while(!q.empty())q.pop();q.push(start);
    finish=0;
    while(!q.empty()){
        int x=q.front();q.pop();
        for(int i=1; i<=n; i++)
            if(g[x][i] && base[i]!=base[x] && Match[x]!=i)
                if(i==start || Match[i]>0 && father[Match[i]]>0)
blossom_contract(x,i);
                else if(father[i]==0){
                    father[i]=x;
                    if(Match[i]>0)q.push(Match[i]);else{
                        finish=i;
                        return;
                    }
                }
    }
}
void augment(){
    int u=finish;
    while(u>0){
        int v=father[u],w=Match[v];
        Match[v]=u;Match[u]=v;
        u=w;
    }
}
void edmonds(){
    memset(Match,0,sizeof(Match));
    for(int k=1; k<=n; k++)if(Match[k]==0){
            start=k;
            find_augmenting_path();
            if(finish>0)augment();
    }
}
```

无项图任意点对最小割（GOMORY-HUtree）

```
#define SLOW_SOLUTION 0
#define Debug 0
#define LET(x,a)   __typeof(a) x(a)
#define IFOR(i,a,b)     for(LET(i,a);i!=(b);++i)
#define EACH(it,v)      IFOR(it,v.begin(),v.end())
#define FOR(i,a,b) for(int i = (a); i < (b); ++i)
```

```
#define REP(i,n)  FOR(i,0,n)
const int INF = 2000000000;
const int MAXN = 150;
int graph[3 * MAXN][3 * MAXN];
class FlowDinic {
    public:
        struct edge {
            int to, cap, back;
        };
        vector<vector<edge> > adj;
        int n;
        FlowDinic(int _n) :n(_n) {
            adj.resize(n);
            REP (i,n) adj[i].clear();
        }
        void Insert(int i, int j, int c) {
            adj[i].push_back((edge) { j, c, adj[j].size() } );
            adj[j].push_back((edge) { i, c, adj[i].size() - 1 } );
        }
        int Dinic(int s, int t) {
            int q[n], prev[n], allflow = 0, qf, qb;
            while (true) {
                memset(prev, -1, sizeof(prev));
                qf = 0; qb = 0;
                prev[q[qb++] = s] = -2;
                while (qb > qf && prev[t] == -1) for (int u = q[qf++],
i = 0, v; i < adj[u].size(); i++)
                    if (prev[v = adj[u][i].to] == -1 && adj[u][i].cap
> 0) prev[q[qb++] = v] = adj[u][i].back;
                if (prev[t] == -1) break;
                for (int i = 0, z; i < adj[t].size(); i++)
                    if (adj[z = adj[t][i].to][adj[t][i].back].cap >
0 && prev[z] != -1) {
                        int flow = adj[z][adj[t][i].back].cap;
                        for (int v = z, u = prev[v]; u >= 0; v =
adj[v][u].to, u = prev[v])
                            flow = min(flow,
adj[adj[v][u].to][adj[v][u].back].cap);
                        if (!flow) continue;
                        adj[z][adj[t][i].back].cap -= flow;
                        adj[t][i].cap += flow;
                        for (int v = z, u = prev[v]; u >= 0; v =
adj[v][u].to, u = prev[v]) {

    adj[adj[v][u].to][adj[v][u].back].cap -= flow;
                            adj[v][u].cap += flow;
                        }
                        allflow += flow;
                    }
                }
                return allflow;
            }
};
void setUpGraph(FlowDinic &G, int n) {
    REP (i,n) FOR (j,i+1,n) if (graph[i][j]) G.Insert(i, j,
graph[i][j]);
}
vector<pair<int,int> > gomoryHU[MAXN];
int minCutCost[MAXN][MAXN], visited[3 * MAXN], cost[3 * MAXN], id[MAXN],
dist[MAXN];
int main() {
    int N; scanf("%d", &N); while (N--) {
        int n, m, query;
        scanf("%d%d", &n, &m);
        memset(graph, 0, sizeof(graph));
        REP (i,m) {
            int x, y, z;
            scanf ("%d%d%d", &x, &y, &z);
            --x; --y;
            graph[x][y] += z;
            graph[y][x] += z;
        }
            /* (n-1) maxflow calls */
            fill(id, id + n, 0);
            REP (i,n) gomoryHU[i].clear();
            REP (i,n) REP (j,n) minCutCost[i][j] = INF;
            for (int step = 0, source, destination, givenId; step < n
- 1; ++step) {
                source = destination = givenId = -1;
                for (int i = 0; i < n && source == -1; ++i)
```

```
                for (int j = i + 1; j < n && destination == -1;
++j)
                    if (id[i] == id[j])
                    source=i,destination=j,givenId=id[i];
            FlowDinic G(n + 2 * step); setUpGraph(G, n + 2 * step);
            int cut = G.Dinic(source, destination);
            gomoryHU[source].push_back(make_pair(destination,
cut));
            gomoryHU[destination].push_back(make_pair(source,
cut));
            fill(visited, visited + n + 2 * step, 0);
            fill(cost, cost + n + 2 * step, 0);
            queue<int> q; q.push(source); visited[source] = 1;
            while (!q.empty()) {
                int cur = q.front(); q.pop();
                EACH (it, G.adj[cur]) {
                    if (it->cap > 0) {
                        if (!visited[it->to]) {
                            visited[it->to] = true;
                            q.push(it->to);
                        }
                    }
                }
            }
            REP (i, n + 2 * step) EACH(it, G.adj[i]) {
                if(visited[i] && !visited[it->to]) {
                    cost[it->to] += graph[i][it->to];
                    cost[i] += graph[i][it->to];
                    graph[i][it->to] = graph[it->to][i] = 0;
                }
            }
            int id1 = source, id2 = destination;
            REP (i,n) if (id[i] == givenId) {
                if (visited[i]) id1 = min(id1, i);
                else id2 = min(id2, i);
            }
            REP (i,n) if (id[i] == givenId) {
                if (visited[i]) id[i] = id1;
                else id[i] = id2;
            }
                REP (i, n + 2 * step) {
                    graph[i][n + 2 * step + 1 - visited[i]] = graph[n
+ 2 * step + 1 - visited[i]][i] = cost[i];
                    graph[i][n + 2 * step + 1 - visited[i]] = graph[n
+ 2 * step + 1 - visited[i]][i] = cost[i];
                }
            }
            REP (i,n) {
                fill(visited, visited + n, 0);
                queue<int> q; q.push(i); visited[i] = 1;
minCutCost[i][i] = INF;
                while (!q.empty()) {
                    int cur = q.front(); q.pop();
                    EACH (it, gomoryHU[cur]) if(!visited[it->first])
{
                        visited[it->first] = true;
                        minCutCost[i][it->first] =
minCutCost[it->first][i] = min(minCutCost[i][cur], it->second);
                        q.push(it->first);
                    }
                }
            }
        scanf("%d", &query);
        for (int i = 0; i < query; ++i) {
            int t, res = 0; scanf("%d", &t);
            REP (j,n) FOR(k, j + 1, n) res += (minCutCost[j][k] <= t);
            printf ("%d\n", res);
        }
        if (N) putchar(10);
    }
}
```

分治法求解树上的限制问题（本例中为控制权A最大化权B）

```
int n,k,ans,K,KK,L;
int start[ran],next[ran*2],vert[ran*2],len[ran*2],val[ran*2];
int num[ran],h[ran],dd[2][ran],bel[ran];
pair<pair<int,int>,int> dist[ran];
bool unavail[ran];
void dfs1(int x,int fat){
    num[x]=0;h[x]=0;
    for(int i=start[x],w; i!=0; i=next[i]){
```

```
          w=vert[i];
          if(fat!=w && !unavail[w]){
               dfs1(w,x);
               num[x]+=num[w]+1;
               h[x]=max(h[x],num[w]+1);
          }
     }
}
void dfs2(int x,int fat,int&ch,int tot){
     h[x]=max(h[x],tot);
     if(h[x]<h[ch])ch=x;
     if(h[ch]<=tot)return;
     for(int i=start[x],w; i!=0; i=next[i]){
          w=vert[i];
          if(fat!=w && !unavail[w])
               dfs2(w,x,ch,tot+num[x]-num[w]);
     }
}
void choose(int p,int&w){
     dfs1(p,p);w=p;dfs2(p,p,w,0);
}
void dfs(int x,int fat,int sgn,int d){
     if(h[x]>k)return;
     if(x!=fat)dist[K++]=make_pair(make_pair(h[x],d),sgn);
     ans=max(ans,d);L=max(L,d);
     for(int i=start[x],w; i!=0; i=next[i]){
          w=vert[i];
          if(fat!=w && !unavail[w]){
               h[w]=h[x]+val[i];
               dfs(w,x,x==fat?w:sgn,d+len[i]);
          }
     }
}
void update(int x,int y,int z){
     if(y>dd[0][x]){
          if(z!=bel[x])
               dd[1][x]=dd[0][x];
          dd[0][x]=y,bel[x]=z;
     }
     else if(y>dd[1][x] && z!=bel[x]) dd[1][x]=y;
}
void solve(int p){
     //in this section, num[] -> the number of nodes (without itself) in
the subtree,
     //h[] -> the maximum subtree size of the node
     int w;choose(p,w);
     //in this section, num[] -> the maximum distance of each subtree
     //h[] -> the crowd value of each node
     h[w]=0;K=0;dfs(w,w,w,0);
     if(L*2<=ans)return;if(K==0)return;
     sort(dist,dist+K);
     //h[] -> the crowd value of each node
     //dd[] -> the distance array
     memset(dd,0,sizeof(int)*(K+5));
     memset(bel,-1,sizeof(int)*(K+5));
     KK=0;
     for(int i=0; i<K; i++){
          if(dist[i].first.first!=h[KK]){
               h[++KK]=dist[i].first.first;
               dd[0][KK]=dd[0][KK-1];
               dd[1][KK]=dd[1][KK-1];
               bel[KK]=bel[KK-1];
          }
          update(KK,dist[i].first.second,dist[i].second);
     }
     for(int i=1,j=KK; i<=KK && h[i]+h[i]<=k; i++){
          while(h[i]+h[j]>k)j--;
          if(bel[i]!=bel[j])
               ans=max(ans,dd[0][i]+dd[0][j]);
          else
               ans=max(ans,max(dd[0][i]+dd[1][j],dd[1][i]+dd[0][j]));
     }
     unavail[w]=true;
     for(int i=start[w]; i!=0; i=next[i])
          if(!unavail[vert[i]])
               solve(vert[i]);
}
int main(){
     int x,y,z,u,_;
     for(scanf("%d",&_); _--;){
```

