NCTU_Yggdarsill

Contents

1 Building Environment

1.1 C++11

```
{
    "shell_cmd": "g++ -std=c++11 -Wall \"${file}\" -o \"${file_path}/${file_base_name}\"",
    "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)?:? (.*)$",
    "working_dir": "${file_path}",
    "selector": "source.c, source.c++",

    "variants":
    [
        {
            "name": "Run",
            "shell_cmd": "g++ -std=c++11 -Wall \"${file}\" -o \"${file_path}/${file_base_name}\" && gnome-terminal -e 'bash -c \"${file_path}/${file_base_name}; echo Press ENTER to continue; read line; exit; exec bash\"'"
        }
    ]
}
```

1.2 Default

```cpp
#define F(n) Fi(i,n)
#define Fi(i,n) Fl(i,0,n)
#define Fl(i,l,n) for(int i=(l);i<(int)(n);++i)
#include <bits/stdc++.h>
#include <bits/extc++.h>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/priority_queue.hpp>
using namespace std;
using namespace __gnu_pbds;
const double PI = acos(-1);
main(){
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cout << fixed << setprecision(7) << PI << endl;
}
```

1.3 Preferences

```
1 {
2    "color_scheme": "Packages/Color Scheme - Default/Monokai Bright.tmTheme",
3    // "font_face": "Courier New",  // Uncomment if defaults is proportional.
4    "font_size": 18
5 }
```

1.4 Print File

```
1 import sublime, sublime_plugin
2 import os
3
4 class print_file(sublime_plugin.TextCommand):
5   def run(self, edit):
6     os.system('cat -n "%s" > tmp.print; lpr tmp.print' % self.view.file_name
         ())
7     self.view.show_popup("JIZZ!!")
```

1.5 Vimrc

```
1 set tabstop=4
2 set autoindent
3
4 map <F9> :w<LF>:!g++ -O2 -std=c++11 -o %.out % && echo "----Start----" &&
     ./%.out<LF>
5 imap <F9> <ESC><F9>
```

2 Convolution

2.1 FFT

```
1 #ifndef SUNMOON_FFT
2 #define SUNMOON_FFT
3 #include<vector>
4 #include<complex>
5 #include<algorithm>
6 template<typename T,typename VT=std::vector<std::complex<T> > >
7 struct FFT{
8   const T pi;
9   FFT(const T pi=acos((T)-1)):pi(pi){}
10  inline unsigned int bit_reverse(unsigned int a,int len){
11    a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
12    a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
13    a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
14    a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
15    a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
16    return a>>(32-len);
17  }
18  inline void fft(bool is_inv,VT &in,VT &out,int N){
19    int bitlen=std::__lg(N),num=is_inv?-1:1;
20    for(int i=0;i<N;++i)out[bit_reverse(i,bitlen)]=in[i];
21    for(int step=2;step<=N;step<<=1){
22      const int mh=step>>1;
23      for(int i=0;i<mh;++i){
24        std::complex<T> wi=exp(std::complex<T>(0,i*num*pi/mh));
25        for(int j=i;j<N;j+=step){
26          int k=j+mh;
27          std::complex<T> u=out[j],t=wi*out[k];
28          out[j]=u+t;
29          out[k]=u-t;
30        }
31      }
32    }
33    if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
34  }
35 };
36 #endif
```

3 GNU Black Magic

3.1 Black Magic

```
1 #include<ext/rope>
2 using namespace std;
3 using namespace __gnu_cxx;
4 const int MAXN = 50000 + 10;
5 crope ro,l[MAXN],tmp;
6 char str[200+10];
7 main(){
8   int T,op,p,c,d=0,cnt=1,v;
9   scanf("%d",&T);
10  while(T--){
11    scanf("%d",&op);
12    if(op==1){
13      scanf("%d%s",&p,str);
14      p-=d;
15      ro.insert(p,str);
16      l[cnt++]=ro;
17    }
18    else if(op==2){
19      scanf("%d%d",&p,&c);
20      p-=d,c-=d;
21      ro.erase(p-1,c);
22      l[cnt++]=ro;
23    }
24    else{
```

```
25          scanf("%d%d%d",&v,&p,&c);
26          p-=d,v-=d,c-=d;
27          tmp=l[v].substr(p-1,c);
28          d+=count(tmp.begin(),tmp.end(),'c');
29          cout<<tmp<<endl;
30      }
31    }
32  }
33  #include<bits/extc++.h>
34  using namespace std;
35  using namespace __gnu_pbds;
36  __gnu_pbds::priority_queue<int> h1,h2;
37  typedef tree<int,null_type,less<int>,rb_tree_tag,
        tree_order_statistics_node_update> set_t;
38
39  int main(){
40    printf("heap:\n");
41    for(int i=1;i<=10;i+=2)h1.push(i);
42    for(int i=2;i<=10;i+=2)h2.push(i);
43
44    printf("%d\n",h1.top());
45    printf("%d\n",h2.top());
46    h1.join(h2);
47    printf("%d\n",h1.size());
48    printf("%d\n",h2.size());
49    printf("%d\n",h1.top());
50
51    printf("\ntree:\n");
52    set_t s;
53    for(int i=0;i<5;i++)s.insert(10*i);
54    printf("%d\n",*s.find_by_order(0));
55    printf("%d\n",*s.find_by_order(3));
56    printf("%d\n",s.find_by_order(5)==s.end());
57
58    printf("%d\n",s.order_of_key(0));
59    printf("%d\n",s.order_of_key(30));
60    printf("%d\n",s.order_of_key(35));
61    printf("%d\n",s.order_of_key(100));
62    return 0;
63  }
```

3.2 GNU Bitwise Operation

```
1  int __builtin_ffs (unsigned int x)
2  int __builtin_ffsl (unsigned long)
3  int __builtin_ffsll (unsigned long long)
4  // 返回右起第一個1的位置
5  // Returns one plus the index of the least significant 1-bit of x, or if x is
        zero, returns zero.
6
7  int __builtin_clz (unsigned int x)
8  int __builtin_clzl (unsigned long)
9  int __builtin_clzll (unsigned long long)
10 // 返回左起第一個1之前0的個數
```

```
11 // Returns the number of leading 0-bits in x, starting at the most
        significant bit position. If x is 0, the result is undefined.
12
13 int __builtin_ctz (unsigned int x)
14 int __builtin_ctzl (unsigned long)
15 int __builtin_ctzll (unsigned long long)
16 // 返回右起第一個1之後的0的個數
17 // Returns the number of trailing 0-bits in x, starting at the least
        significant bit position. If x is 0, the result is undefined.
18
19 int __builtin_popcount (unsigned int x)
20 int __builtin_popcountl (unsigned long)
21 int __builtin_popcountll (unsigned long long)
22 // 返回1的個數
23 // Returns the number of 1-bits in x.
24
25 int __builtin_parity (unsigned int x)
26 int __builtin_parityl (unsigned long)
27 int __builtin_parityll (unsigned long long)
28 // 返回1的個數的奇偶性(1的個數 mod 2的值)
29 // Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
```

4 Graph

4.1 BCC

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int MAXN = 10000;
4  vector <int> adja[MAXN];
5  int gcnt, top, timeStamp, dfn[MAXN], low[MAXN], depth[MAXN];
6  pair<int, int> stk[MAXN],ans[MAXN];
7  set <int> group[MAXN];
8  bool cut[MAXN];
9  void BCC(int now, int nextv){
10     int sf, st;
11     group[gcnt].clear();
12     do{
13         sf = stk[top-1].first, st = stk[top-1].second;
14         group[gcnt].insert(sf);
15         group[gcnt].insert(st);
16         --top;
17     }while(sf != now || st != nextv);
18     ++gcnt;
19 }
20 void tarjan(int now, int parent, int d){
21     int child = 0;
22     dfn[now] = low[now] = ++timeStamp, depth[now] = d;
23     for(int i = 0; i < adja[now].size(); i++){
24         int nextv = adja[now][i];
25         if(nextv == parent) continue;
26         if(dfn[nextv] == 0){
```

```
27          stk[top++] = make_pair(now, nextv);
28          tarjan(nextv, now, d+1);
29          low[now] = min(low[now], low[nextv]);
30          ++child;
31          if( (parent != -1 && low[nextv] >= dfn[now]) || (parent == -1 &&
    child >= 2)){
32              cut[now] = true;
33              if(parent != -1) BCC(now, nextv);
34          }
35          if(parent == -1) BCC(now, nextv);
36      }
37      else if(depth[nextv] < depth[now]-1){
38          stk[top++] = make_pair(now, nextv);
39          low[now] = min(low[now], dfn[nextv]);
40      }
41    }
42 }
43 int main(){
44     int n,m,x,y,cnt=0;
45     while(~scanf("%d",&n)){
46         cnt=timeStamp=top=gcnt=0;
47         memset(cut, 0, sizeof(cut));
48         memset(dfn, 0, sizeof(dfn));
49         for(int i=0;i<n;i++)adja[i].clear();
50         for(int i=0;i<n;i++){
51             scanf("%d ",&x);
52             scanf("(%d)",&m);
53             while(m--){
54                 scanf("%d",&y);
55                 adja[x].push_back(y);
56             }
57         }
58         for(int i=0;i<n;i++)
59             if(dfn[i]==0)tarjan(i, -1, 1);
60         for(int i=0;i<gcnt;i++){
61             if(group[i].size()==2){
62                 //critical links
63             }
64         }
65     }
66 }
```

4.2 MST Directed

