
Template For ICPC



acm International Collegiate
Programming Contest



Author: HKing

Email: 1470042308@qq.com

目录

1	Graph	1
1.1	Dinic	1
1.2	KM	2
1.3	MixedEulerCircuit	4
2	Tree	5
2.1	DivideAndConquerTree	5
2.2	LinkTree	7
2.3	SegmentTree	10
2.4	SplayTree	14
2.5	Treap	17
3	Geometry	19
3.1	BasicStructandAlgorithm	19
3.2	PolygonArea	20
3.3	HalfPlaneIntersection	20
4	Math	20
4.1	ChinaRemainderTheory	20
4.2	Decompose	21
4.3	EulerPhi	22
4.4	ExtendGCD	23
4.5	IntegerInverse	23
4.6	LineMod	23
4.7	LogMod	23
4.8	Lucas	24
4.9	MillerRabin	25
4.10	MulMod	25
4.11	PollardRho	25
4.12	PowMod	26
4.13	PowerMod	26
4.14	PrimitiveRoot	27
4.15	SquareMod	27
5	Others	28
5.1	ExactCover	28
5.2	MatrixFastPower	30
5.3	Polynomial	31
6	字符串	33
6.1	test	33
7	数学	35
7.1	高斯消元	35
7.2	快速幂	35
7.3	欧几里德	36
7.4	扩展欧几里德算法	36

7.5	试除法分解质因数	36
7.6	试除法判定质数	36
7.7	试除法求所有约数	37
7.8	约数个数和约数之和	37
7.9	组合数	37
7.9.1	递推法求组合数	37
7.9.2	通过预处理逆元的方式求组合数	37
7.9.3	Lucas 定理求组合数	38
7.9.4	分解质因数法求组合数	38
7.10	欧拉函数	40
7.10.1	求欧拉函数	40
7.10.2	筛法求欧拉函数	40
7.11	筛法	41
7.11.1	朴素筛法	41
7.11.2	线性筛法	41
8	数据结构	41
8.1	Trie 树	41
8.2	并查集.cpp	42
9	图论	43
9.1	test	43
10	动态规划	45
10.1	test	45
11	计算几何	47
11.1	test	47
12	其他	49
12.1	test	49

1 Graph

1.1 Dinic

```
1  const int inf = 0x3f3f3f3f;
2  const int N = 205;
3  const int M = 1205;
4  struct Edge{
5      int v,f,nxt;
6  };
7  struct Dinic{
8      int src,sink;
9      int g[N],en;
10     Edge e[M*2];
11     int level[N];
12     void _addEdge(int u,int v,int f){
13         e[en].v=v;
14         e[en].f=f;
15         e[en].nxt=g[u];
16         g[u]=en++;
17     }
18     void addEdge(int u,int v,int f){
19         _addEdge(u,v,f);
20         _addEdge(v,u,0);
21     }
22     void init(){
23         en=0;
24         memset(g,-1,sizeof(g));
25     }
26     int q[N],front,rear;
27     bool bfs(){
28         memset(level,0,sizeof(level));
29         level[src]=1;
30         front=0; rear=1;
31         q[0]=src;
32         while(front<rear){
33             int u=q[front++];
34             if(u==sink)return 1;
35             for(int i=g[u];i!=-1;i=e[i].nxt){
36                 int v=e[i].v,f=e[i].f;
37                 if(!level[v]&&f){
38                     level[v]=level[u]+1;
39                     q[rear++]=v;
40                 }
41             }
42         }
43         return 0;
44     }
45     int dfs(int u,int delta){
46         if(u==sink || delta==0)return delta;
47         int ret=0;
48         for(int i=g[u];i!=-1;i=e[i].nxt){
```

```

49     int v=e[i].v, f=e[i].f;
50     if(level[v]==level[u]+1&&f){
51         int minf=min(delta-ret,f);
52         f=dfs(v,minf);
53         e[i].f-=f;
54         e[i^1].f+=f;
55         delta-=f;
56         ret+=f;
57         if(ret==delta)return ret;
58     }
59 }
60 return ret;
61 }
62 int maxflow(int _src,int _sink){
63     src=_src;
64     sink=_sink;
65     int ret=0;
66     while(bfs())ret+=dfs(src,inf);
67     return ret;
68 }
69 }dicnic_solver;

```

1.2 KM

```

1  // eg: soj 1013
2  const int N = 105;
3  const int inf = 1000000000;
4  struct KM{
5      int w[N][N],x[N],y[N];
6      int px[N],py[N],sy[N],sk[N],pr[N];
7      int lx,ly,n;
8      void adjust(int v){
9          sy[v]=py[v];
10         if(px[sy[v]]!=-2)adjust(px[sy[v]]);
11     }
12     int solve(int _n,int _w[][N]){
13         n=_n;
14         memcpy(w,_w,sizeof(w));
15         return km();
16     }
17     bool find(int v){
18         for(int i=0;i<n;++i)if(py[i]==-1){
19             if(sk[i]>x[v]+y[i]-w[v][i]){
20                 sk[i]=x[v]+y[i]-w[v][i];
21                 pr[i]=v;
22             }
23             if(x[v]+y[i]==w[v][i]){
24                 py[i]=v;
25                 if(sy[i]==-1){
26                     adjust(i);
27                     return 1;

```

```

28         }
29         if(px[sy[i]]!=-1)continue;
30         px[sy[i]]=i;
31         if(find(sy[i]))return 1;
32     }
33 }
34 return 0;
35 }
36 int km(){
37     int i,j,m;
38     for(i=0;i<n;++i){
39         sy[i]=-1;
40         y[i]=0;
41     }
42     for(i=0;i<n;++i){
43         x[i]=0;
44         for(j=0;j<n;++j){
45             x[i]=max(x[i],w[i][j]);
46         }
47     }
48     bool f;
49     for(i=0;i<n;++i){
50         for(j=0;j<n;++j){
51             px[j]=py[j]=-1;
52             sk[j]=inf;
53         }
54         px[i]=-2;
55         if(find(i))continue;
56         f=0;
57         while(!f){
58             m=inf;
59             for(j=0;j<n;++j)if(py[j]==-1)m=min(m,sk[j]);
60             for(j=0;j<n;++j){
61                 if(px[j]!=-1)x[j]-=m;
62                 if(py[j]!=-1)y[j]+=m;else sk[j]-=m;
63             }
64             for(j=0;j<n;++j)if(py[j]==-1&&sk[j]){
65                 py[j]=pr[j];
66                 if(sy[j]==-1){
67                     adjust(j);
68                     f=1;
69                     break;
70                 }
71                 px[sy[j]]=j;
72                 if(find(sy[j])){
73                     f=1;
74                     break;
75                 }
76             }
77         }
78     }
79     int ans=0;
80     for(i=0;i<n;++i)ans+=w[sy[i]][i];

```

```
81     return ans;
82 }
83 }km_solver;
```

1.3 MixedEulerCircuit

```
1 // eg: soj 1066
2 const int N = 205;
3 int degree[N],n;
4 void init(){
5     dicnic_solver.init();
6     int m,a,b,c;
7     scanf("%d%d",&n,&m);
8     memset(degree,0,sizeof(degree));
9     while(m--){
10         // c=0,a<->b; c=1,a->b
11         scanf("%d%d%d",&a,&b,&c);
12         a--; b--;
13         degree[a]--;
14         degree[b]++;
15         if(!c)dicnic_solver.addEdge(a,b,1);
16     }
17 }
18 bool work(){
19     int ans=0;
20     for(int i=0;i<n;++i)if(degree[i]&1)return 0;
21     for(int i=0;i<n;++i){
22         if(degree[i]<0){
23             dicnic_solver.addEdge(n,i,-degree[i]/2);
24             ans-=degree[i]/2;
25         }else if(degree[i]>0){
26             dicnic_solver.addEdge(i,n+1,degree[i]/2);
27         }
28     }
29     return dicnic_solver.maxflow(n,n+1)>=ans;
30 }
31 void solve(){
32     puts(work()?"possible":"impossible");
33 }
34 int main(){
35     int t;
36     scanf("%d",&t);
37     while(t--){
38         init();
39         solve();
40     }
41     return 0;
42 }
```

2 Tree

2.1 DivideAndConquerTree

```
1 //hdu 4812 D Tree
2 #include <iostream>
3 #include <cstdio>
4 #include <cstring>
5 #include <vector>
6 #pragma comment(linker, "/STACK:102400000,102400000")
7 using namespace std;
8 const int maxn = 1e5 + 10;
9 const int md = 1e6 + 3;
10 int N,K;
11 vector<int > edge[maxn];
12 void add_edge(int from,int to) {
13     edge[from].push_back(to);
14 }
15 void init() {
16     for(int i = 1;i <= N;i ++) edge[i].clear();
17 }
18 int vi[maxn];
19 int vis[maxn];
20 int root;
21 int mi;
22 int son[maxn];
23 int hash[md + 10];
24 int vers[md + 10];
25 int verc;
26 pair<int , int > ans;
27 int fastpow(int x,int y) {
28     int ret = 1 ,mul = x;
29     while(y) {
30         if(y & 1 ) ret = 1LL * mul * ret % md;
31         mul = 1LL * mul * mul % md;
32         y >>= 1;
33     }
34     return ret;
35 }
36 int comm[md + 10];
37 void inv1() {
38     for(int i = 0;i < md;i ++) {
39         comm[i] = fastpow(i,md - 2);
40     }
41 }
42 int inv(int t) {
43     return comm[t];
44 }
45 void getroot(int t,int sz) {
46     vis[t] = true;
47     son[t] = 1;
48     int mx = 0;
```



```

49   for(int i = 0;i < edge[t].size();i ++) {
50       int nxt = edge[t][i];
51       if(!vis[nxt]) {
52           getroot(nxt,sz);
53           son[t] += son[nxt];
54           mx = max(mx,son[nxt]);
55       }
56   }
57   mx = max(mx,sz - son[t]);
58   if(mx <= mi) {
59       root = t;
60       mi = mx;
61   }
62   vis[t] = false;
63 }
64 void dfs(int t,int mul,int ri) {
65     vis[t] = true;
66     //query
67     mul =1LL * mul * vi[t] % md;
68     if(1LL * mul * ri % md == K) {
69         pair<int ,int > tmp = pair<int ,int > (min(root,t),max(root,t));
70         if(tmp < ans) ans = tmp;
71     }
72     int q = 1LL* inv(1LL * mul * ri % md) * K % md;
73     if(vers[q] == verc && hash[q]!= 0 ) {
74         pair<int ,int > tmp = pair<int ,int > (min(t,hash[q]),max(t,hash[q]));
75         if(tmp < ans) ans = tmp;
76     }
77     son[t] = 1;
78     for(int i = 0;i < edge[t].size();i ++) {
79         int nxt = edge[t][i];
80         if(!vis[nxt]) {
81             dfs(nxt,mul,ri);
82             son[t] += son[nxt];
83         }
84     }
85     //set
86     if(vers[mul] != verc ) {
87         vers[mul] = verc;
88         hash[mul] = t;
89     }
90     hash[mul] = min(hash[mul],t);
91     vis[t] = false;
92 }
93 void work(int t,int sz) {
94     mi = sz;
95     getroot(t,sz);
96     // dfs
97     int rt = root;
98     vis[rt] =true;
99     verc ++;
100    for(int i = 0;i < edge[root].size();i ++) {
101        int nxt = edge[rt][i];