```
        scanf("%d%d",&n,&k);
        memset(start,0,sizeof(start));
        memset(unavail,0,sizeof(unavail));
        for(int i=1,j=1; i<n; i++){
            scanf("%d%d%d%d",&x,&y,&z,&u);
            next[j]=start[x];vert[j]=y;val[j]=z;len[j]=u;
            start[x]=j++;
            next[j]=start[y];vert[j]=x;val[j]=z;len[j]=u;
            start[y]=j++;
        }
        ans=0;solve(1);
        printf("%d\n",ans);
    }
}
```

标号法的常用最大流

```
int g[maxn],to[maxm],np[maxm],cp,m,n,u,v,now,source,sink;
long long int sum,cap[maxm],a[maxn];
int d[maxn],p[maxn],cur[maxn],cnt[maxn];
char cmd[15000],*temp;
#define add_edge(x,y,z)
cap[cp]=z;to[cp]=y;np[cp]=g[x];g[x]=cp++;cap[cp]=0;to[cp]=x;np[cp]=g[
y];g[y]=cp++;
#define add_tedge(x,y,z1,z2)
cap[cp]=z1;to[cp]=y;np[cp]=g[x];g[x]=cp++;cap[cp]=z2;to[cp]=x;np[cp]=
g[y];g[y]=cp++;
int main(){
    int r1,r2,r3;
    cp=2; memset(g,0,sizeof(g));
    scanf("%d%d%*c",&n,&m); source=1; sink=n;
    for(int i=1;i<=m;i++){

    gets(cmd);temp=cmd;r1=atoi(temp);while(isdigit(*temp))temp++;tem
p++;
    r2=atoi(temp);while(isdigit(*temp))temp++;temp++;r3=atoi(temp);w
hile(isdigit(*temp))temp++;temp++;
        add_tedge(r1,r2,r3,r3);
    }
    for(u=1;u<=n;u++)cur[u]=g[u];
    a[u=source]=inf;
    memset(d,0,sizeof(int)*(n+1)); memset(cnt,0,sizeof(int)*(n+1));
    cnt[0]=n;
    while(d[source]<n){
        for(now=cur[u];now;now=np[now])
            if(cap[now]&&d[v=to[now]]+1==d[u])break; cur[u]=now;
        if(now){
            p[v]=now; a[v]=cap[now]; if(a[v]>a[u])a[v]=a[u];
            if((u=v)==sink){
                do{cap[p[u]]-=a[sink]; cap[p[u]^1]+=a[sink];
u=to[p[u]^1];}while(u!=source);
                sum+=a[sink]; a[source]=inf;
            }
        }
        else{
            if(--cnt[d[u]]==0)break; d[u]=n; cur[u]=g[u];
            for(now=g[u];now;now=np[now]) if(cap[now] &&
d[u]>d[to[now]]+1) d[u]=d[to[now]]+1;
            cnt[d[u]]++;
            if(u!=source)u=to[p[u]^1];
        }
    }
    printf("%lld\n",sum);
    return 0;
}
```

度限制最小生成树

```
int n,m,d,sz,ans;
vector<pair<int,int> > e,v[ran];
bool f[ran]; int ace[ran];
pair<int,pair<int,int> > a[100000];
int ancestor(int x){if(ace[x]!=x)ace[x]=ancestor(ace[x]);return
ace[x];}
void del(int x,int y){
    int l=v[x].size();
    for(int i=0; i<l; i++) if(v[x][i].first==y){
        swap(v[x][i],v[x][l-1]);
        v[x].pop_back();return;
    }
}
int cost[ran];pair<int,int> mcost[ran];
void dfs(int x,int fat){
    for(vector<pair<int,int> >::iterator i=v[x].begin(); i!=v[x].end();
```

```
i++){
        if(i->first==fat)continue;
        if(x==1)cost[i->first]=0;
        else{
            if(i->second > cost[x])

cost[i->first]=i->second,mcost[i->first]=make_pair(x,i->first);
            else cost[i->first]=cost[x],mcost[i->first]=mcost[x];
        }
        dfs(i->first,x);
    }
}
int main(){
    int _,u,x,y,z;//temporary variables
    scanf("%d",&_);
    while(_--){
        scanf("%d%d%d",&n,&u,&d);
        for(int i=1; i<=n; i++){
            e.clear();v[i].clear();ace[i]=i;
        }
         m=0;
        while(u--){
            scanf("%d%d%d",&x,&y,&z);
            if(x>y)swap(x,y);if(x==y)continue;
            if(x==1)e.push_back(make_pair(z,y));else
a[m++]=make_pair(z,make_pair(x,y));
        }
        sz=e.size();memset(f,0,sizeof(f));
        //execute the Kruskal's algorithm
        sort(a,a+m);sort(e.begin(),e.end());
        ans=0;u=n;
        for(int i=0; i<m; i++){
            x=a[i].second.first;y=a[i].second.second;
            if(ancestor(x)!=ancestor(y)){
                ans+=a[i].first;ace[ace[x]]=ace[y];
                v[x].push_back(make_pair(y,a[i].first));
                v[y].push_back(make_pair(x,a[i].first));
                u--;
            }
        }
        for(int i=0; i<sz; i++){
            z=e[i].first,y=e[i].second;
            if(ancestor(1)!=ancestor(y)){
                ans+=z;ace[ace[1]]=ace[y];
                v[1].push_back(make_pair(y,z));
                v[y].push_back(make_pair(1,z));
                f[i]=true;u--;d--;
            }
        }
        //extend degree of node 1
        if(u!=1 || d<0){puts("NONE");continue;}
        while(d--){
            dfs(1,1);
            int dec=0,id;pair<int,int> exch;
            for(int i=0; i<sz; i++){
                if(f[i])continue;
                if(cost[e[i].second]-e[i].first>dec){
                    dec=cost[e[i].second]-e[i].first;id=i;
                    exch=mcost[e[i].second];
                }
            }
            if(dec==0)break;ans-=dec; f[id]=true;
            v[e[id].second].push_back(make_pair(1,e[id].first));
            v[1].push_back(make_pair(e[id].second,e[id].first));
            del(exch.first,exch.second);
            del(exch.second,exch.first);
        }printf("%d\n",ans);
    }
}
```

最优二分搜索树

```
const int size = 120000;
int w[size],d[size],q[size],t,m,ans,i,j,k,n;
char buf[size*10],*buff;
void combine(int k){
     int j,d,x;
     m++; w[m]=x=q[k-1]+q[k]; ans+=x; t--;
     for(j=k;j<=t;j++)q[j]=q[j+1];
     for(j=k-2;q[j]<x;j--)q[j+1]=q[j];
     q[j+1]=x;
     while(j>0&&q[j-1]<=x){d=t-j; combine(j); j=t-d;}
```

```
}
int main(){
    for(;;){
        scanf("%d",&n);
        if(!n)break; --n;
        gets(buf); gets(buff=buf);
        for(j=0;j<=n;j++){
            w[j]=0;
            while(!isdigit(*buff))buff++;
            while(isdigit(*buff)){
                w[j]=w[j]*10+*buff-'0';
                buff++;
            }
        }
        if(!n){printf("0\n");continue;}
        m=n; t=1; ans=0; q[0]=1000000000; q[1]=w[0];
        for(k=1;k<=n;k++){
            while(q[t-1]<=w[k])combine(t);
            t++; q[t]=w[k];
        }
        while(t>1)combine(t);
        printf("%d\n",ans);
    }
    return 0;
}
```

以下代码用以实现伸展树基本操作