```
1 #include<cstdio>
2 #include<vector>
3 #include<algorithm>
4 #define N 100100
5 using namespace std;
6 struct edge{
7   edge(){}
8   edge(int _f,int _d):f(_f),d(_d){}
9   int f;
10  int d;
11  bool operator<(const edge &rhs)const{return d<rhs.d;}
12 };
13 struct node{
14   int sz,v,now;
15   node *l,*r;
16   void pull(){sz=1+(l?l->sz:0)+(r?r->sz:0);}
17 }pq[N];
18 int pa[N],sub[N],stk[N],top;
19 bool vis[N],instk[N];
20 vector<edge> rg[N];
21 void init(int n){
22   for(int i=0;i<n;i++){
23     pa[i]=i;
24     sub[i]=0;
25     pq[i].l=pq[i].r=NULL;
26     pq[i].sz=1;
27     pq[i].v=i;
28     pq[i].now=0;
29   }
30 }
31 int find(int x){
32   if(pa[x]==x) return x;
33   int y=find(pa[x]);
34   if(pa[x]!=y) sub[x]+=sub[pa[x]],pa[x]=y;
35   return pa[x];
36 }
37 inline int get_sub(int x){
38   if(x==find(x)) return sub[x];
39   else return sub[x]+sub[pa[x]];
40 }
41 inline int get_cost(const node& a){
42   return rg[a.v][a.now].d-get_sub(a.v);
43 }
44 bool cmp(const node& a,const node& b){
45   return get_cost(a)<get_cost(b);
46 }
47 node* merge(node *a,node *b){
48   if(!a||!b) return a?a:b;
49   if(cmp(*b,*a)) swap(a,b);
50   a->r=merge(a->r,b);
51   if((a->l?a->l->sz:0)<(a->r?a->r->sz:0)) swap(a->l,a->r);
52   a.pull();
53   return a;
54 }
55 int min_cost_arborescence(int r,int n){
56   vis[r]=true;
57   int res=0;
58   for(int i=0;i<n;i++){
59     if(!vis[i]){
60       top=0;
61       int u=i;
62       while(!vis[u]){
63
64       }
65     }
66   }
```

```
67 }
68 int main(){
69   int n,m,r,x,y,w;
70   scanf("%d%d%d",&n,&m,&r);
71   for(int i=0;i<m;i++){
72     scanf("%d%d%d",&x,&y,&w);
73     rg[y].push_back(edge(x,w));
74     sort()
75   }
76 }
```

4.3 SCC

```
 1 #include <cstdlib>
 2 #include <iostream>
 3 #include <vector>
 4 #include <queue>
 5 #define N 300002
 6 using namespace std;
 7 vector<int>go[N],back[N],tree[N];
 8 int hu[N],ST[N],st=0,scc[N],scCo[N],scmx[N];
 9 bool wed[N];
10 int DFS_go(int now){
11     //cout<<now<<" DFS ";
12     wed[now]=true;
13     for(int i=0;i<go[now].size();i++){
14         if(!wed[go[now][i]])
15             DFS_go(go[now][i]);
16     }
17     ST[st++]=now;
18     return 0;
19 }
20 int DFS_back(int now,int id){
21     wed[now]=true;
22     scc[now]=id;
23     int sum=1;
24     if(now==0)sum=0;
25     for(int i=0;i<back[now].size();i++){
26         if(!wed[back[now][i]])
27             sum+=DFS_back(back[now][i],id);
28     }
29     return sum;
30 }
31 int DFS_tree(int now)
32 {
33     if(scmx[now]!=0)return scmx[now];
34     int mx=0,tmp;
35     for(int i=0;i<tree[now].size();i++){
36         tmp=DFS_tree(tree[now][i]);
37         mx=(mx>tmp)? mx:tmp;
38     }
39     scmx[now]=mx+scCo[now];
40     return mx+scCo[now];
41 }
```

```
42 int main(int argc,char *argv[])
43 {
44     ios_base::sync_with_stdio(false);
45     int n,k;
46     char c;
47     cin>>n>>k>>hu[1];
48     go[0].push_back(1);
49     back[1].push_back(0);
50     for(int i=2;i<=n;i++){
51         cin>>hu[i];
52         if(hu[i]>=hu[i-1]){
53             go[i].push_back(i-1);
54             back[i-1].push_back(i);
55         }
56         if(hu[i-1]>=hu[i]){
57             go[i-1].push_back(i);
58             back[i].push_back(i-1);
59         }
60         go[0].push_back(i);
61         back[i].push_back(0);
62     }
63     for(int i=1;i<=n;i++){
64         cin>>c;
65         if(c=='T'){
66             go[i].push_back(0);
67             back[0].push_back(i);
68         }
69     }
70     for(int i=0;i<=n;i++)
71         if(!wed[i])DFS_go(i);
72     //cout<<endl;
73     fill((bool*)wed,(bool*)wed+N,false);
74     int tsc=0;
75     // for(int i=0;i<st;i++)cout<<ST[i]<<" HH ";
76     // cout<<endl;
77     while(st!=0)
78         if(!wed[ST[--st]]){
79             scCo[tsc]=DFS_back(ST[st],tsc);
80             tsc++;
81         }
82     // for(int i=0;i<N;i++)
83     //     while(!back[i].empty())back[i].pop_back();
84     for(int i=0;i<=n;i++)
85         for(int j=0;j<go[i].size();j++){
86             if(scc[i]!=scc[go[i][j]]){
87                 tree[scc[i]].push_back(scc[go[i][j]]);
88             }
89         }
90     // for(int i=0;i<=n;i++)cout<<scc[i]<<" BB ";
91     // cout<<endl;
92     // for(int i=0;i<tsc;i++)cout<<scCo[i]<<" GG ";
93     cout<<DFS_tree(scc[k])<<endl;
94     //system("pause");
95     return 0;
96 }
```

5 Matching

5.1 Bipartite Matching

```cpp
#include<bits/stdc++.h>
#define V 20100
#define inf 0x3f3f3f3f
int mx[V],my[V],dis[V],que[V];
bool vis[V];
vector<int> g[V];
bool DFS(int u){
  vis[u]=true;
  for(int i=0;i<g[u].size();i++){
    int v=my[g[u][i]];
    if(v==-1||!vis[v]&&dis[v]==dis[u]+1&&DFS(v)){
      mx[u]=g[u][i];
      my[g[u][i]]=u;
      return true;
    }
  }
  return false;
}
// n is the size of left hand side
int Hopcroft_Karp(int n){
  int matching=0,qt,qf,sp,i,u,v;
  bool flag=true;
  memset(mx,-1,sizeof(mx));
  memset(my,-1,sizeof(my));
  while(flag){
    flag=false;
    qt=qf=0;
    sp=inf;
    for(i=0;i<n;i++){
      if(mx[i]==-1){
        dis[i]=0;
        que[qt++]=i;
      }
      else dis[i]=inf;
    }
    while(qf<qt){
      u=que[qf++];
      if(dis[u]>=sp) continue;
      for(i=0;i<g[u].size();i++){
        v=my[g[u][i]];
        if(v==-1){
          if(dis[u]+1<sp){
            sp=dis[u]+1;
            flag=true;
          }
        }
        else if(dis[u]+1<dis[v]){
          dis[v]=dis[u]+1;
          que[qt++]=v;
        }
      }
    }
    if(flag){
      memset(vis,0,sizeof(vis));
      for(i=0;i<n;i++){
        if(dis[i]==0&&DFS(i)) matching++;
      }
    }
  }
  return matching;
}
```

5.2 Blossom

```cpp
int V;
bool adj[MAXN][MAXN];
int w[MAXN][MAXN];
int p[MAXN];
int m[MAXN];
int d[MAXN];
int c1[MAXN], c2[MAXN];
int q[MAXN], *qf, *qb;
int pp[MAXN];
int f(int x) {return x == pp[x] ? x : (pp[x] = f(pp[x]));}
void u(int x, int y) {pp[x] = y;}
int v[MAXN];
void path(int r, int x){
    if (r == x) return;
    if (d[x] == 0){
        path(r, p[p[x]]);
        int i = p[x], j = p[p[x]];
        m[i] = j; m[j] = i;
    }
    else if (d[x] == 1){
        path(m[x], c1[x]);
        path(r, c2[x]);
        int i = c1[x], j = c2[x];
        m[i] = j; m[j] = i;
    }
}
int lca(int x, int y, int r){
    int i = f(x), j = f(y);
    while (i != j && v[i] != 2 && v[j] != 1){
        v[i] = 1; v[j] = 2;
        if (i != r) i = f(p[i]);
        if (j != r) j = f(p[j]);
    }
    int b = i, z = j; if (v[j] == 1) swap(b, z);
    for (i = b; i != z; i = f(p[i])) v[i] = -1;
    v[z] = -1;
    return b;
}
void contract_one_side(int x, int y, int b){
    for (int i = f(x); i != b; i = f(p[i])){
        u(i, b);
```