```

```

102     if(!vis[nxt]) {
103         dfs(nxt,1,vi[rt] % md);
104     }
105 }
106 for(int i = 0;i < edge[rt].size();i ++) {
107     int nxt = edge[rt][i];
108     if(!vis[nxt]) {
109         work(nxt,son[rt]);
110     }
111 }
112 }
113 int main() {
114     inv1();
115     verc = 0;
116     while(scanf("%d%d",&N,&K) != EOF) {
117         init();
118         for(int i = 1;i <= N;i ++){
119             scanf("%d",&vi[i]);
120         }
121         for(int i = 0;i < N - 1;i ++){
122             int u,v;
123             scanf("%d%d",&u,&v);
124             add_edge(u,v);
125             add_edge(v,u);
126         }
127         memset(vis,0,sizeof(vis));
128         ans = pair<int ,int > (N+1,N+1);
129         work(1,N);
130         if(ans.first == N+1 && ans.second == N + 1) {
131             puts("No solution");
132         } else {
133             printf("%d %d\n",ans.first,ans.second);
134         }
135     }
136 }

```

2.2 LinkTree

```

1  //HDU 3966
2  //operation1 path c1 to c2 plus k
3  //operation2 path c1 to c2 minus k
4  #include <iostream>
5  #include <cstdio>
6  #include <algorithm>
7  #include <vector>
8  #include <cstring>
9  #pragma comment(linker, "/STACK:1024000000,1024000000")
10 using namespace std;
11 #define lc (o<<1)
12 #define rc (o<<1|1)
13 int N,M,P;

```

```
14 const int maxn = 100010;
15 vector<int > edge[maxn];
16 int ai[maxn];
17 void add_edge(int from,int to) {
18     edge[from].push_back(to);
19 }
20 void init() {
21     for(int i = 1;i <= N;i ++) edge[i].clear();
22 }
23 int son[maxn]; // size of children
24 int fa[maxn];
25 int wn[maxn]; //index in segment
26 int wcnt;
27 int vis[maxn];
28 int dep[maxn]; // depth
29 int top[maxn]; // link fa
30 //Tree link
31 void dfs1(int t,int d) {
32     vis[t] = true;
33     dep[t] = d;
34     son[t] = 1;
35     for(int i = 0;i < edge[t].size();i ++) {
36         int nxt = edge[t][i];
37         if(!vis[nxt]) {
38             fa[nxt] = t;
39             dfs1(nxt,d + 1);
40             son[t] += son[nxt];
41         }
42     }
43     vis[t] = false;
44 }
45 void dfs2(int t) {
46     vis[t] = true;
47     wn[t] = wcnt ++;
48     bool first = true;
49     int index = -1;
50     for(int i = 0;i < edge[t].size();i ++) {
51         int nxt = edge[t][i];
52         if(!vis[nxt]) {
53             if(first) {
54                 first =false;
55                 index = nxt;
56             }
57             if(son[nxt] > son[index]) {
58                 index= nxt;
59             }
60         }
61     }
62     if(!first ) {
63         top[index] = top[t];
64         dfs2(index);
65     }
66     for(int i = 0;i < edge[t].size();i ++) {
67         int nxt = edge[t][i];
```

```
67     if(!vis[nxt] && nxt != index) {
68         top[nxt] = nxt;
69         dfs2(nxt);
70     }
71 }
72 }
73 vis[t] = false;
74 }
75 //segment tree
76 int addv[maxn << 2];
77 void add(int o,int l,int r,int y1,int y2,int v) {
78     if(y1 <= l && r <= y2) {
79         addv[o] += v;
80     } else {
81         int m = (l + r) >> 1;
82         if(y1 <= m) add(lc,l,m,y1,y2,v);
83         if(m < y2) add(rc,m+1,r,y1,y2,v);
84     }
85 }
86 void query(int o,int l,int r,int x,int & ans) {
87     if(l == r && r == x) {
88         ans += addv[o];
89     } else {
90         int m = (l + r ) >> 1;
91         ans += addv[o];
92         if(x <= m ) {
93             query(lc,l,m,x,ans);
94         } else {
95             query(rc,m+1,r,x,ans);
96         }
97     }
98 }
99 void init_seg() {
100     memset(addv,0,sizeof(addv));
101 }
102 char buff[5];
103 int main() {
104     while(~scanf("%d%d%d",&N,&M,&P) ) {
105         init();
106         for(int i = 1;i <= N;i ++ ) {
107             scanf("%d",&ai[i]);
108         }
109         for(int i = 0;i < M;i ++ ) {
110             int u,v;
111             scanf("%d%d",&u,&v);
112             add_edge(u,v);
113             add_edge(v,u);
114         }
115         dfs1(1,1);
116         wcnt = 0;
117         top[1] = 1;
118         dfs2(1);
119         init_seg();
```

```

120 while(P --) {
121     scanf("%s",buff);
122     if(buff[0] == 'I' || buff[0] == 'D') {
123         int c1,c2,k;
124         scanf("%d%d%d",&c1,&c2,&k);
125         if(buff[0] == 'D') k = - k;
126         /// query path
127         while(top[c1] != top[c2]) {
128             int f1 = top[c1];
129             int f2 = top[c2];
130             if(dep[f1] < dep[f2]) {
131                 swap(f1,f2);
132                 swap(c1,c2);
133             }
134             add(1,0,N - 1,wn[f1],wn[c1],k);
135             c1 = fa[f1];
136         }
137         if(dep[c1] < dep[c2]) {
138             swap(c1,c2);
139         }
140         add(1,0,N - 1,wn[c2],wn[c1],k);
141     } else if(buff[0] == 'Q') {
142         int d;
143         scanf("%d",&d);
144         int ans = 0;
145         query(1,0,N-1,wn[d],ans);
146         ans += ai[d];
147         printf("%d\n",ans);
148     }
149 }
150 }
151 }

```

2.3 SegmentTree

```

1 //HDU 4578
2 //segment plus mul power sum
3 #include <cstdio>
4 #include <algorithm>
5 using namespace std;
6 #define lc (o<<1)
7 #define rc (o<<1|1)
8 const int maxn = 100010;
9 const int md = 10007;
10 int sumv1[maxn<<2], sumv2[maxn<<2], sumv3[maxn<<2];
11 int addv[maxn<<2], setv[maxn<<2], timv[maxn<<2];
12 void pushdown(int o) {
13     if (setv[o] >= 0) {
14         setv[lc] = setv[rc] = setv[o];
15         addv[lc] = addv[rc] = 0;
16         timv[lc] = timv[rc] = 1;

```

```
17     setv[o] = -1;
18 }
19 if (timv[o] != 1) {
20     addv[lc] *= timv[o];
21     addv[lc] %= md;
22     addv[rc] *= timv[o];
23     addv[rc] %= md;
24     timv[lc] *= timv[o];
25     timv[lc] %= md;
26     timv[rc] *= timv[o];
27     timv[rc] %= md;
28     timv[o] = 1;
29 }
30 if (addv[o] > 0) {
31     addv[lc] += addv[o];
32     addv[lc] %= md;
33     addv[rc] += addv[o];
34     addv[rc] %= md;
35     addv[o] = 0;
36 }
37 }
38 void maintain(int o,int l,int r) {
39     if (l == r) {
40         if (setv[o] != -1) {
41             sumv1[o] = setv[o];
42             setv[o] = -1;
43         }
44         if (timv[o] != 1) {
45             sumv1[o] *= timv[o];
46             timv[o] = 1;
47             sumv1[o] %= md;
48         }
49         if (addv[o] > 0) {
50             sumv1[o] += addv[o];
51             sumv1[o] %= md;
52             addv[o] = 0;
53         }
54         sumv2[o] = sumv1[o] * sumv1[o] % md;
55         sumv3[o] = sumv1[o] * sumv2[o] % md;
56     } else {
57         sumv1[o] = (sumv1[lc] + sumv1[rc]) % md;
58         sumv2[o] = (sumv2[lc] + sumv2[rc]) % md;
59         sumv3[o] = (sumv3[lc] + sumv3[rc]) % md;
60         if (setv[o] != -1) {
61             sumv1[o] = setv[o] * (r - l + 1) % md;
62             sumv2[o] = setv[o] * setv[o] % md * (r - l + 1) % md;
63             sumv3[o] = setv[o] * setv[o] % md * setv[o] % md * (r - l + 1) % md;
64         }
65         if (timv[o] != 1) {
66             sumv1[o] *= timv[o];
67             sumv1[o] %= md;
68             sumv2[o] *= timv[o] * timv[o] % md;
69             sumv2[o] %= md;
```

```

70     sumv3[o] *= timv[o] * timv[o] % md * timv[o] % md;
71     sumv3[o] %= md;
72 }
73 if (addv[o] > 0) {
74     int tmp1 = sumv1[o];
75     sumv1[o] += addv[o] * (r - 1 + 1) % md;
76     sumv1[o] %= md;
77     int tmp2 = sumv2[o];
78     int tmp3 = sumv3[o];
79     sumv2[o] = (tmp2 + 2*tmp1%md * addv[o]%md + addv[o] * addv[o] %md* (r - 1
      +1)%md) % md;
80     sumv3[o] = tmp3 + 3 * tmp2%md * addv[o] % md + 3 * tmp1 % md * addv[o]%md *
      addv[o] % md + addv[o] * addv[o] % md * addv[o] % md * (r - 1 + 1) %md;
81     sumv3[o] %= md;
82 }
83 }
84 }
85 void setq(int o,int l,int r,int y1,int y2,int v) {
86     if (y1 <= 1 && r <= y2) {
87         setv[o] = v;
88         addv[o] = 0;
89         timv[o] = 1;
90     } else {
91         pushdown(o);
92         int m = (l + r) >> 1;
93         if (y1 <= m) setq(lc,l,m,y1,y2,v);
94         else maintain(lc,l,m);
95         if (m < y2) setq(rc,m+1,r,y1,y2,v);
96         else maintain(rc,m+1,r);
97     }
98     maintain(o,l,r);
99 }
100 void addq(int o,int l,int r,int y1,int y2,int v) {
101     if (y1 <= 1 && r <= y2) {
102         addv[o] += v;
103         addv[o] %= md;
104     } else {
105         pushdown(o);
106         int m = (l + r) >> 1;
107         if (y1 <= m ) addq(lc,l,m,y1,y2,v);
108         else maintain(lc,l,m);
109         if (m < y2) addq(rc,m+1,r,y1,y2,v);
110         else maintain(rc,m+1,r);
111     }
112     maintain(o,l,r);
113 }
114 void timq(int o,int l,int r,int y1,int y2,int v) {
115     if (y1 <= 1 && r <= y2) {
116         timv[o] *= v;
117         timv[o] %= md;
118         addv[o] *= v;
119         addv[o] %= md;
120     } else {

