SOUTH CHINA UNIVERSITY OF TECHNOLOGY

SCUT_GUGUGU

TEMPLATE

0 error(s), 0 warning(s)

Last build at December 6, 2020

Contents

7   Samples   **123**

1 Graph Theory

1.1 Shortest Path

1.1.1 Dijkstra

```cpp
typedef long long LL;
const int MAXN = ;
const int MAXM = ;
const LL DINF = ;
typedef pair<LL, int> P;
struct Edge {
    int to, nxt;
    LL w;
}e[MAXM];
int head[MAXN], ecnt;
LL d[MAXN];
priority_queue<P, vector<P>, greater<P> > q;
inline void addEdge(int x, int y, LL w) {
    e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
}
void dijkstra(int st, int n) {
    for(int i = 0; i <= n; i++) d[i] = DINF;
    d[st] = 0;
    q.push(make_pair(0, st));
    while(!q.empty()) {
        P x = q.top(); q.pop();
        int u = x.second;
        if(d[u] != x.first) continue;
        for(int i = head[u], v; i; i = e[i].nxt) {
            v = e[i].to;
            if(d[v] > d[u] + e[i].w) {
                d[v] = d[u] + e[i].w;
                q.push(make_pair(d[v], v));
            }
        }
    }
}
```

1.1.2 SPFA

```cpp
struct Edge {
    int to, nxt;
    LL w;
}e[MAXE];
int head[MAXN], ecnt;
LL d[MAXN];
bool exist[MAXN];
queue<int> q;
inline void addEdge(int x, int y, LL w) {
    e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
}
void SPFA(int st) {
    memset(d,0x3f,sizeof(d));
    d[st] = 0;
    q.push(st);
    exist[st] = 1;
    while(!q.empty()) {
```

```
18          int u = q.front(); q.pop();
19          exist[u] = 0;
20          for(int i = head[u], v; i; i = e[i].nxt) {
21              v = e[i].to;
22              if(d[v] > d[u] + e[i].w) {
23                  d[v] = d[u] + e[i].w;
24                  //pre[v] = u;
25                  if(!exist[v]) {
26                      q.push(v);
27                      exist[v] = 1;
28                  }
29              }
30          }
31      }
32  }
```

1.1.3 Johnson

```
1  void johnson() {
2      //全源带负权最短路,新建超级节点向全部点连权为0的边，计算D，利用Dij计算点对距离
3      for(int i = 1; i <= n; i++) addEdge(0, i, 0);
4      spfa(0);
5      for(int u = 1; u <= n; u++)
6          for(int i = head[u]; i; i = e[i].nxt)
7              e[i].w += D[e[i].from] - D[e[i].to];
8      dijkstra(s,n);
9      //ans = d[n] - D[s] + D[n];
10
11  }
```

1.1.4 K Shortest Path (A*)

```
1  //可重复走同一条边 利用了反向边表示,ecnt初始化为1
2  //调用dijkstra(ed,n)跑反向图,注意if(i& 1){松弛}
3  int shrt[MAXN];
4  LL A_star(int st, int ed, int k, int n) {
5      if(d[st] == d[0]) return -1;
6      if(st == ed) k++;
7      q.push(make_pair(d[st], st));
8      while(!q.empty()) {
9          P x = q.top(); q.pop();
10         int u = x.second;
11         LL xd = x.first;
12         ++shrt[u];
13         if(u == ed) {
14             if(shrt[ed] == k) return xd;
15         }
16         for(int i = head[u], v; i; i = e[i].nxt)
17         if(!(i & 1)) {
18             v = e[i].to;
19             if(shrt[v] < k)q.push(make_pair(xd-d[u]+e[i].w+d[v], v));
20         }
21     }
22     return -1;
23  }
```

1.1.5 K Shortest Path (Protractable Heap)

```cpp
//可重复走同一条边
typedef double LD;
const int MAXN = ;
const int MAXM = ;
const int MAXLT = MAXM * 20;
const LD DINF = ;
const LD eps = ;

namespace LT{
    int tcnt;
    int ls[MAXLT], rs[MAXLT], dis[MAXLT];
    int to[MAXLT];
    LD val[MAXLT];

    inline int newnode(LD w, int _to) {
        ++tcnt;
        ls[tcnt] = rs[tcnt] = 0; dis[tcnt] = 1;
        val[tcnt] = w; to[tcnt] = _to;
        return tcnt;
    }
    inline int copynode(int id) {
        ++tcnt;
        ls[tcnt] = ls[id]; rs[tcnt] = rs[id]; dis[tcnt] = dis[id];
        val[tcnt] = val[id]; to[tcnt] = to[id];
        return tcnt;
    }
    void push_up(int x) {
        if(dis[ls[x]] < dis[rs[x]]) swap(ls[x], rs[x]);
        dis[x] = dis[rs[x]] + 1;
    }
    int merge(int x, int y) {
        if(!x || !y) return x^y;
        if(val[x] - val[y] > eps) swap(x, y);
        int p = copynode(x);
        rs[p] = merge(rs[p],y);
        push_up(p);
        return p;
    }
}
int rt[MAXN];

typedef pair<LD, int> P;
struct Edge {
    int to, nxt;
    LD w;
}e[MAXM];
int head[MAXN], ecnt;
int stan, sta[MAXN], fa[MAXN];
int vis[MAXN], cov[MAXM];
LD d[MAXN];
priority_queue<P, vector<P>, greater<P> > q;
inline void addEdge(int x, int y, LD w) {
    e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
}
void init(int n, int m) {
    ecnt = 1; stan = 0;
    for(int i = 1; i <= n; i++)
        head[i] = cov[i] = fa[i] = 0;
    for(int i = 1; i <= m; i++) {
```

```
60              int u, v; LD w;
61              scanf("%d%d%lf", &u,&v,&w);
62              addEdge(u, v, w);
63              addEdge(v, u, w);
64          }
65      }
66      void dijkstra(int st, int n) {
67          for(int i = 0; i <= n; i++) {d[i] = DINF; vis[i] = 0;}
68          while(!q.empty()) q.pop();
69          d[st] = 0;q.push(make_pair(0, st));
70          while(!q.empty()) {
71              P x = q.top(); q.pop();
72              int u = x.second;
73              if(vis[u]) continue;
74              vis[u] = 1;
75              for(int i = head[u], v; i; i = e[i].nxt)
76                  if(i & 1) {
77                      v = e[i].to;
78                      if(d[v]- (d[u] + e[i].w) > eps) {
79                          d[v] = d[u] + e[i].w;
80                          q.push(make_pair(d[v], v));
81                      }
82                  }
83          }
84      }
85      void buildT(int u) {
86          sta[++stan] = u;
87          vis[u] = 1;
88          for(int i = head[u], v; i; i = e[i].nxt)
89          if(i & 1) {
90              v = e[i].to;
91              if(fabs(d[v] - (d[u] + e[i].w)) < eps && !vis[v]) {
92                  fa[v] = u; cov[i^1] = 1; buildT(v);
93              }
94          }
95      }
96      void buildH(int st, int n) {
97          buildT(st);
98          for(int i = 2, u, v; i <= ecnt; i += 2) {
99              if(!cov[i]) {
100                 u = e[i^1].to; v = e[i].to;
101                 if(fabs(d[u] - d[0]) < eps || fabs(d[v] - d[0]) < eps) continue;
102                 rt[u] = LT::merge(rt[u], LT::newnode(d[v]+e[i].w-d[u], v));
103             }
104         }
105         for(int i = 2, u; i <= n; i++)
106             if(fa[u = sta[i]]) rt[u] = LT::merge(rt[u], rt[fa[u]]);
107     }
108     //求前k短路径,其和不超过W,问最大k int ans = 0;
109     //求第k短路径,LD ans = 0;
110     void getKth(int st, int ed, LD W) {
111         while(!q.empty()) q.pop();
112         //最短路要记入答案,注意d[st]-W<eps与W-d[st]<eps
113         //!if(d[st] − W > eps) return;
114         //!else{W −= d[st]; ++ans;}
115         //注意st和ed相同
116         //#if(st == ed) k++;
117         //#if(d[st] == d[0]) {ans = −1; return;}
118         //#if(−−k == 0) {ans = d[st]; return;}
119
120         int u = rt[st], v;
```

```
121        if(u) q.push(make_pair(LT::val[u], u));
122        while (!q.empty()) {
123            u = q.top().second; LD cur = q.top().first;
124            q.pop();
125            //!if (cur + d[st] − W > eps) break;
126            //!else {W −= cur + d[st]; ++ans;}
127            //#if (−−k == 0) {ans = cur + d[st]; break;}
128            v = LT::ls[u];
129            if (v) q.push(make_pair(cur - LT::val[u] + LT::val[v], v));
130            v = LT::rs[u];
131            if (v) q.push(make_pair(cur - LT::val[u] + LT::val[v], v));
132            v = rt[LT::to[u]];
133            if (v) q.push(make_pair(cur + LT::val[v], v));
134        }
135 }
136 void sol() {
137     int n, m, st = , ed = ;
138     //LD W;  int k;
139     init(n, m);
140     dijkstra(ed, n);
141     for(int i = 0; i <= n; i++) vis[i] = 0;
142     buildH(ed, n);
143     getKth(st, ed, W);
144     printf("%d\n", ans);
145 }
```

1.2 Network Flow

1.2.1 ISAP

```
1  namespace NWF {
2      struct Edge{
3          int to, nxt;LL f;
4      }e[MAXM << 1];
5      int S, T, tot;
6      int ecnt, head[MAXN], cur[MAXN], pre[MAXN], num[MAXN], dis[MAXN];
7      queue<int> q;
8      void init(int _S, int _T, int _tot){
9          ecnt = 1; S = _S; T = _T; tot = _tot;
10         memset(num,  0, (tot + 1) * sizeof(int));
11         memset(head, 0, (tot + 1) * sizeof(int));
12     }
13     inline void addEdge(int u, int v, LL f) {
14         e[++ecnt] = (Edge) {v, head[u], f}; head[u] = ecnt;
15         e[++ecnt] = (Edge) {u, head[v], 0}; head[v] = ecnt;
16     }
17     void bfs() {
18         memset(dis, 0, (tot + 1) * sizeof(int));
19         q.push(T);
20         dis[T] = 1;
21         while(!q.empty()) {
22             int u = q.front(), v; q.pop();
23             num[dis[u]]++;
24             for(int i = cur[u] = head[u]; i; i = e[i].nxt) {
25                 if(!dis[v = e[i].to]) {
26                     dis[v] = dis[u] + 1;
27                     q.push(v);
28                 }
29             }
```

```
30              }
31          }
32      LL augment() {
33          LL flow = INF;
34          for(int i = S; i != T; i = e[cur[i]].to)
35              flow = min(flow, e[cur[i]].f);
36          for(int i = S; i != T; i = e[cur[i]].to) {
37              e[cur[i]].f -= flow;
38              e[cur[i] ^ 1].f += flow;
39          }
40          return flow;
41      }
42      LL isap() {
43          bfs();
44          int u = S, v;
45          LL flow = 0;
46          while(dis[S] <= tot) {
47              if(u == T) {
48                  flow += augment();
49                  u = S;
50              }
51              bool fg = 0;
52              for(int i = cur[u]; i; i = e[i].nxt) {
53                  if(e[i].f && dis[u] > dis[v = e[i].to]) {
54                      pre[v] = u;
55                      cur[u] = i;
56                      u = v;
57                      fg = 1;
58                      break;
59                  }
60              }
61              if(fg) continue;
62              if(!--num[dis[u]]) break;
63              int maxDis = tot;
64              for(int i = head[u]; i; i = e[i].nxt) {
65                  if(e[i].f && maxDis > dis[v = e[i].to]) {
66                      maxDis = dis[v];
67                      cur[u] = i;
68                  }
69              }
70              num[dis[u] = maxDis + 1]++;
71              if(u != S) u = pre[u];
72          }
73          return flow;
74      }
75  }
```

1.2.2   HLPP

```
1   namespace NWF{
2       struct Edge{
3           int to,nxt;LL f;
4       }e[MAXM << 1];
5       int S, T, tot;
6       int ecnt, head[MAXN], dis[MAXN], num[MAXN];
7       LL sumf[MAXN];
8       queue<int> q;
9       list<int> dep[MAXN];
10      void init(int _S,int _T,int _tot){
11          ecnt = 1;S = _S;T = _T;tot = _tot;
```

```
12          memset(num,  0, (tot + 1) * sizeof(int));
13          memset(head, 0, (tot + 1) * sizeof(int));
14          memset(sumf, 0, (tot + 1) * sizeof(LL));
15      }
16      void addEdge(int u,int v,LL f){
17          e[++ecnt] = (Edge) {v, head[u], f};head[u] = ecnt;
18          e[++ecnt] = (Edge) {u, head[v], 0};head[v] = ecnt;
19      }
20      void bfs(){
21          memset(dis, 0, (tot + 1) * sizeof(int));
22          q.push(T); dis[T] = 1;
23          while(!q.empty()){
24              int u=q.front(), v; q.pop();
25              for(int i = head[u]; i; i = e[i].nxt)
26              if(!dis[v = e[i].to]){
27                  dis[v] = dis[u] + 1;
28                  q.push(v);
29              }
30          }
31      }
32      LL hlpp(){
33          bfs();
34          dis[S] = tot + 1;
35          for(int i = 1;i <= tot; ++i)num[dis[i]]++;
36          for(int i = tot + 1; ~i; --i)dep[i].clear();
37          int maxd = dis[S];LL f;
38          dep[maxd].push_back(S);sumf[S] = INF;
39          for(;;){
40              while(maxd && dep[maxd].empty())maxd--;
41              if(!maxd)break;
42              int u = dep[maxd].back(), v;dep[maxd].pop_back();
43              int minDis = tot + 1;
44              for(int i = head[u]; i;i = e[i].nxt)
45              if(e[i].f){
46                  if(dis[u] > dis[v = e[i].to]){
47                      f = min(sumf[u], e[i].f);
48                      e[i].f -= f;e[i^1].f += f;
49                      if(sumf[u] != INF) sumf[u] -= f;
50                      if(sumf[v] != INF) sumf[v] += f;
51                      if(v!=S && v!=T && sumf[v] == f){
52                          maxd = max(maxd, dis[v]);
53                          dep[dis[v]].push_back(v);
54                      }
55                      if(!sumf[u])break;
56                  }else minDis=min(minDis, dis[v] + 1);
57              }
58              if(sumf[u]){
59                  if(!--num[dis[u]]){
60                      for(int i = dis[u];i <= maxd;++i){
61                          while(!dep[i].empty()){
62                              --num[i];
63                              dis[dep[i].back()] = tot + 1;
64                              dep[i].pop_back();
65                          }
66                      }
67                      maxd = dis[u] - 1;dis[u] = tot + 1;
68                  }else{
69                      dis[u] = minDis;
70                      if(minDis > tot)continue;
71                      num[minDis]++;
72                      maxd = max(maxd, minDis);
```

```
73                       dep[minDis].push_back(u);
74                  }
75              }
76          }
77          return sumf[T];
78      }
79  }
```

1.2.3   Dinic

注意当流为浮点数的时候，要判断 eps 以及不能使用 sumf-=tmpf，否者 1e18 将不会发生改变

```
1   namespace NWF {
2       struct Edge {
3           int to, nxt;LL f;
4       } e[MAXM << 1];
5       int S, T, tot;
6       int ecnt, head[MAXN], cur[MAXN], dis[MAXN];
7       queue<int> q;
8       void init(int _S, int _T, int _tot){
9           ecnt = 1; S = _S; T = _T; tot = _tot;
10          memset(head, 0, (tot + 1) * sizeof(int));
11      }
12      void addEdge(int u, int v, LL f) {
13          e[++ecnt] = (Edge) {v, head[u], f}; head[u] = ecnt;
14          e[++ecnt] = (Edge) {u, head[v], 0}; head[v] = ecnt;
15      }
16      bool bfs() {
17          memset(dis, 0, (tot + 1) * sizeof(int));
18          q.push(S); dis[S] = 1;
19          while (!q.empty()) {
20              int u = q.front(), v; q.pop();
21              for (int i = cur[u] = head[u]; i ; i = e[i].nxt) {
22                  if (e[i].f && !dis[v = e[i].to]) {
23                      q.push(v);
24                      dis[v] = dis[u] + 1;
25                  }
26              }
27          }
28          return dis[T];
29      }
30      LL dfs(int u, LL maxf) {
31          if (u == T) return maxf;
32          LL sumf = maxf;
33          for (int &i = cur[u]; i; i = e[i].nxt) {
34              if (e[i].f && dis[e[i].to] > dis[u]) {
35                  LL tmpf = dfs(e[i].to, min(sumf, e[i].f));
36                  e[i].f -= tmpf; e[i ^ 1].f += tmpf;
37                  sumf -= tmpf;
38                  if (!sumf) return maxf;
39              }
40          }
41          return maxf - sumf;
42      }
43      LL dinic() {
44          LL ret = 0;
45          while (bfs()) ret += dfs(S, INF);
46          return ret;
47      }
48      void rebuild(){
49          //无向图采用 e[i].f = e[i^1].f 的方式建立图
```

```
50          for(int i = 2; i <= ecnt; i+=2) e[i].f = e[i^1].f = (e[i].f + e[i^1].f) >> 1;
51          //有向图
52          //for(int i = 2; i <= ecnt; i+=2) e[i].f += e[i^1].f, e[i^1].f = 0;
53      }
54  }
```