```
struct node{
    int v,size;bool reverse;
    node *lc,*rc,*father;
};
void Resize(node *&p) {/*添加维护信息*/}
void Push(node *p) {/*添加缓冲操作信息*/}
void rl(node *&p) {node *q = p->lc;p->lc = q->rc;q->rc = p;Resize(p);p
= q;}
void rr(node *&p) {node *q = p->rc;p->rc = q->lc;q->lc = p;Resize(p);p
= q;}
inline int kind(node *p) {if (p == p->father->lc) return -1; else return
1;}
inline int Size(node *p) {return (p == NULL ? 0 : p->size);}
void Zg(node *&p) {
    node *temp = p->father->father;
    int t;
    if (p->father->father != NULL) {if (p->father ==
p->father->father->lc) t = -1; else t = 1;}
    if (p == p->father->lc) rl(p->father); else rr(p->father);
    p->father = temp;
    if (p->father != NULL) if (t == -1) p->father->lc = p;else
p->father->rc = p;
}

void Splay(node *&p) {
    while (p->father != NULL){
        if (p->father->father == NULL) Zg(p);else{
            if (kind(p->father) == kind(p)) {
                Zg(p->father);Zg(p);
            } else {Zg(p); Zg(p);}
        }
    } Resize(p);
}
```

左偏树核心代码（伪码）

```
Function Merge(A, B)
    If A = NULL Then return B
    If B = NULL Then return A
    If key(B) < key(A) Then swap(A, B)
    right(A) ←Merge(right(A), B)
    If dist(right(A)) > dist(left(A)) Then
        swap(left(A), right(A))
    If right(A) = NULL Then dist(A) ←0
    Else dist(A) ←dist(right(A)) + 1
    return A
End Function
```

后缀数组及高度数组

```
char data[M],_data[M];
int SA[M],rank[M],h[M],height[M];
int n,value;
#define Rank(a)((a)>n?0:rank[a])
bool cmp(int a,int b){return data[a] < data[b];}
int c[M],a[M],_SA[M],_rank[M];
void Double(){
```

```
    memset(c,0,sizeof(int) * (n + 1));int i;
    for (i=1;i<=n;i++) c[Rank(i + value)]++;
    a[0] = 1;for (i=1;i<=n;i++) a[i] = a[i - 1] + c[i - 1];
    for (i=1;i<=n;i++) _SA[a[Rank(i + value)]++] = i;
    memset(c,0,sizeof(int) * (n + 1));
    for (i=1;i<=n;i++) c[Rank(i)]++;
    a[0] = 1;for (i=1;i<=n;i++) a[i] = a[i - 1] + c[i - 1];
    for (i=1;i<=n;i++) SA[a[Rank(_SA[i])]++] = _SA[i];
    _rank[SA[1]] = 1;
    for (i=2;i<=n;i++) if (Rank(SA[i]) == Rank(SA[i - 1]) && Rank(SA[i]
+ value) == Rank(SA[i - 1] + value)) _rank[SA[i]] = _rank[SA[i - 1]]; else
_rank[SA[i]] = _rank[SA[i - 1]] + 1;
    memcpy(rank,_rank,sizeof(int) * (n + 1));
    value <<= 1;
}
void make_SA(){
    int i;
    for (i=1;i<=n;i++) SA[i] = i;
    sort(SA + 1,SA + n + 1,cmp);rank[SA[1]] = 1;
    for (i=2;i<=n;i++) if (data[SA[i]] == data[SA[i - 1]]) rank[SA[i]]
= rank[SA[i - 1]]; else rank[SA[i]] = rank[SA[i - 1]] + 1;
    value = 1; while (value < n) Double();
}
void make_Height(){
    int i;
    memset(h,0,sizeof(int) * (n + 1));
    for (i=1;i<=n;i++) if (rank[i] == 1) h[i] = 0;
    else if (i == 1 || h[i - 1] <= 1) while (data[i + h[i]] ==
data[SA[rank[i] - 1] + h[i]]) h[i]++;
    else{
        h[i] = h[i - 1] - 1;
        while (data[i + h[i]] == data[SA[rank[i] - 1] + h[i]]) h[i]++;
    }
    for (i=1;i<=n;i++) height[rank[i]] = h[i];
}
```

以下代码用以构建后缀数组，使用skew算法

```
inline bool leq(int a1, int a2, int b1, int b2){return a1 < b1 || a1 ==
b1 && a2 <= b2;}
inline bool leq(int a1, int a2, int a3, int b1, int b2, int b3){return
a1 < b1 || a1 == b1 && leq(a2,a3, b2,b3);}
static void radixPass(int* a, int* b, int* r, int n, int K){
    for (int i = 0; i <= K; i++) c[i] = 0;
    for (int i = 0; i < n; i++) c[r[a[i]]]++;
    for (int i = 0, sum = 0; i <= K; i++){
        int t = c[i];c[i] = sum;sum += t;
    }
    for (int i = 0; i < n; i++)b[c[r[a[i]]]++] = a[i];
}
void suffixArray(int* T, int* SA, int n, int K) {
    int n0=(n+2)/3, n1=(n+1)/3, n2=n/3, n02=n0+n2;
    int* R = new int[n02 + 3]; R[n02]= R[n02+1]= R[n02+2]=0;
    int* SA12 = new int[n02 + 3]; SA12[n02]=SA12[n02+1]=SA12[n02+2]=0;
    int* R0 = new int[n0];
    int* SA0 = new int[n0];
    for (int i=0, j=0; i < n+(n0-n1); i++) if (i%3 != 0) R[j++] = i;
    radixPass(R , SA12, T+2, n02, K);
    radixPass(SA12, R , T+1, n02, K);
    radixPass(R , SA12, T , n02, K);
    int name = 0, c0 = -1, c1 = -1, c2 = -1;
    for (int i = 0; i < n02; i++) {
        if (T[SA12[i]] != c0 || T[SA12[i]+1] != c1 || T[SA12[i]+2] !=
c2)
        { name++; c0 = T[SA12[i]]; c1 = T[SA12[i]+1]; c2 =
T[SA12[i]+2]; }
        if (SA12[i] % 3 == 1) { R[SA12[i]/3] = name; }
        else { R[SA12[i]/3 + n0] = name; }
    }
    if (name < n02) {
        suffixArray(R, SA12, n02, name);
        for (int i = 0; i < n02; i++) R[SA12[i]] = i + 1;
    } else for (int i = 0; i < n02; i++) SA12[R[i] - 1] = i;
    for (int i=0, j=0; i < n02; i++) if (SA12[i] < n0) R0[j++] = 3*SA12[i];
    radixPass(R0, SA0, T, n0, K);
    for (int p=0, t=n0-n1, k=0; k < n; k++) {
#define GetI() (SA12[t] < n0 ? SA12[t] * 3 + 1 : (SA12[t] - n0) * 3 + 2)
        int i = GetI(),j = SA0[p];
        if (SA12[t] < n0 ?
        leq(T[i], R[SA12[t] + n0], T[j], R[j/3]) :
        leq(T[i],T[i+1],R[SA12[t]-n0+1], T[j],T[j+1],R[j/3+n0])){
            SA[k] = i; t++;
```

```
            if (t == n02) for (k++; p < n0; p++, k++) SA[k] = SA0[p];
        }else {
            SA[k] = j; p++;
            if (p == n0) for (k++; t < n02; t++, k++) SA[k] = GetI();
        }
    }
    delete [] R; delete [] SA12; delete [] SA0; delete [] R0;
}
```

字符串模式匹配KMP算法

```
void makenext(){
    int i = 1,j = 0;
    while (i<=lms)
    if((j==0)||(ms[i]==ms[j])) next[++i]=++j;
    else j=next[j];
}
int kmp(int s){
    int p = s,j = 1;
    while ((p<=l)&&(j<=lms))
        if ((j==0)||(str[p]==ms[j])) {p++;j++;}  else j=next[j];
    if (j>lms) return p-lms;
    else return 0;
}
```

以下代码用以求三角形外心

```
void Circumcenter (CPoint p0 , CPoint p1 , CPoint p2 , CPoint &cp)
{
    double a1=p1.x-p0.x,b1 = p1.y - p0.y,c1 = (sqr(a1) + sqr(b1)) / 2;
    double a2=p2.x-p0.x,b2 = p2.y - p0.y,c2 = (sqr(a2) + sqr(b2)) / 2;
    double d = a1 * b2 - a2 * b1;
    cp.x = p0.x + (c1 * b2 - c2 * b1) / d;
    cp.y = p0.y + (a1 * c2  - a2 * c1) / d;
}
```

以下代码用以计算正交六面体表面两点最短距离问题

```
int r,L,H,W,x1,y1,z1,x2,y2,z2;
void turn(int i,int j,int x,int y,int z,int x0,int y0,int L,int W,int H){
    if(z==0){int R=x*x+y*y; if(R<r)r=R;}
    else{
        if(i>=0 && i<2) turn(i+1,j,x0+L+z,y,x0+L-x,x0+L,y0,H,W,L);
        if(j>=0 && j<2) turn(i,j+1,x,y0+W+z,y0+W-y,x0,y0+W,L,H,W);
        if(i<=0 && i>-2) turn(i-1,j,x0-z,y,x-x0,x0-H,y0,H,W,L);
        if(j<=0 && j>-2) turn(i,j-1,x,y0-z,y-y0,x0,y0-H,L,H,W);
    }
}
int main(){
    while(cin>>L>>W>>H>>x1>>y1>>z1>>x2>>y2>>z2){
        if(z1!=0 && z1!=H)
        if(y1==0 || y1==W){
            swap(y1,z1);swap(y2,z2);swap(W,H);
        }
        else{
            swap(x1,z1);swap(x2,z2);swap(L,H);
        }
        if(z1==H)z1=0, z2=H-z2;
        r=0x3fffffff; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
        cout<<r<<endl;
    }
}
```

以下代码求两圆交点