```

```

121     pushdown(o);
122     int m = (l + r) >> 1;
123     if (y1 <= m) timq(lc,l,m,y1,y2,v);
124     else maintain(lc,l,m);
125     if (m < y2) timq(rc,m+1,r,y1,y2,v);
126     else maintain(rc,m+1,r);
127 }
128 maintain(o,l,r);
129 }
130 int ans1, ans2, ans3;
131 void query(int o,int l,int r,int y1,int y2,int add,int ti) {
132     if (setv[o] > 0) {
133         add = ti * addv[o] % md + add;
134         ti = ti * timv[o] % md;
135         int len = min(r,y2) - max(y1,l) + 1;
136         int tmp1 = setv[o] * len % md * ti % md;
137         int tmp2 = setv[o] * setv[o] % md * len % md * ti % md * ti % md;
138         int tmp3 = setv[o] * setv[o] % md * setv[o] % md * len % md * ti % md * ti % md *
            ti % md;
139         int _sum1 = tmp1 + add * len % md;
140         _sum1 %= md;
141         int _sum2 = (tmp2 + 2 * tmp1 * add % md + add * add % md * len % md) % md;
142         int _sum3 = (tmp3 + 3 * tmp2 * add % md + 3 * tmp1 * add % md * add % md + len
            * add % md * add % md * add % md) % md;
143         ans1 = (ans1 + _sum1) % md;
144         ans2 = (ans2 + _sum2) % md;
145         ans3 = (ans3 + _sum3) % md;
146         return ;
147     }
148     if (y1 <= l && r <= y2) {
149         int tmp1 = sumv1[o] * ti % md;
150         int tmp2 = sumv2[o] * ti % md * ti % md;
151         int tmp3 = sumv3[o] * ti % md * ti % md * ti % md;
152         int _sum = tmp1 + add * (r - l + 1) % md;
153         int _sum2 = tmp2 + 2 * tmp1 * add % md + add * add % md * (r - l + 1) % md;
154         int _sum3 = tmp3 + 3 * tmp2 * add % md + 3 * tmp1 * add % md * add % md * add %
            md + add * add % md * add % md * (r - l + 1) % md;
155         _sum %= md;
156         _sum2 %= md;
157         _sum3 %= md;
158         ans1 = (ans1 + _sum) % md;
159         ans2 = (ans2 + _sum2) % md;
160         ans3 = (ans3 + _sum3) % md;
161     }
162     else {
163         int m = (l + r) >> 1;
164         if (y1 <= m) query(lc,l,m,y1,y2,(ti * addv[o] % md + add) % md,ti * timv[o] %
            md);
165         if (m < y2) query(rc,m+1,r,y1,y2,(ti * addv[o] % md + add) % md,ti * timv[o] %
            md);
166     }
167 }
168 void init(int o,int l,int r) {

```



```

169   setv[o] = -1;
170   timv[o] = 1;
171   addv[o] = 0;
172   sumv1[o] = sumv2[o] = sumv3[o] = 0;
173   if (l == r) {
174   } else {
175       int m = (l + r) >> 1;
176       init(lc,l,m);
177       init(rc,m+1,r);
178   }
179 }
180 int main() {
181     int N,M;
182     while (scanf("%d%d",&N,&M)==2 && N && M) {
183         init(1,1,N);
184         while (M --) {
185             int cmd,x,y,c;
186             scanf("%d%d%d",&cmd,&x,&y,&c);
187             if(cmd == 1) {
188                 c %= md;
189                 addq(1,1,N,x,y,c);
190             } else if(cmd == 2) {
191                 c %= md;
192                 timq(1,1,N,x,y,c);
193             } else if(cmd == 3) {
194                 c %= md;
195                 setq(1,1,N,x,y,c);
196             } else if(cmd == 4) {
197                 ans1 = ans2 = ans3 = 0;
198                 query(1,1,N,x,y,0,1);
199                 if(c == 1) {
200                     printf("%d\n",ans1);
201                 } else if(c == 2){
202                     printf("%d\n",ans2);
203                 } else if(c == 3) {
204                     printf("%d\n",ans3);
205                 }
206             }
207         }
208     }
209 }

```

2.4 SplayTree

```

1  #include <cstdio>
2  #include <iostream>
3  using namespace std;
4  struct Node {
5      Node* ch[2];
6      int v, s, flip;
7      void maintain() {

```

```

8     s = 1 + ch[0]→s + ch[1]→s;
9 }
10 void pushdown() {
11     if (flip) {
12         flip = 0;
13         swap(ch[0], ch[1]);
14         ch[0]→flip ^= 1;
15         ch[1]→flip ^= 1;
16     }
17 }
18 int cmp(int k) const {
19     int d = k - ch[0]→s;
20     if (d == 1) return -1;
21     return d <= 0 ? 0 : 1;
22 }
23 };
24 Node* null = new Node();
25 void rotate(Node* &o, int d) {
26     Node* k = o→ch[d^1];
27     o→ch[d^1] = k→ch[d];
28     k→ch[d] = o;
29     o→maintain();
30     k→maintain();
31     o = k;
32 }
33 void splay(Node* &o, int k) {
34     o→pushdown();
35     int d = o→cmp(k);
36     if (d == 1) k -= o→ch[0]→s + 1;
37     if (d != -1) {
38         Node* p = o→ch[d];
39         p→pushdown();
40         int d2 = p→cmp(k);
41         int k2 = (d2 == 0) ? k : k - p→ch[0]→s - 1;
42         if (d2 != -1) {
43             splay(p→ch[d2], k2);
44             if (d == d2) {
45                 rotate(o, d^1);
46             } else {
47                 rotate(o→ch[d], d);
48             }
49         }
50         rotate(o, d^1);
51     }
52 }
53 Node* merge(Node* left, Node* right) { // make sure left != null
54     splay(left, left→s);
55     left→ch[1] = right;
56     left→maintain();
57     return left;
58 }
59 void split(Node* o, int k, Node* &left, Node* &right) { // make sure 1 <= k <= o→s
60     splay(o, k);

```

```

61     left = o;
62     right = o->ch[1];
63     o->ch[1] = null;
64     left->maintain();
65 }
66 const int maxn = 300000 + 10;
67 struct SS {
68     int n;
69     Node seq[maxn];
70     Node* root;
71     Node* build(int sz) {
72         if (!sz) return null;
73         Node* L = build(sz/2);
74         Node* o = &seq[++n];
75         o->v = n-1;
76         o->flip = 0;
77         o->ch[0] = L;
78         o->ch[1] = build(sz - sz/2 - 1);
79         o->maintain();
80         return o;
81     }
82     void init(int sz) {
83         n = 0;
84         null->s = null->flip = 0;
85         root = build(sz);
86     }
87     void print(Node *o) {
88         if (o != null) {
89             o->pushdown();
90             print(o->ch[0]);
91             if (o->v) {
92                 if (o->v != 1) putchar(' ');
93                 printf("%d", o->v);
94             }
95             print(o->ch[1]);
96         }
97     }
98 } ss;
99 int n, m, a, b, c;
100 char op[10];
101 int main() {
102     while (scanf("%d%d", &n, &m) == 2 && n != -1 && m != -1) {
103         ss.init(n+1);
104         Node *t1, *t2, *t3;
105         while(m--){
106             scanf("%s", op);
107             if(op[0]=='C'){ // split [a,b], put it after c
108                 scanf("%d%d%d", &a, &b, &c);
109                 split(ss.root, b+1, t1, t2);
110                 split(t1, a, t1, t3);
111                 ss.root = merge(t1, t2);
112                 split(ss.root, c+1, t1, t2);
113                 ss.root = merge(merge(t1, t3), t2);

```

```

114     } else { // flip [a,b]
115         scanf("%d%d",&a,&b);
116         split(ss.root, b+1, t1, t3);
117         split(t1, a, t1, t2);
118         t2->flip ^= 1;
119         ss.root = merge(merge(t1, t2), t3);
120     }
121 }
122 ss.print(ss.root);
123 puts("\n");
124 }
125 }

```

2.5 Treap

```

1 struct Node {
2     Node *ch[2]; // 0-left 1-right
3     int r, v, s; // rank, val, #node
4     Node(int v): v(v) {
5         ch[0] = ch[1] = NULL;
6         r = rand();
7         s = 1;
8     }
9     int cmp(int x) const {
10         if (x == v) return -1;
11         return x < v ? 0 : 1;
12     }
13     void maintain() { // maintain #node
14         s = 1;
15         if (ch[0] != NULL) s += ch[0]->s;
16         if (ch[1] != NULL) s += ch[1]->s;
17     }
18 };
19 void rotate(Node* &o, int d) {
20     Node* k = o->ch[d^1];
21     o->ch[d^1] = k->ch[d];
22     k->ch[d] = o;
23     o->maintain();
24     k->maintain();
25     o = k;
26 }
27 void insert(Node* &o, int x) {
28     if (o == NULL) {
29         o = new Node(x);
30     } else {
31         int d = o->cmp(x);
32         if (d != -1) { // same ele won't be inserted
33             insert(o->ch[d], x);
34             if (o->ch[d]->r > o->r) rotate(o, d^1);
35         }
36     }

```

```
37 o->maintain();
38 }
39 void remove(Node* &o, int x) {
40     if (o == NULL) return ; // ele to be removed not exist
41     int d = o->cmp(x);
42     if (d == -1) {
43         Node* ret = o;
44         if (o->ch[0] != NULL && o->ch[1] != NULL) {
45             int d2 = (o->ch[0]->r > o->ch[1]->r ? 1 : 0);
46             rotate(o, d2);
47             remove(o->ch[d2], x);
48         } else {
49             if (o->ch[0] == NULL) o = o->ch[1];
50             else o = o->ch[0];
51             delete ret;
52         }
53     } else {
54         remove(o->ch[d], x);
55     }
56     if (o) o->maintain();
57 }
58 int find(Node* o, int x) {
59     while (o != NULL) {
60         int d = o->cmp(x);
61         if (d == -1) return 1;
62         else o = o->ch[d];
63     }
64     return 0;
65 }
66 int kth_big(Node* o, int k) {
67     if (o == NULL || k <= 0 || k > o->s) return 0;
68     int s = o->ch[1] == NULL ? 0 : o->ch[1]->s;
69     if (k == s+1) return o->v;
70     else if (k <= s) return kth_big(o->ch[1], k);
71     else return kth_big(o->ch[0], k-s-1);
72 }
73 int kth_small(Node* o, int k) {
74     if (o == NULL || k <= 0 || k > o->s) return 0;
75     int s = o->ch[0] == NULL ? 0 : o->ch[0]->s;
76     if (k == s) return o->v;
77     else if (k < s) return kth_small(o->ch[0], k);
78     else return kth_small(o->ch[1], k-s-1);
79 }
80 void merge(Node* &src, Node* &dest) {
81     if (src == NULL) return ;
82     merge(src->ch[0], dest);
83     merge(src->ch[1], dest);
84     insert(dest, src->v);
85     delete src;
86     src = NULL;
87 }
88 void clear(Node* &o) {
89     if (o == NULL) return ;
```

```

90 clear(o->ch[0]);
91 clear(o->ch[1]);
92 delete o;
93 o = NULL;
94 }