1.2.4   Bound Flow

```
1   namespace NWF{
2       //在Edge中添加下限，delta_flow[i]:节点i的入流−出流，bound_flow注意要处理除了SS和TT以
        外的点
3       int delta_flow[MAXN];
4       void addEdge(int u, int v, int mxf, int mnf) {
5           addEdge(u, v, mxf-mnf);
6           delta_flow[u] -= mnf; delta_flow[v] += mnf;
7       }
8       void delEdge(int u) {
9           for(int i = head[u]; i; i = e[i].nxt) e[i].f = e[i^1].f = 0;
10      }
11      int bound_flow() {
12          int SS = ++tot, TT = ++ tot, sum = 0;
13          head[SS] = head[TT] = 0;
14          for(int i =1; i <= tot;++i) {
15              if(delta_flow[i]<0) addEdge(i, TT, -delta_flow[i]);
16              if(delta_flow[i]>0){
17                  sum+=delta_flow[i];
18                  addEdge(SS, i, delta_flow[i]);
19              }
20          }
21          addEdge(T,S, INF);
22          int tS = S, tT = T;
23          S = SS; T = TT;
24          if (dinic()==sum) {
25              delEdge(SS); delEdge(TT);
26              int flow = e[ecnt].f;
27              e[ecnt].f = e[ecnt^1].f = 0;
28              //S = tS; T = tT;// 有上下界有源汇最大流
29              //return flow + dinic();
30              S = tT; T = tS;// 有上下界有源汇最小流
31              return flow - dinic();
32          }else{
33              return -1;
34          }
35      }
36  }
```

1.2.5   Modeling Optimization

利用分治优化建模，每个点 i 向 j 连边，费用为 |ai-aj|

```
1   int pos[MAXN];
2   pair<int, int> tmp[MAXN];
3   void CDQ(int L, int R) {
4       if (L == R) return;
5       int mid = (L + R) >> 1;
6       CDQ(L, mid); CDQ(mid + 1, R);
7       inplace_merge(tmp + L, tmp + mid + 1, tmp + R + 1);
8       for (int i = L; i <= R; ++i) pos[tmp[i].second] = i;
9       for (int i = 2; i <= R - L + 1; ++i) {
```

```
10              addEdge(tot + i, tot + i - 1, INF, tmp[L + i - 1].first - tmp[L + i - 2].first);
11              addEdge(tot + i - 1, tot + i, INF, tmp[L + i - 1].first - tmp[L + i - 2].first);
12          }
13          for (int i = L; i <= R; ++i) {
14              if (i <= mid)
15                  addEdge(i+i-1, tot + pos[i] - L + 1, 1, 0);
16              else
17                  addEdge(tot + pos[i] - L + 1, i+i, 1, 0);
18          }
19          tot += R - L + 1;
20      }
```

1.2.6 Gomory-Hu Tree

两点间的割可以转为树上两点的距离

```
1  namespace NWF{
2      Edge Te[MAXN];
3      int Tcnt, Thead[MAXN];
4      void TaddEdge(int u, int v, LL f) {
5          Te[++Tcnt] = (Edge) {v, Thead[u], f}; Thead[u] = Tcnt;
6          Te[++Tcnt] = (Edge) {u, Thead[v], f}; Thead[v] = Tcnt;
7      }
8      int node[MAXN], tmp[MAXN];
9      void build(int l, int r) {
10         if (l == r) return;
11         S = node[l]; T = node[l+1];
12         rebuild();
13         LL cut = dinic();
14         TaddEdge(S, T, cut);
15         int tl = l, tr = r;
16         for(int i = l; i <= r; i++) {
17             if(dis[node[i]]) tmp[tl++] = node[i]; else tmp[tr--] = node[i];
18         }
19         for(int i=l; i<=r; i++) node[i] = tmp[i];
20         build(l,tl-1); build(tr+1,r);
21     }
22     int log2n;
23     int dep[MAXN], anc[MAXN][MAXS];LL mnl[MAXN][MAXS];//anc: 祖先; mnl: 最小边
24     void lca_dfs(int u, int _fa) {
25         for(int i=Thead[u], v; i; i=Te[i].nxt) {
26             if((v = Te[i].to) == _fa) continue;
27             dep[v] = dep[u] + 1;
28             anc[v][0] = u;
29             mnl[v][0] = Te[i].f;
30             lca_dfs(v, u);
31         }
32     }
33     void work() {
34         if(tot == 0) return;
35         log2n = log2(tot)+1; Tcnt = 1;
36         for(int i=1; i<=tot; i++) node[i]=i, Thead[i]=0;
37         build(1, tot);
38         dep[1] = 1; anc[1][0] = 0; mnl[1][0] = INF;
39         lca_dfs(1, -1);
40         for(int j = 1; j <= log2n; j++) {
41             for(int i = 1; i <= tot; i++) {
42                 anc[i][j] = anc[anc[i][j-1]][j-1];
43                 mnl[i][j] = min(mnl[i][j-1], mnl[anc[i][j-1]][j-1]);
44             }
45         }
```

```
46        }
47        LL get_cut(int u,int v) {
48            LL res=INF;
49            if(dep[u] < dep[v]) swap(u, v);
50            for(int i = log2n; i >= 0; i--) {
51                if(dep[anc[u][i]] >= dep[v]){
52                    res = min(res, mnl[u][i]);
53                    u = anc[u][i];
54                }
55            }
56            if(u == v) return res;
57            for(int i = log2n; i>=0 ; i--) {
58                if(anc[u][i] != anc[v][i]) {
59                    res = min(res, mnl[u][i]);
60                    res = min(res, mnl[v][i]);
61                    u = anc[u][i];
62                    v = anc[v][i];
63                }
64            }
65            res = min(res, mnl[u][0]);
66            res = min(res, mnl[v][0]);
67            return res;
68        }
69    }
```

1.2.7   MCMF

```
1    namespace NWF{
2        struct Edge {
3            int to, nxt;LL f, c;
4        } e[MAXM << 1];
5        int S, T, tot;
6        int ecnt, head[MAXN], cur[MAXN];LL dis[MAXN];
7        bool exist[MAXN];
8        queue<int> q;
9        void init(int _S, int _T, int _tot){
10            ecnt = 1; S = _S; T = _T; tot = _tot;
11            memset(head, 0, (tot + 1) * sizeof(int));
12        }
13        void addEdge(int u, int v, LL f, LL c) {
14            e[++ecnt] = (Edge) {v, head[u], f, c}; head[u] = ecnt;
15            e[++ecnt] = (Edge) {u, head[v], 0,-c}; head[v] = ecnt;
16        }
17        bool spfa() {
18            for(int i = 0;i <= tot; ++i){
19                dis[i] = INF; cur[i] = exist[i] = 0;
20            }
21            q.push(S);dis[S] = 0;exist[S] = 1;
22            while(!q.empty()) {
23                int u = q.front(), v; q.pop();exist[u] = 0;
24                for(int i = head[u]; i; i = e[i].nxt) {
25                    if(e[i].f && dis[v = e[i].to] > dis[u] + e[i].c) {
26                        dis[v] = dis[u] + e[i].c;
27                        cur[v] = i;
28                        if(!exist[v]) {
29                            q.push(v);
30                            exist[v] = 1;
31                        }
32                    }
33                }
```

```
34          }
35          return dis[T] != INF;
36      }
37      LL mcmf() {
38          LL cost = 0;
39          //while(spfa() && dis[T] < 0) {//最小费用可行流
40          while(spfa()) {
41              LL flow = INF;
42              for(int i = T; i != S; i = e[cur[i] ^ 1].to)
43                  flow = min(flow, e[cur[i]].f);
44              for(int i = T; i != S; i = e[cur[i] ^ 1].to) {
45                  e[cur[i]].f -= flow;
46                  e[cur[i] ^ 1].f += flow;
47              }
48              cost += flow * dis[T];
49          }
50          return cost;
51      }
52 }
```

1.3 Tree Related

1.3.1 Union Set

```
1  int fa[MAXN], rnk[MAXN];
2  int Find(int x) { return x == fa[x] ? x : fa[x] = Find(fa[x]); }
3  bool same(int x, int y){ return Find(x) == Find(y); }
4  void unite(int x, int y)
5  {
6      x = Find(x);
7      y = Find(y);
8      if(x == y) return;
9      if(rnk[x] < rnk[y]) {
10         fa[x] = y;
11     }
12     else {
13         fa[y] = x;
14         if(rnk[x] == rnk[y]) rnk[x]++;
15     }
16 }
```

1.3.2 Kruskal

```
1  namespace MST{
2      struct Edge{
3          int u,v; LL w;
4          bool operator < (const Edge& x) const { return w < x.w; }
5      }e[MAXM];
6      int ecnt, fa[MAXN];
7      void addEdge(int u, int v, LL w) {
8          e[++ecnt] = (Edge){v, u, w}; //headp[u] = ecnt;
9      }
10     int Find(int x) { return x == fa[x] ? x : fa[x] = Find(fa[x]); }
11     LL kruskal(int n) {
12         sort(e + 1, e + ecnt + 1);
13         for(int i = 1; i <= n; i++) fa[i] = i;
14         LL sum = 0;
15         for (int i = 1; i <= ecnt; i++){
```

```
16          int fu = Find(e[i].u), fv = Find(e[i].v);
17          if(fu != fv){
18              fa[fu] = fv;
19              sum += e[i].w;
20          }
21       }
22       return sum;
23    }
24 }
```

1.3.3 Prim

```
1  namespace MST {
2      struct Edge{
3          int to,nxt; LL w;
4      }e[MAXM];
5      int ecnt, head[MAXN], vis[MAXN]; // pre[MAXN];
6      LL dis[MAXN];
7      void addEdge(int u, int v, LL w){
8          e[++ecnt] = (Edge){v, head[u], w}; head[u] = ecnt;
9          e[++ecnt] = (Edge){u, head[v], w}; head[v] = ecnt;
10     }
11     LL Prim(int n){
12         for (int i = 1; i <= n; i++){
13             //pre[i] = 0;
14             vis[i] = 0;
15             dis[i] = INF;
16         }
17         vis[1] = 1;
18         LL sum = 0;
19         for (int i = head[1]; i; i = e[i].nxt)
20             dis[e[i].to] = min(dis[e[i].to],e[i].w);
21         for (int j = 1; j < n; j++){
22             int u; LL minDis = INF;
23             for (int i = 1; i <= n; ++i)
24                 if (!vis[i] && dis[i] < minDis){
25                     minDis = dis[i];
26                     u = i;
27                 }
28             if (minDis == INF) return -1;
29             vis[u] = 1;
30             sum += minDis;
31             for (int i = head[u], v; i; i = e[i].nxt)
32                 if (!vis[v = e[i].to] && e[i].w < dis[v]){
33                     //pre[u] = v;
34                     dis[v] = e[i].w;
35                 }
36         }
37         return sum;
38     }
39 }
```

1.3.4 Spanning Tree Calculation

关联矩阵 B:n*m 的矩阵, 其中 ek=(vi,vj),Bik 和 Bjk 一个为 1 一个为-1, 第 k 列其他元素为 0 度数矩阵 D:n*n 的矩阵, 其中 i!=j 时,D[i][j]=0;i==j 时,D[i][j]=vi 的度邻接矩阵 A:n*n 的矩阵,vi,vj 有边相连, 为 1, 否则为 0 Kirchhoff 矩阵: B*BT = D - A 即: 如果 i=j, 那么 aij 为点 i(j) 的度数。如果 i!=j, 那么 Aij 为 i 到 j 的边数的相反数。生成树个数:Kirchhoff 矩阵 n-1 阶主子式的行列式值构造 Kirchhoff 矩阵, 调用 det(n)

```
 1  LL a[MAXN][MAXN];
 2  void getSTC(int n, int m) {
 3      for(int i = 1; i <= n; i++) {
 4          for(int j = 1; j <= n; j++)
 5              a[i][j] = a[j][i] = 0;
 6      }
 7      for(int i = 1,u, v; i <= m; i++) {
 8          scanf("%d%d", &u,&v);
 9          if(u == v) continue;
10          a[u][v] = --a[v][u];
11      }
12      for(int i = 1; i <= n; i++) {
13          int t = 0;
14          for(int j = 1; j <= n; j++)
15              t += a[i][j];
16          a[i][i] = - t;
17      }
18      LL ans = det();//删掉一行一列以后求行列式的值
19  }
```

1.3.5   Minimum Spanning Tree Calculation

```
 1  typedef long long LL;
 2  const int MAXN = ;
 3  const int MAXM = ;
 4  int sum,ans1,ans2=1, Mod=;
 5  int fa1[MAXN],fa2[MAXN];
 6  bool vis[MAXN];
 7  struct Edge{int v,u,val;}e[MAXM];
 8  bool cmp(Edge A,Edge B){return A.val<B.val;}
 9  int getfa1(int *fa,int x){return fa[x]=fa[x]==x?x:getfa1(fa,fa[x]);}
10  int getfa2(int *fa,int x){return fa[x]==x?x:getfa2(fa,fa[x]);}
11  void dfs(int tot,int l,int r){
12      if(tot==0){++sum;return;}
13      for(int i=l,fx,fy;i<r;++i)
14          if(!vis[i]){
15              vis[i]=true;
16              fx=getfa2(fa2,e[i].u);fy=getfa2(fa2,e[i].v);
17              if(fx!=fy){
18                  fa2[fx]=fy;
19                  dfs(tot-1,i+1,r);
20                  fa2[fx]=fx;
21              }
22              vis[i]=false;
23          }
24  }
25  void sol(){
26      int n,m;
27      scanf("%d %d",&n,&m);
28      for(int i=1;i<=n;++i)fa1[i]=fa2[i]=i;
29      for(int i=1;i<=m;++i)
30          scanf("%d %d %d",&e[i].v,&e[i].u,&e[i].val);
31      std::sort(e+1,e+m+1,cmp);e[m+1].val=-1;
32      for(int i=1,j=1,fx,fy,tot;i<=m;++i){
33          for(;e[i].val==e[j].val;++j);
34          tot=0;
35          for(int k=i;k<j;++k){
36              fx=getfa1(fa1,e[k].u);fy=getfa1(fa1,e[k].v);
37              if(fx!=fy){++tot;ans1++;fa1[fx]=fy;}
38          }
```

```
39          if(!tot)continue;
40          sum=0;dfs(tot,i,j);
41          (ans2*=sum)%=Mod;
42          for(int k=i;k<j;++k){
43              fx=getfa1(fa2,e[k].u);fy=getfa1(fa2,e[k].v);
44              if(fx!=fy)fa2[fx]=fy;
45          }
46      }
47      if(ans1!=n-1)puts("0");else printf("%d",ans2);
48  }
```