```
void calc(int a,int b)
{
    point A = c[a], B = c[b];
    double d = dist(A,B);
    if(r[a] + r[b] < d - eps) return;
    double u=ACOS((r[b] * r[b] - r[a] * r[a] - d * d) / -2 / r[a] / d);
    double v = atan2(B.y-A.y,B.x-A.x);
    point res = A;
    res.x += r[a] * cos(v+u); res.y += r[a] * sin(v+u);
    if(check(res,a,b)) list[cnt++] = res;
    res = A;
    res.x += r[a] * cos(v-u); res.y += r[a] * sin(v-u);
    if(check(res,a,b)) list[cnt++] = res;
}
```

圆面积并(spoj ORZ)

```
#define max(a, b) ((a) > (b) ? (a) : (b))
#define min(a, b) ((a) < (b) ? (a) : (b))
int const maxN = 256;
double const eps = 1e-6;
double const pi = 3.14159265358979323846264433;

struct circle{
    double ux, dx;
    int type, id;
};
```

```
int N1, N2, N, aSize;
double MaxX, MaxY;
double app[maxN * maxN];
double X[maxN], Y[maxN], R[maxN];
circle C[maxN * 2];
int equal(double a, double b){ return fabs(a - b) <= eps; }
int vaild(double y){ return y > eps && y < MaxY - eps; }
double sqr(double x){ return x * x; }
double dist(double dx, double dy){ return sqrt(sqr(dx) + sqr(dy)); }

int qCmp(void const *a, void const *b){
    circle *c1 = (circle *)a, *c2 = (circle *)b;
    if (equal(c1 -> ux + c1 -> dx, c2 -> ux + c2 -> dx)){
        if (c1 -> type != c2 -> type) return c1 -> type - c2 -> type;
        else if (R[c1 -> id] < R[c2 -> id]){
            if (c1 -> type == 0) return -1;
            else return 1;
        }  else {
            if (c1 -> type == 0) return 1;
            else return -1;
        }
    }  else if (c1 -> ux + c1 -> dx < c2 -> ux + c2 -> dx) return -1;
    else return 1;
}

int qCmp0(void const *a, void const *b){
    if (*(double *)a > *(double *)b) return 1;
    else return -1;
}

double upperGet(int id, double y1, double y2){
    double alpha = asin(min((Y[id] - y1) / R[id], 1.0)), beta =
asin(min((Y[id] - y2) / R[id], 1.0));
    double uDist = sqrt(sqr(R[id]) - sqr(y1 - Y[id]));
    if (sqr(R[id]) < sqr(y1 - Y[id])) uDist = 0.0;
    double dDist = sqrt(sqr(R[id]) - sqr(y2 - Y[id]));
    if (sqr(R[id]) < sqr(y2 - Y[id])) dDist = 0.0;
    double sTriAlpha = uDist * (Y[id] - y1) / 2.0;
    double s1 = alpha * sqr(R[id]) / 2.0 - sTriAlpha;
    double sTriBeta = dDist * (Y[id] - y2) / 2.0;
    double s2 = beta * sqr(R[id]) / 2.0 - sTriBeta;
    s2 += (dDist - uDist) * (Y[id] - y2);
    return s1 - s2;
}

double getS(circle &c, double y1, double y2){
    int id = c.id;
    if (Y[id] >= y2){
        return upperGet(id, y1, y2);
    }  else if (Y[id] <= y1){
        return upperGet(id, Y[id] - (y2 - Y[id]), Y[id] - (y1 - Y[id]));
    }  else {
        double ret = getS(c, y1, Y[id]) + getS(c, Y[id], y2);
        if (c.type == 0){
            if (c.ux < c.dx) ret += (c.dx - c.ux) * (Y[id] - y1);
            else ret += (c.ux - c.dx) * (y2 - Y[id]);
        }  else {
            if (c.ux < c.dx) ret += (c.dx - c.ux) * (y2 - Y[id]);
            else ret += (c.ux - c.dx) * (Y[id] - y1);
        }
        return ret;
    }
}

int main(void){
    for (scanf("%lf%lf%d%d", &MaxX, &MaxY, &N1, &N2); MaxX > 0.0;
scanf("%lf%lf%d%d", &MaxX, &MaxY, &N1, &N2)){
        aSize = 0;
        for (int i = 0; i < N1; i ++)
            scanf("%lf%lf", &X[i], &Y[i]), R[i] = .58;
        for (int i = N1; i < N1 + N2; i ++)
            scanf("%lf%lf", &X[i], &Y[i]), R[i] = 1.31;
        N = N1 + N2;
        app[aSize ++] = 0.0;
        app[aSize ++] = MaxY;
        for (int i = 0; i < N; i ++){
            for (int j = i + 1; j < N; j ++){
                if (R[i] + R[j] > dist(X[i] - X[j], Y[i] - Y[j]) &&
dist(X[i] - X[j], Y[i] - Y[j]) > fabs(R[i] - R[j])){
                    if (equal(dist(X[i] - X[j], Y[i] - Y[j]), 0.0))
continue;
                    double d2 = (sqr(X[i] - X[j]) + sqr(Y[i] - Y[j])
+ sqr(R[i]) - sqr(R[j])) / 2.0 / dist(X[i] - X[j], Y[i] - Y[j]);
                    double d = sqrt(sqr(R[i]) - sqr(d2));
                    if (sqr(R[i]) < sqr(d2)) d = 0.0;
                    double vy = X[i] - X[j];
                    vy *= d / dist(X[i] - X[j], Y[i] - Y[j]);
                    double vyy = (Y[j] - Y[i]) * d2 / dist(X[i] - X[j],
Y[i] - Y[j]);
                    if (vaild(Y[i] + vyy + vy)) app[aSize ++] = Y[i]
+ vyy + vy;
                    if (vaild(Y[i] + vyy - vy)) app[aSize ++] = Y[i]
+ vyy - vy;
                }
```

```
            }
        }
        for (int i = 0; i < N; i ++){
            if (vaild(Y[i] - R[i])) app[aSize ++] = Y[i] - R[i];
            if (vaild(Y[i] + R[i])) app[aSize ++] = Y[i] + R[i];
            if (X[i] + R[i] > MaxX){
                if (vaild(Y[i] + sqrt(sqr(R[i]) - sqr(MaxX - X[i]))))
app[aSize ++] = Y[i] + sqrt(sqr(R[i]) - sqr(MaxX - X[i]));
                if (vaild(Y[i] - sqrt(sqr(R[i]) - sqr(MaxX - X[i]))))
app[aSize ++] = Y[i] - sqrt(sqr(R[i]) - sqr(MaxX - X[i]));
            }
            if (X[i] - R[i] < 0.0){
                if  (vaild(Y[i]  +  sqrt(sqr(R[i])  -  sqr(X[i]))))
app[aSize ++] = Y[i] + sqrt(sqr(R[i]) - sqr(X[i]));
                if  (vaild(Y[i]  -  sqrt(sqr(R[i])  -  sqr(X[i]))))
app[aSize ++] = Y[i] - sqrt(sqr(R[i]) - sqr(X[i]));
            }
        }
        qsort(app, aSize, sizeof(double), qCmp0);
        int aSize2 = 1;
        for (int i = 1; i < aSize; i ++)
            if (!equal(app[i], app[i - 1])) app[aSize2 ++] = app[i];
        aSize = aSize2;
        //for (int i = 0; i < aSize; i ++) printf("%lf\n", app[i]);
        double ans = 0.0;
        for (int i = 1; i < aSize; i ++){
            double mid = (app[i] + app[i - 1]) / 2.0;
            int cSize = 0;
            for (int j = 0; j < N; j ++){
                if (mid <= Y[j] + R[j] && mid >= Y[j] - R[j]){
                    double d1 = sqrt(sqr(R[j]) - sqr(app[i - 1] -
Y[j]));
                    if (R[j] < fabs(app[i - 1] - Y[j])) d1 = 0.0;
                    double d2 = sqrt(sqr(R[j]) - sqr(app[i] - Y[j]));
                    if (R[j] < fabs(app[i] - Y[j])) d2 = 0.0;
                    C[cSize].type = 0;
                    C[cSize].id = j;
                    C[cSize].ux = X[j] - d1;
                    C[cSize].dx = X[j] - d2;
                    cSize ++;
                    C[cSize].type = 1;
                    C[cSize].id = j;
                    C[cSize].ux = X[j] + d1;
                    C[cSize].dx = X[j] + d2;
                    cSize ++;
                }
            }
            qsort(C, cSize, sizeof(C[0]), qCmp);
            int j = 0;
            while (j < cSize){
                int l = j, r = j + 1, deg = 1;
                for (; deg; r ++){
                    if (C[r].type == 0) deg ++;
                    else deg --;
                }
                r --;
                j = r + 1;
                double lx = max(C[l].ux, C[l].dx), rx = min(C[r].ux,
C[r].dx);
                ans += (min(rx, MaxX) - max(lx, 0.0)) * (app[i] - app[i
- 1]);
                if (X[C[r].id] + R[C[r].id] <= MaxX || C[r].ux < MaxX
- eps || C[r].dx < MaxX - eps) ans += getS(C[r], app[i - 1], app[i]);
                if (X[C[l].id] - R[C[l].id] >= 0.0 || C[l].ux > eps
|| C[l].dx > eps) ans += getS(C[l], app[i - 1], app[i]);
            }
        }
        printf("%.2lf\n", MaxX * MaxY - ans);
    }
}
```

以下代码实现扩展欧几里德算法