```
42          if (d[i] == 1) c1[i] = x, c2[i] = y, *qb++ = i;
43      }
44 }
45 bool BFS(int r){
46      for (int i=0; i<V; ++i) pp[i] = i;
47      memset(v, -1, sizeof(v));
48      memset(d, -1, sizeof(d));
49      d[r] = 0;
50      qf = qb = q;
51      *qb++ = r;
52      while (qf < qb)
53          for (int x=*qf++, y=0; y<V; ++y)
54              if (adj[x][y] && m[y] != y && f(x) != f(y))
55                  if (d[y] == -1)
56                      if (m[y] == -1){
57                          path(r, x);
58                          m[x] = y; m[y] = x;
59                          return true;
60                      }
61                      else{
62                          p[y] = x; p[m[y]] = y;
63                          d[y] = 1; d[m[y]] = 0;
64                          *qb++ = m[y];
65                      }
66                  else
67                      if (d[f(y)] == 0) {
68                          int b = lca(x, y, r);
69                          contract_one_side(x, y, b);
70                          contract_one_side(y, x, b);
71                      }
72      return false;
73 }
74 int match_result(){
75      int res=0;
76      memset(m,-1,sizeof(m));
77      for(int i=0;i<V;i++){
78          if(m[i]==-1){
79              if(BFS(i))res++;
80              else m[i]=i;
81          }
82      }
83      return res;
84 }
85 int num[10000 + 10],top;
86 int main(){
87      int T,Case=0,n;
88      scanf("%d",&T);
89      while(T--){
90          scanf("%d",&n);
91          V=(1<<n);
92          top=0;
93          for(int i=0;i<V;i++){
94              for(int j=i+1;j<V;j++){
95                  scanf("%d",&w[i][j]);
96                  num[top++]=w[i][j];
97              }
```

```
98          }
99          sort(num,num+top);
100         top = (unique(num,num+top)-num);
101         int l=0,r=top-1,mid;
102         while(r>l){
103             mid=(l+r+1)/2;
104             memset(adj,false,sizeof(adj));
105             for(int i=0;i<V;i++){
106                 for(int j=i+1;j<V;j++){
107                     if(w[i][j]>=num[mid])adj[i][j]=adj[j][i]=true;
108                 }
109             }
110             int res=match_result();
111             if(res==V/2)l=mid;
112             else r=mid-1;
113         }
114         printf("Case %d: %d\n",++Case,num[l]);
115     }
116 }
```

5.3 Dinic

```
1 //Dinic
2 #define V 1000
3 struct edge{
4     edge(){}
5     edge(int a,int b,int c):to(a),cap(b),rev(c){}
6     int to,cap,rev;
7 };
8 vector<edge> g[V];
9 int level[V];
10 int iter[V];
11 void add_edge(int from,int to,int cap){
12     g[from].push_back(edge(to,cap,g[to].size()));
13     g[to].push_back(edge(from,0,g[from].size()-1));
14 }
15 void bfs(int s){
16     memset(level,-1,sizeof(level));
17     queue<int>que;
18     level[s]=0;
19     que.push(s);
20     while(!que.empty()){
21         int v=que.front();
22         que.pop();
23         for(int q=0;q<g[v].size();q++){
24             edge &e=g[v][q];
25             if(e.cap>0&&level[e.to]<0){
26                 level[e.to]=level[v]+1;
27                 que.push(e.to);
28             }
29         }
30     }
31 }
32 int dfs(int v,int t,int f){
```

```
33        if(v==t)return f;
34        for(int &q=iter[v];q<g[v].size();++q){
35            edge &e=g[v][q];
36            if(e.cap>0&&level[v]<level[e.to]){
37                int d=dfs(e.to,t,min(f,e.cap));
38                if(d>0){
39                    e.cap-=d;
40                    g[e.to][e.rev].cap+=d;
41                    return d;
42                }
43            }
44        }
45        return 0;
46 }
47 int max_flow(int s,int t){
48        int flow=0;
49        for(;;){
50            bfs(s);
51            if(level[t]<0)return flow;
52            memset(iter,0,sizeof(iter));
53            int f;
54            while((f=dfs(s,t,1e9))>0)
55                flow+=f;
56        }
57 }
```

5.4　General Weighted Matching

```
 1 #include <iostream>
 2 #include <cstdio>
 3 #include <algorithm>
 4 #include <vector>
 5 using namespace std;
 6
 7 typedef long long s64;
 8
 9 const int INF = 2147483647;
10
11 const int MaxN = 400;
12 const int MaxM = 79800;
13
14 template <class T>
15 inline void tension(T &a, const T &b)
16 {
17   if (b < a)
18     a = b;
19 }
20 template <class T>
21 inline void relax(T &a, const T &b)
22 {
23   if (b > a)
24     a = b;
25 }
26 template <class T>
```

```
27 inline int size(const T &a)
28 {
29   return (int)a.size();
30 }
31
32 inline int getint()
33 {
34   char c;
35   while (c = getchar(), '0' > c || c > '9');
36
37   int res = c - '0';
38   while (c = getchar(), '0' <= c && c <= '9')
39     res = res * 10 + c - '0';
40   return res;
41 }
42
43 const int MaxNX = MaxN + MaxN;
44
45 struct edge
46 {
47   int v, u, w;
48
49   edge(){}
50   edge(const int &_v, const int &_u, const int &_w)
51     : v(_v), u(_u), w(_w){}
52 };
53
54 int n, m;
55 edge mat[MaxNX + 1][MaxNX + 1];
56
57 int n_matches;
58 s64 tot_weight;
59 int mate[MaxNX + 1];
60 int lab[MaxNX + 1];
61
62 int q_n, q[MaxN];
63 int fa[MaxNX + 1], col[MaxNX + 1];
64 int slackv[MaxNX + 1];
65
66 int n_x;
67 int bel[MaxNX + 1], blofrom[MaxNX + 1][MaxN + 1];
68 vector<int> bloch[MaxNX + 1];
69
70 inline int e_delta(const edge &e) // does not work inside blossoms
71 {
72   return lab[e.v] + lab[e.u] - mat[e.v][e.u].w * 2;
73 }
74 inline void update_slackv(int v, int x)
75 {
76   if (!slackv[x] || e_delta(mat[v][x]) < e_delta(mat[slackv[x]][x]))
77     slackv[x] = v;
78 }
79 inline void calc_slackv(int x)
80 {
81   slackv[x] = 0;
82   for (int v = 1; v <= n; v++)
```

```
 83        if (mat[v][x].w > 0 && bel[v] != x && col[bel[v]] == 0)
 84          update_slackv(v, x);
 85    }
 86
 87    inline void q_push(int x)
 88    {
 89      if (x <= n)
 90        q[q_n++] = x;
 91      else
 92      {
 93        for (int i = 0; i < size(bloch[x]); i++)
 94          q_push(bloch[x][i]);
 95      }
 96    }
 97    inline void set_mate(int xv, int xu)
 98    {
 99      mate[xv] = mat[xv][xu].u;
100      if (xv > n)
101      {
102        edge e = mat[xv][xu];
103        int xr = blofrom[xv][e.v];
104        int pr = find(bloch[xv].begin(), bloch[xv].end(), xr) - bloch[xv].begin()
              ;
105        if (pr % 2 == 1)
106        {
107          reverse(bloch[xv].begin() + 1, bloch[xv].end());
108          pr = size(bloch[xv]) - pr;
109        }
110
111        for (int i = 0; i < pr; i++)
112          set_mate(bloch[xv][i], bloch[xv][i ^ 1]);
113        set_mate(xr, xu);
114
115        rotate(bloch[xv].begin(), bloch[xv].begin() + pr, bloch[xv].end());
116      }
117    }
118    inline void set_bel(int x, int b)
119    {
120      bel[x] = b;
121      if (x > n)
122      {
123        for (int i = 0; i < size(bloch[x]); i++)
124          set_bel(bloch[x][i], b);
125      }
126    }
127
128    inline void augment(int xv, int xu)
129    {
130      while (true)
131      {
132        int xnu = bel[mate[xv]];
133        set_mate(xv, xu);
134        if (!xnu)
135          return;
136        set_mate(xnu, bel[fa[xnu]]);
137        xv = bel[fa[xnu]], xu = xnu;
138      }
139    }
140    inline int get_lca(int xv, int xu)
141    {
142      static bool book[MaxNX + 1];
143      for (int x = 1; x <= n_x; x++)
144        book[x] = false;
145      while (xv || xu)
146      {
147        if (xv)
148        {
149          if (book[xv])
150            return xv;
151          book[xv] = true;
152          xv = bel[mate[xv]];
153          if (xv)
154            xv = bel[fa[xv]];
155        }
156        swap(xv, xu);
157      }
158      return 0;
159    }
160
161    inline void add_blossom(int xv, int xa, int xu)
162    {
163      int b = n + 1;
164      while (b <= n_x && bel[b])
165        b++;
166      if (b > n_x)
167        n_x++;
168
169      lab[b] = 0;
170      col[b] = 0;
171
172      mate[b] = mate[xa];
173
174      bloch[b].clear();
175      bloch[b].push_back(xa);
176      for (int x = xv; x != xa; x = bel[fa[bel[mate[x]]]])
177        bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(bel[mate[
            x]]);
178      reverse(bloch[b].begin() + 1, bloch[b].end());
179      for (int x = xu; x != xa; x = bel[fa[bel[mate[x]]]])
180        bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(bel[mate[
            x]]);
181
182      set_bel(b, b);
183
184      for (int x = 1; x <= n_x; x++)
185      {
186        mat[b][x].w = mat[x][b].w = 0;
187        blofrom[b][x] = 0;
188      }
189      for (int i = 0; i < size(bloch[b]); i++)
190      {
191        int xs = bloch[b][i];
```