1.3.6   Steiner Tree

```
1   const int MAXH = 128;
2   const int MAXW = 128;
3   const int MAXST = 1256;
4   namespace SteinerTree{
5       const int dx[4] = {0,0,1,-1};
6       const int dy[4] = {1,-1,0,0};
7       int n, m, k, stn;
8       bool vis[MAXH][MAXW];
9       LL G[MAXH][MAXW],f[MAXST][MAXH][MAXW];
10      queue<pair<int, int> > q;
11      struct PRE{int s, x, y;} pre[MAXST][MAXH][MAXW];
12
13      void spfa(int st) {
14          while(!q.empty()) {
15              int vx = q.front().first,vy = q.front().second;
16              q.pop();
17              vis[vx][vy] = 0;
18              for (int i = 0, ux,uy; i < 4; ++i) {
19                  ux = vx + dx[i];
20                  uy = vy + dy[i];
21                  if (ux==0||uy==0||ux==n+1||uy==m+1)continue;
22                  if (f[st][vx][vy]+G[ux][uy]<f[st][ux][uy]) {
23                      f[st][ux][uy] = f[st][vx][vy] + G[ux][uy];
24                      pre[st][ux][uy] = (PRE) {st, vx, vy};
25                      if (!vis[ux][uy]) {
26                          vis[ux][uy]=1;
27                          q.push(make_pair(ux,uy));
28                      }
29                  }
30              }
31          }
32      }
33      LL sum = 0;
34      void init() {
35          k = 0;
36          for(int i = 1; i <= n; ++i)
37              for(int j = 1; j <= m; ++j) {
38                  scanf("%d", &G[i][j]);
39                  if(G[i][j] == 0)  k++;
40              }
41          stn = 1<<k;
42          for (int st = 0; st < stn; ++st)
43              for (int i = 1; i <= n; ++i)
44                  for (int j = 1; j <= m; ++j)
45                      f[st][i][j] = INF;
46          int tk = 0;
47          for(int i = 1; i <= n; ++i)
```

```
48              for(int j = 1; j <= m; ++j) {
49                  vis[i][j] = 0;
50                  if(G[i][j] == 0) {f[1<<tk][i][j] = 0; tk++;}
51              }
52      }
53      void dfs(int st, int x, int y) {
54          vis[x][y] = 1;
55          PRE tmp = pre[st][x][y];
56          if (tmp.x == 0 && tmp.y == 0) return ;
57          dfs(tmp.s, tmp.x, tmp.y);
58          if (tmp.x == x && tmp.y == y) dfs(st - tmp.s, tmp.x, tmp.y);
59      }
60      void sol(int _n, int _m) {
61          n = _n; m = _m;
62          init();
63          for (int st = 0; st < stn; ++st) {
64              for (int i = 1; i <= n; ++i)
65                  for (int j = 1; j <= m; ++j) {
66                      for (int s = st&(st-1); s; s = st&(s-1))
67                          if(f[st-s][i][j]+f[s][i][j]-G[i][j] < f[st][i][j]){
68                              f[st][i][j] = f[st-s][i][j]+f[s][i][j]-G[i][j];
69                              pre[st][i][j] = (PRE) {s, i, j};
70                          }
71                      if (f[st][i][j]!=INF) {
72                          q.push(make_pair(i,j));
73                          vis[i][j]=1;
74                      }
75                  }
76              spfa(st);
77          }
78          int ansx, ansy, fg = 0;
79          for(int i = 1; i <= n && !fg; ++i)
80              for(int j = 1; j <= m; ++j)
81              if(!G[i][j]) {ansx = i; ansy = j; fg = 1; break;}
82          printf("%d\n", f[stn-1][ansx][ansy]);
83          memset(vis, 0, sizeof vis);
84          dfs(stn-1, ansx, ansy);
85          for(int i = 1; i <= n; i++, puts("")) {
86              for(int j = 1; j <= m; j++) {
87                  if(G[i][j] == 0) putchar('x');
88                  else if(vis[i][j]) putchar('o');
89                  else putchar('_');
90              }
91          }
92      }
93 }
```

1.3.7 Tree Divide and Conquer

```
1 struct Edge {
2     int to, nxt, w;
3 }e[MAXM];
4 int head[MAXN], ecnt;
5 int sz[MAXN];
6 int d[MAXN], t[5], ans;
7 bool vis[MAXN];
8 inline void add_edge(int u, int v, int w) {
9     e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
10    e[++ecnt] = (Edge) {u, head[v], w}; head[v] = ecnt;
11 }
```

```
12  int getsz(int x, int fa) {
13      sz[x] = 1;
14      for(int i = head[x]; i; i = e[i].nxt) {
15          int y = e[i].to;
16          if(vis[y] || y == fa) continue;
17          sz[x] += getsz(y, x);
18      }
19      return sz[x];
20  }
21  int getrt(int x) {
22      int tot = getsz(x, 0) >> 1;
23      while(1) {
24          int u = -1;
25          for(int i = head[x]; i; i = e[i].nxt) {
26              int y = e[i].to;
27              if(vis[y] || sz[y] > sz[x]) continue;
28              if(u == -1 || sz[y] > sz[u]) u = y;
29          }
30          if(~u && sz[u] > tot) x = u;
31          else break;
32      }
33      return x;
34  }
35  void getdep(int x, int fa) {
36      t[d[x]]++;
37      for(int i = head[x]; i; i = e[i].nxt) {
38          int y = e[i].to;
39          if(vis[y] || y == fa) continue;
40          d[y] = (d[x] + e[i].w) % 3;
41          getdep(y, x);
42      }
43  }
44  int cal(int x, int v) {
45      t[0] = t[1] = t[2] = 0;
46      d[x] = v % 3;
47      getdep(x, 0);
48      return t[0] * t[0] + t[1] * t[2] * 2;
49  }
50  void solve(int x) {
51      vis[x] = 1;
52      ans += cal(x, 0);
53      for(int i = head[x]; i; i = e[i].nxt) {
54          int y = e[i].to;
55          if(vis[y]) continue;
56          ans -= cal(y, e[i].w);
57          solve(getrt(y));
58      }
59  }
60  int main() {
61      solve(getrt(1));
62  }
```

1.3.8 Dominator Tree

```
1  #define LL long long
2  #define FILE "dagch"
3  using namespace std;
4
5  const int N = 200010;
6  struct Node{int to,next;}E[N<<1];
```

```
7   int n,m,q,head[N],tot,dfn[N],clo,rev[N],fa[N],semi[N],Ans[N];
8   vector<int>G[N];
9   struct Union_Merge_Set{
10      int fa[N],Mi[N];
11      inline void init(){
12          for(int i=0;i<=n;++i)
13              fa[i]=Mi[i]=semi[i]=i;
14      }
15      inline int find(int x){
16          if(x==fa[x])return x;
17          int fx=fa[x],y=find(fa[x]);
18          if(dfn[semi[Mi[fx]]]<dfn[semi[Mi[x]]])Mi[x]=Mi[fx];
19          return fa[x]=y;
20      }
21  }uset;
22
23  inline void tarjan(int u) {
24      rev[dfn[u] = ++tarjan_time] = u;
25      for(auto v : G[u])
26      if(!dfn[v]) {
27          fa[v] = u;
28          tarjan(v);
29      }
30  }
31  inline void get_semi() {
32      for(int i = tarjan_time; i >= 2; i--) {
33          int u = rev[i], tsemi = n;
34          for(auto v : rG[u]) {
35              if(!dfn[v]) continue;
36              if(dfn[v] < dfn[u]) tsemi = min(tsemi, dfn[x]);
37              else{
38                  uset.find(x);
39                  tsemi = min(tsemi, dfn[semi[uset.Mi[x]]]);
40              }
41          }
42          uset.fa[y] = fa[y];
43          semi[y] = rev[tsemi];
44          Ans[rev[tsemi]]++;
45      }
46  }
47
48  inline void solve() {
49      scanf("%d %d %d", &n, &m, &q);
50      fa[1]=1;
51      for(int i = 1, u, v; i <= m; ++i){
52          scanf("%d%d", &u,&v);
53          link(v,u);
54          G[u].push_back(v);
55      }
56      for(int i = 1; i <= n; i++)
57          if(G[i].size())
58              sort(G[i].begin(), G[i].end());
59
60      uset.init();
61
62      tarjan(1);
63      build();
64      for(int i=1;i<=q;++i)
65          printf("%d ",Ans[gi()]);
66      printf("\n");
67      for(int i=0;i<=n;++i){
```

```
68          G[i].clear();head[i]=0;
69          Ans[i]=semi[i]=fa[i]=0;
70      }
71      clo=tot=0;
72  }
73
74  int main() {
75      int T; scanf("%d", &T);
76      while(T--) solve();
77      return 0;
78  }
```

1.4   LCA

1.4.1   Tree Decomposition LCA

见树链剖分

1.4.2   Tarjan LCA

```
1   vector< pair<int,int> > G[MAXN],ask[MAXN];
2   int fa[MAXN], ans[MAXN], vis[MAXN] ,dis[MAXN];
3   int Find(int x){
4       return x == fa[x] ? x : fa[x] = Find(fa[x]);
5   }
6   void init(int n){
7       memset(ans, 0,sizeof ans);
8       memset(vis, 0,sizeof vis);
9       for(int i = 0; i <= n; i++){
10          G[i].clear();
11          ask[i].clear();
12      }
13  }
14  void LCA(int u){
15      int v;
16      fa[u] = u;
17      vis[u] = true;
18      for(auto it : ask[u])
19          if(vis[v = it.first])
20              ans[it.second] = dis[u] + dis[v] - 2 * dis[Find(it.first)];
21      for(auto it : G[u])
22      if(!vis[v = it.first]){
23          dis[v] = dis[u] + it.second;
24          LCA(v);
25          fa[v] = u;
26      }
27  }
```

1.5   Tarjan

1.5.1   SCC

```
1   namespace SCC{
2       vector<int> G[MAXN];
3       int dfs_clock, scc_cnt, dfn[MAXN], low[MAXN], sccno[MAXN];
4       stack<int> S;
5       void addEdge(int u, int v) {
```

```
6              G[u].push_back(v);
7          }
8      void tarjan(int u) {
9          dfn[u] = low[u] = ++dfs_clock;
10         S.push(u);
11         for(auto v : G[u]) {
12             if(!dfn[v]) {
13                 tarjan(v);
14                 low[u] = min(low[u], low[v]);
15             }else if(!sccno[v]) {
16                 low[u] = min(low[u], dfn[v]);
17             }
18         }
19         if(dfn[u] == low[u]) {
20             scc_cnt++;
21             for(;;) {
22                 int v = S.top(); S.pop();
23                 sccno[v] = scc_cnt;
24                 if(v == u) break;
25             }
26         }
27     }
28     void findSCC(int n) {
29         for(int i = 1; i <= n; i++)
30             if(!dfn[i]) tarjan(i);
31     }
32     void init(int n){
33         dfs_clock = scc_cnt = 0;
34         for(int i = 0;i <= n;++i){
35             dfn[i] = low[i] = sccno[i] = 0;
36             G[i].clear();
37         }
38     }
39 }
```

1.5.2   BCC

```
1  namespace BCC{
2      struct Edge {
3          int to, nxt;
4      }e[MAXM << 1];
5      int ecnt, head[MAXN];
6      int dfs_clock, dfn[MAXN], low[MAXN];
7
8      int is_vertex[MAXN], vbcc_cnt, vbccno[MAXN];
9      vector<int> vbcc[MAXN];
10     stack<int> vS;
11
12     int ebcc_cnt, ebccno[MAXN];
13     stack<int> eS;
14
15     inline void addEdge(int u, int v) {
16         e[++ecnt] = (Edge) {v, head[u]}; head[u] = ecnt;
17         e[++ecnt] = (Edge) {u, head[v]}; head[v] = ecnt;
18     }
19     inline void init(int n) {
20         ecnt = 1;
21         dfs_clock = 0;
22         vbcc_cnt = 0;
23         ebcc_cnt = 0;
```

```
24          for(int i = 1; i <= n; ++i){
25              head[i] = dfn[i] = low[i] = 0;
26              is_vertex[i] = 0;
27              vbccno[i] = 0;
28              ebccno[i] = 0;
29          }
30          while(!vS.empty()) vS.pop();
31      }
32      //root's edge = -1;
33      void tarjan(int u, int edge) {
34          dfn[u] = low[u] = ++dfs_clock;
35          int ch = 0;
36          vS.push(u);
37          eS.push(u);
38          for(int i = head[u], v; i; i = e[i].nxt) {
39              if(!dfn[v = e[i].to]) {
40                  tarjan(v, i ^ 1);
41                  low[u] = min(low[u], low[v]);
42                  if(low[v] >= dfn[u]) {
43                      ++ch;
44                      if(edge > 0 || ch > 1) is_vertex[u] = 1;
45                      vbcc[++vbcc_cnt].clear();
46                      vbcc[vbcc_cnt].push_back(u);
47                      for(int x;;){
48                          x = vS.top();vS.pop();
49                          vbcc[vbcc_cnt].push_back(x);
50                          vbccno[x] = vbcc_cnt;
51                          if(x == v)break;
52                      }
53                  }
54                  if(low[v] > dfn[u]) {
55                      // i && i ^ 1 is bridge
56                  }
57              }
58              else if(dfn[v] < dfn[u] && i != edge)
59                  low[u] = min(low[u], dfn[v]);
60          }
61          if(dfn[u] == low[u]) {
62              ebcc_cnt++;
63              for(int v;;) {
64                  v = eS.top(); eS.pop();
65                  ebccno[v] = ebcc_cnt;
66                  if(v == u) break;
67              }
68          }
69      }
70      void findBCC(int n){
71          for(int i = 1; i <= n; i++)
72              if(!dfn[i]) tarjan(i, -1);
73
74          //findBridge
75          for(int u = 1; u <= n; u++) {
76              for(int i = head[u], v; i; i = e[i].nxt)
77              if(ebccno[u] != ebccno[v = e[i].to]) {
78                  //is bridge
79              }
80          }
81      }
82  }
```

1.6   Cactus

1.6.1   Circle-Square Tree

```
 1  #include <bits/stdc++.h>
 2  using namespace std;
 3  typedef pair<int, int> P;
 4  const int MAXN = 2e4 + 5;
 5  const int S = 15;
 6  namespace Tree {
 7      struct Edge {
 8          int to, nxt, w;
 9      }e[MAXN << 1];
10      int ecnt, head[MAXN];
11      int rt, isrt[MAXN], fa[MAXN][S + 3];
12      int sz[MAXN];
13      inline void addEdge(int u, int v, int w) {
14          e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
15          fa[v][0] = u;
16      }
17  }
18  int n, m, Q;
19  namespace BCC {
20      struct Edge {
21          int to, nxt, w;
22      }e[MAXN << 1];
23      int ecnt, head[MAXN];
24      int dfs_clock, dfn[MAXN], low[MAXN];
25      int is_vertex[MAXN], vbcc_cnt, vbccno[MAXN];
26      vector<P> vbcc[MAXN];
27      stack<P> vs;
28      int tag[MAXN];
29      inline void addEdge(int u, int v, int w) {
30          e[++ecnt] = (Edge) {v, head[u], w}; head[u] = ecnt;
31          e[++ecnt] = (Edge) {u, head[v], w}; head[v] = ecnt;
32      }
33      inline void init(int n) {
34          ecnt = 1;
35          dfs_clock = 0;
36          vbcc_cnt = 0;
37          for(int i = 0; i <= 2 * n; i++){
38              head[i] = dfn[i] = low[i] = 0;
39              vbccno[i] = 0;
40              tag[i] = 0;
41          }
42          while(!vs.empty()) vs.pop();
43      }
44      //root's edge = -1;
45      void tarjan(int u, int edge) {
46          dfn[u] = low[u] = ++dfs_clock;
47          vs.push(P(u, e[edge ^ 1].w));
48          for(int i = head[u], v; i; i = e[i].nxt) {
49              if(!dfn[v = e[i].to]) {
50                  tarjan(v, i ^ 1);
51                  low[u] = min(low[u], low[v]);
52                  if(low[v] >= dfn[u]) {
53                      if(vs.top().first == v) {
54                          Tree::addEdge(u, v, vs.top().second);
55                          vs.pop();
56                          continue;
```

```
57                          }
58                          vbcc[++vbcc_cnt].clear();
59                          vbcc[vbcc_cnt].push_back(P(u, 0));
60                          Tree::isrt[u] = 1;
61                          int &sz = Tree::sz[n + vbcc_cnt];
62                          tag[vs.top().first] = n + vbcc_cnt;
63                          //Tree::addEdge(u, rt, 0);
64                          for(P x;;) {
65                              x = vs.top(); vs.pop();
66                              sz += x.second;
67                              //Tree::addEdge(rt, x.first, sz);
68                              vbcc[vbcc_cnt].push_back(x);
69                              vbccno[x.first] = vbcc_cnt;
70                              if(x.first == v) break;
71                          }
72                      }
73                  }
74              else if(dfn[v] < dfn[u] && i != edge)
75                  low[u] = min(low[u], dfn[v]);
76          }
77          for(int i = head[u], v; i; i = e[i].nxt) {
78              if(tag[v = e[i].to]) {
79                  int r = tag[v]; Tree::sz[r] += e[i].w;
80                  tag[v] = 0;
81              }
82          }
83      }
84      void findBCC(int n) {
85          for(int i = 1; i <= n; i++)
86              if(!dfn[i]) tarjan(i, -1);
87      }
88  }
89  namespace Tree {
90      int dis[MAXN], dep[MAXN], len[MAXN];
91      inline void init(int n) {
92          BCC::init(n);
93          rt = n;
94          ecnt = 1;
95          for(int i = 0; i <= 2 * n; i++) {
96              head[i] = 0;
97              fa[i][0] = isrt[i] = dis[i] = dep[i]  = len[i] = 0;
98          }
99      }
100     void dfs(int x) {
101         for(int i = head[x], y; i; i = e[i].nxt) {
102             if(!dep[y = e[i].to]) {
103                 dep[y] = dep[x] + 1;
104                 dis[y] = dis[x] + e[i].w;
105                 dfs(y);
106             }
107         }
108     }
109     void pre() {
110         for(int k = 1; k <= BCC::vbcc_cnt; k++) {
111             rt++;
112             vector<P> &E = BCC::vbcc[k];
113             addEdge(E[0].first, rt, 0);
114             int cnt = 0;
115             for(int i = E.size() - 1; i >= 1; i--) {
116                 cnt += E[i].second;
117                 len[E[i].first] = cnt;
```

```
118                    addEdge(rt, E[i].first, min(cnt, sz[rt] - cnt));
119                }
120            }
121            for(int k = 1; k <= S; k++) {
122                for(int i = 1; i <= rt; i++) {
123                    fa[i][k] = fa[fa[i][k - 1]][k - 1];
124                }
125            }
126            dep[1] = 1;
127            dfs(1);
128        }
129        int up(int x, int d) {
130            for(int i = S; i >= 0; i--) {
131                if(dep[fa[x][i]] >= d) x = fa[x][i];
132            }
133            return x;
134        }
135        int lca(int u, int v) {
136            if(dep[u] > dep[v]) swap(u, v);
137            v = up(v, dep[u]);
138            if(u == v) return u;
139            for(int i = S; i >= 0; i--) {
140                if(fa[u][i] != fa[v][i]) {
141                    u = fa[u][i], v = fa[v][i];
142                }
143            }
144            return fa[u][0];
145        }
146        int query(int u, int v) {
147            int l = lca(u, v);
148            if(l <= n) return dis[u] + dis[v] - 2 * dis[l];
149            int x = up(u, dep[l] + 1), y = up(v, dep[l] + 1);
150            int res = dis[u] - dis[x] + dis[v] - dis[y];
151            int tmp = abs(len[x] - len[y]);
152            return res + min(tmp, sz[l] - tmp);
153        }
154    }
155
156    int main() {
157        ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
158        using namespace Tree;
159        cin >> n >> m >> Q;
160        init(n);
161        for(int i = 1, u, v, w; i <= m; i++) {
162            cin >> u >> v >> w;
163            BCC::addEdge(u, v, w);
164        }
165        BCC::findBCC(n);
166        pre();
167        int u, v;
168        while(Q--) {
169            cin >> u >> v;
170            cout << query(u, v) << endl;
171        }
172        return 0;
173    }
```