```

3 Geometry

3.1 BasicStructandAlgorithm

```

1 struct Point {
2     double x, y;
3     Point(double x=0, double y=0):x(x),y(y){}
4 };
5
6 typedef Point Vector;
7
8 Vector operator + (const Vector &A, const Vector &B) { return Vector(A.x+B.x,
9     A.y+B.y); }
9 Vector operator - (const Point &A, const Point &B) { return Vector(A.x-B.x,
10     A.y-B.y); }
10 Vector operator * (const Vector &A, double p) { return Vector(A.x*p, A.y*p); }
11 double Dot(const Vector &A, const Vector &B) { return A.x*B.x + A.y*B.y; }
12 double Cross(const Vector &A, const Vector &B) { return A.x*B.y - A.y*B.x; }
13 double Length(const Vector &A) { return sqrt(Dot(A, A)); }
14 Vector Normal(const Vector &A) { double L = Length(A); return Vector(-A.y/L,
15     A.x/L); }
16
16 struct Line {
17     Point P;
18     Vector v;
19     double ang;
20     Line() {}
21     Line(Point P, Vector v):P(P),v(v){ ang = atan2(v.y, v.x); }
22     bool operator < (const Line &L) const {
23         return ang < L.ang;
24     }
25 };
26
27 // if $p$ is on the left side of $L$
28 bool OnLeft(const Line &L, const Point &p) {
29     return Cross(L.v, p-L.P) > 0;
30 }
31
32 // intersection of line $a$ and $b$
33 Point GetLineIntersection(const Line &a, const Line &b) {
34     Vector u = a.P-b.P;
35     double t = Cross(b.v, u) / Cross(a.v, b.v);
36     return a.P+a.v*t;
37 }

```

3.2 PolygonArea

```

1 double PolygonArea(vector<Point> p) {
2     int n = p.size();
3     double area = 0;
4     for(int i = 1; i < n-1; i++)
5         area += Cross(p[i]-p[0], p[i+1]-p[0]);
6     return area/2;
7 }

```

3.3 HalfPlaneIntersection

```

1 const double eps = 1e-6;
2 // intersection of areas (leftside of lines)
3 vector<Point> HalfplaneIntersection(vector<Line> L) {
4     int n = L.size();
5     sort(L.begin(), L.end());
6     int first, last;
7     vector<Point> p(n);
8     vector<Line> q(n);
9     vector<Point> ans;
10    q[first=last=0] = L[0];
11    for(int i = 1; i < n; i++) {
12        while(first < last && !OnLeft(L[i], p[last-1])) last--;
13        while(first < last && !OnLeft(L[i], p[first])) first++;
14        q[++last] = L[i];
15        if(fabs(Cross(q[last].v, q[last-1].v)) < eps) {
16            last--;
17            if(OnLeft(q[last], L[i].P)) q[last] = L[i];
18        }
19        if(first < last) p[last-1] = GetLineIntersection(q[last-1], q[last]);
20    }
21    while(first < last && !OnLeft(q[first], p[last-1])) last--;
22    if(last - first <= 1) return ans;
23    p[last] = GetLineIntersection(q[last], q[first]);
24    for(int i = first; i <= last; i++) ans.push_back(p[i]);
25    return ans;
26 }

```

4 Math

4.1 ChinaRemainderTheory

```

1 // china remainder theory, no matter whether gcd(m[i],m[j])=1
2 LL CRT(const vector<LL>&m, const vector<LL> &b){
3     bool flag = false;
4     LL x, y, i, d, result, a1, m1, a2, m2, Size = m.size();
5     m1 = m[0], a1 = b[0];
6     for(int i = 1; i < Size; i++){

```

```

7   m2 = m[i], a2 = b[i];
8   d = exgcd(m1, m2, x, y);
9   if ((a2 - a1) % d != 0) flag = true;
10  result = (mul_mod(x, (a2 - a1) / d, m2) % m2 + m2) % m2;
11  LL tmp = m1;
12  m1 = m1 / d * m2;
13  a1 = (a1 + mul_mod(tmp, result, m1)) % m1;
14  a1 = (a1 % m1 + m1) % m1;
15  }
16  if (flag) return -1;
17  else return a1;
18  }

```

4.2 Decompose

```

1  // eg: poj 3471
2  const int maxn = 10000000;
3  const int maxp = 700000; // about maxn/log(maxn)
4  struct Factor{ // factor as p^num
5      int p, num;
6  };
7  struct DeComposer {
8      DeComposer() { gen_primes(); }
9      bool vis[maxn+5];
10     int pn, prime[maxp];
11     void sieve() {
12         int m = (int)sqrt(maxn+0.5);
13         memset(vis,0,sizeof(vis));
14         for(int i=2;i<=m;++i)if(!vis[i])
15             for(int j=i*i;j<=maxn;j+=i)vis[j]=1;
16     }
17     void gen_primes() {
18         sieve();
19         pn = 0;
20         for (int i = 2; i <= maxn; ++ i) {
21             if (!vis[i]) prime[pn++] = i;
22         }
23     }
24     int fcn;
25     Factor fc[64]; // x = p1^a1 * p2^a2 * ...
26     int fn, factor[maxp]; // all y satisfy y|x
27     void decompose2(int x,int d){
28         if(d==fcn){
29             factor[fn++] = x;
30         } else {
31             for(int i = 0; i <= fc[d].num; ++ i) {
32                 decompose2(x, d+1);
33                 x /= fc[d].p;
34             }
35         }
36     }

```



```

37 void decompose1(int x) {
38     fcn = 0;
39     for(int i = 0; i < pn && prime[i] * prime[i] <= x; ++ i) if (x % prime[i] == 0)
40     {
41         fc[fcn].p = prime[i];
42         fc[fcn].num = 0;
43         while(x % prime[i] == 0) {
44             fc[fcn].num ++;
45             x /= prime[i];
46         }
47         fcn ++;
48     }
49     if (x > 1) {
50         fc[fcn].p = x;
51         fc[fcn].num = 1;
52         fcn ++;
53     }
54 void decompose(int x){
55     decompose1(x);
56     fn = 0;
57     decompose2(1,0);
58 }
59 } dc_solver;

```

4.3 EulerPhi

```

1 // #x that x<=n && gcd(x,n)==1
2 int euler_phi(int n) {
3     int m = (int)sqrt(n+0.5);
4     int ans = n;
5     for (int i = 2; i <= m; ++ i) if (n % i == 0) {
6         ans = ans / i * (i-1);
7         while (n%i == 0) n /= i;
8     }
9     if (n > 1) ans = ans / n * (n-1);
10    return ans;
11 }
12 int phi[maxn];
13 void phi_table(int n) {
14     for (int i = 2; i <= n; ++ i) phi[i] = 0;
15     phi[1] = 1;
16     for (int i = 2; i <= n; ++ i) {
17         if (!phi[i]) {
18             for (int j = i; j <= n; j += i) {
19                 if (!phi[j]) phi[j] = j;
20                 phi[j] = phi[j] / i * (i-1);
21             }
22         }
23         phi[i] += phi[i-1];
24     }

```

25 }

4.4 ExtendGCD

```

1 // a * x + b * y = d, |x| + |y| get the minimum
2 LL exgcd(LL a, LL b, LL &d, LL &x, LL &y){
3     if (a) { x = 0; y = 1; return a; }
4     else { exgcd(b, a%b, d, y, x); y -= x*(a/b); }
5 }

```

4.5 IntegerInverse

```

1 LL inv1(LL a, LL n) { // a-1 under n
2     LL d, x, y;
3     gcd(a,n,d,x,y);
4     return d == 1 ? (x+n)%n : -1;
5 }
6 LL inv2(LL a, LL p) { // in case that p is a prime
7     return pow_mod(a, p-2, p);
8 }

```

4.6 LineMod

```

1 // ax = b (mod n)
2 // let d = gcd(a,n), use exgcd to solve ax + ny = d
3 // if b|d, then there are #ans=d, otherwise, no solution
4 vector<LL> line_mod(LL a, LL b, LL n) {
5     LL x, y;
6     exgcd(a,n,x,y);
7     vector<LL>ans;
8     ans.clear();
9     if(b%d==0){
10         x%=n; x+=n; x%=n;
11         ans.push_back(x*(b/d)%(n/d));
12         for(LL i=1;i<d;++i){
13             ans.push_back((ans[0]+i*n/d)%n);
14         }
15     }
16     return ans;
17 }

```

4.7 LogMod

```

1 // eg: hdu 2815
2 // d*a(x-c) = b (mod n), make sure that (a,n) = 1 and (d,n) = 1
3 map<LL,LL>f;
4 LL log_mod(LL a, LL b, LL n, LL c, LL d) {

```

```

5  LL m, v, e=1, i, x, y, dd;
6  m = ceil( sqrt(n + 0.5) );
7  f.clear();
8  f[1] = m;
9  for(i = 1; i < m; ++ i) {
10     e = e*a%n;
11     if (!f[e]) f[e] = i;
12 }
13 e = (e*a)%n;
14 for (i = 0; i < m; ++ i) {
15     exgcd(d,n,dd,x,y);
16     x = (x*b%n + n) % n;
17     if (f[x]) {
18         LL num = f[x];
19         return c + i*m + (num==m ? 0 : num);
20     }
21     d = (d*e) % n;
22 }
23 return -1;
24 }
25 // a^x = b (mod n), no restriction
26 LL log_mod(LL a, LL b, LL n) {
27     b%=n;
28     LL c = 0, d = 1, t;
29     while((t=__gcd(a,n))!=1){
30         if(b%t) return -1;
31         c++;
32         n/=t;
33         b/=t;
34         d=d*a/t%n;
35         if(d==b)return c;
36     }
37     return log_mod(a,b,n,c,d);
38 }

```

4.8 Lucas

```

1  // C(n,m) % p, make sure p is prime, p <= 10^5
2  // n = n[k] * p^k + n[k-1] * p^(k-1) + .. + n[0]
3  // m = m[k] * p^k + m[k-1] * p^(k-1) + .. + m[0]
4  // then, C(n,m) = C(n[k],m[k])*C(n[k-1],m[k-1])*...*C(n[0],m[0]) (mod p)
5  // C(n,m) = C(n%p, m%p) * C(n/p, m/p) (mod p)
6  // eg: hdu3037
7  LL Lucas(LL n, LL m, LL p) {
8      LL ret = 1;
9      while(n && m) {
10         LL np = n%p, mp = m%p;
11         if(np < mp) return 0;
12         ret = ret * factorial(np) % p * reverse(factorial(mp), p) % p *
            reverse(factorial(np-mp), p) % p;
13         n /= p;