```
192      for (int x = 1; x <= n_x; x++)
193        if (mat[b][x].w == 0 || e_delta(mat[xs][x]) < e_delta(mat[b][x]))
194          mat[b][x] = mat[xs][x], mat[x][b] = mat[x][xs];
195      for (int x = 1; x <= n_x; x++)
196        if (blofrom[xs][x])
197          blofrom[b][x] = xs;
198    }
199    calc_slackv(b);
200  }
201  inline void expand_blossom1(int b) // lab[b] == 1
202  {
203    for (int i = 0; i < size(bloch[b]); i++)
204      set_bel(bloch[b][i], bloch[b][i]);
205
206    int xr = blofrom[b][mat[b][fa[b]].v];
207    int pr = find(bloch[b].begin(), bloch[b].end(), xr) - bloch[b].begin();
208    if (pr % 2 == 1)
209    {
210      reverse(bloch[b].begin() + 1, bloch[b].end());
211      pr = size(bloch[b]) - pr;
212    }
213
214    for (int i = 0; i < pr; i += 2)
215    {
216      int xs = bloch[b][i], xns = bloch[b][i + 1];
217      fa[xs] = mat[xns][xs].v;
218      col[xs] = 1, col[xns] = 0;
219      slackv[xs] = 0, calc_slackv(xns);
220      q_push(xns);
221    }
222    col[xr] = 1;
223    fa[xr] = fa[b];
224    for (int i = pr + 1; i < size(bloch[b]); i++)
225    {
226      int xs = bloch[b][i];
227      col[xs] = -1;
228      calc_slackv(xs);
229    }
230
231    bel[b] = 0;
232  }
233  inline void expand_blossom_final(int b) // at the final stage
234  {
235    for (int i = 0; i < size(bloch[b]); i++)
236    {
237      if (bloch[b][i] > n && lab[bloch[b][i]] == 0)
238        expand_blossom_final(bloch[b][i]);
239      else
240        set_bel(bloch[b][i], bloch[b][i]);
241    }
242    bel[b] = 0;
243  }
244
245  inline bool on_found_edge(const edge &e)
246  {
247    int xv = bel[e.v], xu = bel[e.u];
248    if (col[xu] == -1)
249    {
250      int nv = bel[mate[xu]];
251      fa[xu] = e.v;
252      col[xu] = 1, col[nv] = 0;
253      slackv[xu] = slackv[nv] = 0;
254      q_push(nv);
255    }
256    else if (col[xu] == 0)
257    {
258      int xa = get_lca(xv, xu);
259      if (!xa)
260      {
261        augment(xv, xu), augment(xu, xv);
262        for (int b = n + 1; b <= n_x; b++)
263          if (bel[b] == b && lab[b] == 0)
264            expand_blossom_final(b);
265        return true;
266      }
267      else
268        add_blossom(xv, xa, xu);
269    }
270    return false;
271  }
272
273  bool match()
274  {
275    for (int x = 1; x <= n_x; x++)
276      col[x] = -1, slackv[x] = 0;
277
278    q_n = 0;
279    for (int x = 1; x <= n_x; x++)
280      if (bel[x] == x && !mate[x])
281        fa[x] = 0, col[x] = 0, slackv[x] = 0, q_push(x);
282    if (q_n == 0)
283      return false;
284
285    while (true)
286    {
287      for (int i = 0; i < q_n; i++)
288      {
289        int v = q[i];
290        for (int u = 1; u <= n; u++)
291          if (mat[v][u].w > 0 && bel[v] != bel[u])
292          {
293            int d = e_delta(mat[v][u]);
294            if (d == 0)
295            {
296              if (on_found_edge(mat[v][u]))
297                return true;
298            }
299            else if (col[bel[u]] == -1 || col[bel[u]] == 0)
300              update_slackv(v, bel[u]);
301          }
302      }
303
```

```
304      int d = INF;
305      for (int v = 1; v <= n; v++)
306        if (col[bel[v]] == 0)
307          tension(d, lab[v]);
308      for (int b = n + 1; b <= n_x; b++)
309        if (bel[b] == b && col[b] == 1)
310          tension(d, lab[b] / 2);
311      for (int x = 1; x <= n_x; x++)
312        if (bel[x] == x && slackv[x])
313          {
314            if (col[x] == -1)
315              tension(d, e_delta(mat[slackv[x]][x]));
316            else if (col[x] == 0)
317              tension(d, e_delta(mat[slackv[x]][x]) / 2);
318          }
319
320      for (int v = 1; v <= n; v++)
321        {
322          if (col[bel[v]] == 0)
323            lab[v] -= d;
324          else if (col[bel[v]] == 1)
325            lab[v] += d;
326        }
327      for (int b = n + 1; b <= n_x; b++)
328        if (bel[b] == b)
329          {
330            if (col[bel[b]] == 0)
331              lab[b] += d * 2;
332            else if (col[bel[b]] == 1)
333              lab[b] -= d * 2;
334          }
335
336      q_n = 0;
337      for (int v = 1; v <= n; v++)
338        if (lab[v] == 0) // all unmatched vertices' labels are zero! cheers!
339          return false;
340      for (int x = 1; x <= n_x; x++)
341        if (bel[x] == x && slackv[x] && bel[slackv[x]] != x && e_delta(mat[
     slackv[x]][x]) == 0)
342          {
343            if (on_found_edge(mat[slackv[x]][x]))
344              return true;
345          }
346      for (int b = n + 1; b <= n_x; b++)
347        if (bel[b] == b && col[b] == 1 && lab[b] == 0)
348          expand_blossom1(b);
349    }
350    return false;
351 }
352
353 void calc_max_weight_match()
354 {
355    for (int v = 1; v <= n; v++)
356      mate[v] = 0;
357
358    n_x = n;
```

```
359    n_matches = 0;
360    tot_weight = 0;
361
362    bel[0] = 0;
363    for (int v = 1; v <= n; v++)
364      bel[v] = v, bloch[v].clear();
365    for (int v = 1; v <= n; v++)
366      for (int u = 1; u <= n; u++)
367        blofrom[v][u] = v == u ? v : 0;
368
369    int w_max = 0;
370    for (int v = 1; v <= n; v++)
371      for (int u = 1; u <= n; u++)
372        relax(w_max, mat[v][u].w);
373    for (int v = 1; v <= n; v++)
374      lab[v] = w_max;
375
376    while (match())
377      n_matches++;
378
379    for (int v = 1; v <= n; v++)
380      if (mate[v] && mate[v] < v)
381        tot_weight += mat[v][mate[v]].w;
382 }
383
384 int main()
385 {
386    n = getint(), m = getint();
387
388    for (int v = 1; v <= n; v++)
389      for (int u = 1; u <= n; u++)
390        mat[v][u] = edge(v, u, 0);
391
392    for (int i = 0; i < m; i++)
393      {
394        int v = getint(), u = getint(), w = getint();
395        mat[v][u].w = mat[u][v].w = w;
396      }
397
398    calc_max_weight_match();
399
400    printf("%lld\n", tot_weight);
401    for (int v = 1; v <= n; v++)
402      printf("%d ", mate[v]);
403    printf("\n");
404
405    return 0;
406 }
```

5.5 KM

```
1 #define MAXN 100
2 #define INF INT_MAX
3 int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[MAXN];
```

```
 4  int px[MAXN],py[MAXN],match_y[MAXN],par[MAXN];
 5  int n;
 6  void adjust(int y){//把增廣路上所有邊反轉
 7    match_y[y]=py[y];
 8    if(px[match_y[y]]!=-2)
 9      adjust(px[match_y[y]]);
10  }
11  bool dfs(int x){//DFS找增廣路
12    for(int y=0;y<n;++y){
13      if(py[y]!=-1)continue;
14      int t=lx[x]+ly[y]-g[x][y];
15      if(t==0){
16        py[y]=x;
17        if(match_y[y]==-1){
18          adjust(y);
19          return 1;
20        }
21        if(px[match_y[y]]!=-1)continue;
22        px[match_y[y]]=y;
23        if(dfs(match_y[y]))return 1;
24      }else if(slack_y[y]>t){
25        slack_y[y]=t;
26        par[y]=x;
27      }
28    }
29    return 0;
30  }
31  inline int km(){
32    memset(ly,0,sizeof(int)*n);
33    memset(match_y,-1,sizeof(int)*n);
34    for(int x=0;x<n;++x){
35      lx[x]=-INF;
36      for(int y=0;y<n;++y){
37        lx[x]=max(lx[x],g[x][y]);
38      }
39    }
40    for(int x=0;x<n;++x){
41      for(int y=0;y<n;++y)slack_y[y]=INF;
42      memset(px,-1,sizeof(int)*n);
43      memset(py,-1,sizeof(int)*n);
44      px[x]=-2;
45      if(dfs(x))continue;
46      bool flag=1;
47      while(flag){
48        int cut=INF;
49        for(int y=0;y<n;++y)
50          if(py[y]==-1&&cut>slack_y[y])cut=slack_y[y];
51        for(int j=0;j<n;++j){
52          if(px[j]!=-1)lx[j]-=cut;
53          if(py[j]!=-1)ly[j]+=cut;
54          else slack_y[j]-=cut;
55        }
56        for(int y=0;y<n;++y){
57          if(py[y]==-1&&slack_y[y]==0){
58            py[y]=par[y];
59            if(match_y[y]==-1){
60              adjust(y);
61              flag=0;
62              break;
63            }
64            px[match_y[y]]=y;
65            if(dfs(match_y[y])){
66              flag=0;
67              break;
68            }
69          }
70        }
71      }
72    }
73    int ans=0;
74    for(int y=0;y<n;++y)if(g[match_y[y]][y]!=-INF)ans+=g[match_y[y]][y];
75    return ans;
76  }
```

5.6　MinCostFlow