2 Data Structures

2.1 Basic Structures

2.1.1 RMQ

```
1  struct RMQ {
2      int d[MAXN][S + 2];
3      inline void init(int *a, int n) {
4          for(int i = 1; i <= n; i++) d[i][0] = a[i];
5          for(int k = 1; (1 << k) <= n; k++)
6              for(int i = 1; i + (1 << k) - 1 <= n; i++)
7                  d[i][k] = min(d[i][k - 1], d[i + (1 << (k - 1))][k - 1]);
8      }
9      inline int query(int l, int r) {
10         if(l > r) swap(l, r);
11         int k = 0;
12         while((1 << (k + 1)) <= r - l + 1) k++;
13         return min(d[l][k], d[r - (1 << k) + 1][k]);
14     }
15 }rmq;
16 const int MAXM = 2e5 + 5, MAXN = 3e6 + 5, S = 22;
17 const LL INF = 1e18;
18 #define belong(x) (x / S + 1)
19 #define pos(x) (x % S + 1)
20 int Log[MAXN], sz;
21 struct RMQ {
22     LL a[MAXN];
23     LL d[MAXM][S + 2];
24     LL pre[MAXM][S + 2], aft[MAXM][S + 2];
25     inline void init(int n) {
26         sz = n / S + 1;
27         Log[0] = -1; for(int i = 1; i <= n; i++) Log[i] = Log[i / 2] + 1;
28         for(int i = 1; i <= sz; i++) {
29             pre[i][0] = aft[i][S + 1] = INF;
30         }
31         for(int i = 1; i <= n; i++) {
32             pre[belong(i)][pos(i)] = min(pre[belong(i)][pos(i) - 1], a[i]);
33         }
34         for(int i = n; i >= 1; i--) {
35             aft[belong(i)][pos(i)] = min(aft[belong(i)][pos(i) + 1], a[i]);
36         }
37         for(int i = 1; i <= sz; i++) {
38             d[i][0] = aft[i][1];
39         }
40         for(int k = 1; k <= S; k++)
41             for(int i = 1; i + (1 << k) <= sz; i++)
42                 d[i][k] = min(d[i][k - 1], d[i + (1 << (k - 1))][k - 1]);
43     }
44     inline LL ask(int l, int r) {
45         assert(l <= r);
46         LL res = INF;
47         if(belong(l) == belong(r)) {
48             for(int i = l; i <= r; i++) res = min(res, a[i]);
49             return res;
50         }
51         res = min(aft[belong(l)][pos(l)], pre[belong(r)][pos(r)]);
52         int k = Log[belong(r) - belong(l) - 1];
53         if(~k) {
```

```
54              res = min(res, d[belong(l) + 1][k]);
55              res = min(res, d[belong(r) - (1 << k)][k]);
56          }
57          return res;
58      }
59  }rmq;
```

2.1.2   Divide Blocks

```
1   int belong[MAXN], l[MAXN], r[MAXN];
2   int sz, num;
3   void build(int n) {
4       sz = sqrt(n);
5       num = n / sz; if(n % sz) num++;
6       for(int i = 1; i <= num; i++) {
7           l[i] = (i - 1) * sz + 1;
8           r[i] = i * sz;
9       }
10      r[num] = n;
11      for(int i = 1; i <= n; i++) {
12          belong[i] = (i - 1) / sz + 1;
13      }
14  }
```

2.2   Heap Structures

2.2.1   Leftist Tree

```
1   const int MAXN = ;
2   namespace LeftistTree{
3       int ls[MAXN], rs[MAXN];
4       int dis[MAXN];
5       int fg[MAXN], sfa[MAXN], rt[MAXN];//利用rt得到堆根节点
6       int val[MAXN];
7
8       void push_down(int x) {};
9       void push_up(int x) {
10          if(dis[ls[x]] < dis[rs[x]]) swap(ls[x], rs[x]);
11          dis[x] = dis[rs[x]] + 1;
12      }
13      int merge(int x, int y) {
14          if(!x || !y) return x^y;
15          if(val[x] > val[y] || (val[x] == val[y] && x > y)) swap(x, y);
16          push_down(x);
17          rs[x]  = merge(rs[x], y);
18          push_up(x);
19          return x;
20      }
21      int getSfa(int x) {return sfa[x] == x ? x : sfa[x] = getSfa(sfa[x]);}
22      int uni(int x, int y) {//返回合并后的根
23          if(!x || !y) return x^y;
24          if(fg[x] || fg[y]) return;
25          x = getSfa(x); y = getSfa(y);
26          if(x == y) return;
27          int z = merge(x, y);
28          return sfa[x] = sfa[y] = sfa[z] = z;
29      }
30      void uni2(int a, int b) {
```

```
31            //val[a] >>= 1;
32            int c = merge(ls[a], rs[a]);
33            ls[a] = rs[a] = dis[a] = 0;
34            int a1 = merge(c, a);
35            //val[b] >>= 1;
36            c = merge(ls[b], rs[b]);
37            ls[b] = rs[b] = dis[b] = 0;
38            int b1 = merge(c, b);
39            c = merge(a1, b1);
40            sfa[a] = sfa[b] = sfa[c] = c;
41            printf("%d\n", val[c])
42        }
43        int pop(int x) {//返回堆顶值,也可以用于返回根
44            if(!x || fg[x]) return -1;
45            x = getSfa(x); fg[x] = 1;
46            push_down(x);//在删除堆顶时要下传标记
47            int y = merge(ls[x], rs[x]);
48            sfa[x] = sfa[y] = y;
49            return val[x];
50        }
51        void init(int n) {
52            for(int i = 1; i <= n; i++) {
53                sfa[i] = i;
54                ls[i] = rs[i] = dis[i] = fg[i] = 0;
55            }
56        }
57    }
58    //可持久化版本见k短路
```

2.3   Sequence Structures

2.3.1   Cartesian Tree

```
1   struct CartesianTree{
2       int rt, fa[MAXN], ls[MAXN], rs[MAXN];
3       int top, st[MAXN];
4       int cnt[MAXN];
5       void build(LL *a,int n) {
6           top = rt = 0;
7           for(int i = 1; i <= n; i++) {
8               ls[i] = rs[i] = fa[i] = 0;
9               while(top && a[st[top]] > a[i]) ls[i] = st[top--];
10              fa[i] = st[top];
11              if(ls[i]) fa[ls[i]] = i;
12              if(fa[i]) rs[fa[i]] = i; else rt = i;
13              st[++top] = i;
14          }
15      }
16      void dfs(int x) {
17          cnt[x] = 1;
18          if(ls[x]) {dfs(ls[x]); cnt[x] += cnt[ls[x]];}
19          if(rs[x]) {dfs(rs[x]); cnt[x] += cnt[rs[x]];}
20      }
21      LL getAns(LL *a, int n) {
22          //dfs(rt);
23          //————
24          return res;
25      }
26  }T;
```

2.3.2 TreeArray

```
1  //树状数组上二分
2  int BS(int x) {
3      int res = 0;
4      for (int i = 1 << 18; i; i >>= 1)
5          if ((res | i) <= Tn && T[res | i] <= x)
6              x -= T[res |= i];
7      return res;
8  }
```

2.3.3 Segment Tree

```
1  #define Ls(x) (x << 1)
2
3  #define Rs(x) (x << 1 | 1)
4  struct Tree {
5      int l, r, lazy;
6      LL sum, mx;
7  }tree[MAXN << 2];
8  int A[MAXN];
9  void push_up(int x) {
10     tree[x].sum = tree[Ls(x)].sum + tree[Rs(x)].sum;
11     tree[x].mx = max(tree[Ls(x)].mx, tree[Rs(x)].mx);
12 }
13 void push_down(int x) {
14     if(tree[x].lazy) {
15         tree[Ls(x)].sum += tree[x].lazy * (tree[Ls(x)].r - tree[Ls(x)].l + 1);
16         tree[Rs(x)].sum += tree[x].lazy * (tree[Rs(x)].r - tree[Rs(x)].l + 1);
17         tree[Ls(x)].mx += tree[x].lazy;
18         tree[Rs(x)].mx += tree[x].lazy;
19         tree[Ls(x)].lazy += tree[x].lazy;
20         tree[Rs(x)].lazy += tree[x].lazy;
21         tree[x].lazy = 0;
22     }
23 }
24 void build(int x, int L, int R) {
25     tree[x].lazy = 0;
26     tree[x].l = L; tree[x].r = R;
27     if(L == R) {
28         tree[x].sum = A[L];
29         tree[x].mx = A[L];
30
31         return;
32     }
33     int mid = (L + R) >> 1;
34     build(Ls(x), L, mid);
35     build(Rs(x), mid + 1, R);
36     push_up(x);
37 }
38 void update(int x, int L, int R, LL val) {
39     if(tree[x].l >= L && tree[x].r <= R) {
40         tree[x].lazy += val;
41         tree[x].sum += val * (tree[x].r - tree[x].l + 1);
42         tree[x].mx += val;
43         return;
44     }
45     push_down(x);
46     int mid = (tree[x].l + tree[x].r) >> 1;
```

```
47        if(L <= mid) update(Ls(x), L, R, val);
48        if(R > mid) update(Rs(x), L, R, val);
49        push_up(x);
50    }
51    LL query(int x, int L, int R) {
52        if(tree[x].l >= L && tree[x].r <= R)
53            return tree[x].sum;
54        push_down(x);
55        int mid = (tree[x].l + tree[x].r) >> 1;
56        LL res = 0;
57        if(L <= mid) res += query(Ls(x), L, R);
58        if(R > mid) res += query(Rs(x), L, R);
59
60        return res;
61    }
62    LL query2(int x, int L, int R) {
63        if(tree[x].l >= L && tree[x].r <= R)
64            return tree[x].mx;
65        push_down(x);
66        int mid = (tree[x].l + tree[x].r) >> 1;
67        LL res = -INF;
68        if(L <= mid) res = max(res, query2(Ls(x), L, R));
69        if(R > mid) res = max(res, query2(Rs(x), L, R));
70        return res;
71    }
```

2.3.4   LiChao Tree

```
1    const double eps = 1e-12;
2    namespace LiT{
3        const int MLIMIT = 40000;
4        typedef double LD;
5        struct line{LD k,b;int l,r,id;} T[MAXN << 2];
6        //inline LD calc(line &a,int pos) {return a.k*vec[pos]+a.b;}
7        inline LD calc(line &a,int pos) {return a.k*pos+a.b;}
8        inline double cross(line &a,line &b) {
9            if(b.k == a.k) return -1e9;
10           return (double)(a.b-b.b)/(b.k-a.k);
11       }
12       void build(int v, int l, int r) {
13           T[v].k = 0;T[v].b = -1e18;
14           T[v].l = 0;T[v].r = MLIMIT;
15           T[v].id = 0;
16           if(l == r)return;
17           int mid = (l+r)>>1;
18           build(v<<1,l,mid);
19           build(v<<1|1,mid+1,r);
20       }
21       void ins(int v,int l,int r, line k) {
22           if(k.l <= l && r <= k.r) {
23               LD fl = calc(k, l), fr = calc(k, r);
24               LD gl = calc(T[v], l), gr = calc(T[v], r);
25               if(fl - gl > eps && fr - gr > eps) T[v] = k;
26               else if(fl - gl > eps || fr - gr > eps) {
27                   int mid = (l+r)>>1;
28                   if(calc(k, mid) - calc(T[v], mid) > eps) swap(k, T[v]);
29                   //if(vec[mid] - cross(k, T[v]) > eps)
30                   if(mid - cross(k, T[v]) > eps)
31                       ins(v<<1, l, mid, k);else ins(v<<1|1, mid+1, r, k);
32               }
```

```
33          return;
34        }
35        int mid=(l+r)>>1;
36        if(k.l <= mid) ins(v<<1, l, mid, k);
37        if(mid < k.r)  ins(v<<1|1, mid+1, r, k);
38      }
39      LD ans;int ansid;
40      void que(int v, int l, int r, int x) {
41        LD tmp = calc(T[v], x);
42        if(tmp > ans || (tmp == ans && T[v].id < ansid)) {
43            ans = tmp;
44            ansid = T[v].id;
45        }
46        if(l == r) return;
47        int mid = (l+r)>>1;
48        if(x <= mid) que(v<<1,l,mid,x);else que(v<<1|1,mid+1,r,x);
49      }
50    };
51    //左闭右闭
```