```
void extended_gcd(ll a,ll b,ll&x,ll&y)
{
    if(b==0)x=1,y=0;else
    {
        extended_gcd(b,a%b,y,x);
        y-=a/b*x;
    }
}
```

Hash实数精度下的点

```
struct Point{
    double x,y;
    inline Point(){}
    inline Point(double x,double y):x(x),y(y){}
    inline unsigned operator()(const Point &a)const{
        return (unsigned)(a.x+eps)*2741+(unsigned)(a.y+eps);
    }
    inline bool operator==(const Point &a)const{
        return !(x+eps<a.x||a.x+eps<x||y+eps<a.y||a.y+eps<y);
    }
    inline bool operator<(const Point &a)const{
```

```
        return x+eps<a.x||x<=a.x+eps&&y+eps<a.y;
    }
};
```

D-Triangle with MST

```
typedef long long real;
struct point { real x,y; struct edge *e; };
struct edge {
  point *o, *d;
  struct edge *on, *op, *dn, *dp;
};
typedef struct point point;
typedef struct edge edge;
#define  0p(e,p)  ((e)->o==p?(e)->d:(e)->o)
#define  Next(e,p)  ((e)->o==p?(e)->on:(e)->dn)
#define  Prev(e,p)  ((e)->o==p?(e)->op:(e)->dp)

edge *make_edge(point *u, point *v){
  edge *e = new edge();
  e->on=e->op=e->dn=e->dp=e;
  e->o=u;e->d=v;
  if (u->e==NULL) u->e=e;
  if (v->e==NULL) v->e=e;
  return e;
}
void delete_edge(edge *e){
  point *u=e->o, *v=e->d;
  if (u->e==e) u->e=e->on;
  if (v->e==e) v->e=e->dn;
  if (e->on->o==u) e->on->op=e->op; else e->on->dp=e->op;
  if (e->op->o==u) e->op->on=e->on; else e->op->dn=e->on;
  if (e->dn->o==v) e->dn->op=e->dp; else e->dn->dp=e->dp;
  if (e->dp->o==v) e->dp->on=e->dn; else e->dp->dn=e->dn;
  delete e;
}
void splice(edge *a, edge *b, point *v){
  edge *n;
  if (a->o==v) { n=a->on; a->on=b; } else { n=a->dn; a->dn=b; }
  if (n->o==v) n->op=b; else n->dp=b;
  if (b->o==v) { b->on=n; b->op=a; } else { b->dn=n; b->dp=a; }
}
edge *join(edge *a, point *u, edge *b, point *v, int s){
  edge *e = make_edge(u,v);
  if (s == 0){ if (a->o==u) splice(a->op,e,u); else splice(a->dp,e,u);
splice(b,e,v); }
  else { splice(a,e,u); if (b->o==v) splice(b->op,e,v); else
splice(b->dp,e,v); }
  return e;
}

#define Vector(p1,p2,u,v)     (u=p2->x-p1->x,v=p2->y-p1->y)
#define cross_v(u1,v1,u2,v2) (u1*v2-v1*u2)
#define cross_p(p1,p2,p3)
((p2.x-p1.x)*(p3.y-p1.y)-(p2.y-p1.y)*(p3.x-p1.x))
#define dot_v(u1,v1,u2,v2)    (u1*u2+v1*v2)
const double eps = 1e-10;

void lower_tangent(edge *l, point *s, edge *r, point *u, edge **ll, point
**llo, edge **rl, point **rlo){
  point *ol=s, *dl=0p(l,s);
  point *oor=u, *dr=0p(r,u);
  while(1)
       if (cross_p((*ol),(*dl),(*oor))>0) { l=Prev(l,dl); ol=dl;
dl=0p(l,ol); }
       else if (cross_p((*oor),(*dr),(*ol))<0) { r=Next(r, dr); oor=dr;
dr=0p(r,oor); }
       else break;
  *ll=l; *rl=r; *llo=ol; *rlo=oor;
}

static void merge(edge *r_cw_l, point *s, edge *l_ccw_r, point *u, edge
**l_tangent){
  edge *b, *lc, *rc, *ll, *rl, *next, *prev;
  point *ob, *db, *dlc, *drc, *orl, *oll, *dest_next, *dest_prev;
  real ulcob, vlcob, ulcdb, vlcdb, urcob, vrcob, urcdb, vrcdb, cplc, cprc,
dplc, dprc;
  real unob, vnob, undb, vndb, cpn, dpn, upob, vpob, updb, vpdb, cpp, dpp;
  real alc, arc, an, ap;
  double crc, clc;

  lower_tangent(r_cw_l, s, l_ccw_r, u, &ll, &oll, &rl, &orl);
```

```
  b = join(ll, oll, rl, orl, 1);
  ob = oll; db = orl;

  *l_tangent = b;

  do{
    lc=Next(b,ob); rc=Prev(b,db); dlc=Op(lc,ob); drc=Op(rc,db);
    Vector(dlc,ob,ulcob,vlcob); Vector(dlc,db,ulcdb,vlcdb);
Vector(drc,ob,urcob,vrcob); Vector(drc,db,urcdb,vrcdb);
    cplc=cross_v(ulcob, vlcob, ulcdb, vlcdb); cprc=cross_v(urcob, vrcob,
urcdb, vrcdb);
    alc=(cplc > 0); arc=(cprc > 0);
    if (!alc && !arc) break;
    if (alc){
      dplc = dot_v(ulcob, vlcob, ulcdb, vlcdb);
      clc = (double)dplc / cplc;
      do{
        next = Next(lc, ob);
        dest_next = Op(next, ob);
        Vector(dest_next, ob, unob, vnob);
        Vector(dest_next, db, undb, vndb);
        cpn = cross_v(unob, vnob, undb, vndb);
        an = (cpn > 0);
        if (!an) break;
        dpn = dot_v(unob, vnob, undb, vndb);
        double cn = (double)dpn / cpn;
        if (cn > clc) break;
        delete_edge(lc);
        lc = next;
        clc = cn;
      }while(1);
    }
    if (arc){
      dprc = dot_v(urcob, vrcob, urcdb, vrcdb);
      crc = (double)dprc / cprc;
      do{
        prev = Prev(rc, db);
        dest_prev = Op(prev, db);
        Vector(dest_prev, ob, upob, vpob);
        Vector(dest_prev, db, updb, vpdb);
        cpp = cross_v(upob, vpob, updb, vpdb);
        ap = (cpp > 0);
        if (!ap) break;
        dpp = dot_v(upob, vpob, updb, vpdb);
        double cp = (double)dpp / cpp;
        if (cp > crc) break;
        delete_edge(rc);
        rc = prev;
        crc = cp;
      } while (1);
    }
    dlc=Op(lc, ob); drc=Op(rc, db);
    if (!alc || (alc && arc && crc < clc)){ b = join(b, ob, rc, drc, 1);
db = drc; }
     else { b = join(lc, dlc, b, db, 1); ob = dlc; }
  }while(1);
}

void divide(point *p, int l, int r, edge **l_ccw, edge **r_cw){
  int n=r-l+1;
  edge *l_ccw_l, *r_cw_l, *l_ccw_r, *r_cw_r, *l_tangent, *a, *b, *c;
  if (n == 2) {
    *l_ccw = *r_cw = make_edge(&p[l], &p[r]);
  }else if (n == 3) {
    a = make_edge(&p[l], &p[l+1]);
    b = make_edge(&p[l+1], &p[r]);
    splice(a,b,&p[l+1]);
    real c_p = cross_p(p[l], p[l+1], p[r]);
    if (c_p>0){ c=join(a,&p[l],b,&p[r],1); *l_ccw=a; *r_cw=b; }
     else if (c_p<0) { c=join(a,&p[l],b,&p[r],0); *l_ccw=c; *r_cw=c; }
     else { *l_ccw=a; *r_cw=b; }
  }else if (n > 3){
    int split=(l+r)/2;
    divide(p,l,split,&l_ccw_l,&r_cw_l);
    divide(p,split+1,r,&l_ccw_r,&r_cw_r);
    merge(r_cw_l,&p[split],l_ccw_r,&p[split+1],&l_tangent);
    if(l_tangent->o == &p[l]) l_ccw_l=l_tangent;
    if(l_tangent->d == &p[r]) r_cw_r=l_tangent;
    *l_ccw=l_ccw_l; *r_cw=r_cw_r;
  }
```