```
 1  #define maxnode (1000+10)
 2  #define maxedge (40000+10)
 3  #define INF 1023456789
 4  #include<bits/stdc++.h>
 5  using namespace std;
 6  int node, src, dest, nedge;
 7  int head[maxnode], point[maxedge], nxt[maxedge], flow[maxedge], capa[maxedge
      ], wt[maxedge];
 8  int dist[maxnode], in[maxnode], from[maxnode], mf[maxnode];
 9  //set number of node, source, and destination (one base)
10  void init(int _node, int _src, int _dest) {
11      node = _node;
12      src = _src;
13      dest = _dest;
14      nedge = 0;
15      memset(point, -1, sizeof(point));
16      for (int i = 1; i <= node; i++) head[i] = -1;
17      nedge = 0;
18  }
19  void add_edge(int u, int v, int c1, int w) {
20      point[nedge] = v, capa[nedge] = c1, flow[nedge] = 0, nxt[nedge] = head[u
        ], wt[nedge]=w, head[u] = (nedge++);
21      point[nedge] = u, capa[nedge] = 0, flow[nedge] = 0, nxt[nedge] = head[v],
         wt[nedge]=-w, head[v] = (nedge++);
22  }
23  int sp(int &left){
24    for(int i=1;i<=node;i++) dist[i]=INF;
25    queue<int> que;
26    que.push(src);
27    in[src]=1;
28    mf[src]=left;
29    dist[src]=0;
30    while(!que.empty()){
31      int u=que.front();
```

```
32      que.pop();
33      in[u]=0;
34      if(dist[u]>=dist[dest]) continue;
35      for(int v=head[u];v!=-1;v=nxt[v]){
36        if(flow[v]==capa[v]) continue;
37        if(dist[u]+wt[v]<dist[point[v]]){
38          dist[point[v]]=dist[u]+wt[v];
39          from[point[v]]=v;
40          mf[point[v]]=min(mf[u],capa[v]-flow[v]);
41          if(!in[point[v]]){
42            in[point[v]]=1;
43            que.push(point[v]);
44          }
45        }
46      }
47    }
48    left-=mf[dest];
49    if(dist[dest]<INF){
50      for(int u=dest;u!=src;u=point[from[u]^1]){
51        flow[from[u]]+=mf[dest];
52        flow[from[u]^1]-=mf[dest];
53      }
54    }
55    return dist[dest];
56 }
57 int min_cost_flow(){
58    int res=0,tmp,maxflow=2;
59    while(maxflow&&(tmp=sp(maxflow))<INF) res+=tmp;
60    return res;
61 }
62 int main(){
63    int n,m,x,y,z;
64    while(scanf("%d%d",&n,&m)==2){
65      init(n,1,n);
66      for(int i=0;i<m;i++){
67        scanf("%d%d%d",&x,&y,&z);
68        add_edge(x,y,1,z);
69        add_edge(y,x,1,z); //undirected
70      }
71      printf("%d\n",min_cost_flow());
72    }
73    return 0;
74 }
```

5.7 Stable Marriage

```
 1 #define F(n) Fi(i, n)
 2 #define Fi(i, n) Fl(i, 0, n)
 3 #define Fl(i, l, n) for(int i = l ; i < n ; ++i)
 4 #include <bits/stdc++.h>
 5 using namespace std;
 6 int D, quota[205], weight[205][5];
 7 int S, scoretodep[12005][205], score[5];
 8 int P, prefer[12005][85], iter[12005];
```

```
 9 int ans[12005];
10 typedef pair<int, int> PII;
11 map<int, int> samescore[205];
12 typedef priority_queue<PII, vector<PII>, greater<PII>> QQQ;
13 QQQ pri[205];
14 void check(int d) {
15    PII t = pri[d].top();
16    int v;
17    if (pri[d].size() - samescore[d][t.first] + 1 <= quota[d]) return;
18    while (pri[d].top().first == t.first) {
19      v = pri[d].top().second;
20      ans[v] = -1;
21      --samescore[d][t.first];
22      pri[d].pop();
23    }
24 }
25 void push(int s, int d) {
26    if (pri[d].size() < quota[d]) {
27      pri[d].push(PII(scoretodep[s][d], s));
28      ans[s] = d;
29      ++samescore[s][scoretodep[s][d]];
30    } else if (scoretodep[s][d] >= pri[d].top().first) {
31      pri[d].push(PII(scoretodep[s][d], s));
32      ans[s] = d;
33      ++samescore[s][scoretodep[s][d]];
34      check(d);
35    }
36 }
37 void f() {
38    int over;
39    while (true) {
40      over = 1;
41      Fi (q, S) {
42        if (ans[q] != -1 || iter[q] >= P) continue;
43        push(q, prefer[q][iter[q]++]);
44        over = 0;
45      }
46      if (over) break;
47    }
48 }
49 main() {
50    ios::sync_with_stdio(false);
51    cin.tie(NULL);
52    int sadmit, stof, dexceed, dfew;
53    while (cin >> D, D) { // Beware of the input format or judge may troll us.
54      sadmit = stof = dexceed = dfew = 0;
55      memset(iter, 0, sizeof(iter));
56      memset(ans, 0, sizeof(ans));
57      Fi (q, 205) {
58        pri[q] = QQQ();
59        samescore[q].clear();
60      }
61      cin >> S >> P;
62      Fi (q, D) {
63        cin >> quota[q];
64        Fi (w, 5) cin >> weight[q][w];
```

```
65      }
66    Fi (q, S) {
67      Fi (w, 5) cin >> score[w];
68      Fi (w, D) {
69        scoretodep[q][w] = 0;
70        F (5) scoretodep[q][w] += weight[w][i] * score[i];
71      }
72    }
73    Fi (q, S) Fi (w, P) {
74      cin >> prefer[q][w];
75      --prefer[q][w];
76    }
77    f();
78    Fi (q, D) sadmit += pri[q].size();
79    Fi (q, S) if (ans[q] == prefer[q][0]) ++stof;
80    Fi (q, D) if (pri[q].size() > quota[q]) ++dexceed;
81    Fi (q, D) if (pri[q].size() < quota[q]) ++dfew;
82    cout << sadmit << ' ' << stof << ' ' << dexceed << ' ' << dfew << '\n';
83  }
84 }
```

6 Mathematics

6.1 Extgcd

```
1 long long extgcd(long long a,long long b,long long &x,long long &y){
2    long long d=a;
3    if(b!=0){
4        d=extgcd(b,a%b,y,x);
5        y-=(a/b)*x;
6    }
7    else x=1,y=0;
8    return d;
9 }
10 int main(){
11    int T;
12    long long a,b,m,GCD,x,y;
13    while(~scanf("%d",&T))
14        while(T--){
15            scanf("%lld%lld%lld",&m,&a,&b);
16            GCD=extgcd(a,m,x,y);
17            if(GCD!=1)printf("No inverse, gcd(a,m)=%lld\n",GCD);
18            else{
19                b=((-b*x)%m+m)%m;
20                printf("%lld %lld\n",(x%m+m)%m,b);
21            }
22        }
23 }
```

6.2 Miller-Rabin

```
1 inline long long mod_mul(long long a,long long b,long long m){
2    a%=m,b%=m;
3    long long y=(long long)((double)a*b/m+0.5);/* fast for m < 2^58 */
4    long long r=(a*b-y*m)%m;
5    return r<0?r+m:r;
6 }
7 template<typename T>
8 inline T pow(T a,T b,T mod){//a^b%mod
9    T ans=1;
10    for(;b;a=mod_mul(a,a,mod),b>>=1)
11        if(b&1)ans=mod_mul(ans,a,mod);
12    return ans;
13 }
14 int sprp[3]={2,7,61};//int範圍可解
15 int llsprp[7]={2,325,9375,28178,450775,9780504,1795265022};//至少unsigned
        long long範圍
16 template<typename T>
17 inline bool isprime(T n,int *sprp,int num){
18    if(n==2)return 1;
19    if(n<2||n%2==0)return 0;
20    int t=0;
21    T u=n-1;
22    for(;u%2==0;++t)u>>=1;
23    for(int i=0;i<num;++i){
24        T a=sprp[i]%n;
25        if(a==0||a==1||a==n-1)continue;
26        T x=pow(a,u,n);
27        if(x==1||x==n-1)continue;
28        for(int j=0;j<t;++j){
29            x=mod_mul(x,x,n);
30            if(x==1)return 0;
31            if(x==n-1)break;
32        }
33        if(x==n-1)continue;
34        return 0;
35    }
36    return 1;
37 }
```

7 String

7.1 AC Automaton

```
1 #ifndef SUNMOON_AHO_CORASICK_AUTOMATON
2 #define SUNMOON_AHO_CORASICK_AUTOMATON
3 #include<queue>
4 #include<vector>
5 template<char L='a',char R='z'>
6 class ac_automaton{
```