```

```

14     m /= p;
15 }
16 return ret;
17 }

```

4.9 MillerRabin

```

1 // prime test
2 bool Witness(LL n, LL a) {
3     LL m = n-1, j = 0;
4     while(!(m&1)) m >>= 1, j++;
5     LL ans = pow_mod(a, m, n);
6     while (j --) {
7         LL tmp = mul_mod(ans, ans, n);
8         if (tmp == 1 && ans != 1 && ans != n-1) return 1;
9         ans = tmp;
10    }
11    return ans != 1;
12 }
13 bool Miller_Rabin(LL n) {
14     if (n < 2) return 0;
15     if (n == 2) return 1;
16     if (!(n&1)) return 0;
17     for (int i = 0; i < max_test; ++ i) {
18         ll a = rand() % (n-2) + 2;
19         if (Witness(n,a)) return 0;
20     }
21     return 1;
22 }

```

4.10 MulMod

```

1 // x*y % n
2 LL mul_mod(LL x, LL y, LL n) {
3     LL T = floor(sqrt(n) + 0.5);
4     LL t = T * T - n;
5     LL a = x / T, b = x % T;
6     LL c = y / T, d = y % T;
7     LL e = a * c / T, f = a * c % T;
8     LL v = ((a*d + b*c) % n + e*t) % n;
9     LL g = v / T, h = v % T;
10    LL ret = (((f+g)*t % n + b*d) % n + h*T) % n;
11    return (ret % n + n) % n;
12 }

```

4.11 PollardRho

```

1 // get a factor of n in log(n)
2 LL Pollard_Rho(LL n, LL c=1) {

```

```

3  LL i=1, k=2, x=rand()%(n-1)+1, y=x, d;
4  while(1) {
5      i++;
6      x = (mul_mod(x,x,n)+c)%n;
7      d=__gcd(n,y-x);
8      if(d>1 && d<n) return d;
9      if(y==x) return n;
10     if(i==k){
11         k<<=1;
12         y=x;
13     }
14 }
15 }

```

4.12 PowMod

```

1  // a^x % n
2  LL pow_mod(LL a, LL x, LL n) {
3      LL ret = 1, mul = a;
4      while (x) {
5          if (x&1) ret = mul_mod(ret, mul, n);
6          mul = mul_mod(mul, mul, n);
7          x >>= 1;
8      }
9      return ret;
10 }

```

4.13 PowerMod

```

1  // x^n = a (mod p), make sure that p is prime
2  // let g be a primitive root of p, x = g^y, a = g^m
3  // use log_mod to get m, g^(yn) = g^m (mod p)
4  // thus yn = m (mod p-1), use exgcd to solve and get back
5  vector<int> power_mod(int a, int n, int p) {
6      int g = primitive_root(p);
7      LL m = log_mod(g, a, p);
8      vector<int> ret;
9      if(a==0){
10         ret.push_back(0);
11         return ;
12     }
13     if(m==-1)return ret;
14     LL A=n,B=p-1,C=m,x,y;
15     LL d = exgcd(A,B,x,y);
16     if(C%d!=0)return ret;
17     x=x*(C/d)%B;
18     LL delta=B/d;
19     for(int i=0;i<d;++i){
20         x=((x+delta)%B+B)%B;
21         ret.push_back((int)pow_mod(g,x,p));

```

```

22 }
23 sort(ret.begin(),ret.end());
24 ret.erase(unique(ret.begin(),ret.end()), ret.end());
25 return ret;
26 }

```

4.14 PrimitiveRoot

```

1 // eg: SGU 511
2 struct PR {
3     // make sure that p is prime
4     // if p = 2, solve the prob. without PR
5     int divs[N+5];
6     int primitive_root(const int p) {
7         if (p == 2) return 1;
8         int cnt = 0, m = p-1;
9         for (int i = 2; i*i <= m; ++ i) if (m%i == 0) {
10             divs[cnt++] = i;
11             if (i*i < m) divs[cnt++] = m/i;
12         }
13         int r = 2, j = 0;
14         while (1) {
15             for (j = 0; j < cnt; ++ j) {
16                 if (fastpow(r, divs[j], p) == 1) break;
17             }
18             if (j >= cnt) return r;
19             r ++;
20         }
21         return -1;
22     }
23 } pr_solver;

```

4.15 SquareMod

```

1 // x*x = a (mod n), make sure that n is prime
2 // be careful there is a single sol. when n = 2
3 // otherwise, x and n-x are both okay
4 // eg: ural 1132
5 LL modsqr(LL a, LL n) {
6     LL b, k, i, x;
7     if (n == 2) return a % n;
8     if (pow_mod(a, (n-1)/2, n) == 1) {
9         if (n%4 == 3) {
10             x = pow_mod(a, (n+1)/4, n);
11         }else{
12             for(b=1; pow_mod(b, (n-1)/2, n) == 1; b ++);
13             i = (n-1)/2;
14             k = 0;
15             do {
16                 i/=2;

```

```

17     k/=2;
18     if((pow_mod(a,i,n) * pow_mod(b,k,n)+1) %n == 0) {
19         k += (n-1)/2;
20     }
21     } while(i%2 == 0);
22     x = (pow_mod(a,(i+1)/2,n) * pow_mod(b,k/2,n)) %n;
23     }
24     if(x*2 > n) x = n-x;
25     return x;
26 }
27 return -1;
28 }

```

5 Others

5.1 ExactCover

```

1  // la 2659
2  #include <cstdio>
3  #include <vector>
4  using namespace std;
5  const int MROW = 16*16*16 + 5;
6  const int MCOL = 16*16*4 + 5;
7  const int NODE = 16*16*16*4 + 5;
8  struct DLX {
9      int n, sz;
10     int S[MCOL];
11     int row[NODE], col[NODE];
12     int ansd, ans[MROW];
13     int L[NODE], R[NODE], U[NODE], D[NODE];
14     void init(int n) {
15         this->n = n;
16         for (int i = 0; i <= n; ++ i) {
17             U[i] = D[i] = i;
18             L[i] = i-1; R[i] = i+1;
19             S[i] = 0;
20         }
21         R[n] = 0; L[0] = n;
22         sz = n+1;
23     }
24     void addRow(int r, const vector<int> &columns) {
25         int first = sz;
26         for (int i = 0; i < columns.size(); ++ i) {
27             int c = columns[i];
28             L[sz] = sz-1; R[sz] = sz+1;
29             D[sz] = c; U[sz] = U[c];
30             D[U[c]] = sz; U[c] = sz;
31             row[sz] = r; col[sz] = c;
32             S[c] ++; sz ++;
33         }
34         R[sz-1] = first; L[first] = sz-1;

```

```

35     }
36     #define FOR(i,A,s) for(int i=A[s];i!=s;i=A[i])
37     void remove(int c) {
38         L[R[c]] = L[c]; R[L[c]] = R[c];
39         FOR(i,D,c)
40             FOR(j,R,i) { U[D[j]] = U[j]; D[U[j]] = D[j]; -- S[col[j]]; }
41     }
42     void restore(int c) {
43         FOR(i,U,c)
44             FOR(j,L,i) { ++S[col[j]]; U[D[j]]=j; D[U[j]]=j; }
45         L[R[c]] = c; R[L[c]] = c;
46     }
47     bool dfs(int d) {
48         if (R[0] == 0) {
49             ansd = d;
50             return 1;
51         }
52         int c = R[0];
53         FOR(i,R,0) if(S[i]<S[c]) c=i;
54         remove(c);
55         FOR(i,D,c) {
56             ans[d] = row[i];
57             FOR(j,R,i) remove(col[j]);
58             if(dfs(d+1)) return 1;
59             FOR(j,L,i) restore(col[j]);
60         }
61         restore(c);
62         return 0;
63     }
64     bool solve(vector<int>&v) {
65         v.clear();
66         if (!dfs(0)) return 0;
67         for (int i = 0; i < ansd; ++ i) v.push_back(ans[i]);
68         return 1;
69     }
70 } dlx;
71 char data[18][18];
72 bool input() {
73     for (int i = 0; i < 16; ++ i) {
74         if (scanf("%s",data[i]) == EOF) return 0;
75     }
76     return 1;
77 }
78 enum { SLOT=0, ROW, COL, BLOK };
79 int encode(int i, int j, int k) {
80     return i*256 + j*16 + k + 1;
81 }
82 int block(int i, int j) {
83     return 4*(i/4) + (j/4);
84 }
85 void decode(int x, int &a, int &b, int &c) {
86     x --;
87     c = x % 16; x /= 16;

```



```

88     b = x % 16; x /= 16;
89     a = x;
90 }
91 vector<int>columns;
92 void solve() {
93     dlx.init(16*16*4);
94     for (int i = 0; i < 16; ++ i) {
95         for (int j = 0; j < 16; ++ j) {
96             for (int k = 0; k < 16; ++ k) {
97                 if (data[i][j] == '-' || data[i][j] == k+'A') {
98                     columns.clear();
99                     columns.push_back(encode(SLOT, i, j));
100                     columns.push_back(encode(ROW, i, k));
101                     columns.push_back(encode(COL, j, k));
102                     columns.push_back(encode(BLOK, block(i,j), k));
103                     dlx.addRow(encode(i,j,k), columns);
104                 }
105             }
106         }
107     }
108     columns.clear();
109     dlx.solve(columns);
110     for (int i = 0; i < columns.size(); ++ i) {
111         int r, c, v;
112         decode(columns[i], r, c, v);
113         data[r][c] = char('A' + v);
114     }
115     for (int i = 0; i < 16; ++ i) {
116         printf("%s\n", data[i]);
117     }
118 }
119 int main() {
120     int kcase = 0;
121     while (input()) {
122         if (kcase) puts("");
123         kcase ++;
124         solve();
125     }
126 }

```

5.2 MatrixFastPower

```

1 struct Matrix {
2     int n, a[N][N];
3     Matrix operator * (const Matrix &b) const {
4         Matrix ret; ret.clear();
5         ret.n = n;
6         for (int i = 0; i < n; ++ i) {
7             for (int k = 0; k < n; ++ k) if (a[i][k]) {
8                 for (int j = 0; j < n; ++ j) {
9                     ret.a[i][j] += a[i][k] * b.a[k][j];

```

```

10     ret.a[i][j] %= mod;
11 }
12 }
13 }
14 return ret;
15 }
16 void clear() {
17     memset(a,0,sizeof(a));
18 }
19 };
20 Matrix matrix_one(int n) {
21     Matrix ret; ret.clear();
22     ret.n = n;
23     for (int i = 0; i < n; ++ i) {
24         ret.a[i][i] = 1;
25     }
26     return ret;
27 }
28 Matrix matrix_pow(Matrix x, int n) {
29     Matrix ret = matrix_one(x.n), mul = x;
30     while (n) {
31         if (n&1) ret = ret * mul;
32         mul = mul * mul;
33         n >>= 1;
34     }
35     return ret;
36 }