```
}
int n, m;
point *p;
int*eu, *ev, *pr, *r;
double *ew;

double dis(point p1, point p2){ return
sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)); }

void enum_edges(int n){
  edge *e_start, *e;
  point *u, *v;
  eu = new int[n*3]; ev = new int[n*3]; ew = new double[n*3];
  m = 0;
  for (int i = 0; i < n; i++) {
    u = &p[i];
    e_start = e = u->e;
    do{
      v = Op(e, u);
      if (u < v){ eu[m] = u-p; ev[m] = v-p; ew[m]=dis(*u, *v); m++; }
      e = Next(e, u);
    } while (e!=e_start);
  }
}

int cmp(const point& a, const point& b){ return (a.x < b.x || a.x == b.x
&& a.y < b.y); }
int cmp_e(const point& a, const point& b){ return a.x == b.x && a.y ==
b.y; }
int cmp2(const int& i, const int& j){ return ew[i]<ew[j]; }
int find(int x){ return x==pr[x]?x:pr[x]=find(pr[x]); }

double kruskal(){
  double ans=0.0;
  int i, j=0, e, u, v;
  pr = new int[n]; r = new int[m];
  for(i=0;i<n;i++) pr[i]=i;
  for(i=0;i<m;i++) r[i]=i;
  sort(r,r+m,cmp2);
  for(i=0; i<n-1; i++){
    do{ e=r[j++]; u=find(eu[e]); v=find(ev[e]); }while(u==v);
    ans+=ew[e]; pr[u]=v;
  }
  return ans;
}

int main(){
    edge* l_cw, *r_ccw;
    int i;
    scanf("%d", &n);
    p = new point[n];
    for (i = 0; i < n; i++){
      scanf("%I64d%I64d", &p[i].x, &p[i].y);
      p[i].e = NULL;
    }
    sort(p, p+n, cmp);
    n=unique(p, p+n, cmp_e)-p;
    if(n == 1)
        printf("%.4lf\n", 0.0);
    else{
        divide(p, 0, n-1, &l_cw, &r_ccw);
        enum_edges(n);
        printf("%.4lf\n", kruskal());
    }
    return 0;
}
```

单纯形代码（需保证收敛）

```
/*
说明：
    本来变量都应放在class里面的，但是由于在里面开大内存会RE，所以暂时
先放外面。
    N[0]代表N中的元素个数，B[0]代表B中的元素个数。
    读入格式（在文件名为inputName的文件中读入）：
        首先两个数n, m，表示未知数的数量和约束的数量。
        接下来一行n个数，为目标函数的系数。
        然后m行，每行m+1个数，表示一个约束。前m个数是系数，最后一个是
常数项。
    输出格式（在文件名为outputName的文件中输出）：
        如果无解，只有一行"Infeasible"。
        如果解可以无穷大，只有一行"Unbounded"。
```

否则，第一行为最大的目标函数值，接下来是每个未知数的值。
*/

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

const double eps = 1e-10;
const int MAXSIZE = 2000;
const int oo = 1000000000;

double A[MAXSIZE+1][MAXSIZE+1], tA[MAXSIZE+1][MAXSIZE+1];
double b[MAXSIZE+1], tb[MAXSIZE+1], c[MAXSIZE+1], tc[MAXSIZE+1];
int N[MAXSIZE+1+1], B[MAXSIZE+1+1];
int n, m;
double v;

class LinearProgramming
{

    void read()
    {
        scanf("%d%d", &n, &m);
        for(int i=1; i<=n; i++)
            scanf("%lf", &c[i]);
        for(int i=1; i<=m; i++)
        {
            for(int j=1; j<=n; j++)
                scanf("%lf", &A[n+i][j]);
            scanf("%lf", &b[n+i]);
        }
    }

    void pivot(int l, int e)
    {
        tb[e] = b[l]/A[l][e];
        tA[e][l] = 1/A[l][e];
        for(int i=1; i<=N[0]; i++)
            if (N[i] != e)
                tA[e][N[i]] = A[l][N[i]]/A[l][e];
        for(int i=1; i<=B[0]; i++)
        {
            tb[B[i]] = b[B[i]]-A[B[i]][e]*tb[e];
            tA[B[i]][l] = -A[B[i]][e]*tA[e][l];
            for(int j=1; j<=N[0]; j++)
                if (N[j] != e)
                    tA[B[i]][N[j]] =
A[B[i]][N[j]]-tA[e][N[j]]*A[B[i]][e];
        }

        v += tb[e]*c[e];
        tc[l] = -tA[e][l]*c[e];
        for(int i=1; i<=N[0]; i++)
            if (N[i] != e)
                tc[N[i]] = c[N[i]]-tA[e][N[i]]*c[e];

        for(int i=1; i<=N[0]; i++)
            if (N[i] == e) N[i] = l;
        for(int i=1; i<=B[0]; i++)
            if (B[i] == l) B[i] = e;
        for(int i=1; i<=B[0]; i++)
        {
            for(int j=1; j<=N[0]; j++)
                A[B[i]][N[j]] = tA[B[i]][N[j]];
            b[B[i]] = tb[B[i]];
        }
        for(int i=1; i<=N[0]; i++)
            c[N[i]] = tc[N[i]];
    }

    bool opt()//false stands for unbounded
    {
        while (true)
        {
            int l, e;
            double maxUp = -1;//不能是!
            for(int ie=1; ie<=N[0]; ie++)
            {
                int te = N[ie];
```

```
                if (c[te] <= eps) continue;
                double delta = oo;
                int tl = MAXSIZE+1;
                for(int i=1; i<=B[0]; i++)
                    if (A[B[i]][te] > eps)
                    {
                        double temp = b[B[i]]/A[B[i]][te];
                        if (delta == oo || temp < delta || temp ==
delta && B[i] < tl)
                        {
                            delta = temp;
                            tl = B[i];
                        }
                    }
                if (tl == MAXSIZE+1) return false;
                if (delta*c[te] > maxUp)
                {
                    maxUp = delta*c[te];
                    l = tl;
                    e = te;
                }
            }
            if (maxUp == -1) break;
            pivot(l, e);
        }
        return true;
    }

    void delete0()
    {
        int p;
        for(p=1; p<=B[0]; p++)
            if (B[p] == 0) break;
        if (p <= B[0]) pivot(0, N[1]);
        for(p=1; p<=N[0]; p++)
            if (N[p] == 0) break;
        for(int i=p; i<N[0]; i++)
            N[i] = N[i+1];
        N[0]--;
    }
```

```
bool initialize()
{
    N[0] = B[0] = 0;
    for(int i=1; i<=n; i++)
        N[++N[0]] = i;
    for(int i=1; i<=m; i++)
        B[++B[0]] = n+i;
    v = 0;

    int l = B[1];
    for(int i=2; i<=B[0]; i++)
        if (b[B[i]] < b[l])
            l = B[i];
    if (b[l] >= 0) return true;

    double origC[MAXSIZE+1];
    memcpy(origC, c, sizeof(double)*(n+m+1));
    N[++N[0]] = 0;
    for(int i=1; i<=B[0]; i++)
        A[B[i]][0] = -1;
    memset(c, 0, sizeof(double)*(n+m+1));
    c[0] = -1;
    pivot(l, 0);
    opt();//unbounded????
    if (v < -eps) return false;
    delete0();

    memcpy(c, origC, sizeof(double)*(n+m+1));
    bool inB[MAXSIZE+1];
    memset(inB, false, sizeof(bool)*(n+m+1));
    for(int i=1; i<=B[0]; i++)
        inB[B[i]] = true;
    for(int i=1; i<=n+m; i++)
        if (inB[i] && c[i] != 0)
        {
            v += c[i]*b[i];
            for(int j=1; j<=N[0]; j++)
                c[N[j]] -= A[i][N[j]]*c[i];
            c[i] = 0;
```

```
            }
        return true;
    }

    public: void simplex(string inputName, string outputName)
    {
        freopen(inputName.c_str(), "r", stdin);
        freopen(outputName.c_str(), "w", stdout);
        read();
        if (!initialize())
        {
            printf("Infeasible\n");
            return;
        }
        if (!opt())
        {
            printf("Unbounded\n");
            return;
        }
        else printf("Max value is %lf\n", v);

        bool inN[MAXSIZE+1];
        memset(inN, false, sizeof(bool)*(n+m+1));
        for(int i=1; i<=N[0]; i++)
            inN[N[i]] = true;
        for(int i=1; i<=n; i++)
            if (inN[i]) printf("x%d = %lf\n", i, 0.0);
            else printf("x%d = %lf\n", i, b[i]);
    }
};

int main()
{
    LinearProgramming test;
    test.simplex("a.in", "a.out");
}
```

以下代码用以计算树计数，源码出自PT07-contest

Given two integer n, p. 4 kinds of query is needed to solve:

1. Counting the number of labeled unrooted trees with n nodes
2. Counting the number of labeled rooted trees with n nodes
3. Counting the number of unlabeled rooted trees with n nodes
4. Counting the number of unlabeled unrooted trees with n nodes