```cpp
 7    private:
 8      struct joe{
 9        int next[R-L+1],fail,efl,ed,cnt_dp,vis;
10        joe():ed(0),cnt_dp(0),vis(0){
11          for(int i=0;i<=R-L;++i)next[i]=0;
12        }
13      };
14    public:
15      std::vector<joe> S;
16      std::vector<int> q;
17      int qs,qe,vt;
18      ac_automaton():S(1),qs(0),qe(0),vt(0){}
19      inline void clear(){
20        q.clear();
21        S.resize(1);
22        for(int i=0;i<=R-L;++i)S[0].next[i]=0;
23        S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
24      }
25      inline void insert(const char *s){
26        int o=0;
27        for(int i=0,id;s[i];++i){
28          id=s[i]-L;
29          if(!S[o].next[id]){
30            S.push_back(joe());
31            S[o].next[id]=S.size()-1;
32          }
33          o=S[o].next[id];
34        }
35        ++S[o].ed;
36      }
37      inline void build_fail(){
38        S[0].fail=S[0].efl=-1;
39        q.clear();
40        q.push_back(0);
41        ++qe;
42        while(qs!=qe){
43          int pa=q[qs++],id,t;
44          for(int i=0;i<=R-L;++i){
45            t=S[pa].next[i];
46            if(!t)continue;
47            id=S[pa].fail;
48            while(~id&&!S[id].next[i])id=S[id].fail;
49            S[t].fail=~id?S[id].next[i]:0;
50            S[t].efl=S[S[t].fail].ed?S[t].fail:S[S[t].fail].efl;
51            q.push_back(t);
52            ++qe;
53          }
54        }
55      }
56      /*DP出每個前綴在字串s出現的次數並傳回所有字串被s匹配成功的次數O(N+M)*/
57      inline int match_0(const char *s){
58        int ans=0,id,p=0,i;
59        for(i=0;s[i];++i){
60          id=s[i]-L;
61          while(!S[p].next[id]&&p)p=S[p].fail;
62          if(!S[p].next[id])continue;
```

```cpp
63          p=S[p].next[id];
64          ++S[p].cnt_dp;/*匹配成功則它所有後綴都可以被匹配(DP計算)*/
65        }
66        for(i=qe-1;i>=0;--i){
67          ans+=S[q[i]].cnt_dp*S[q[i]].ed;
68          if(~S[q[i]].fail)S[S[q[i]].fail].cnt_dp+=S[q[i]].cnt_dp;
69        }
70        return ans;
71      }
72      /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)*/
73      inline int match_1(const char *s)const{
74        int ans=0,id,p=0,t;
75        for(int i=0;s[i];++i){
76          id=s[i]-L;
77          while(!S[p].next[id]&&p)p=S[p].fail;
78          if(!S[p].next[id])continue;
79          p=S[p].next[id];
80          if(S[p].ed)ans+=S[p].ed;
81          for(t=S[p].efl;~t;t=S[t].efl){
82            ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
83          }
84        }
85        return ans;
86      }
87      /*枚舉(s的子字串∩A)的所有相異字串各恰一次並傳回次數O(N*M^(1/3))*/
88      inline int match_2(const char *s){
89        int ans=0,id,p=0,t;
90        ++vt;
91        /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成false
92        這種利用vt的方法可以O(1)歸零vis陣列*/
93        for(int i=0;s[i];++i){
94          id=s[i]-L;
95          while(!S[p].next[id]&&p)p=S[p].fail;
96          if(!S[p].next[id])continue;
97          p=S[p].next[id];
98          if(S[p].ed&&S[p].vis!=vt){
99            S[p].vis=vt;
100            ans+=S[p].ed;
101          }
102          for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
103            S[t].vis=vt;
104            ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
105          }
106        }
107        return ans;
108      }
109      /*把AC自動機變成真的自動機*/
110      inline void evolution(){
111        for(qs=1;qs!=qe;){
112          int p=q[qs++];
113          for(int i=0;i<=R-L;++i)
114            if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];
115        }
116      }
117    };
118    #endif
```

7.2 Suffix Array

```
1  //should initialize s and n first
2  #define N 301000
3  using namespace std;
4  char s[N]; //string=s,suffix array=sar,longest common prefix=lcp
5  int rk[2][N],id[2][N];
6  int n,p;
7  int cnt[N];
8  int len[N],od[N],sar[N];
9  inline int sr(int i,int t){ //rank of shifted position
10   return i+t<n?rk[p][i+t]:-1;
11 }
12 inline bool check_same(int i,int j,int t){
13   return rk[p][i]==rk[p][j]&&sr(i,t)==sr(j,t);
14 }
15 bool cmp(int i,int j){
16   return s[i]<s[j];
17 }
18 void sa(){ //length of array s
19   int i,t,now,pre;
20   memset(cnt,0,sizeof(cnt));
21   for(i=0;i<n;i++){
22     id[p][i]=i;
23     rk[p][i]=s[i];
24     cnt[s[i]]++;
25   }
26   for(i=1;i<128;i++) cnt[i]+=cnt[i-1];
27   sort(id[p],id[p]+n,cmp);
28   for(t=1;t<n;t<<=1){
29       //least significant bit is already sorted
30     for(i=n-1;i>=0;i--){
31           now=id[p][i]-t;
32       if(now>=0) id[p^1][--cnt[rk[p][now]]]=now;
33     }
34     for(i=n-t;i<n;i++){
35           id[p^1][--cnt[rk[p][i]]]=i;
36     }
37     memset(cnt,0,sizeof(cnt));
38     now=id[p^1][0];
39     rk[p^1][now]=0;
40     cnt[0]++;
41     for(i=1;i<n;i++){
42       pre=now;
43       now=id[p^1][i];
44       if(check_same(pre,now,t)){
45         rk[p^1][now]=rk[p^1][pre];
46       }
47       else{
48         rk[p^1][now]=rk[p^1][pre]+1;
49       }
50       cnt[rk[p^1][now]]++;
51     }
52     p^=1;
53     if(rk[p][now]==n-1) break;
54     for(i=1;i<n;i++) cnt[i]+=cnt[i-1];
```

```
55   }
56   memcpy(sar,id[p],sizeof(sar));
57 }
58 void lcp(){
59     int i,l,pre;
60     for(i=0;i<n;i++) od[sar[i]]=i;
61     for(i=0;i<n;i++){
62         if(i) l=len[od[i-1]]?len[od[i-1]]-1:0;
63         else l=0;
64         if(od[i]){
65             pre=sar[od[i]-1];
66             while(pre+l<n&&i+l<n&&s[pre+l]==s[i+l]) l++;
67             len[od[i]]=l;
68         }
69         else len[0]=0;
70     }
71 }
```

7.3 Suffix Automaton

```
1  #include<bits/stdc++.h>
2  #define C 96
3  #define N 200100
4  using namespace std;
5  struct SAM{
6    struct node{
7      node *nxt[C],*pre;
8      int len;
9      vector<int> pos;
10   };
11   node mem[N*2],*root,*ed;
12   int top;
13   SAM(){
14     top = 0;
15     root = new_node(0);
16     ed = root;
17   }
18   node *new_node(int l){
19     for(int i=0;i<C;i++) mem[top].nxt[i]=NULL;
20     mem[top].pre=NULL;
21     mem[top].len=l;
22     mem[top].pos.clear();
23     return mem+(top++);
24   }
25   node *split_node(int l,node *p){
26     for(int i=0;i<C;i++) mem[top].nxt[i]=p->nxt[i];
27     mem[top].pre = p->pre;
28     mem[top].len = l;
29     mem[top].pos.assign()
30     p->pre = mem+top;
31     return mem+(top++);
32   }
33   void push(char c){
34     node *nw = new_node(ed->len+1),*ptr=ed->pre;
```

```
35      ed->nxt[c] = nw;
36      nw->pos.push_back(ed->len);
37      for(;ptr;ptr=ptr->pre){
38        if(ptr->nxt[c]){
39          if(ptr->nxt[c]->len==ptr->len+1){
40            nw->pre = ptr->nxt[c];
41          }
42          else{
43            node *tmp=ptr->nxt[c];
44            nw->pre = split_node(ptr->len+1,tmp);
45            while(ptr && ptr->nxt[c]==tmp){
46              ptr->nxt[c] = nw->pre;
47              ptr = ptr->pre;
48            }
49          }
50          break;
51        }
52        else{
53          ptr->nxt[c] = nw;
54        }
55      }
56      if(!nw->pre) nw->pre = root;
57      ed = ed->nxt[c];
58    }
59    void init(){
60      while(top){
61        mem[--top].pos.clear();
62      }
63      root = new_node(0);
64      ed = root;
65    }
66    void push(char *s){
67      for(int i=0;s[i];i++) push(s[i]-32);
68    }
69    long long count(){
70      long long ans=0;
71      for(int i=1;i<top;i++){
72        ans+=mem[i].len-mem[i].pre->len;
73      }
74      return ans;
75    }
76  }sam;
77  char S[N];
78  int main(){
79    int T;
80    scanf("%d",&T);
81    while(T--){
82      scanf("%s",S);
83      sam.build(S);
84      printf("%lld\n",sam.count());
85    }
86    return 0;
87  }
```

7.4 Z Algorithm

```
1  void Zalg(char *s, int *z, int n) {
2    z[0]=n;
3    for(int L=0, R=0, i=1; i<n; i++) {
4      if(i<=R && z[i-L]<=R-i) z[i]=z[i-L];
5      else {
6        L=i;
7        if(i>R) R=i;
8        while(R<n && s[R-L]==s[R]) R++;
9        z[i]=(R--)-L;
10     }
11   }
12 }
```

8 Struct

8.1 Splay Tree