2.3.5 Splay Tree

```
1    namespace splay{
2      int n, m, sz, rt;
3      int val[MAXN], id[MAXN];
4      int tr[MAXN][2], size[MAXN], fa[MAXN], rev[MAXN], s[MAXN], lazy[MAXN];
5      void push_up(int x) {
6        int l = tr[x][0], r = tr[x][1];
7        s[x] = max(val[x], max(s[l], s[r]));
8        size[x] = size[l] + size[r] + 1;
9      }
10     void push_down(int x) {
11       int l = tr[x][0], r = tr[x][1];
12       if(lazy[x]) {
13           if(l) {
14               lazy[l] += lazy[x];
15               s[l] += lazy[x];
16               val[l] += lazy[x];
17           }
18           if(r) {
19               lazy[r] += lazy[x];
20               s[r] += lazy[x];
21               val[r] += lazy[x];
22           }
23           lazy[x] = 0;
24       }
25       if(rev[x]) {
26           rev[x] = 0;
27           rev[l] ^= 1; rev[r] ^= 1;
28           swap(tr[x][0], tr[x][1]);
29       }
30     }
31     void rotate(int x, int &k) {
32       int y = fa[x];
33       int z = fa[y];
34       int l, r;
35       if(tr[y][0] == x) l = 0;
36       else l = 1;
37       r = l ^ 1;
38       if(y == k) k = x;
```

```
39            else {
40                if(tr[z][0] == y) tr[z][0] = x;
41                else tr[z][1] = x;
42            }
43            fa[x] = z; fa[y] = x; fa[tr[x][r]] = y;
44            tr[y][l] = tr[x][r]; tr[x][r] = y;
45            push_up(y); push_up(x);
46        }
47        void splay(int x, int &k) {
48            int y, z;
49            while(x != k) {
50                y = fa[x];
51                z = fa[y];
52                if(y != k) {
53                    if((tr[y][0] == x) ^ (tr[z][0] == y)) rotate(x, k);
54
55                    else rotate(y, k);
56                }
57                rotate(x, k);
58            }
59        }
60        int find(int x, int rank) {
61            push_down(x);
62
63            int l = tr[x][0], r = tr[x][1];
64            if(size[l] + 1 == rank) return x;
65            else if(size[l] >= rank) return find(l, rank);
66            else return find(r, rank - size[l] - 1);
67        }
68        void update(int l, int r, int v) {
69            int x = find(rt, l), y = find(rt, r + 2);
70            splay(x, rt); splay(y, tr[x][1]);
71            int z = tr[y][0];
72            lazy[z] += v;
73            val[z] += v;
74            s[z] += v;
75        }
76        void reverse(int l, int r) {
77            int x = find(rt, l), y = find(rt, r + 2);
78            splay(x, rt); splay(y, tr[x][1]);
79            int z = tr[y][0];
80            rev[z] ^= 1;
81        }
82        void query(int l, int r) {
83            int x = find(rt, l), y = find(rt, r + 2);
84            splay(x, rt); splay(y, tr[x][1]);
85            int z = tr[y][0];
86            printf("%d\n", s[z]);
87        }
88        void build(int l, int r, int f) {
89            if(l > r) return;
90            int now = id[l], last = id[f];
91            if(l == r) {
92                fa[now] = last; size[now] = 1;
93                if(l < f) tr[last][0] = now;
94                else tr[last][1] = now;
95                return;
96            }
97            int mid = (l + r) >> 1; now = id[mid];
98            build(l, mid - 1, mid); build(mid + 1, r, mid);
99            fa[now] = last;
```

```
100              push_up(now);
101              if(mid < f) tr[last][0] = now;
102              else tr[last][1] = now;
103          }
104          void init() {
105              s[0] = -INF;
106              scanf("%d%d", &n, &m);
107              for(int i = 1; i <= n + 2; i++) id[i] = ++sz;
108              build(1, n + 2, 0); rt = (n + 3) >> 1;
109          }
110      }
111      namespace splay{
112          //内存回收池见fhq_treap
113          int tcnt, root;
114          int sz[MAXN];
115          int tr[MAXN][2], fa[MAXN];
116          int val[MAXN];
117          int newnode(int w) {
118              ++tcnt;
119              sz[tcnt] = 1;
120              fa[tcnt] = tr[tcnt][0] = tr[tcnt][1] = 0;
121              //val[tcnt] = w;
122              return tcnt;
123          }
124          void push_up(int v) {
125              int l = tr[v][0], r = tr[v][1];
126              sz[v] = sz[l] + 1 + sz[r];
127          }
128          void push_down(int v) {
129              if(!v) return;
130          }
131          void init() {
132              tcnt = 2;
133              tr[root = fa[1] = 2][0] = 1;
134              sz[1] = 1; sz[2] = 2;
135              //val[1] = -INF; val[2] = INF;//权值平衡树
136              //val[1] = val[2] = 0;//位置平衡树
137              //1,2为哨兵节点,根据题意也可设置为n+1,n+2或1,n+1
138          }
139          void rotate(int x) {
140              int y = fa[x], z = fa[y];
141              push_down(y);push_down(x);
142              int lr = tr[y][1] == x;
143              if(z) tr[z][tr[z][1]==y] = x;
144              fa[x] = z;
145              fa[tr[y][lr] = tr[x][lr^1]] = y;
146              fa[tr[x][lr^1] = y] = x;
147              push_up(y); push_up(x);
148          }
149          void splay(int x, int k) {
150              for(int y, z; (y = fa[x]) != k; rotate(x)) {
151                  if((z = fa[y]) != k) {
152                      if((tr[y][0] == x) ^ (tr[z][0] == y))
153                          rotate(x); else rotate(y);
154                  }
155              }
156              if(!k) root = x;
157          }
158          int find(int x, int rank) {
159              push_down(x);
160              int l = tr[x][0], r = tr[x][1];
```

```
161        if(sz[l] + 1 == rank) return x;
162        if(sz[l] >= rank) return find(l, rank);
163        return find(r, rank - sz[l] - 1);
164    }
165    int build(int l, int r) {
166        if(l > r) return 0;
167        if(l == r) {
168            int num;scanf("%d", &num);
169            return newnode(num);
170        }
171        int mid = (l + r) >> 1;
172        int ls = build(l, mid-1);
173        int num; scanf("%d", &num);
174        int v = newnode(num);
175        int rs = build(mid+1, r);
176        if(ls) fa[ls] = v;
177        tr[v][0] = ls;
178        if(rs) fa[rs] = v;
179        tr[v][1] = rs;
180        push_up(v);
181        return v;
182    }
183    void insert(int pos, ...) {
184        int x = find(root, pos+1), y = find(root, pos+2);
185        splay(x, 0); splay(y, x);
186        //int z = newnode(w);//插入一个节点
187        //int z = build(1, n);//插入n个节点
188        fa[tr[y][0] = z] = y;
189        splay(z,0);
190    }
191    void modifyOrQuery(int l, int r, int v) {
192        int x = find(root, l), y = find(root, r + 2);
193        splay(x, 0); splay(y, x);
194        int z = tr[y][0];
195        if(!z) return;
196        //标记对本身无效,处理时将z点重新计算
197        splay(z,0);
198    }
199    void display(int v) {
200        if(!v) return;
201        push_down(v);
202        display(tr[v][0]);
203        if(val[v]) printf("%d ", val[v]);
204        display(tr[v][1]);
205    }
206    /*int findValue(int v) {
207        int res = root;
208        for(int cur = root;cur; res = cur, cur = tr[cur][val[cur] <= v]);
209        return res;
210    }
211    void insert(int w) {
212        int y = findValue(w);
213        int z = newnode(w);
214        fa[tr[y][val[y] <= w] = z] = y;
215        splay(z,0);
216    }*/
217    /*void split(int v) {//splay维护区间[l,r],区间分裂为[l,k-1],[k,k],[k+1,r];
218        //ump查看标号是否出现,mp维护子区间左端点
219        if(ump.find(v) == ump.end()) {
220            auto it = mp.upper_bound(v); --it;
221            int z = it->second;
```

```
222                splay(z, 0);
223                int pos = sz[tr[z][0]];
224                int x = find(root,pos), y = find(root,pos+rc[z]-lc[z]+2);
225                splay(x, 0); splay(y, x);
226                z = tr[y][0];
227                if(lc[z] != v) {
228                    tr[z][0] = newnode(lc[z],v-1);
229                    fa[tr[z][0]] = z;
230                    mp[lc[z]] = tr[z][0];
231                }
232                if(rc[z] != v) {
233                    tr[z][1] = newnode(v+1,rc[z]);
234                    fa[tr[z][1]] = z;
235                    mp[v+1] = tr[z][1];
236                }
237                lc[z] = rc[z] = v;
238                splay(z,0);
239                mp[v] = z;
240                ump[v] = z;
241            }
242        }*/
243
244 }
```

2.3.6 Scapegoat Tree

```
1  struct ScapegoatTree{
2      int Tsn; queue<int> q;
3      int val[MAXM], ext[MAXM];
4      int sz[MAXM], tsz[MAXM];
5      int fa[MAXM], tr[MAXM][2];
6      int root;
7      double alp;
8      void init() {
9          root = 0;
10         alp = 0.7;
11     }
12     int newnode(int x) {
13         if(q.empty()) q.push(++Tsn);
14         int tcnt = q.front(); q.pop();
15         val[tcnt] = x; ext[tcnt] = 1;
16         fa[tcnt] = tr[tcnt][0] = tr[tcnt][1] = 0;
17         sz[tcnt] = tsz[tcnt] = 0;
18         return tcnt;
19     }
20     void push_up(int v) {
21         sz[v] = ext[v];tsz[v] = 1;
22         if(tr[v][0]) {
23             sz[v] += sz[tr[v][0]];
24             tsz[v] += tsz[tr[v][0]];
25         }
26         if(tr[v][1]) {
27             sz[v] += sz[tr[v][1]];
28             tsz[v] += tsz[tr[v][1]];
29         }
30     }
31     bool isBad(int v) {
32         return (double(tsz[ tr[v][0] ]) > double(tsz[v]) * alp) ||
33             (double(tsz[ tr[v][1] ]) > double(tsz[v]) * alp) ||
34             (sz[v] * 2 < tsz[v]);
```

```
35          }
36          vector<int> vec;
37          void rRecycle(int v) {
38              if(tr[v][0]) rRecycle(tr[v][0]);
39              if(ext[v]) vec.push_back(v); else q.push(v);
40              if(tr[v][1]) rRecycle(tr[v][1]);
41          }
42          int rBuild(int l, int r) {
43              int mid = (l + r) >> 1, v = vec[mid];
44              tr[v][0] = (l <= mid-1) ? rBuild(l, mid - 1) : 0;
45              if(tr[v][0]) fa[tr[v][0]] = v;
46              tr[v][1] = (mid+1 <= r) ? rBuild(mid + 1, r) : 0;
47              if(tr[v][1]) fa[tr[v][1]] = v;
48              push_up(v);
49              return v;
50          }
51          void rebuild(int x) {
52              int v = 0;
53              for(;x; x= fa[x]) {
54                  push_up(x);
55                  if(isBad(x)) v = x;
56              }
57              if(v && isBad(v)){
58                  vec.clear();
59                  int u = fa[v], lr = tr[u][1] == v;
60                  rRecycle(v);
61                  if(vec.size()) v = rBuild(0, vec.size() - 1); else v = 0;
62                  if(u == 0) fa[root = v] = 0;
63                  else{
64                      tr[u][lr] = v;
65                      if(v) fa[v] = u;
66                  }
67              }
68          }
69          void ins(int x) {
70              int p = root, q = root;
71              for(;p && val[p] != x; q = p, p = tr[p][ val[p] < x]) ;
72              if(!q) {
73                  p = root = newnode(x);
74              }else if(p) {
75                  ext[p]++;
76              }else{
77                  fa[p = tr[q][val[q] < x] = newnode(x) ] = q;
78              }
79              rebuild(p);
80          }
81          void del(int x) {
82              int p = root;
83              for(;p && val[p] != x; p = tr[p][ val[p] < x]);
84              if(p && ext[p]){
85                  --ext[p];
86                  rebuild(p);
87              }
88          }
89          int get_rank(int x) {
90              int ret = 0;
91              for(int p = root;p;) {
92                  if(val[p] < x) {
93                      ret += sz[tr[p][0]] + ext[p];
94                      p = tr[p][1];
95                  }else p = tr[p][0];
```

```
96              }
97              return ret + 1;
98          }
99          int get_Kth(int p, int k) {
100             if(sz[tr[p][0]] >= k) return get_Kth(tr[p][0] ,k);
101             k -= sz[tr[p][0]];
102             if(ext[p] >= k) return val[p];
103             k -= ext[p];
104             return get_Kth(tr[p][1], k);
105         }
106         int pre(int x) {
107             int id = get_rank(x);
108             return get_Kth(root, id - 1);
109         }
110         int nxt(int x) {
111             int id = get_rank(x + 1);
112             return get_Kth(root ,id);
113         }
114         void display(int v) {
115             if(tr[v][0]) display(tr[v][0]);
116             cerr<<val[v]<<" ";
117             if(tr[v][1]) display(tr[v][1]);
118         }
119 }T;
```

2.3.7 FHQ Treap

```
1   namespace fhq_treap{
2       int Tsz; queue<int> q; //内存回收池
3       int tcnt, root;
4       //int rt[MAXN]; //可持久化时使用rt,维护版本号(int &root)，空间开大
5       int sz[MAXN], rnd[MAXN];
6       int tr[MAXN][2];
7       //int fa[MAXN];//维护fa时除了在pushup更新v节点左右孩子父节点信息还要在split和merge结
        束时维护root的fa信息,fa[root]=0;
8       int val[MAXN], rev[MAXN];
9       void init() {
10          srand(time(0));
11          Tsz = tcnt = root = 0;
12      }
13      int newnode(int v) {
14          if(q.empty()) q.push(++Tsz);
15          tcnt = q.front(); q.pop();
16          sz[tcnt] = 1;
17          rnd[tcnt] = rand();
18          tr[tcnt][0] = tr[tcnt][1] = 0;
19          //val[tcnt] = v;
20          return tcnt;
21      }
22      /*int copynode(int id) {
23          //++tcnt;//获取一个新的节点编号
24          sz[tcnt] = sz[id];
25          rnd[tcnt] = rnd[id];
26          tr[tcnt][0] = tr[id][0];
27          tr[tcnt][1] = tr[id][1];
28          //val[tcnt] = val[id];
29          return tcnt;
30      }*/
31      void push_up(int v) {
32          int l = tr[v][0], r = tr[v][1];
```

```
33              sz[v] = sz[l] + 1 + sz[r];
34         }
35     void push_down(int v) {
36         if(!v) return;
37         int l = tr[v][0], r = tr[v][1];
38         //if(l) ;
39         //if(r) ;
40         //swap时候注意交换tr[v][0]和tr[v][1],而不是l和r;
41     }
42     /* //可持久化在push_down是要新建节点,否者历史版本有可能被多次下传,以rev为例
43     void push_down(int v) {
44         if(!v || !rev[v]) return;
45         int &l = tr[v][0], &r = tr[v][1];
46         if(l) {
47             l = copynode(l);
48             rev[l]^=1;
49             swap(tr[l][0],tr[l][1]);
50         };
51         if(r) {
52             r = copynode(r);
53             rev[r]^=1;
54             swap(tr[r][0],tr[r][1]);
55         };
56         rev[v] = 0;
57     }*/
58     void split(int v,int k,int &x,int &y) {
59         if(!v) {x=y=0;return;}
60         push_down(v);
61         //v = copynode(v);//可持久化时复制节点
62         /*if(k > sz[tr[v][0]]) {
63             x = v;
64             split(tr[v][1], k-sz[tr[v][0]]-1, tr[v][1], y);
65         }else{
66             y = v;
67             split(tr[v][0], k , x ,tr[v][0]);
68         }*/
69         if(val[v] <= k) {
70             x = v;
71             split(tr[v][1], k, tr[v][1], y);
72         }else{
73             y = v;
74             split(tr[v][0], k, x, tr[v][0]);
75         }
76         push_up(v);
77     }
78     int merge(int x,int y) {//x堆所有值均小于y堆
79         if(!x || !y) return x|y;
80         push_down(x); push_down(y);
81         if(rnd[x]<rnd[y]){
82             //x = copynode(x);//可持久化时复制节点,可不写
83             tr[x][1] = merge(tr[x][1],y);
84             push_up(x);
85             return x;
86         }else{
87             //y = copynode(y);//可持久化时复制节点,可不写
88             tr[y][0] = merge(x,tr[y][0]);
89             push_up(y);
90             return y;
91         }
92     }
93     void insert(int k) {
```

```
 94          int x,y;
 95          split(root,k,x,y);
 96          root = merge(merge(x,newnode(k)),y);
 97      }
 98      void recycle(int v) {//回收一颗treap上所有节点
 99          if(!v) return;
100          q.push(v);
101          recycle(tr[v][0]); recycle(tr[v][1]);
102      }
103      void erase(int k) {
104          int x,y,z;
105          split(root,k,x,y);
106          split(x,k-1,x,z);
107          z = merge(tr[z][0],tr[z][1]);
108          root = merge(x,merge(z,y));
109      }
110      void krank(int k) {
111          int x,y;
112          split(root,k-1,x,y);
113          printf("%d\n",sz[x]+1);
114          root = merge(x,y);
115      }
116      int find(int v,int k) {
117          if(sz[tr[v][0]]==k-1) return val[v];
118          if(sz[tr[v][0]]>=k) return find(tr[v][0],k);
119          return find(tr[v][1],k-sz[tr[v][0]]-1);
120      }
121      void pre(int k) {
122          int x,y;
123          split(root,k-1,x,y);
124          printf("%d\n",find(x,sz[x]));
125          root=merge(x,y);
126      }
127      void nxt(int k){
128          int x,y;
129          split(root,k,x,y);
130          printf("%d\n",find(y,1));
131          root=merge(x,y);
132      }
133      void reverse(int l,int r){
134          int x,y,z;
135          split(root, r, x, y);
136          split(x, l-1, x, z);
137          //rev[z] ^= 1;标记对本身无效,处理时将z点重新计算
138          root = merge(merge(x,z),y);
139      }
140      /*int getRank(int S) {
141          int res = sz[tr[S][0]]+1;
142          for(;fa[S];S = fa[S])
143              if(tr[fa[S]][1] == S) res += sz[tr[fa[S]][0]] + 1;
144          return res;
145      }*/
146      void display(int v) {
147          if(!v) return;
148          push_down(v);
149          display(tr[v][0]);
150          printf("%d ",val[v]);
151          display(tr[v][1]);
152      }
153  }
```

154   *//一种可持久化平衡树的替代(非强制在线),由历史版本向当前版本连边,在dfs遍历中利用权值树状*
       数组,普通平衡树等获取答案

2.4 Persistent Data Structures

2.4.1 Chairman Tree

```
1  struct Node {
2      int l, r;
3      LL sum;
4  }t[MAXN * 40];
5  int cnt, n;
6  int rt[MAXN];
7  void update(int pre, int &x, int l, int r, int v) {
8      x = ++cnt; t[x] = t[pre]; t[x].sum++;
9      if(l == r) return;
10     int mid = (l + r) >> 1;
11     if(v <= mid) update(t[pre].l, t[x].l, l, mid, v);
12     else update(t[pre].r, t[x].r, mid + 1, r, v);
13 }
14 int query(int x, int y, int l, int r, int v) {
15     if(l == r) return l;
16     int mid = (l + r) >> 1;
17     int sum = t[t[y].l].sum - t[t[x].l].sum;
18     if(sum >= v) return query(t[x].l, t[y].l, l, mid, v);
19     else return query(t[x].r, t[y].r, mid + 1, r, v - sum);
20 }
```