Calculate the answer modulo p.

```
int a[r],s[r][r],k,n,p;
int pow(int a,int b,int p){
    int ans=1;a%=p;
    while(b>0){
        if(b&1)
        ans=ans*a%p;
        a=a*a%p;b/=2;
    }
    return ans;
}
void extended_gcd(int a,int b,int c,int &x,int &y){
    if(b==0)x=c/a,y=0;
    else{
        extended_gcd(b,a%b,c,y,x);
        y-=a/b*x;
    }
}
int main()
{
    int tmp,__;
    while(scanf("%d%d%d",&k,&n,&p)!=-1){
        switch(k){
        case 1:printf("%d\n",pow(n,n-2,p)); break;
        case 2:printf("%d\n",pow(n,n-1,p)); break;
        default:
              a[1]=1;
            for(int N=1; N<n; N++){
                a[N+1]=0;
                for(int i=1; i<=N; i++){
                    if(N-i<i) s[N][i]=a[N+1-i];
                    else s[N][i]=(s[N-i][i]+a[N+1-i])%p;
                    a[N+1]+=s[N][i]*i%p*a[i]%p;
                }
                __=a[N+1]%p;
                extended_gcd(N,p,__,a[N+1],tmp);
                if(a[N+1]>=0) a[N+1]%=p;
```

```
            else a[N+1]=p-(-a[N+1])%p;
        }
        if(k==4){
            for(int i=1; i<=n/2; i++)a[n]=(a[n]+p-a[i]*a[n-i]%p)%p;
            if((n&1)==0) a[n]+=a[n/2]*(a[n/2]+1)/2;
        }
        printf("%d\n",a[n]%p);
        break;
      }
   }
}
```

以下代码用以求解最小环排列问题

```
char s[10010];int n;
int MCP(){
    int i, j, x, y, u, v;
    for(x=0,y=1; y<n; y++)if(s[y]<=s[x];{
        i=u=x;j=v=y;
        while(s[i]==s[j]){
            ++u;if(++i==n)i=0;++v;
            if(++j==n)j=0;if(i==x)break;
        }
        if(s[i]<=s[j])y=v;
        else{x=y;if(u>y)y=u;}
    }
    return x;
}
```

斯坦纳MST核心代码

```
for(int i=0; i<256; i++){
    if((i&(i-1))==0)continue;
    for(int j=0; j<n; j++){
        dp[j][i]=oo;
        for(int k=1; k<i; k++)
        if((i|k)==i)
        dp[j][i]=min(dp[j][i],dp[j][k]+dp[j][i-k]);
    }
    memset(f,0,sizeof(f));
    for(int j=0; j<n; j++){
        t1=oo;
        for(int k=0; k<n; k++) if(dp[k][i]<t1 && !f[k]) t1=dp[x=k][i];
        f[x]=1;
        for(int k=0; k<n; k++) dp[k][i]=min(dp[k][i],t1+a[x][k]);
    }
}
```

RHO & MULMOD

```
int64 mulmod(int64 a, int64 b, int64 p){
    int64 y = (int64)((double)a*(double)b/p+0.5);
    int64 r = a*b-y*p;
    if (r<0) r = r+p;
    return r;
}
bool millar(int64 n,int base)  {
    int64 n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=powmod(base,n2,n);
    if((res==1)||(res==n-1)) return true; s--;
    while(s>=0) {
        res=mulmod(res,res,n);
        if(res==n-1)  return true;
        s--;
    }
    return false;
}
int64 rho(int64 n){
    int64 x,y,d,c=1;
    for(int i=2;i<=100;i++) if(n%i==0) return i;
    while(1) {
        x = y = 2;
        while(1) {
            x = mulmod(x,x,n); x = (x+c) % n;
            y = mulmod(y,y,n); y = (y+c) % n;
            y = mulmod(y,y,n); y = (y+c) % n;
            d = gcd(abs(x-y),n);
            if(d==n) break;else if(d>1) return d;
        }
        c++;
    }
}
```

求解平方剩余，先要特判勒让得符号为-1与p=2的情况。

```
if(p%4==3)ans=powmod(n,(p+1)/4,p);else{
    q=p-1,s=0;
    while(q%2==0)q/=2,s++;
    for(w=1; powmod(w,(p-1)/2,p)==1; w++);
    r=powmod(n,(q+1)/2,p);
    v=powmod(w,q,p);
    _n=powmod(n,p-2,p);
    while(1)
    {
        for(i=0,t=r*r%p*_n%p; t!=1; t=t*t%p,i++);
        if(i==0)
        {
            ans=r;
            break;
        }
        r=r*powmod(v,1LL<<((int)(s-i-1)),p)%p;
    }
}
```

Given a arithmetical progression z=a*y+b, y->[x..x+n]
Calculate the number of z, z%m lies in [c,d]

```
#include<stdio.h>
#define ll "ll"
#define int64 long long int
int64 calc_single(int64 u,int64 r,int64 n,int64 d){
    if(d<u)
        return 0;
    if(u+r*(n-1)<=d)
        return n;
    return (d-u)/r+1;
}
int64 calc(int64 u,int64 r,int64 n,int64 d,int64 m)
{
    int64 ret=0,n1,u1,tot,newd,newr;
    u%=m,r%=m;
    if(u+r*(n-1)<m)//Only one single - and not complete - chuck.
        return calc_single(u,r,n,d);
    //The first chuck.
    n1=(m-1-u)/r+1;
    ret+=calc_single(u,r,n1,d);
    u=(u+n1*r)%m;
    n-=n1;
    //The last chuck. variable "newr" is a temporary one.
    u1=(u+(n-1)*r)%m,newr=u1%r;
    n1=(u1-newr)/r+1;
    ret+=calc_single(newr,r,n1,d);
    n-=n1;
    if(n==0)
        return ret;
    //now we come to the chucks in the middle.
    //We calculate the number of chucks first.
    tot=(u+(n-1)*r)/m+1;
    ret+=d/r*tot;
    //Last step. Recursion.
    newd=d%r,newr=((m-1)/r+1)*r%m;
    if(newr<=r/2)
        ret+=calc(u,newr,tot,newd,r);
    else
        //the inverse(?) of the original task. See the code below.
        ret+=calc((u+newr*(tot-1))%r,r-newr,tot,newd,r);
    return ret;
}
int main(){
    int _;
    int64 a,b,x,n,c,d,m;
    for(scanf("%d",&_); _--;)
    {

    scanf("%"ll"d%"ll"d%"ll"d%"ll"d%"ll"d%"ll"d%"ll"d",&a,&b,&x,&n,&
c,&d,&m);
        //We transform this progression to z=a*y+b, y->[0..n]
        b=a*x+b;n++;
        printf("%"ll"d\n",calc(b,a,n,d,m)-(c?calc(b,a,n,c-1,m):0));
    }
    return 0;
}
```

求解高次方程

```
double power(double x,double y){return x<0?-pow(-x,y):pow(x,y);}
double sqr(double x){return x*x}
double findRoot3(double a,double b,double c){
    //x^3+a*x^2+b*x+c=0;let x=y-a/3,y^3+p*y+q=0
```

```
    double p=b-a*a/3;
    double q=2*a*a*a/27-a*b/3+c;
    double d=sqr(q/2)+power(q/3,3.0);
    if(d>=0)
        return
power(-q/2+sqrt(d),1.0/3)+power(-q/2-sqrt(d),1.0/3)+a/3;
    a=-q/2; b=sqrt(-d);
    return 2*cos(atan2(abs(b),a)/3)*pow(a*a+b*b,1.0/6);
}
void findRoot2(vector<double>&ans,double b,double c){
    if(b*b-4*a*c>=0){
        ans.push_back((-b+sqrt(b*b-4*a*c))/2/a);
        ans.push_back((-b-sqrt(b*b-4*a*c))/2/a);
    }
}
void findRoot4(vector<double>&ans,double a,double b,double c,double
d,double e){
    if(!a)return ans;//degenerates
    b/=a,c/=a,d/=a,e/=a;
    double y0=findRoot3(-c,b*d-4*e,-b*b*e+4*c*e-d*d);
    findRoot2(ans,1,(b+sqrt(b*b-4*c+4*y0))/2,
              (y0+(b*y0-2*d)/sqrt(4*y0+b*b-4*c))/2);
    findRoot2(ans,1,(b+sqrt(b*b-4*c+4*y0))/2,
              (y0+(b*y0-2*d)/sqrt(4*y0+b*b-4*c))/2);
}
```

快速傅立叶变换

```
// n should be the power of 2
// if it's reverse FFT, theta should be -2 * PI / n, otherwise 2 * PI /
n
// a[] is the Source and the Destination of FFT
void fft(int n, double theta, Complex a[]) {
    int u = 1;
    for(int m=n; m>=2; m>>=1) {
        int mh = m >> 1;
        for(int i=0; i<mh; i++) {
            Complex w = exp(i*theta*I);
            for(int j=i; j<n; j+=m) {
                int k = j + mh;
                Complex x = a[j] - a[k];
                a[j] += a[k];
                a[k] = w * x;
            }
        }
        theta *= 2;
    }
    int i = 0;
    for(int j=1; j<n-1; j++) {
        for(int k=n>>1; k>(i^=k);k>>=1);
        if(j<i) swap(a[i],a[j]);
    }
}
```