```
1  #include<cstdio>
2  #include<string>
3  using namespace std;
4  struct node{
5    node *ch[2],*par;
6    long long sum;
7    int val,sz,add;
8    node(){}
9    node(int x):par(NULL),val(x),sum(x),add(0),sz(1){ch[0]=ch[1]=NULL;}
10   bool dir(){return !par||par->ch[1]==this;}
11   void pull();
12   void push();
13 }pool[100100];
14 inline long long qsum(node *x){
15   return x?1LL*x->add*x->sz+x->sum:0;
16 }
17 inline int qsz(node *x){return x?x->sz:0;}
18 void node::pull(){
19   sum=val+qsum(ch[0])+qsum(ch[1]);
20   sz=1+qsz(ch[0])+qsz(ch[1]);
21 }
22 void node::push(){
23   if(add){
24     val+=add;
25     sum+=add*sz;
26     if(ch[0]) ch[0]->add+=add;
27     if(ch[1]) ch[1]->add+=add;
28     add=0;
29   }
30 }
31 inline void con(node *p,node *c,bool d){
32   p->ch[d]=c;
```

```cpp
33      if(c) c->par=p;
34  }
35  void splay(node *x){
36      x->push();
37      while(x->par){
38          node *p=x->par,*g=p->par;
39          bool d=x->dir(),pd=p->dir();;
40          con(p,x->ch[d^1],d);
41          con(x,p,d^1);
42          if(g){
43              if(g->par) con(g->par,x,g->dir());
44              else x->par=NULL;
45              if(d^pd){
46                  con(g,x->ch[d],pd);
47                  con(x,g,pd^1);
48              }
49              else{
50                  con(g,p->ch[pd^1],pd);
51                  con(p,g,pd^1);
52              }
53              g->pull();
54          }
55          else x->par=NULL;
56          p->pull();
57          x->pull();
58      }
59  }
60  void check_tree(node *t,int d){
61      if(!t) return;
62      check_tree(t->ch[0],d+1);
63      for(int i=0;i<d;i++) printf("\t");
64      printf("%d\n",t->val);
65      check_tree(t->ch[1],d+1);
66  }
67  void split(node *t,int k,node *&a,node *&b){
68      if(!k){
69          a=NULL; b=t; return;
70      }
71      int rod;
72      while( k != (rod=qsz(t->ch[0])+1) ){
73          t->push();
74          if(k>rod) k-=rod,t=t->ch[1];
75          else t=t->ch[0];
76      }
77      splay(t);
78      a=t;
79      a->push();
80      b=a->ch[1];
81      a->ch[1]=NULL;
82      a->pull();
83      if(b) b->par=NULL;
84  }
85  node* merge(node *a,node *b){
86      if(!a) return b;
87      while(a->ch[1]){
88          a->push();
```

```cpp
89          a=a->ch[1];
90      }
91      splay(a);
92      con(a,b,1);
93      a->pull();
94      return a;
95  }
96  int main(){
97      int n,q,x;
98      node *root=NULL,*a,*b,*c;
99      scanf("%d%d",&n,&q);
100     for(int i=0;i<n;i++){
101         scanf("%d",&x);
102         node *tmp=new (pool+i) node(x);
103         root=merge(root,tmp);
104     }
105     for(int i=0;i<q;i++){
106         char tp;
107         int x,y,z;
108         scanf(" %c%d%d",&tp,&x,&y);
109         split(root,x-1,a,b);
110         split(b,y-x+1,b,c);
111         if(tp=='C'){
112             scanf("%d",&z);
113             b->add+=z;
114         }
115         else printf("%lld\n",qsum(b));
116         root=merge(a,merge(b,c));
117     }
118     return 0;
119 }
```

8.2　Treap

```cpp
1  struct Treap{
2      Treap *l,*r;
3      int pri,sz,val,add;
4      Treap(int _val):pri(rand()),sz(1),val(_val),add(0),l(NULL),r(NULL){}
5  };
6
7  int size(Treap *t){
8      return t?t->sz:0;
9  }
10 void pull(Treap *t){
11     t->sz=size(t->l)+size(t->r)+1;
12 }
13 void push(Treap *t){
14     t->val+=t->add;
15     if(t->l) t->l->add+=t->add;
16     if(t->r) t->r->add+=t->add;
17     t->add=0;
18 }
19 Treap* merge(Treap *a,Treap *b){
20     if(!a||!b) return a?a:b;
```

```
21    if(a->pri > b->pri){
22      push(a);
23      a->r = merge(a->r,b);
24      pull(a);
25      return a;
26    }
27    else{
28      push(b);
29      b->l = merge(a,b->l);
30      pull(b);
31      return b;
32    }
33 }
34 void split(Treap *t,int k,Treap *&a,Treap *&b){
35    if(!t) a=b=NULL;
36    else{
37      push(t);
38      if(size(t->l) < k){
39        a=t;
40        split(t->r,k-size(t->l)-1,a->r,b);
41        pull(a);
42      }
43      else{
44        b=t;
45        split(t->l,k,a,b->l);
46        pull(b);
47      }
48    }
49 }
```

9 Tree

9.1 Heavy Light Decomposition

```
 1 //with set value && query sum, 1-based with n points
 2 //remove vis in DFS, add it back if something weird happen(I don't think it
       's required)
 3 using namespace std;
 4 int sz[N],top[N],up[N],dep[N];
 5 int lightval[N];   //value on light edge
 6 struct node{
 7   node(){}
 8   node(int _l,int _r):val(1),l(_l),r(_r),lc(NULL),rc(NULL){}
 9   int l,r;
10   node *lc,*rc;
11   int sum;
12   int val;
13   int qsum(){return val>=0?val*(r-l):sum;}
14   void push(){
15     if(val>=0){
16       sum=val*(r-l);
17       lc->val=rc->val=val;
```

```
18       val=-1;
19     }
20   }
21   void pull(){
22     sum=lc->qsum()+rc->qsum();
23   }
24 };
25 node* tr[N];
26 node* build(int l,int r){
27   node *now=new node(l,r);
28   if(r-l>1){
29     now->lc=build(l,(l+r)/2);
30     now->rc=build((l+r)/2,r);
31   }
32   return now;
33 }
34 //partial
35 int qry(node* now,int l,int r){
36   if(l>=r) return 0;
37   if(l==now->l&&r==now->r){
38     return now->qsum();
39   }
40   int m=(now->l+now->r)/2;
41   now->push();
42   if(l>=m){
43     return qry(now->rc,l,r);
44   }
45   else if(r<=m){
46     return qry(now->lc,l,r);
47   }
48   else return qry(now->lc,l,m)+qry(now->rc,m,r);
49 }
50 void set0(node *now,int l,int r){
51   if(l>=r) return;
52   if(l==now->l&&r==now->r){
53     now->val=0;
54     return;
55   }
56   int m=(now->l+now->r)/2;
57   now->push();
58   if(l>=m){
59     set0(now->rc,l,r);
60   }
61   else if(r<=m){
62     set0(now->lc,l,r);
63   }
64   else{
65     set0(now->lc,l,m);
66     set0(now->rc,m,r);
67   }
68   now->pull();
69 }
70 vector<int> g[N];
71 void DFS(int u,int p,int d){
72   dep[u]=d;
73   sz[u]=1;
```

```
74      for(int i=0;i<g[u].size();i++){
75        int v=g[u][i];
76        if(v==p) continue;
77        DFS(v,u,d+1);
78        sz[u]+=sz[v];
79  }
80  }
81  void decom(int u,int p,bool istop){
82    bool ed=true;
83    if(istop) top[u]=u,up[u]=p,lightval[u]=1;
84    else top[u]=top[p],up[u]=up[p];
85    for(int i=0;i<g[u].size();i++){
86      int v=g[u][i];
87      if(v==p) continue;
88      if(sz[v]>=sz[u]-sz[v]){
89        decom(v,u,false);
90        ed=false;
91      }
92      else decom(v,u,true);
93    }
94    if(ed){
95      tr[top[u]]=build(dep[top[u]],dep[u]);
96    }
97  }
98  //global
99  int qry(int u,int v){
100   int res=0;
101   while(top[u]!=top[v]){
102     if(dep[top[u]]>dep[top[v]]) swap(u,v);
103     res+=qry(tr[top[v]],dep[top[v]],dep[v]);
104     res+=lightval[top[v]];
105     v=up[top[v]];
106   }
107   if(dep[u]>dep[v]) swap(u,v);
108   res+=qry(tr[top[v]],dep[u],dep[v]);
109   return res;
110 }
111 void set0(int u,int v){
112   while(top[u]!=top[v]){
113     if(dep[top[u]]>dep[top[v]]) swap(u,v);
114     set0(tr[top[v]],dep[top[v]],dep[v]);
115     lightval[top[v]]=0;
116     v=up[top[v]];
117   }
118   if(dep[u]>dep[v]) swap(u,v);
119   set0(tr[top[v]],dep[u],dep[v]);
120 }
121 int main(){
122   DFS(1,0,0);
123   decom(1,0,true);
124 }
```