2.4.2 Unite Chairman Tree

```
1  //Q x到y路径第k大
2  //L link(x, y)
3  #include <bits/stdc++.h>
4  using namespace std;
5  typedef int LL;
6  const int MAXN = 8e4 + 5;
7  const int S = 18;
8  struct Node {
9      int l, r;
10     LL sum;
11 }t[MAXN * 800]; //2 * log^2(n)
12 int n, m, Q;
13 int cnt;
14 int rt[MAXN], sz[MAXN];
15 void update(int pre, int &x, int l, int r, int v) {
16     x = ++cnt; t[x] = t[pre]; t[x].sum++;
17     if(l == r) return;
18     int mid = (l + r) >> 1;
19     if(v <= mid) update(t[pre].l, t[x].l, l, mid, v);
20     else update(t[pre].r, t[x].r, mid + 1, r, v);
21 }
22 int query(int x, int y, int z, int w, int l, int r, int v) {
23     if(l == r) return l;
24     int mid = (l + r) >> 1;
25     int sum = t[t[x].l].sum + t[t[y].l].sum - t[t[z].l].sum - t[t[w].l].sum;
26     if(sum >= v) return query(t[x].l, t[y].l, t[z].l, t[w].l, l, mid, v);
27     return query(t[x].r, t[y].r, t[z].r, t[w].r, mid + 1, r, v - sum);
28 }
```

```cpp
29  int fa[MAXN][S + 3], dep[MAXN];
30  int val[MAXN];
31  vector<int> G[MAXN];
32  inline void addEdge(int x, int y) {
33      G[x].push_back(y);
34      G[y].push_back(x);
35  }
36  inline void upd(int x) {
37      update(rt[fa[x][0]], rt[x], 1, n, val[x]);
38      for(int i = 1; i <= S; i++) fa[x][i] = fa[fa[x][i - 1]][i - 1];
39  }
40  inline void Go(int &x, int step) {
41      for(int i = S; i >= 0; i--) if(step >> i & 1) x = fa[x][i];
42  }
43  int lca(int x, int y) {
44      if(dep[x] < dep[y]) swap(x, y);
45      Go(x, dep[x] - dep[y]);
46      if(x == y) return x;
47      for(int i = S; i >= 0; i--) if(fa[x][i] != fa[y][i]) {
48          x = fa[x][i], y = fa[y][i];
49      }
50      return fa[x][0];
51  }
52  int get_rt(int x) {
53      for(int i = S; i >= 0; i--) if(fa[x][i]) x = fa[x][i];
54      return x;
55  }
56  void dfs(int x, int f) {
57      fa[x][0] = f;
58      dep[x] = dep[f] + 1;
59      upd(x);
60      for(auto y : G[x]) {
61          if(y == f) continue;
62          dfs(y, x);
63      }
64  }
65  void unite(int x, int y) {
66      int rx = get_rt(x), ry = get_rt(y);
67      if(sz[rx] > sz[ry]) swap(x, y), swap(rx, ry);
68      addEdge(x, y);
69      dfs(x, y);
70      sz[ry] += cnt - rt[x] + 1;
71  }
72  void init() {
73      cnt = 0;
74      for(int i = 1; i <= n; i++) {
75          rt[i] = sz[i] = 0;
76          G[i].clear();
77          dep[i] = 0;
78      }
79  }
80  int main() {
81      ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
82      int T; cin >> T;
83      while(T--) {
84          cin >> n >> m >> Q;
85          init();
86          vector<int> b;
87          map<int, int> mp;
88          for(int i = 1; i <= n; i++) {
89              cin >> val[i];
```

```
 90                        b.push_back(val[i]);
 91                }
 92                sort(b.begin(), b.end());
 93                b.erase(unique(b.begin(), b.end()), b.end());
 94                for(int i = 1, tmp; i <= n; i++) {
 95                        tmp = val[i];
 96                        val[i] = lower_bound(b.begin(), b.end(), val[i]) - b.begin() + 1;
 97                        mp[val[i]] = tmp;
 98                }
 99                for(int i = 1, u, v; i <= m; i++) {
100                        cin >> u >> v;
101                        addEdge(u, v);
102                }
103                for(int i = 1; i <= n; i++) if(!dep[i]) {
104                        dep[i] = 1;
105                        dfs(i, 0);
106                        sz[i] = cnt - rt[i] + 1;
107                }
108                char s[3]; int x, y, z, k, ans = 0;
109                while(Q--) {
110                        cin >> s >> x >> y;
111                        x ^= ans; y ^= ans;
112                        if(s[0] == 'Q') {
113                                cin >> k; k ^= ans;
114                                z = lca(x, y);
115                                ans = query(rt[x], rt[y], rt[z], rt[fa[z][0]], 1, n, k);
116                                ans = mp[ans];
117                                cout << ans << endl;
118                        }
119                        else {
120                                unite(x, y);
121                        }
122                }
123        }
124        return 0;
125 }
```

2.4.3 Persistent Trie

```
 1  //区间异或最值查询
 2  const int N=5e4+10;
 3  int t[N];
 4  int ch[N*32][2],val[N*32];
 5  int cnt;
 6  void init(){
 7      mem(ch,0);
 8      mem(val,0);
 9      cnt=1;
10  }
11  int add(int root,int x){
12      int newroot=cnt++,ret=newroot;
13      for(int i=30;i>=0;i--){
14          ch[newroot][0]=ch[root][0];
15          ch[newroot][1]=ch[root][1];
16          int now=(x>>i)&1;
17          root=ch[root][now];
18
19          ch[newroot][now]=cnt++;
20          newroot=ch[newroot][now];
21          val[newroot]=val[root]+1;
```

```
22        }
23
24        return ret;
25  }
26  int query(int lt,int rt,int x){
27        int ans=0;
28        for(int i=30;i>=0;i--){
29            int now=(x>>i)&1;
30            if(val[ch[rt][now^1]]-val[ch[lt][now^1]]){
31                ans|=(1<<i);
32                rt=ch[rt][now^1];
33                lt=ch[lt][now^1];
34                } else{
35                rt=ch[rt][now];
36                lt=ch[lt][now];
37            }
38        }
39        return ans;
40  }
```

2.4.4 SGT in BBST

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int MAXN = 1e5;
4   const int MAXM = 2e7;
5   const int LM = 0;
6   const int RM = 70005;
7   namespace T{
8       int Tsz; queue<int> q;
9       int ls[MAXM], rs[MAXM], val[MAXM];
10      int Tan, Ta[105], Tbn, Tb[105];
11      int newnode() {
12          if(q.empty()) q.push(++Tsz);
13          int x = q.front(); q.pop();
14          ls[x] = rs[x] = val[x] = 0;
15          return x;
16      }
17      void insert(int &x, int l, int r, int k, int f) {
18          if(!x) x = newnode();
19          val[x] += f;
20          if(l == r) return;
21          int mid = (l + r) >> 1;
22          if(k <= mid) {
23              insert(ls[x], l, mid, k,f);
24          }else{
25              insert(rs[x], mid+1, r, k,f);
26          }
27      }
28      int query(int l, int r, int y) {
29          if(l == r) return l;
30          int mid = (l + r) >> 1;
31          int sum = 0;
32          for(int i = 0; i <= Tan; i++) sum -= val[ls[Ta[i]]];
33          for(int i = 0; i <= Tbn; i++) sum += val[ls[Tb[i]]];
34
35          if(y <= sum) {
36              for(int i = 0; i <= Tan; i++) Ta[i] = ls[Ta[i]];
37              for(int i = 0; i <= Tbn; i++) Tb[i] = ls[Tb[i]];
38              return query(l, mid, y);
```

```
39                }
40                for(int i = 0; i <= Tan; i++) Ta[i] = rs[Ta[i]];
41                for(int i = 0; i <= Tbn; i++) Tb[i] = rs[Tb[i]];
42                return query(mid+1, r, y - sum);
43            }
44        void recycle(int v) {
45            if(!v) return;
46            q.push(v);
47            recycle(ls[v]);
48            recycle(rs[v]);
49        }
50    };
51    namespace TT{
52        int Tsn;
53        int val[MAXN], Trt[MAXN];
54        int sz[MAXN];
55        int fa[MAXN], tr[MAXN][2];
56
57        int root;
58        double alp;
59
60        int newnode(int x) {
61            int tcnt = ++Tsn;
62            val[tcnt] = x; Trt[tcnt] = 0;
63            fa[tcnt] = tr[tcnt][0] = tr[tcnt][1] = 0;
64            sz[tcnt] = 0;
65            return tcnt;
66        }
67        bool isBad(int v) {
68            return (double(sz[ tr[v][0] ]) > double(sz[v]) * alp) ||
69                (double(sz[ tr[v][1] ]) > double(sz[v]) * alp);
70        }
71        vector<int> vec;
72        void rRecycle(int v) {
73            if(tr[v][0]) rRecycle(tr[v][0]);
74            vec.push_back(v);
75            T::recycle(Trt[v]);
76            if(tr[v][1]) rRecycle(tr[v][1]);
77        }
78        int rBuild(int l, int r) {
79            int mid = (l + r) >> 1, v = vec[mid];
80            Trt[v] = 0; sz[v] = r - l + 1;
81            for(int i = l; i <= mid; i++) {
82                T::insert(Trt[v], LM, RM, val[vec[i]], 1);
83            }
84            tr[v][0] = (l <= mid-1) ? rBuild(l, mid - 1) : 0;
85            if(tr[v][0]) fa[tr[v][0]] = v;
86            tr[v][1] = (mid+1 <= r) ? rBuild(mid + 1, r) : 0;
87            if(tr[v][1]) fa[tr[v][1]] = v;
88            return v;
89        }
90        void rebuild(int v) {
91            if(isBad(v)) {
92                vec.clear();
93                int u = fa[v], lr = tr[u][1] == v;
94                rRecycle(v);
95                if(vec.size()) v = rBuild(0, vec.size() - 1); else v = 0;
96                if(u == 0) fa[root = v] = 0;
97                else{
98                    tr[u][lr] = v;
99                    if(v) fa[v] = u;
```

```
100                }
101            }
102        }
103        int find(int x, int k) {
104            int l = tr[x][0], r = tr[x][1];
105            if(sz[l] + 1 == k) return x;
106            if(sz[l] >= k) return find(l, k);
107            return find(r, k - sz[l] - 1);
108        }
109        void ins(int x, int y) {
110            int v = find(root, x);
111            int p = tr[v][1], q = v;
112            if(p) {
113                for(;p; q = p, p = tr[p][0]);
114                fa[p = tr[q][0] = newnode(y)] = q;
115            }else {
116                fa[p = tr[q][1] = newnode(y)] = q;
117            }
118            int fg = 0;
119            T::insert(Trt[p], LM, RM, y, 1);
120            sz[p] = 1;
121            for(;fa[p]; p = fa[p]) {
122                if(tr[fa[p]][0] == p)
123                    T::insert(Trt[fa[p]], LM, RM, y, 1);
124                sz[fa[p]]++;
125                if(isBad(fa[p])) fg = fa[p];
126            }
127            rebuild(fg);
128        }
129        void upd(int x, int y) {
130            int p = find(root, x+1);
131            int ty = val[p]; val[p] = y;
132            T::insert(Trt[p], LM, RM, ty, -1);
133            T::insert(Trt[p], LM, RM, y, 1);
134            for(;fa[p]; p = fa[p]) {
135                if(tr[fa[p]][0] == p) {
136                    T::insert(Trt[fa[p]], LM, RM, ty, -1);
137                    T::insert(Trt[fa[p]], LM, RM, y, 1);
138                }
139            }
140        }
141        int que(int x, int y,int z) {
142            x = find(root, x);
143            T::Ta[T::Tan = 0] = Trt[x];
144            for(;fa[x]; x = fa[x])
145                if(tr[fa[x]][1] == x) {
146                    T::Ta[++T::Tan] = Trt[fa[x]];
147                }
148            y = find(root, y+1);
149            T::Tb[T::Tbn = 0] = Trt[y];
150            for(;fa[y]; y = fa[y]) {
151                if(tr[fa[y]][1] == y) {
152                    T::Tb[++T::Tbn] = Trt[fa[y]];
153                }
154            }
155            return T::query(LM, RM, z);
156        }
157        void init(int n) {
158            alp = 0.7;
159            vec.clear();
160            vec.push_back(newnode(RM));
```

```
161        for(int i = 1, a; i <= n; i++) {
162            scanf("%d", &a);
163            vec.push_back(newnode(a));
164        }
165        root = rBuild(0,vec.size() - 1);
166        fa[root] = 0;
167    }
168    void display(int v) {
169        if(tr[v][0]) display(tr[v][0]);
170        cerr<<val[v]<<" ";
171        if(tr[v][1]) display(tr[v][1]);
172    }
173 };
```

2.5   Tree Structures

2.5.1   dsu on tree

```
1  const int MAXN = 1e5 + 7;
2  vector<int> G[MAXN];
3  int bgison, dfs_clock, sz[MAXN], st[MAXN], bt[MAXN], et[MAXN];
4  int fg[MAXN], col[MAXN];
5  long long ans[MAXN];
6  void dfs1(int u, int fa) {
7      sz[u] = 1;
8      st[bt[u] = ++dfs_clock] = u;
9      for(auto v : G[u])
10     if(v != fa) {
11         dfs1(v, u);
12         sz[u] += sz[v];
13     }
14     et[u] = dfs_clock;
15 }
16 int maxx = 0;
17 void dfs2(int u, int fa, int keep) {
18     int mx = -1, bigson = -1;
19     for(auto &v : G[u])
20     if(v != fa) {
21         if(sz[v] > mx)
22             mx = sz[v], bigson = v;
23     }
24     for(auto &v : G[u])
25     if(v != fa && v != bigson)
26         dfs2(v,u,0);
27     if(bigson != -1) {
28         dfs2(bigson, u, 1);
29         ans[u] = ans[bigson];
30         for(int &v : G[u])
31             if(v != fa && v != bigson)
32             for(int i = bt[v]; i <= et[v]; i++) {
33                 ++fg[col[st[i]]];
34                 if(fg[ col[st[i]] ] > maxx) maxx=fg[col[st[i]]], ans[u] = 0;
35                 if(fg[ col[st[i]] ] == maxx) ans[u] += col[st[i]];
36             }
37     }
38     ++fg[col[u]];
39     if(fg[col[u]] > maxx) maxx = fg[col[u]], ans[u] = 0;
40     if(fg[col[u]] == maxx) ans[u] += col[u];
41     if(keep == 0) {
```

```
42          maxx = 0;
43          for(int i = bt[u]; i <= et[u]; i++)
44              fg[col[st[i]]] = 0;
45      }
46  }
```

2.5.2   Vitural Tree

```
1   const int MAXN = ;
2   const int MAXM = ;
3   const LL INF = ;
4   const int S = 19;
5   int ecnt, head[MAXN];
6   struct Edge{int to, nxt; LL w;} e[MAXM], ve[MAXM];
7   inline void addEdgeT(int x, int y, LL w) {
8       e[++ecnt] = (Edge) {y, head[x], w}; head[x] = ecnt;
9   }
10  int dep[MAXN], dfn_time, dfn[MAXN], fa[MAXN][S+1];
11  LL dis[MAXN][S+1];
12  void dfs(int v,int _fa) {
13      dfn[v] = ++dfn_time;
14      dep[v] = dep[_fa] + 1;
15      fa[v][0] = _fa;
16      for(int i = 1; i <= S; i++) {
17          fa[v][i] = fa[fa[v][i-1]][i-1];
18          dis[v][i] = min(dis[v][i-1], dis[fa[v][i-1]][i-1]);
19      }
20      for(int i = head[v], u; i; i = e[i].nxt)
21      if((u = e[i].to) != _fa) {
22          dis[u][0] = e[i].w;
23          dfs(u, v);
24      }
25  }
26  int getLca(int u, int v) {
27      if(dep[u] < dep[v]) swap(u, v);
28      for(int i = S; i >= 0; i--)
29          if(dep[fa[u][i]] >= dep[v]) u = fa[u][i];
30      if(u == v) return u;
31      for(int i = S; i >= 0; i--)
32          if(fa[u][i] != fa[v][i])
33              u = fa[u][i], v = fa[v][i];
34      return fa[u][0];
35  }
36  LL getDis(int u, int v) {
37      if(dep[u] < dep[v]) swap(u, v);
38      LL res = INF;
39      for(int i = S; i >= 0; i--)
40          if(dep[fa[u][i]] >= dep[v]) {
41              res = min(res, dis[u][i]);
42              u = fa[u][i];
43          }
44      return res;
45  }
46  namespace VituralTree{
47      int hn, h[MAXN];
48      int vecnt, vhead[MAXN];
49      Edge ve[MAXM];
50      int top, st[MAXN], cln, cl[MAXN];
51      int fgn, fg[MAXN];//利用 fg[i]==fgn 判断是否为当前有效点
52      void addEdgeVT(int x, int y, LL w) {
```