```

5.3 Polynomial

```

1 // eg: UVALive 4305
2 const int MAXN = 500;
3 const double EPS = 1e-10;
4 inline int sgn(const double &a) { return a > EPS ? 1 : (a < -EPS ? -1 : 0); }
5 struct Polynomial {
6     double data[MAXN];
7     int n;
8     Polynomial() {}
9     Polynomial(int _n) : n(_n) {
10         memset(data, 0, sizeof(data));
11     }
12     Polynomial(double *_data, int _n) {
13         memset(data, 0, sizeof(data));
14         n = _n;
15         for (int i = n; i >= 0; i--) data[i] = *_data[i];
16     }
17     Polynomial operator + (const Polynomial &a) {
18         Polynomial c(max(n, a.n));
19         for (int i = c.n; i >= 0; i--) c.data[i] = data[i] + a.data[i];
20         while (sgn(c.data[c.n]) == 0 && c.n) c.n--;
21         return c;

```

```

22 }
23 Polynomial operator - (const Polynomial &a) {
24     Polynomial c(max(n, a.n));
25     for (int i = c.n; i >= 0; i--) c.data[i] = data[i] - a.data[i];
26     while (sgn(c.data[c.n]) == 0 && c.n) c.n--;
27     return c;
28 }
29 Polynomial operator * (const Polynomial &a) {
30     Polynomial c(n + a.n);
31     for (int i = n; i >= 0; i--) for (int j = a.n; j >= 0; j--) c.data[i + j] +=
        data[i] * a.data[j];
32     return c;
33 }
34 Polynomial operator / (const Polynomial &a) {
35     if (n < a.n) return *this;
36     else {
37         Polynomial c(n - a.n);
38         for (int i = c.n; i >= 0; i--) c.data[i] = data[i + a.n];
39         for (int i = c.n; i >= 0; i--) {
40             c.data[i] /= a.data[a.n];
41             for (int j = i - 1; a.n - i + j >= 0 && j >= 0; j--) c.data[j] -=
                c.data[i] * a.data[a.n - i + j];
42         }
43         return c;
44     }
45 }
46 Polynomial operator % (const Polynomial &a) {
47     Polynomial c = *this - *this / a * a;
48     while (sgn(c.data[c.n]) == 0 && c.n) c.n--;
49     return c;
50 }
51 bool iszero() {
52     return n == 0 && sgn(data[0]) == 0;
53 }
54 bool isconst() {
55     return n > 0;
56 }
57 Polynomial derivative() {
58     Polynomial a(n - 1);
59     for (int i = n - 1; i >= 0; i--) a.data[i] = data[i + 1] * (double)(i + 1);
60     return a;
61 }
62 Polynomial integral() {
63     Polynomial a(n + 1);
64     for (int i = n + 1; i >= 1; i--) a.data[i] = data[i - 1] / (double)i;
65     return a;
66 }
67 void show() {
68     for (int i = n; i >= 0; i--) {
69         printf("%.6f", data[i], i);
70         if (i != 0) printf(" x");
71         if (i != 1 && i != 0) printf(" ^ %d", i);
72         if (i != 0) printf(" + ");

```

```

73     else printf("\n");
74     }
75     }
76 };
77 Polynomial gcd(Polynomial a , Polynomial b) {
78     if (b.iszero()) return a;
79     else return gcd(b, a % b);
80 }

```

6 字符串

6.1 test

```

1  /*****
2  > File Name: test.cpp
3  > Author: HKing
4  > Mail: 1470042308@qq.com
5  > Created Time: 2021年05月25日 星期二 15时20分17秒
6  *****/
7
8  #include <algorithm>
9  #include <cmath>
10 #include <cstring>
11 #include <iostream>
12 #include <map>
13 #include <queue>
14 #include <set>
15 #include <stack>
16 #include <string>
17 #include <vector>
18 #define IOS ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
19 #define endl '\n'
20 #define out(n) cout << n << ' '
21 #define outl(n) cout << n << endl
22 #define sd(n) scanf("%d", &n)
23 #define sdd(n, m) scanf("%d%d", &n, &m)
24 #define sddd(n, m, k) scanf("%d%d%d", &n, &m, &k)
25 #define pd(n) printf("%d\n", (n))
26 #define pdd(n, m) printf("%d %d\n", n, m)
27 #define pddd(n, m, k) printf("%d %d %d\n", n, m, k)
28 #define slld(n) scanf("%lld", &n)
29 #define slldd(n, m) scanf("%lld%lld", &n, &m)
30 #define slddd(n, m, k) scanf("%lld%lld%lld", &n, &m, &k)
31 #define plld(n) printf("%lld\n", n)
32 #define pldd(n, m) printf("%lld %lld\n", n, m)
33 #define plddd(n, m, k) printf("%lld %lld %lld\n", n, m, k)
34 #define sf(n) scanf("%lf", &n)
35 #define sff(n, m) scanf("%lf%lf", &n, &m)
36 #define sfff(n, m, k) scanf("%lf%lf%lf", &n, &m, &k)
37 #define ss(str) scanf("%s", str)
38 #define ps(str) printf("%s", str)

```

```

39 #define x first
40 #define y second
41 #define pi acos(-1)
42 #define de(c, n) \
43     for (int i = 0; i < n; ++i) \
44         cout << c; \
45         cout << endl
46 #define debug(a) cout << #a << '=' << a << endl
47 #define INF_INT 0x3f3f3f3f;
48 #define INF_LONG 4557430888798830399
49 #define mem(ar, num) memset(ar, num, sizeof(ar))
50 #define me(ar) memset(ar, 0, sizeof(ar))
51 #define all(v) v.begin(), v.end()
52 #define max(a, b, c) max(a, max(b, c))
53 #define lowbit(x) (x & (-x))
54 #define gcd(a, b) __gcd(a, b)
55 #define lcm(a, b) a / gcd(a, b) * b
56 #define qpow(a, k, p) \
57     ({ \
58         LL s = 1; \
59         while (k > 0) { \
60             if (k & 1) \
61                 s = s * a % p; \
62             a = a * a % p; \
63             k >>= 1; \
64         } \
65         s; \
66     })
67 #define inv(a, p) \
68     ({ \
69         LL q = p - 2; \
70         qpow(a, q, p); \
71     })
72 #define W(t) \
73     cin >> t; \
74     while (t--)
75 using namespace std;
76 typedef long long LL;
77 typedef unsigned long long ULL;
78 typedef pair<int, int> PII;
79 typedef pair<int, PII> PIII;
80 typedef pair<LL, LL> PLL;
81 typedef pair<LL, PLL> PLLL;
82
83 int main() {
84     IOS;
85
86     return 0;
87 }

```

7 数学

7.1 高斯消元

```
1 // a[N][N]是增广矩阵
2 int gauss() {
3     int c, r;
4     for (c = 0, r = 0; c < n; c++) {
5         int t = r;
6         for (int i = r; i < n; i++) // 找到绝对值最大的行
7             if (fabs(a[i][c]) > fabs(a[t][c]))
8                 t = i;
9
10        if (fabs(a[t][c]) < eps)
11            continue;
12
13        for (int i = c; i <= n; i++)
14            swap(a[t][i], a[r][i]); // 将绝对值最大的行换到最顶端
15        for (int i = n; i >= c; i--)
16            a[r][i] /= a[r][c]; // 将当前行的首位变成1
17        for (int i = r + 1; i < n; i++) // 用当前行将下面所有的列消成0
18            if (fabs(a[i][c]) > eps)
19                for (int j = n; j >= c; j--)
20                    a[i][j] -= a[r][j] * a[i][c];
21
22        r++;
23    }
24
25    if (r < n) {
26        for (int i = r; i < n; i++)
27            if (fabs(a[i][n]) > eps)
28                return 2; // 无解
29        return 1; // 有无穷多组解
30    }
31
32    for (int i = n - 1; i >= 0; i--)
33        for (int j = i + 1; j < n; j++)
34            a[i][n] -= a[i][j] * a[j][n];
35
36    return 0; // 有唯一解
37 }
```

7.2 快速幂

```
1 //求  $m^k \bmod p$ , 时间复杂度  $O(\log k)$ 。
2
3 int qmi(int m, int k, int p) {
4     int res = 1 % p, t = m;
5     while (k) {
6         if (k & 1)
7             res = res * t % p;
```

```
8     t = t * t % p;
9     k >>= 1;
10 }
11 return res;
12 }
```

7.3 欧几里德

```
1 { return b ? gcd(b, a % b) : a; }
2 }
```

7.4 扩展欧几里德算法

```
1 // 求x, y, 使得ax + by = gcd(a, b)
2 int exgcd(int a, int b, int &x, int &y) {
3     if (!b) {
4         x = 1;
5         y = 0;
6         return a;
7     }
8     int d = exgcd(b, a % b, y, x);
9     y -= (a / b) * x;
10    return d;
11 }
```

7.5 试除法分解质因数

```
1 void divide(int x) {
2     for (int i = 2; i <= x / i; i++)
3         if (x % i == 0) {
4             int s = 0;
5             while (x % i == 0)
6                 x /= i, s++;
7             cout << i << ' ' << s << endl;
8         }
9     if (x > 1)
10        cout << x << ' ' << 1 << endl;
11    cout << endl;
12 }
```

7.6 试除法判定质数

```
1 bool is_prime(int x) {
2     if (x < 2)
3         return false;
4     for (int i = 2; i <= x / i; i++)
5         if (x % i == 0)
```

```

6     return false;
7     return true;
8 }

```

7.7 试除法求所有约数

```

1 vector<int> get_divisors(int x) {
2     vector<int> res;
3     for (int i = 1; i <= x / i; i++)
4         if (x % i == 0) {
5             res.push_back(i);
6             if (i != x / i)
7                 res.push_back(x / i);
8         }
9     sort(res.begin(), res.end());
10    return res;
11 }

```

7.8 约数个数和约数之和

```

1 //如果  $N = p_1^{c_1} * p_2^{c_2} * \dots * p_k^{c_k}$ 
2 //约数个数:  $(c_1 + 1) * (c_2 + 1) * \dots * (c_k + 1)$ 
3 //约数之和:  $(p_1^0 + p_1^1 + \dots + p_1^{c_1}) * \dots * (p_k^0 + p_k^1 + \dots + p_k^{c_k})$ 

```

7.9 组合数

7.9.1 递推法求组合数

```

1 //  $c[a][b]$  表示从  $a$  个苹果中选  $b$  个的方案数
2 for (int i = 0; i < N; i++)
3     for (int j = 0; j <= i; j++)
4         if (!j)
5             c[i][j] = 1;
6         else
7             c[i][j] = (c[i - 1][j] + c[i - 1][j - 1]) % mod;

```

7.9.2 通过预处理逆元的方式求组合数

```

1 //首先预处理出所有阶乘取模的余数fact[N]，以及所有阶乘取模的逆元inifact[N]
2 //如果取模的数是质数，可以用费马小定理求逆元
3 int qmi(int a, int k, int p) // 快速幂模板
4 {
5     int res = 1;
6     while (k) {
7         if (k & 1)
8             res = (LL)res * a % p;
9         a = (LL)a * a % p;
10        k >>= 1;