快速傅立叶变换递归版

```
// if it's reverse FFT, theta should be -2 * PI / n, otherwise 2 * PI /
n
// a[] is the Source and the Destination of FFT
Complex c[20];
void dft(int n, Complex theta, Complex*a)
{
    for(int i=0; i<n; i++)
    {
        c[i]=0;
        Complex w = pow(theta,i);
        for(int j=n-1; j>=0; j--)
            c[i]=c[i]*w+a[j];
    }
    for(int i=0; i<n; i++)
        a[i]=c[i];
}
void fft(int n, Complex theta, Complex*a) {
    int n1,n2;
    if(n==1)return;
    for(int t=0;;t++)
        if(n%p[t]==0)
        {
            n1=p[t],n2=n/p[t];
            break;
        }
    Complex*b=a+n,*c=a+n+n;
    for(int i=0; i<n1; i++)
    {
```

```
        for(int j=i,k=0; j<n; j+=n1,k++)
            b[k]=a[j];
        fft(n2,pow(theta,n1),b);
        for(int j=i,k=0; j<n; j+=n1,k++)
            a[j]=b[k];
    }
    for(int i=0; i<n2; i++)
    {
        for(int j=i*n1,k=0; j<(i+1)*n1; j++,k++)
            b[k]=a[j]*pow(theta,i*k);
        dft(n1,pow(theta,n2),b);
        for(int j=0;j<n1;j++) c[j*n2+i]=b[j];
    }
    for(int i=0; i<n; i++) a[i] = c[i];
}
```

连通性dp，源码出自楼教主男人程序

```
#define int64 long long
#define get(a,p) (((a)>>((p)*2))&3)
#define set2(a,p,v) (((a)&~(15<<(p)*2))|((v)<<(p)*2))
#define set(a,p,v) (((a)&~(3<<(p)*2))|((v)<<(p)*2))
const int maxn = 12;const int maxh = 2003;
struct hash
{
    int p[maxh]; int64 v[maxh];
    int64 &operator[](int k){
        int kk=k%maxh;
        while(v[kk]&&p[kk]!=k)kk=kk+1==maxh?0:kk+1;
        p[kk]=k;
        return v[kk];
    }
    void clear(){memset(v,0,sizeof(v));}
}dp[2];
int n,m,s,a,p1,p2;int64 v;char map[maxn][maxn];
int main(){
    while(scanf("%d%d",&n,&m)&&n&&m){
        for(int i=0;i<n;i++)scanf("%s",map[i]);
        if(map[n-1][0]=='#'||map[n-1][m-1]=='#'){printf("0\n");conti
        nue;
        }
        if(n==1&&m==1){printf("1\n");continue;}
        for(int j=0;j<m;j++)map[n][j]=map[n+1][j]='.';
        for(int j=1;j+1<m;j++)map[n][j]='#';
        n+=2;
        dp[s=0].clear();dp[s][0]=1;
        for(int i=0;i<n;i++)
            for(int j=0;j<m;j++,s^=1){
                dp[s^1].clear();
                for(int k=0;k<maxh;k++){
                    if(dp[s].v[k]==0)continue;
                    a=dp[s].p[k];v=dp[s].v[k];
                    if(!j){
                        if(get(a,m))continue;
                        a<<=2;
                    }
                    p1=get(a,j); p2=get(a,j+1);
                    if(map[i][j]=='#'){
                        if(p1|p2)continue;
                        dp[s^1][a]+=v;
                    }
                    else if(!(p1||p2)) dp[s^1][set2(a,j,6)]+=v;
                    else if(!(p1&&p2))
dp[s^1][set2(a,j,(p1|p2))]+=v, dp[s^1][set2(a,j,(p1|p2)<<2)]+=v;
                    else if(p1!=p2){

    if(p1==1||i+j==n+m-2)dp[s^1][set2(a,j,0)]+=v;
                    }
                    else if(p1==2){
                        int c=1,d=j+1;
                        while(c){
                            d++;
                            if(get(a,d)==2)c++;
                            else if(get(a,d)==1)c--;
                        }
                        dp[s^1][set(set2(a,j,0),d,2)]+=v;
                    }
                    else if(p1==1){
                        int c=1,d=j;
                        while(c){
                            d--;
                            if(get(a,d)==2)c--;
```

```
                        else if(get(a,d)==1)c++;
                    }
                    dp[s^1][set(set2(a,j,0),d,1)]+=v;
                }
            }
        }printf("%I64d\n",dp[s][0]);
    }return 0;
}
```

自然语言资料：

一：各数量级素数

11657,104729,1076143,9763393,67867967,166666666666667,499999999999993

二：常用常数

e= 2.71828 18284 59045 23536

Euler = 0.57721566490153286060651209008240243104 2

Catalan数列通项:(2n)! / (n! * (n+1)!)

三：常用算法描述

4.1 旋转坐标轴公式

(x′ cos -y′ sin)i+(x′ sin +y′ cos)j

4.2 稳定婚姻问题

延迟认可算法：（女士最优）

从每位女士被标记落选开始。

当存在落选女士时，做：

(1) 每位落选女士在所有尚未拒绝她的男士中选择最偏向的男士。

(2) 每位男士在选择他的女士且未被拒绝的女士中挑选他最偏向的，对她延迟决定，并拒绝其他女士。

4.3 铺砖问题:

如果p*q能覆盖a*b，当且仅当它们必满足以下条件之一：

1、a,b中一个是p的整数倍，另一个是q的整数倍。

2、a,b中一个是p*q的整数倍，另一个能表示成mp+nq(m,n为非负整数)的形式。

4.4 平面镶嵌

一个一般图形能够被非旋转对称平面镶嵌当且仅当下面两个条件满足一个：

1.可以找到四个点A,B,C,D，图形的边界从A->B可以与D->C完全重合，B->C可以与A->D完全重合；

2.可以找到六个点A,B,C,D,E,F，图形的边界从A->B,B->C,C->D可以对应与E->D,F->E,A->F完全重合。

这个比较容易理解，因为对于一个凸多边形来说，只有四边形和六边形可以进行这样的平面镶嵌。

并且可以注意到这里的诸如A->B，B->C的向量就是图形镶嵌的偏移量。并且容易知道偏移量的叉积应该等于图形的面积，因为可以把这种图形等效地看做六边形或者四边形。

对于格点多边形的情况，只需要两个偏移向量u和v即可，六边形的情况的另外一个向量可以用u+v代替。

4.5 输入一有向无环图，求一个最大的点集，使点集中的点互不可达。

解答：为了更直观地表示顶点的关系，我们先用floyd算法求出任意两个顶点之间是否可达的信息，从而构造出一个新图，此图仍为有向无环图。

定理：图中的最小路径覆盖（路径覆盖所有顶点，任意两条路径没有公共顶点）数等于题目所求点集的大小。

4.6 线性时间求解表达式

在线性时间求解表达式需要维护一个栈，栈每个元素是一个数字和他后面接着的操作符。

顺序处理整个表达式，在遇到一个操作符或者前括号的时候将当前的值和这个操作符入栈，入栈之前首先将栈顶的优先级低于和等于本操作符的所有操作弹出并进行计算（如果是右结合则没有等于的情况）。然后将计算获得的新值入栈。遇到后括号的时候将栈顶直到第一个前括号的元素全部出栈按照上面的方法进行计算。

我每次只有遇到一个操作符或者是前括号的时候才会将当前的操作数入栈，其他时候都只是维护当前的操作数，并不入栈。

对于单元运算符可以看做是和一个无意义的变量进行操作的双元运算符。

可以将表达式最外面添加一对括号，以方便最后获得表达式的值。

4.7 多塔问题

题目描述：N个盘子M个柱子的汉诺塔问题，输出一组方案。

算法：

dp[n,m] = min(dp[k][m]*2+dp[n-k][m-1])

4.8 求原根的算法

If the multiplicative order of a number m modulo n is equal to fai(n), then it is a primitive root. In fact the converse is true: If m is a primitive root modulo n then the multiplicative order of m is fai(n). We can use this to test for primitive roots.

First, compute fai(n). Then determine the different prime factors of fai(n), say p(1),...,p(k). Now, for every element m, compute m^(fai(n)/p(i))%n for i=1,2,...,k

using a fast algorithm for modular exponentiation such as exponentiation by squaring. A number m for which these k results are all different from 1 is a primitive root.

The number of primitive roots modulo n, if there are any, is equal to fai(fai(n)).

4.9 vim RC

```
set sw=4   set ts=4
set go=e   set nu
set autoindent   set cindent
```

set backspace=indent,eol,start syntax on

4.10 给出两个有标号的儿子顺序固定的有根树T和T'，要求计算出T和T'之间的"距离"——把T变成T'的最小操作次数，操作包括：

1.改变点的标号

2.删除一个点（将其儿子接到父亲上，替代它的位置）

3.新建一个点

Solution:

O(n^2*m^2) dynamic programming. Insertions and deletions are reversible, so you can consider only deleting from either the 1st or 2nd tree rather than ever inserting. In each dp state you're trying to turn one forest into another and can either: (a) delete the root of the rightmost tree in the left forest and recurse, (b) delete the root of the rightmost tree in the right forest and recurse, or (c) recurse on both turning the rightmost tree in the left forest into the rightmost tree in the right forest as well as the remaining part of the left forest into the remaining part of the right forest. If you label the nodes in each initial tree by post-order traversal, the states of the dp are just pairs of contiguous intervals (each forest is an interval of node labels).

4.11 将文件映射到内存buf

fread(str,sizeof(char),1<<23,stdin);

4.12 Linux常用命令

cat,chmod,cmp,diff,rm,mv,cp

cd,mkdir,ls

ps,kill,su,sudo

bash的第一行必须为 #!/bin/bash, 参数为 $0 $1 $2