```
53            ve[++vecnt] = (Edge) {y, vhead[x], w}; vhead[x] = vecnt;
54        }
55        inline void link(int u, int v) {
56            LL w = getDis(u, v);
57            addEdgeVT(u, v, w);
58            addEdgeVT(v, u, w);
59        }
60        inline bool cmp(int a, int b) {return dfn[a] < dfn[b];}
61        void build() {
62            ++fgn;
63            for(int i = 1; i <= hn; i++) fg[h[i]] = fgn;
64            sort(h + 1, h + hn + 1, cmp);
65            cl[cln = 1] = st[top = 1] = 1;
66            for(int i = 1; i <= hn; i++) {
67                int rem = getLca(st[top], h[i]);
68                if(rem == st[top]) {
69                    if(rem != st[top]) cl[++cln] = st[++top] = h[i];
70                    continue;
71                }
72                while(top > 1 && dep[st[top - 1]] >= dep[rem]) {
73                    link(st[top - 1], st[top]); top--;
74                }
75                if(dep[st[top]] > dep[rem]) {
76                    link(rem, st[top]), top --;
77                }
78                if(rem != st[top]) cl[++cln] = st[++top] = rem;
79                if(h[i] != st[top]) cl[++cln] = st[++ top] = h[i];
80            }
81            while(top > 1) {
82                link(st[top - 1], st[top]), top --;
83            }
84        }
85        void clear() {
86            vecnt = 0;
87            for(;cln; --cln) vhead[cl[cln]] = 0;
88        }
89        void sol() {
90            build();
91            //注意一号节点可能在虚树外面
92            clear();
93        }
94 }
```

2.5.3   Tree Decomposition

```
1  int sz[MAXN], dep[MAXN], top[MAXN], fa[MAXN], son[MAXN], num[MAXN], totw;
2  struct Edge {
3      int to, nxt;
4  }e[MAXN << 1];
5  int head[MAXN], ecnt;
6  int n, m, Q;
7  #define Ls(x) (x << 1)
8  #define Rs(x) (x << 1 | 1)
9  struct Tree {
10     int l, r, lazy;
11     LL sum, mx;
12 }tree[MAXN << 2];
13 int A[MAXN], B[MAXN];
14 void push_up(int x) {
15     tree[x].sum = tree[Ls(x)].sum + tree[Rs(x)].sum;
```

```
16        tree[x].mx = max(tree[Ls(x)].mx, tree[Rs(x)].mx);
17  }
18  void push_down(int x) {
19      if(tree[x].lazy) {
20          tree[Ls(x)].sum += tree[x].lazy * (tree[Ls(x)].r - tree[Ls(x)].l + 1);
21          tree[Rs(x)].sum += tree[x].lazy * (tree[Rs(x)].r - tree[Rs(x)].l + 1);
22          tree[Ls(x)].mx += tree[x].lazy;
23          tree[Rs(x)].mx += tree[x].lazy;
24          tree[Ls(x)].lazy += tree[x].lazy;
25          tree[Rs(x)].lazy += tree[x].lazy;
26          tree[x].lazy = 0;
27      }
28  }
29  void build(int x, int L, int R) {
30      tree[x].lazy = 0;
31      tree[x].l = L; tree[x].r = R;
32      if(L == R) {
33          tree[x].sum = B[L];
34          tree[x].mx = B[L];
35          return;
36      }
37      int mid = (L + R) >> 1;
38      build(Ls(x), L, mid);
39      build(Rs(x), mid + 1, R);
40      push_up(x);
41  }
42  void update(int x, int L, int R, LL val) {
43      if(tree[x].l >= L && tree[x].r <= R) {
44          tree[x].lazy += val;
45          tree[x].sum += val * (tree[x].r - tree[x].l + 1);
46          tree[x].mx += val;
47          return;
48      }
49      push_down(x);
50      int mid = (tree[x].l + tree[x].r) >> 1;
51      if(L <= mid) update(Ls(x), L, R, val);
52      if(R > mid) update(Rs(x), L, R, val);
53      push_up(x);
54  }
55  LL query(int x, int L, int R) {
56      if(tree[x].l >= L && tree[x].r <= R)
57          return tree[x].sum;
58      push_down(x);
59      int mid = (tree[x].l + tree[x].r) >> 1;
60      LL res = 0;
61      if(L <= mid) res += query(Ls(x), L, R);
62      if(R > mid) res += query(Rs(x), L, R);
63      return res;
64  }
65  LL query2(int x, int L, int R) {
66      if(tree[x].l >= L && tree[x].r <= R)
67          return tree[x].mx;
68      push_down(x);
69      int mid = (tree[x].l + tree[x].r) >> 1;
70      LL res = -INF;
71      if(L <= mid) res = max(res, query2(Ls(x), L, R));
72      if(R > mid) res = max(res, query2(Rs(x), L, R));
73      return res;
74  }
75  inline void add_edge(int x, int y) {
76      e[++ecnt] = (Edge) {y, head[x]}; head[x] = ecnt;
```

```
77  }
78  void dfs1(int x) {
79      sz[x] = 1; son[x] = 0;
80      for(int i = head[x]; i; i = e[i].nxt) {
81          int v = e[i].to;
82          if(v == fa[x]) continue;
83          fa[v] = x;
84          dep[v] = dep[x] + 1;
85          dfs1(v);
86          sz[x] += sz[v];
87          if(sz[v] > sz[son[x]]) son[x] = v;
88      }
89  }
90  void dfs2(int x) {
91      B[num[x]] = A[x];
92      if(son[x]) {
93          top[son[x]] = top[x];
94          num[son[x]] = ++totw;
95          dfs2(son[x]);
96      }
97      for(int i = head[x]; i; i = e[i].nxt) {
98          int v = e[i].to;
99          if(v == fa[x] || v == son[x]) continue;
100         top[v] = v;
101         num[v] = ++totw;
102         dfs2(v);
103     }
104 }
105 void up(int a, int b, int c) {
106     int f1 = top[a], f2 = top[b];
107     while(f1 != f2) {
108         if(dep[f1] < dep[f2]) { swap(a, b); swap(f1, f2); }
109         update(1, num[f1], num[a], c);
110         a = fa[f1];
111         f1 = top[a];
112     }
113     if(dep[a] > dep[b]) swap(a, b);
114     update(1, num[a], num[b], c);
115 }
116 int qsum(int a, int b) {
117     if(a == b) return query(1, num[a], num[a]);
118     int f1 = top[a], f2 = top[b];
119     int res = 0;
120     while(f1 != f2) {
121         if(dep[f1] < dep[f2]) { swap(a, b); swap(f1, f2); }
122         res += query(1, num[f1], num[a]);
123         a = fa[f1];
124         f1 = top[a];
125     }
126     if(dep[a] > dep[b]) swap(a, b);
127     res += query(1, num[a], num[b]);
128     return res;
129 }
130 int qmax(int a, int b) {
131     if(a == b) return query2(1, num[a], num[a]);
132     int f1 = top[a], f2 = top[b];
133     int res = -1000000000;
134     while(f1 != f2) {
135         if(dep[f1] < dep[f2]) { swap(a, b); swap(f1, f2); }
136         res = max(res, query2(1, num[f1], num[a]));
137         a = fa[f1];
```

```
138            f1 = top[a];
139        }
140        if(dep[a] > dep[b]) swap(a, b);
141        res = max(res, query2(1, num[a], num[b]));
142        return res;
143 }
144 inline void init() {
145        memset(head, 0, sizeof(head)); ecnt = 0;
146        fa[1] = 0; dep[1] = 1; top[1] = 1; num[1] = 1; totw = 1;
147 }
148 inline void pre() {
149        dfs1(1); dfs2(1); build(1, 1, totw);
150 }
151 //———————————————————————————————————————————————————————
152 const int MAXN = ;
153 const int INF = ;
154 int A[MAXN], B[MAXN], C[MAXN];
155 struct SGT{
156        int sL, sR;
157        LL sW, mx[MAXN << 2];
158        void push_down(int v) {
159        }
160        void push_up(int  v) {
161            mx[v] = max(mx[v << 1], mx[v<<1|1]);
162        }
163        void build(int v, int l, int r) {
164            if(l == r) {
165                mx[v] = B[l];
166                return;
167            }
168            int mid = (l + r) >> 1;
169            build(v << 1, l, mid);
170            build(v<<1|1, mid+1,r);
171            push_up(v);
172        }
173        void upd(int v, int l, int r) {
174            if(sL <= l && r <= sR) {
175                //sum[v] = sW *(r − l + 1);
176                mx[v] = sW;
177                return;
178            }
179            push_down(v);
180            int mid = (l + r) >> 1;
181            if(sL <= mid) upd(v << 1, l, mid);
182            if(mid < sR)  upd(v<<1|1, mid+1,r);
183            push_up(v);
184        }
185        void qmax(int v, int l, int r) {
186            if(sL <= l && r <= sR) {
187                sW = max(sW, mx[v]);
188                return;
189            }
190            push_down(v);
191            int mid = (l + r) >> 1;
192            if(sL <= mid) qmax(v << 1, l, mid);
193            if(mid < sR)  qmax(v<<1|1, mid+1, r);
194        }
195 }T;
196 namespace TD{
197        struct Edge {int to, nxt;}e[MAXN << 1];
198        int ecnt, head[MAXN];
```

```
199        inline void addEdge(int x, int y) {
200            e[++ecnt] = (Edge) {y, head[x]}; head[x] = ecnt;
201            e[++ecnt] = (Edge) {x, head[y]}; head[y] = ecnt;
202        }
203        int fa[MAXN], sz[MAXN], dep[MAXN], son[MAXN], top[MAXN];
204        int dfn_time, dfn[MAXN], rnk[MAXN];
205        void dfs1(int u) {
206            sz[u] = 1; son[u] = -1;
207            for(int i = head[u], v; i; i = e[i].nxt) {
208                v = e[i].to;
209                if(v == fa[u]) continue;
210                fa[v] = u;
211                dep[v] = dep[u] + 1;
212                dfs1(v);
213                sz[u] += sz[v];
214                if(son[u] == -1 || sz[v] > sz[son[u]]) son[u] = v;
215            }
216        }
217        void dfs2(int u) {
218            dfn[u] = ++dfn_time;
219            //rnk[dfn_time] = u;
220            B[dfn[u]] = A[u];
221            if(son[u] == -1) return;
222            top[son[u]] = top[u];
223            dfs2(son[u]);
224            for(int i = head[u], v; i; i = e[i].nxt) {
225                v = e[i].to;
226                if(v == fa[u] || v == son[u]) continue;
227                top[v] = v;
228                dfs2(v);
229            }
230        }
231        inline void init(int n) {
232            ecnt = 1;
233            for(int i = 0; i <= n; i++) head[i] = 0;
234        }
235        inline void pre(int rt) {
236            fa[rt] = -1; dep[rt] = 1;
237            top[rt] = rt; dfn_time = 0;
238            dfs1(rt); dfs2(rt);
239            // edge -> point
240            //for(int i = 2; i <= ecnt; i += 2) {
241            //    if(e[i].to == fa[e[i^1].to]) {
242            //        C[i / 2] = e[i^1].to;
243            //    }else{
244            //        C[i / 2] = e[i].to;
245            //    }
246            //    B[dfn[C[i / 2]]] = e[i].w;
247            //}
248            T.build(1, 1, dfn_time);
249        }
250        int lca(int u, int v) {
251            while(top[u] != top[v]) {
252                if(dep[top[u]] > dep[top[v]]) swap(u, v);
253                v = fa[top[v]];
254            }
255            if(dep[u] > dep[v]) swap(u, v);
256            return u;
257        }
258        void upd(int a, int b, int c) {
259            int ta = top[a], tb = top[b];
```

```
260          while(ta != tb) {
261              if(dep[ta] < dep[tb]) { swap(a, b); swap(ta, tb); }
262              T.sL = dfn[ta]; T.sR = dfn[a]; T.sW = c;
263              T.upd(1, 1, dfn_time);
264              a = fa[ta]; ta = top[a];
265          }
266          if(dep[a] > dep[b]) swap(a, b);              // point
267          //if(a == b) return;                         // edge
268          //if(dep[a] > dep[b]) swap(a, b);            // edge
269          //a = son[a];                                // edge
270          T.sL = dfn[a]; T.sR = dfn[b]; T.sW = c;
271          T.upd(1, 1, dfn_time);
272      }
273      //更新子树, 由于dfs2中有if(son[u] == -1) return; 小心使用end[u];
274      void upd2(int a, int c) {
275          T.sL = dfn[a]; T.sR = dfn[a] + sz[a] - 1; T.sW = c;
276          T.upd(1, 1, dfn_time);
277      }
278      int qmax(int a, int b) {
279          int ta = top[a], tb = top[b]; int res = - INF;
280          while(ta != tb) {
281              if(dep[ta] < dep[tb]) { swap(a, b); swap(ta, tb); }
282              T.sL = dfn[ta]; T.sR = dfn[a]; T.sW = - INF;
283              T.qmax(1, 1, dfn_time); res = max(T.sW, res);
284              a = fa[ta];ta = top[a];
285          }
286          if(dep[a] > dep[b]) swap(a, b);              // point
287          //if(a == b) return res;                     // edge
288          //if(dep[a] > dep[b]) swap(a, b);            // edge
289          //a = son[a];                                // edge
290          T.sL = dfn[a]; T.sR = dfn[b]; T.sW = -INF;
291          T.qmax(1, 1, dfn_time); res = max(T.sW, res);
292          return res;
293      }
294 }
```

2.5.4 Link-Cut Tree

```
1  namespace LCT {
2      int tcnt; //动态开点
3      int fa[MAXN], tr[MAXN][2], rev[MAXN];
4      //int val[MAXN], sval[MAXN];
5      //void clear(int id) { //或者改为int newnode() {}
6      //    fa[id] = tr[id][0] = tr[id][1] = 0;
7      //    rev[id] = 0; //sz[id] = 1;
8      //}
9      void Rev(int x) {
10         rev[x] ^= 1; swap(tr[x][0], tr[x][1]);
11     }
12     void push_up(int x) {
13         //int l = tr[x][0], r = tr[x][1];
14         //sval[x] = sval[l] + val[x] + sval[r];
15     }
16     void push_down(int x) {
17         int l = tr[x][0], r = tr[x][1];
18         if(rev[x]) {
19             if(l) Rev(l);
20             if(r) Rev(r);
21             rev[x] = 0;
22         }
```

```
23          }
24          bool isroot(int x) {
25              return tr[fa[x]][0] != x && tr[fa[x]][1] != x;
26          }
27          void pre(int x) {
28              if(!isroot(x)) pre(fa[x]);
29              push_down(x);
30          }
31          void rotate(int x) {
32              int y = fa[x], z = fa[y], lr = tr[y][1] == x;
33              if(!isroot(y)) tr[z][tr[z][1] == y] = x;
34              fa[x] = z;
35              fa[tr[y][lr] = tr[x][lr^1]] = y;
36              fa[tr[x][lr^1] = y] = x;
37              push_up(y);
38          }
39          inline void splay(int x) {
40              pre(x);
41              for (int y, z; !isroot(x); rotate(x)) {
42                  y = fa[x]; z = fa[y];
43                  if (!isroot(y)) rotate((tr[z][0] == y) ^ (tr[y][0] == x) ? x : y);
44              }
45              push_up(x);
46          }
47          inline int access(int x) {
48              int y = 0;
49              for (; x; y = x, x = fa[x]) {
50                  splay(x);
51                  //sz2[x] += sz[tr[x][1]] - sz[y]; //subtree
52                  tr[x][1] = y;
53                  push_up(x);
54              }
55              return y;//不求LCA不必
56          }
57          inline void makeroot(int x) {
58              access(x); splay(x); Rev(x);
59          }
60          inline int findroot(int x) {
61              access(x); splay(x);
62              for(;tr[x][0]; x = tr[x][0]) push_down(x);
63              splay(x);
64              return x;
65          }
66          inline void lnk(int x, int y) {
67              makeroot(x);
68              if(findroot(y) != x) fa[x] = y;
69              //sz2[y] += sz[x]; //subtree
70          }
71          inline void cut(int x,int y) {
72              makeroot(x); //access(y); splay(x);
73              if(findroot(y) == x&&fa[y] == x&&!tr[y][0]){
74                  fa[y] = tr[x][1] = 0;
75                  push_up(x);
76              }
77          }
78          inline void cut(int y) { //有根树断开与父节点连边
79              access(y); splay(y);
80              fa[tr[y][0]] = 0;
81              tr[y][0] = 0;
82              push_up(y);
83          }
```