```
1   const int MAXN = 100000 + 10;
2   int tree[30][MAXN]={},sorted[MAXN]={},toleft[30][MAXN]={};
3   void build(int l,int r,int dep){
4       if(l==r)return;
5       int mid=(l+r)>>1;
6       int same=mid-l+1;
7       for(int i=l;i<=r;i++)if(tree[dep][i]<sorted[mid])same--;
8       int lpos=l,rpos=mid+1;
9       for(int i=l;i<=r;i++){
10          if(tree[dep][i]<sorted[mid])tree[dep+1][lpos++]=tree[dep][i];
11          else if(tree[dep][i]==sorted[mid]&&same>0)tree[dep+1][lpos++]=tree[
        dep][i],same--;
12          else tree[dep+1][rpos++]=tree[dep][i];
13          toleft[dep][i]=toleft[dep][l-1]+lpos-l;
14      }
15      build(l,mid,dep+1);
16      build(mid+1,r,dep+1);
17  }
18  int query(int L,int R,int l,int r,int dep,int k){
19      if(l==r)return tree[dep][l];
20      int mid=(L+R)>>1;
21      int cnt=toleft[dep][r]-toleft[dep][l-1];
22      if(cnt>=k){
23          int newl=L+toleft[dep][l-1]-toleft[dep][L-1];
24          int newr=newl+cnt-1;
25          return query(L,mid,newl,newr,dep+1,k);
26      }
27      else{
28          int newr=r+toleft[dep][R]-toleft[dep][r];
29          int newl=newr-(r-l-cnt);
30          return query(mid+1,R,newl,newr,dep+1,k-cnt);
31      }
32  }
33  int main(){
34      int n,m,a,b,c;
35      while(~scanf("%d%d",&n,&m)){
36          for(int i=1;i<=n;i++){
37              scanf("%d",&tree[0][i]);
38              sorted[i]=tree[0][i];
39          }
40          sort(sorted+1,sorted+n+1);
41          build(1,n,0);
42          while(m--){
43              scanf("%d%d%d",&a,&b,&c);
44              printf("%d\n",query(1,n,a,b,0,c));
45          }
46      }
47      return 0;
48  }
```

9.2   Huafen

9.3   KDtree Insert

```
1   #include<algorithm>
```

```
 2  #include<cmath>
 3  #include<cstdio>
 4  #include<queue>
 5  #include<cstdlib>
 6  #include<vector>
 7  #define MAXN 50100
 8  using namespace std;
 9  inline long long sq(long long x){return x*x;}
10  const double alpha=0.75;
11  int W,H,rx[MAXN],ry[MAXN];
12  namespace KDTree{
13    struct Point {
14      int x,y;
15      int index;
16      long long distance(const Point &b)const{
17        return sq(x-b.x) + sq(y-b.y);
18      }
19      bool operator==(const Point& rhs){return index==rhs.index;}
20    };
21    struct qnode{
22      Point p;
23      long long dis;
24      qnode(){}
25      qnode(Point _p,long long _dis){
26        p = _p;
27        dis = _dis;
28      }
29      bool operator <(const qnode &b)const{
30        if(dis != b.dis)return dis < b.dis;
31        else return p.index < b.p.index;
32      }
33    };
34    priority_queue<qnode>q;
35    inline bool cmpX(const Point &a,const Point &b){
36      return a.x < b.x || (a.x == b.x && a.y < b.y) || (a.x == b.x && a.y == b.
        y && a.index < b.index);
37    }
38    inline bool cmpY(const Point &a,const Point &b){
39      return a.y < b.y || (a.y == b.y && a.x < b.x) || (a.y == b.y && a.x == b.
        x && a.index < b.index);
40    }
41    bool cmp(const Point &a,const Point &b,bool div){
42      return div?cmpY(a,b):cmpX(a,b);
43    }
44    struct Node{
45      Point e;
46      Node *lc,*rc;
47      int size;
48      bool div;
49      inline void pull(){
50        size = 1 + lc->size + rc->size;
51      }
52      inline bool isBad(){
53        return lc->size > alpha*size || rc->size > alpha*size;
54      }
55    }pool[MAXN],*tail,*root,*recycle[MAXN],*null;
56    int rc_cnt;
57    void init(){
58      tail = pool;
59      null = tail++;
60      null->lc = null->rc = null;
61      null->size = 0;
62      rc_cnt = 0;
63      root = null;
64    }
65    Node *newNode(Point e){
66      Node *p;
67      if(rc_cnt)p = recycle[--rc_cnt];
68      else p = tail++;
69      p->e = e;
70      p->lc = p->rc = null;
71      p->size = 1;
72      return p;
73    }
74    Node *build(Point *a,int l,int r,bool div){
75      if(l >= r)return null;
76      int mid = (l+r)/2;
77      nth_element(a+l,a+mid,a+r,div?cmpY:cmpX);
78      Node *p = newNode(a[mid]);
79      p->div = div;
80      p->lc = build(a,l,mid,!div);
81      p->rc = build(a,mid+1,r,!div);
82      p->pull();
83      return p;
84    }
85    void getTree(Node *p,vector<Point>& v){
86      if(p==null) return;
87      getTree(p->lc,v);
88      v.push_back(p->e);
89      recycle[rc_cnt++]=p;
90      getTree(p->rc,v);
91    }
92    Node *rebuild(vector<Point>& v,int l,int r,bool div){
93      if(l>=r) return null;
94      int mid = (l+r)/2;
95      nth_element(v.begin()+l,v.begin()+mid,v.begin()+r,div?cmpY:cmpX);
96      Node *p = newNode(v[mid]);
97      p->div = div;
98      p->lc = rebuild(v,l,mid,!div);
99      p->rc = rebuild(v,mid+1,r,!div);
100     p->pull();
101     return p;
102   }
103   void rebuild(Node *&p){
104     vector<Point> v;
105     getTree(p,v);
106     p = rebuild(v,0,v.size(),p->div);
107   }
108   Node **insert(Node *&p,Point a,bool div){
109     if(p==null){
110       p = newNode(a);
111       p->div = div;
```

```
112        return &null;
113      }
114    else{
115      Node **res;
116      if(cmp(a,p->e,div)) res=insert(p->lc,a,!div);
117      else res=insert(p->rc,a,!div);
118      p->pull();
119      if(p->isBad()) res=&p;
120      return res;
121    }
122  }
123  void insert(Point e){
124    Node **p = insert(root,e,0);
125    if(*p!=null) rebuild(*p);
126  }
127  Node **get_min(Node *&p,bool div){
128    if(p->div==div){
129      if(p->lc!=null) return get_min(p->lc,div);
130      else return &p;
131    }
132    else{
133      Node **res=&p,**tmp;
134      if(p->lc!=null){
135        tmp = get_min(p->lc,div);
136        if(cmp((*tmp)->e,(*res)->e,div)) res=tmp;
137      }
138      if(p->rc!=null){
139        tmp = get_min(p->rc,div);
140        if(cmp((*tmp)->e,(*res)->e,div)) res=tmp;
141      }
142      return res;
143    }
144  }
145  void del(Node *&p){
146    Node **nxt;
147    if(p->rc!=null){
148      nxt = get_min(p->rc,p->div);
149      p->e = (*nxt)->e;
150      del(*nxt);
151    }
152    else if(p->lc!=null){
153      nxt = get_min(p->lc,p->div);
154      p->e = (*nxt)->e;
155      del(*nxt);
156      p->rc = p->lc;
157      p->lc = null;
158    }
159    else{
160      recycle[rc_cnt++]=p;
161      p=null;
162    }
163  }
164  void del(Node *&p,Point d){
165    if(p->e==d){
166      del(p);
167    }
```

```
168    else if(cmp(d,p->e,p->div)) del(p->lc,d);
169    else del(p->rc,d);
170  }
171  void search(Point p,Node *t,bool div,int m){
172    if(!t)return;
173    if(cmp(p,t->e,div)){
174      search(p,t->lc,!div,m);
175      if(q.size() < m){
176        q.push(qnode(t->e,p.distance(t->e)));
177        search(p,t->rc,!div,m);
178      }
179      else {
180        if(p.distance(t->e) <= q.top().dis){
181          q.push(qnode(t->e,p.distance(t->e)));
182          q.pop();
183        }
184        if(!div){
185          if(sq(t->e.x-p.x) <= q.top().dis)
186            search(p,t->rc,!div,m);
187        }
188        else {
189          if(sq(t->e.y-p.y) <= q.top().dis)
190            search(p,t->rc,!div,m);
191        }
192      }
193    }
194    else {
195      search(p,t->rc,!div,m);
196      if(q.size() < m){
197        q.push(qnode(t->e,p.distance(t->e)));
198        search(p,t->lc,!div,m);
199      }
200      else {
201        if(p.distance(t->e) <= q.top().dis){
202          q.push(qnode(t->e,p.distance(t->e)));
203          q.pop();
204        }
205        if(!div){
206          if(sq(t->e.x-p.x) <= q.top().dis)
207            search(p,t->lc,!div,m);
208        }
209        else {
210          if(sq(t->e.y-p.y) <= q.top().dis)
211            search(p,t->lc,!div,m);
212        }
213      }
214    }
215  }
216  void search(Point p,int m){
217    while(!q.empty())q.pop();
218    search(p,root,0,m);
219  }
220  void getRange(Node *p,vector<Point>& v,int x1,int x2,int y1,int y2){
221    if(p==null) return;
222    if(x1<=p->e.x && p->e.x<=x2 && y1<=p->e.y && p->e.y<=y2) v.push_back(p->e
       );
```

```
223      if(p->div ? y1<=p->e.y : x1<=p->e.x) getRange(p->lc,v,x1,x2,y1,y2);
224      if(p->div ? y2>=p->e.y : x2>=p->e.x) getRange(p->rc,v,x1,x2,y1,y2);
225    }
226    void solve(Point p){
227      del(root,p);
228      insert(p);
229    }
230 };
231 KDTree::Point p[MAXN];
232 int main(){
233    KDTree::init();
234    KDTree::root = KDTree::build(p,0,n,0);
235    while(q--){
236      KDTree::Point tmp,p1,p2;
237      scanf("%d%d",&tmp.x,&tmp.y);
238      search(tmp,2);
239      p1=KDTree::q.top().p;
240      KDTree::q.pop();
241      p2=KDTree::q.top().p;
242      KDTree::q.pop();
243    }
244    return 0;
245 }
```