```



```

11 }
12 return res;
13 }
14
15 // 预处理阶乘的余数和阶乘逆元的余数
16 fact[0] = infact[0] = 1;
17 for (int i = 1; i < N; i++) {
18     fact[i] = (LL)fact[i - 1] * i % mod;
19     infact[i] = (LL)infact[i - 1] * qmi(i, mod - 2, mod) % mod;
20 }

```

7.9.3 Lucas 定理求组合数

```

1 //若p是质数, 则对于任意整数  $1 \leq m \leq n$ , 有:
2 //  $C(n, m) = C(n \% p, m \% p) * C(n / p, m / p) \pmod{p}$ 
3
4 int qmi(int a, int k, int p) // 快速幂模板
5 {
6     int res = 1 % p;
7     while (k) {
8         if (k & 1)
9             res = (LL)res * a % p;
10        a = (LL)a * a % p;
11        k >>= 1;
12    }
13    return res;
14 }
15
16 int C(int a, int b, int p) // 通过定理求组合数C(a, b)
17 {
18     if (a < b)
19         return 0;
20
21     LL x = 1, y = 1; // x是分子, y是分母
22     for (int i = a, j = 1; j <= b; i--, j++) {
23         x = (LL)x * i % p;
24         y = (LL)y * j % p;
25     }
26
27     return x * (LL)qmi(y, p - 2, p) % p;
28 }
29
30 int lucas(LL a, LL b, int p) {
31     if (a < p && b < p)
32         return C(a, b, p);
33     return (LL)C(a % p, b % p, p) * lucas(a / p, b / p, p) % p;
34 }

```

7.9.4 分解质因数法求组合数

```
1 //当我们要求出组合数的真实值，而非对某个数的余数时，分解质因数的方式比较好用：
2 // 1. 筛法求出范围内的所有质数
3 // 2. 通过  $C(a, b) = a! / b! / (a - b)!$  这个公式求出每个质因子的次数。  $n!$ 
4 // 中  $p$  的次数是  $n / p + n / p^2 + n / p^3 + \dots$ 
5 // 3. 用高精度乘法将所有质因子相乘
6
7 int primes[N], cnt; // 存储所有质数
8 int sum[N]; // 存储每个质数的次数
9 bool st[N]; // 存储每个数是否已被筛掉
10
11 void get_primes(int n) // 线性筛法求素数
12 {
13     for (int i = 2; i <= n; i++) {
14         if (!st[i])
15             primes[cnt++] = i;
16         for (int j = 0; primes[j] <= n / i; j++) {
17             st[primes[j] * i] = true;
18             if (i % primes[j] == 0)
19                 break;
20         }
21     }
22 }
23
24 int get(int n, int p) // 求  $n!$  中的次数
25 {
26     int res = 0;
27     while (n) {
28         res += n / p;
29         n /= p;
30     }
31     return res;
32 }
33
34 vector<int> mul(vector<int> a, int b) // 高精度乘低精度模板
35 {
36     vector<int> c;
37     int t = 0;
38     for (int i = 0; i < a.size(); i++) {
39         t += a[i] * b;
40         c.push_back(t % 10);
41         t /= 10;
42     }
43
44     while (t) {
45         c.push_back(t % 10);
46         t /= 10;
47     }
48
49     return c;
50 }
51
52 get_primes(a); // 预处理范围内的所有质数
53
```

```
54 for (int i = 0; i < cnt; i++) // 求每个质因数的次数
55 {
56     int p = primes[i];
57     sum[i] = get(a, p) - get(b, p) - get(a - b, p);
58 }
59
60 vector<int> res;
61 res.push_back(1);
62
63 for (int i = 0; i < cnt; i++) // 用高精度乘法将所有质因子相乘
64     for (int j = 0; j < sum[i]; j++)
65         res = mul(res, primes[i]);
```

7.10 欧拉函数

7.10.1 求欧拉函数

```
1 int phi(int x) {
2     int res = x;
3     for (int i = 2; i <= x / i; i++)
4         if (x % i == 0) {
5             res = res / i * (i - 1);
6             while (x % i == 0)
7                 x /= i;
8         }
9     if (x > 1)
10         res = res / x * (x - 1);
11
12     return res;
13 }
```

7.10.2 筛法求欧拉函数

```
1 int primes[N], cnt; // primes[]存储所有素数
2 int euler[N]; // 存储每个数的欧拉函数
3 bool st[N]; // st[x]存储x是否被筛掉
4
5 void get_eulers(int n) {
6     euler[1] = 1;
7     for (int i = 2; i <= n; i++) {
8         if (!st[i]) {
9             primes[cnt++] = i;
10            euler[i] = i - 1;
11        }
12        for (int j = 0; primes[j] <= n / i; j++) {
13            int t = primes[j] * i;
14            st[t] = true;
15            if (i % primes[j] == 0) {
16                euler[t] = euler[i] * primes[j];
17                break;
18            }
19        }
20    }
```

```
18     }
19     euler[t] = euler[i] * (primes[j] - 1);
20 }
21 }
22 }
```

7.11 筛法

7.11.1 朴素筛法

```
1 int primes[N], cnt; // primes[]存储所有素数
2 bool st[N]; // st[x]存储x是否被筛掉
3
4 void get_primes(int n) {
5     for (int i = 2; i <= n; i++) {
6         if (st[i])
7             continue;
8         primes[cnt++] = i;
9         for (int j = i + i; j <= n; j += i)
10             st[j] = true;
11     }
12 }
```

7.11.2 线性筛法

```
1 int primes[N], cnt; // primes[]存储所有素数
2 bool st[N]; // st[x]存储x是否被筛掉
3
4 void get_primes(int n) {
5     for (int i = 2; i <= n; i++) {
6         if (!st[i])
7             primes[cnt++] = i;
8         for (int j = 0; primes[j] <= n / i; j++) {
9             st[primes[j] * i] = true;
10             if (i % primes[j] == 0)
11                 break;
12         }
13     }
14 }
```

8 数据结构

8.1 Trie 树

```
1 int son[N][26], cnt[N], idx;
2 // 0号点既是根节点，又是空节点
3 // son[][]存储树中每个节点的子节点
4 // cnt[]存储以每个节点结尾的单词数量
```

```
5
6 // 插入一个字符串
7 void insert(char *str) {
8     int p = 0;
9     for (int i = 0; str[i]; i++) {
10         int u = str[i] - 'a';
11         if (!son[p][u])
12             son[p][u] = ++idx;
13         p = son[p][u];
14     }
15     cnt[p]++;
16 }
17
18 // 查询字符串出现的次数
19 int query(char *str) {
20     int p = 0;
21     for (int i = 0; str[i]; i++) {
22         int u = str[i] - 'a';
23         if (!son[p][u])
24             return 0;
25         p = son[p][u];
26     }
27     return cnt[p];
28 }
```

8.2 并查集.cpp

```
1 (1) 朴素并查集:
2
3     int p[N]; //存储每个点的祖宗节点
4
5 // 返回x的祖宗节点
6 int find(int x) {
7     if (p[x] != x)
8         p[x] = find(p[x]);
9     return p[x];
10 }
11
12 // 初始化, 假定节点编号是1~n
13 for (int i = 1; i <= n; i++)
14     p[i] = i;
15
16 // 合并a和b所在的两个集合:
17 p[find(a)] = find(b);
18
19 (2) 维护size的并查集:
20
21     int p[N],
22         size[N];
23 // p[]存储每个点的祖宗节点,
24 // size[]只有祖宗节点的有意义, 表示祖宗节点所在集合中的点的数量
```

```

25
26 // 返回x的祖宗节点
27 int find(int x) {
28     if (p[x] != x)
29         p[x] = find(p[x]);
30     return p[x];
31 }
32
33 // 初始化, 假定节点编号是1~n
34 for (int i = 1; i <= n; i++) {
35     p[i] = i;
36     size[i] = 1;
37 }
38
39 // 合并a和b所在的两个集合:
40 size[find(b)] += size[find(a)];
41 p[find(a)] = find(b);
42
43 (3) 维护到祖宗节点距离的并查集:
44
45     int p[N],
46     d[N];
47 // p[]存储每个点的祖宗节点, d[x]存储x到p[x]的距离
48
49 // 返回x的祖宗节点
50 int find(int x) {
51     if (p[x] != x) {
52         int u = find(p[x]);
53         d[x] += d[p[x]];
54         p[x] = u;
55     }
56     return p[x];
57 }
58
59 // 初始化, 假定节点编号是1~n
60 for (int i = 1; i <= n; i++) {
61     p[i] = i;
62     d[i] = 0;
63 }
64
65 // 合并a和b所在的两个集合:
66 p[find(a)] = find(b);
67 d[find(a)] = distance; // 根据具体问题, 初始化find(a)的偏移量

```

9 图论

9.1 test

```

1 /*****
2 > File Name: test.cpp
3 > Author: HKing

```

```

4 > Mail: 1470042308@qq.com
5 > Created Time: 2021年05月25日 星期二 15时21分42秒
6 *****/
7
8 #include <algorithm>
9 #include <cmath>
10 #include <cstring>
11 #include <iostream>
12 #include <map>
13 #include <queue>
14 #include <set>
15 #include <stack>
16 #include <string>
17 #include <vector>
18 #define IOS ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
19 #define endl '\n'
20 #define out(n) cout << n << ' '
21 #define outl(n) cout << n << endl
22 #define sd(n) scanf("%d", &n)
23 #define sdd(n, m) scanf("%d%d", &n, &m)
24 #define sddd(n, m, k) scanf("%d%d%d", &n, &m, &k)
25 #define pd(n) printf("%d\n", (n))
26 #define pdd(n, m) printf("%d %d\n", n, m)
27 #define pddd(n, m, k) printf("%d %d %d\n", n, m, k)
28 #define slld(n) scanf("%lld", &n)
29 #define sldd(n, m) scanf("%lld%lld", &n, &m)
30 #define slddd(n, m, k) scanf("%lld%lld%lld", &n, &m, &k)
31 #define plld(n) printf("%lld\n", n)
32 #define pldd(n, m) printf("%lld %lld\n", n, m)
33 #define plddd(n, m, k) printf("%lld %lld %lld\n", n, m, k)
34 #define sf(n) scanf("%lf", &n)
35 #define sff(n, m) scanf("%lf%lf", &n, &m)
36 #define sfff(n, m, k) scanf("%lf%lf%lf", &n, &m, &k)
37 #define ss(str) scanf("%s", str)
38 #define ps(str) printf("%s", str)
39 #define x first
40 #define y second
41 #define pi acos(-1)
42 #define de(c, n) \
43     for (int i = 0; i < n; ++i) \
44         cout << c; \
45         cout << endl
46 #define debug(a) cout << #a << '=' << a << endl
47 #define INF_INT 0x3f3f3f3f;
48 #define INF_LONG 4557430888798830399
49 #define mem(ar, num) memset(ar, num, sizeof(ar))
50 #define me(ar) memset(ar, 0, sizeof(ar))
51 #define all(v) v.begin(), v.end()
52 #define max(a, b, c) max(a, max(b, c))
53 #define lowbit(x) (x & (-x))
54 #define gcd(a, b) __gcd(a, b)
55 #define lcm(a, b) a / gcd(a, b) * b
56 #define qpow(a, k, p) \

```

```

57  ({ \
58      LL s = 1; \
59      while (k > 0) { \
60          if (k & 1) \
61              s = s * a % p; \
62              a = a * a % p; \
63              k >>= 1; \
64          } \
65          s; \
66      })
67  #define inv(a, p) \
68      ({ \
69          LL q = p - 2; \
70          qpow(a, q, p); \
71      })
72  #define W(t) \
73      cin >> t; \
74      while (t--)
75  using namespace std;
76  typedef long long LL;
77  typedef unsigned long long ULL;
78  typedef pair<int, int> PII;
79  typedef pair<int, PII> PIII;
80  typedef pair<LL, LL> PLL;
81  typedef pair<LL, PLL> PLLL;
82
83  int main() {
84      IOS;
85
86      return 0;
87  }

```

10 动态规划

10.1 test

```

1  /*****
2  > File Name: test.cpp
3  > Author: HKing
4  > Mail: 1470042308@qq.com
5  > Created Time: 2021年05月25日 星期二 15时23分33秒
6  *****/
7
8  #include <algorithm>
9  #include <cmath>
10 #include <cstring>
11 #include <iostream>
12 #include <map>
13 #include <queue>
14 #include <set>
15 #include <stack>

```



```

16 #include <string>
17 #include <vector>
18 #define IOS ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
19 #define endl '\n'
20 #define out(n) cout << n << ' '
21 #define outl(n) cout << n << endl
22 #define sd(n) scanf("%d", &n)
23 #define sdd(n, m) scanf("%d%d", &n, &m)
24 #define sddd(n, m, k) scanf("%d%d%d", &n, &m, &k)
25 #define pd(n) printf("%d\n", (n))
26 #define pdd(n, m) printf("%d %d\n", n, m)
27 #define pddd(n, m, k) printf("%d %d %d\n", n, m, k)
28 #define sld(n) scanf("%lld", &n)
29 #define sldd(n, m) scanf("%lld%lld", &n, &m)
30 #define slddd(n, m, k) scanf("%lld%lld%lld", &n, &m, &k)
31 #define pld(n) printf("%lld\n", n)
32 #define pldd(n, m) printf("%lld %lld\n", n, m)
33 #define plddd(n, m, k) printf("%lld %lld %lld\n", n, m, k)
34 #define sf(n) scanf("%lf", &n)
35 #define sff(n, m) scanf("%lf%lf", &n, &m)
36 #define sfff(n, m, k) scanf("%lf%lf%lf", &n, &m, &k)
37 #define ss(str) scanf("%s", str)
38 #define ps(str) printf("%s", str)
39 #define x first
40 #define y second
41 #define pi acos(-1)
42 #define de(c, n) \
43     for (int i = 0; i < n; ++i) \
44         cout << c; \
45         cout << endl
46 #define debug(a) cout << #a << '=' << a << endl
47 #define INF_INT 0x3f3f3f3f;
48 #define INF_LONG 4557430888798830399
49 #define mem(ar, num) memset(ar, num, sizeof(ar))
50 #define me(ar) memset(ar, 0, sizeof(ar))
51 #define all(v) v.begin(), v.end()
52 #define max(a, b, c) max(a, max(b, c))
53 #define lowbit(x) (x & (-x))
54 #define gcd(a, b) __gcd(a, b)
55 #define lcm(a, b) a / gcd(a, b) * b
56 #define qpow(a, k, p) \
57     ({ \
58         LL s = 1; \
59         while (k > 0) { \
60             if (k & 1) \
61                 s = s * a % p; \
62             a = a * a % p; \
63             k >>= 1; \
64         } \
65         s; \
66     })
67 #define inv(a, p) \
68     ({ \

```

```

69     LL q = p - 2; \
70     qpow(a, q, p); \
71 }
72 #define W(t) \
73     cin >> t; \
74     while (t--)
75 using namespace std;
76 typedef long long LL;
77 typedef unsigned long long ULL;
78 typedef pair<int, int> PII;
79 typedef pair<int, PII> PIII;
80 typedef pair<LL, LL> PLL;
81 typedef pair<LL, PLL> PLLL;
82
83 int main() {
84     IOS;
85
86     return 0;
87 }

```

11 计算几何

11.1 test

```

1  /*****
2  > File Name: test.cpp
3  > Author: HKing
4  > Mail: 1470042308@qq.com
5  > Created Time: 2021年05月25日 星期二 15时23分57秒
6  *****/
7
8  #include <algorithm>
9  #include <cmath>
10 #include <cstring>
11 #include <iostream>
12 #include <map>
13 #include <queue>
14 #include <set>
15 #include <stack>
16 #include <string>
17 #include <vector>
18 #define IOS ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
19 #define endl '\n'
20 #define out(n) cout << n << ' '
21 #define outl(n) cout << n << endl
22 #define sd(n) scanf("%d", &n)
23 #define sdd(n, m) scanf("%d%d", &n, &m)
24 #define sddd(n, m, k) scanf("%d%d%d", &n, &m, &k)
25 #define pd(n) printf("%d\n", (n))
26 #define pdd(n, m) printf("%d %d\n", n, m)
27 #define pddd(n, m, k) printf("%d %d %d\n", n, m, k)

```

```

28 #define sld(n) scanf("%lld", &n)
29 #define sldd(n, m) scanf("%lld%lld", &n, &m)
30 #define slddd(n, m, k) scanf("%lld%lld%lld", &n, &m, &k)
31 #define pld(n) printf("%lld\n", n)
32 #define pldd(n, m) printf("%lld %lld\n", n, m)
33 #define plddd(n, m, k) printf("%lld %lld %lld\n", n, m, k)
34 #define sf(n) scanf("%lf", &n)
35 #define sff(n, m) scanf("%lf%lf", &n, &m)
36 #define sfff(n, m, k) scanf("%lf%lf%lf", &n, &m, &k)
37 #define ss(str) scanf("%s", str)
38 #define ps(str) printf("%s", str)
39 #define x first
40 #define y second
41 #define pi acos(-1)
42 #define de(c, n) \
43     for (int i = 0; i < n; ++i) \
44         cout << c; \
45         cout << endl
46 #define debug(a) cout << #a << '=' << a << endl
47 #define INF_INT 0x3f3f3f3f;
48 #define INF_LONG 4557430888798830399
49 #define mem(ar, num) memset(ar, num, sizeof(ar))
50 #define me(ar) memset(ar, 0, sizeof(ar))
51 #define all(v) v.begin(), v.end()
52 #define max(a, b, c) max(a, max(b, c))
53 #define lowbit(x) (x & (-x))
54 #define gcd(a, b) __gcd(a, b)
55 #define lcm(a, b) a / gcd(a, b) * b
56 #define qpow(a, k, p) \
57     ({ \
58         LL s = 1; \
59         while (k > 0) { \
60             if (k & 1) \
61                 s = s * a % p; \
62             a = a * a % p; \
63             k >>= 1; \
64         } \
65         s; \
66     })
67 #define inv(a, p) \
68     ({ \
69         LL q = p - 2; \
70         qpow(a, q, p); \
71     })
72 #define W(t) \
73     cin >> t; \
74     while (t--)
75 using namespace std;
76 typedef long long LL;
77 typedef unsigned long long ULL;
78 typedef pair<int, int> PII;
79 typedef pair<int, PII> PIII;
80 typedef pair<LL, LL> PLL;

```

```

81 typedef pair<LL, PLL> PLLL;
82
83 int main() {
84     IOS;
85
86     return 0;
87 }

```

12 其他

12.1 test

```

1  /*****
2  > File Name: test.cpp
3  > Author: HKing
4  > Mail: 1470042308@qq.com
5  > Created Time: 2021年05月25日 星期二 15时24分28秒
6  *****/
7
8  #include <algorithm>
9  #include <cmath>
10 #include <cstring>
11 #include <iostream>
12 #include <map>
13 #include <queue>
14 #include <set>
15 #include <stack>
16 #include <string>
17 #include <vector>
18 #define IOS ios::sync_with_stdio(0), cin.tie(0), cout.tie(0)
19 #define endl '\n'
20 #define out(n) cout << n << ' '
21 #define outl(n) cout << n << endl
22 #define sd(n) scanf("%d", &n)
23 #define sdd(n, m) scanf("%d%d", &n, &m)
24 #define sddd(n, m, k) scanf("%d%d%d", &n, &m, &k)
25 #define pd(n) printf("%d\n", (n))
26 #define pdd(n, m) printf("%d %d\n", n, m)
27 #define pddd(n, m, k) printf("%d %d %d\n", n, m, k)
28 #define slld(n) scanf("%lld", &n)
29 #define slldd(n, m) scanf("%lld%lld", &n, &m)
30 #define slddd(n, m, k) scanf("%lld%lld%lld", &n, &m, &k)
31 #define plld(n) printf("%lld\n", n)
32 #define plldd(n, m) printf("%lld %lld\n", n, m)
33 #define plddd(n, m, k) printf("%lld %lld %lld\n", n, m, k)
34 #define sf(n) scanf("%lf", &n)
35 #define sff(n, m) scanf("%lf%lf", &n, &m)
36 #define sfff(n, m, k) scanf("%lf%lf%lf", &n, &m, &k)
37 #define ss(str) scanf("%s", str)
38 #define ps(str) printf("%s", str)
39 #define x first

```

```
40 #define y second
41 #define pi acos(-1)
42 #define de(c, n) \
43     for (int i = 0; i < n; ++i) \
44         cout << c; \
45         cout << endl
46 #define debug(a) cout << #a << '=' << a << endl
47 #define INF_INT 0x3f3f3f3f;
48 #define INF_LONG 4557430888798830399
49 #define mem(ar, num) memset(ar, num, sizeof(ar))
50 #define me(ar) memset(ar, 0, sizeof(ar))
51 #define all(v) v.begin(), v.end()
52 #define max(a, b, c) max(a, max(b, c))
53 #define lowbit(x) (x & (-x))
54 #define gcd(a, b) __gcd(a, b)
55 #define lcm(a, b) a / gcd(a, b) * b
56 #define qpow(a, k, p) \
57     ({ \
58         LL s = 1; \
59         while (k > 0) { \
60             if (k & 1) \
61                 s = s * a % p; \
62             a = a * a % p; \
63             k >>= 1; \
64         } \
65         s; \
66     })
67 #define inv(a, p) \
68     ({ \
69         LL q = p - 2; \
70         qpow(a, q, p); \
71     })
72 #define W(t) \
73     cin >> t; \
74     while (t--)
75 using namespace std;
76 typedef long long LL;
77 typedef unsigned long long ULL;
78 typedef pair<int, int> PII;
79 typedef pair<int, PII> PIII;
80 typedef pair<LL, LL> PLL;
81 typedef pair<LL, PLL> PLLL;
82
83 int main() {
84     IOS;
85
86     return 0;
87 }
```