```
 84        inline int lca(int u, int v) {
 85            access(u);
 86            return access(v);
 87        }
 88        void split(int x, int y) {
 89            makeroot(x); access(y); splay(y);
 90        }
 91        //维护节点或者维护路径
 92        //例如：染色（注意tag_rev）、tag_add的区间信息、splay维护连续端最远位置
 93        void upd(int x, int y) {
 94            makeroot(x); val[x] = y; push_up(x);
 95        }
 96        int que(int x, int y) {
 97            split(x, y);
 98            //return sval[y];
 99        }
100        //维护边权(y -> eid -> x)，需要初始化 vcnt，ecnt，可用map(注意双向维护)维护eid信息
101        //注意 if(x == y) continue;
102        struct LCTEdge{int u, v; int w;} e[MAXN];
103        void addEdge(int eid) { //e[eid = ++ecnt] = (Edge){x, y, w};
104            lnk(e[eid].u, vcnt + eid);
105            lnk(vcnt + eid, e[eid].v);
106        }
107        void delEdge(int eid) {
108            cut(e[eid].u, vcnt + eid);
109            cut(vcnt + eid, e[eid].v);
110        }
111        //维护边双连通分量
112        //并查集,所有的fa[x]改为Find(fa[x]),public调用函数使用前Find(x)，需要保证只在shink进
               行过合并
113        //维护边双时，节点自身信息在unite中维护，路径等信息在push_up和push_down中维护
114        int Rt[MAXN];
115        int findroot(int x) {return Rt[x] == x ? x : Rt[x] = findroot(Rt[x]);}
116        void unite_dfs(int x) {
117            push_down(x);
118            if (tr[x][0]) unite_dfs(tr[x][0]), unite(tr[x][0], x);
119            if (tr[x][1]) unite_dfs(tr[x][1]), unite(tr[x][1], x);
120        }
121        void shrink(int x, int y) {
122            split(x, y);
123            unite_dfs(y);
124            int z = Find(y);
125            fa[z] = fa[y]; tr[z][0] = tr[z][1] = 0;
126            push_up(z);
127        }
128        void addEdge(int x, int y) {
129            x = Find(x);y = Find(y);
130            if(findroot(x) != findroot(y)) {
131                lnk(x, y);  Rt[findroot(x)] = Rt[findroot(y)];
132            }else shrink(x, y);
133        }
134 };
135 //未连成树、初始化节点时，需要调用LCT::push_up 维护节点其余信息
136 //维护的信息要有 可减性 ，如子树结点数，子树权值和，但不能直接维护子树最大最小值，因为在
        将一条虚边变成实边时要排除原先虚边的贡献。
137 //新建一个附加值存储虚子树的贡献，在统计时将其加入本结点答案，在改变边的虚实时及时维护。
138 //其余部分同普通 LCT，在统计子树信息时一定将其作为根节点。
139 //如果维护的信息没有可减性，如维护区间最值，可以对每个结点开一个平衡树维护结点的虚子树中
        的最值。
140 if(x == y) continue;
141 if(Find(x) != Find(y)) {
```

```
142        unite(x, y);
143        addEdge(mp[{x, y}]);
144    }else{
145        int eid = que(x, y);
146        int id = mp[{x, y}];
147        if(val[eid] > e[id].w) {
148            delEdge(eid - vcnt);
149            addEdge(id);
150        }
151    }
```

2.5.5 Divide Combine Tree

1.[i,i+1] 构造依赖, 利用线段树辅助建图, 利用 tarjan 求 scc, 利用 rmq 求最左最右边界 2. 定义 (i,j) 为一个好二元组，当且仅当 a[i]-a[j]=1 这样的两项的二元组在 [l,r] 中恰好有 r-l 个线段树维护 val + l = r，其中 val 是区间 [l,r] 中好二元组的个数离线 3. 析合树维护 mx - mn = r - l <=> fx = (mx - mn) - (r - l)

```
 1  namespace DCT{
 2      struct RMQ {
 3          int lg[MAXN], mn[MAXN][S+1], mx[MAXN][S+1];
 4          inline void init(int *a, int n) {
 5              for (int i = 2; i <= n; i++) lg[i] = lg[i >> 1] + 1;
 6              for (int i = 1; i <= n; i++) mn[i][0] = mx[i][0] = a[i];
 7              for (int k = 1; (1 << k) <= n; k++)
 8                  for (int i = 1; i + (1 << k) - 1 <= n; i++) {
 9                      mn[i][k] = min(mn[i][k - 1], mn[i + (1 << (k - 1))][k - 1]);
10                      mx[i][k] = max(mx[i][k - 1], mx[i + (1 << (k - 1))][k - 1]);
11                  }
12          }
13          inline int Min(int l, int r) {
14              int len = lg[r - l + 1];
15              return min(mn[l][len], mn[r - (1 << len) + 1][len]);
16          }
17          inline int Max(int l, int r) {
18              int len = lg[r - l + 1];
19              return max(mx[l][len], mx[r - (1 << len) + 1][len]);
20          }
21      } D;
22
23      struct SEG {
24          int setL, setR, setW;
25          int mn[MAXN << 2], tag[MAXN << 2];
26
27          inline void pushup(int x) {
28              mn[x] = min(mn[x << 1], mn[x << 1 | 1]);
29          }
30          inline void pushdown(int x) {
31              if(!tag[x]) return;
32              mn[x << 1] += tag[x]; mn[x << 1 | 1] += tag[x];
33              tag[x << 1] += tag[x]; tag[x << 1 | 1] += tag[x]; tag[x] = 0;
34          }
35          void init(int x, int l, int r) {
36              mn[x] = tag[x] = 0;
37              if (l == r) return;
38              int mid = (l + r) >> 1;
39              init(x << 1, l, mid);
40              init(x << 1 | 1, mid + 1, r);
41          }
42          void upt(int x, int l, int r) {
43              if (setL <= l && r <= setR) {
```

```
44                     tag[x] += setW; mn[x] += setW;
45                     return;
46                 }
47                 pushdown(x);
48                 int mid = (l + r) >> 1;
49                 if (setL <= mid) upt(x << 1, l, mid);
50                 if (mid < setR ) upt(x << 1 | 1, mid+1, r);
51                 pushup(x);
52             }
53         int que(int x, int l, int r) {
54                 if (l == r) return l;
55                 pushdown(x);
56                 int mid = (l+r)>>1;
57                 if (!mn[x << 1]) return que(x << 1, l, mid);
58                 return que(x << 1 | 1, mid+1, r);
59             }
60     } T;
61
62     int tpmn, stmn[MAXN], tpmx, stmx[MAXN], tpk, stk[MAXN];
63     int ncnt, type[MAXN<<1], L[MAXN<<1], R[MAXN<<1], M[MAXN<<1];
64     int dep[MAXN<<1], fa[MAXN<<1][S+1], C[MAXN<<1];
65     int id[MAXN << 1];
66     int newnode(int _type, int _L, int _R, int _M = 0) {
67         ++ncnt; type[ncnt] = _type;
68         L[ncnt] = _L; R[ncnt] = _R; M[ncnt] = _M;
69         C[ncnt] = 0;
70         return ncnt;
71     }
72
73     inline bool judge(int l, int r) {
74         return D.Max(l, r) - D.Min(l, r) == r - l;
75     }
76
77     int ecnt, head[MAXN << 1];
78     struct Edge{int to, nxt;} e[MAXN<<1];
79     inline void addEdge(int x, int y) {
80         e[++ecnt] = (Edge) {y, head[x]}; head[x] = ecnt;
81         fa[y][0] = x; C[x]++;
82     }
83     void dfs(int u) {
84         for(int j = 0; j < S; j++) fa[u][j+1] = fa[fa[u][j]][j];
85         for(int i = head[u]; i; i = e[i].nxt) {
86             dep[e[i].to] = dep[u] + 1;
87             dfs(e[i].to);
88         }
89     }
90
91     inline void init(int n) {
92         ecnt = 0;
93         for(int i = 0; i <= n; i++) head[i] = 0;
94     }
95     void buildT(int *a, int n) {
96         init(n);
97         D.init(a, n);
98         T.init(1, 1, n);
99         tpmn = tpmx = tpk = 0;
100         stmn[0] = stmx[0] = stk[0] = 0;
101         for (int i = 1; i <= n; i++) {
102             for (;tpmn && a[i] <= a[stmn[tpmn]]; --tpmn) {
103                 T.setL = stmn[tpmn - 1] + 1; T.setR = stmn[tpmn]; T.setW = a[stmn[tpmn
     ]];
```

```
104                    T.upt(1, 1, n);
105                }
106                T.setL = stmn[tpmn] + 1; T.setR = i; T.setW = -a[i];
107                T.upt(1, 1, n);
108                stmn[++tpmn] = i;
109
110                for (;tpmx && a[i] >= a[stmx[tpmx]]; --tpmx) {
111                    T.setL = stmx[tpmx - 1] + 1; T.setR = stmx[tpmx]; T.setW = -a[stmx[tpmx
      ]];
112                    T.upt(1, 1, n);
113                }
114                T.setL = stmx[tpmx] + 1; T.setR = i; T.setW = a[i];
115                T.upt(1, 1, n);
116                stmx[++tpmx] = i;
117
118                int Li = T.que(1, 1, n), np = id[i] = newnode(0, i, i), nq, nw;
119                while (tpk && L[nq = stk[tpk]] >= Li) {
120                    if (type[nq] && judge(M[nq], i)) {
121                        R[nq] = i;
122                        addEdge(nq, np);
123                        np = nq; tpk--;
124                    } else if (judge(L[nq], i)) {
125                        nw = newnode(1, L[nq], i, L[np]);
126                        addEdge(nw, nq); addEdge(nw, np);
127                        np = nw; tpk--;
128                    } else {
129                        nw = newnode(0, -1, i);
130                        addEdge(nw, np);
131                        do {
132                            addEdge(nw, nq);
133                            nq = stk[--tpk];
134                        } while (tpk && !judge(L[nq], i));
135                        addEdge(nw, nq);
136                        L[nw] = L[nq]; R[nw] = i;
137                        np = nw; --tpk;
138                    }
139                }
140                stk[++tpk] = np;
141                T.setL = 1; T.setR = i; T.setW = -1;
142                T.upt(1, 1, n);
143            }
144        assert(tpk == 1);
145        dfs(stk[tpk]);
146    }
147    void lca(int u, int v, int &aL, int &bR) {
148        if(u == v) {
149            aL = L[u]; bR = R[v];
150            return;
151        }
152        if(dep[u] > dep[v]) swap(u, v);
153        for(int i = S; i >= 0; i--)
154            if(dep[fa[v][i]] >= dep[u]) v = fa[v][i];
155        assert(u != v);
156        for(int i = S; i >= 0; i--)
157            if(fa[u][i] != fa[v][i]) {
158                u = fa[u][i]; v = fa[v][i];
159            }
160        if(type[fa[u][0]]) {
161            aL = min(L[v], L[u]);
162            bR = max(R[v], R[u]);
163        }else{
```

```
164            aL = L[fa[u][0]];
165            bR = R[fa[u][0]];
166        }
167    }
168 };
```

3 String

3.1 Basics

3.1.1 Hash

```
const LL p1 = 201, p2 = 301, mod1 = 1200000319, mod2 = 2147483647;
struct Hash {
    LL a, b;
    void append(Hash pre, int v) {
        a = (pre.a * p1 + v) % mod1;
        b = (pre.b * p2 + v) % mod2;
    }
    void init(string S) {
        a = b = 0;
        for(int i = 0; i < S.size(); i++) append(*this, S[i]);
    }
    bool operator == (const Hash &x) const {
        return a == x.a && b == x.b;
    }
    bool operator < (const Hash &x) const {
        return a < x.a || (a == x.a && b < x.b);
    }
};
```

3.1.2 Minimum String

```
namespace minstring{
    int getmin(char *s, int sn) {
        int i = 0, j = 1, k = 0, t;
        while(i < sn && j < sn && k < sn) {
            t = s[(i + k) % sn] - s[(j + k) % sn];
            if(!t) k++;
            else {
                if(t > 0) i += k + 1; else j += k + 1;
                if(i == j) j++;
                k = 0;
            }
        }
        return i < j ? i : j;
    }
}
```

3.2 String Matching

3.2.1 Bitset Match

```
namespace BitMatch{
    const int S = 26;
    bitset<MAXN> bs[S], ret;
    char s[MAXN];
    inline int idx(char c) { return c - 'a'; }
    inline void init() {
        for(int i = 0; i < 26; i++) bs[i].reset();
        scanf("%s", s);
        int sn = strlen(s);
```

```
10          for(int i = 0; i < sn; i++) bs[idx(s[i])].set(i);
11      }
12      void modify(int p, char ch) {
13          bs[idx(s[p])].reset(p);
14          s[p] = ch;
15          bs[idx(s[p])].set(p);
16      }
17      int match(char *t, int tn) {//返回t串在s串中出现的次数
18          ret = bs[idx(t[0])];
19          for(int i = 1; i < tn; i++) {
20              ret <<= 1;
21              ret &= bs[idx(t[i])];
22          }
23          return ret.count();
24      }
25  }
```

3.2.2   KMP && exKMP

```
1  判断循环子串的充要条件: i/(i-fa[i]) > 1 && i%(i-fa[i])==0 (i是长度)且去除下面fa的优化
2  namespace KMP {
3      int fa[MAXN];
4      void get_fail(char* t, int tn) {
5          fa[0] = -1;
6          int i = 0, j = -1;
7          while(i < tn) {
8              if (j == -1 || t[i] == t[j]) {
9                  ++i; ++j;
10                 fa[i] = t[i] != t[j] ? j : fa[j];
11             }else{
12                 j = fa[j];
13             }
14         }
15     }
16     void kmp(char* s, int sn, char* t, int tn) {
17         int i = 0, j = 0;
18         while(i < sn) {
19             if (j == -1 || s[i] == t[j]) {
20                 i++;j++;
21                 if(j == tn) {
22                 }
23             }else j = fa[j];
24         }
25     }
26 }
27 nxt[i]是t和从i开始的t的后缀的最大公共前缀长度。
28 t,s的第一个字符下标为0，最后一个字符下标为n-1。
29 namespace exKMP {
30     int nxt[MAXN], ext[MAXN];
31     void get_nxt(char* t, int tn) {
32         int j = 0, mx = 0;
33         nxt[0] = tn;
34         for(int i = 1; i < tn; i++) {
35             if(i >= mx || i + nxt[i - j] >= mx) {
36                 if(i > mx) mx = i;
37                 while(mx < tn && t[mx] == t[mx - i]) mx++;
38                 nxt[i] = mx - i;
39                 j = i;
40             }else nxt[i] = nxt[i - j];
41         }
```

```
42          }
43      void exkmp(char *s, int sn, char *t, int tn) {
44          int j = 0, mx = 0;
45          for(int i = 0; i < sn; i++) {
46              if(i >= mx || i + nxt[i - j] >= mx) {
47                  if(i > mx) mx = i;
48                  while(mx < sn && mx - i < tn && s[mx] == t[mx - i]) mx++;
49                  ext[i] = mx - i;
50                  j = i;
51              }else ext[i] = nxt[i - j];
52          }
53      }
54  }
```

3.2.3   AC Automaton

```
1   namespace AC {
2       int ch[MAXN][sigma_size], last[MAXN];
3       int val[MAXN], f[MAXN], sz;
4       inline void init() { sz = 1; memset(ch[0], 0, sizeof(ch[0])); }
5       inline int idx(char c) { return c - 'a'; }
6       void insert(string s, int v) {
7           int u = 0;
8           for(int i = 0; i < s.size(); i++) {
9               int c = idx(s[i]);
10              if(!ch[u][c]) {
11                  memset(ch[sz], 0, sizeof(ch[sz]));
12                  val[sz] = 0;
13                  ch[u][c] = sz++;
14              }
15              u = ch[u][c];
16          }
17          val[u] = v;
18      }
19      void get_fail() {
20          queue<int> q;
21          f[0] = 0;
22          for(int c = 0; c < sigma_size; c++) {
23              int u = ch[0][c];
24              if(u) { f[u] = 0; q.push(u); last[u] = 0; }
25          }
26          while(!q.empty()) {
27              int r = q.front(); q.pop();
28              for(int c = 0; c < sigma_size; c++) {
29                  int u = ch[r][c];
30                  if(!u) { ch[r][c] = ch[f[r]][c]; continue; }
31                  q.push(u);
32                  int v = f[r];
33                  while(v && !ch[v][c]) v = f[v];
34                  f[u] = ch[v][c];
35                  last[u] = val[f[u]] ? f[u] : last[f[u]];
36              }
37          }
38      }
39      inline void solve(int j) {
40          if(j) {
41              ans += val[j];
42              solve(last[j]);
43          }
44      }
```

```
45      void find(string T) {
46          int j = 0;
47          for(int i = 0; i < T.size(); i++) {
48              int c = idx(T[i]);
49              j = ch[j][c];
50              if(val[j]) solve(j);
51              else if(last[j]) solve(last[j]);
52          }
53      }
54  }
55  namespace AC {
56      int root, tcnt;
57      int ch[MAXN][sigma_size], fa[MAXN];
58      inline int newnode() {
59          fa[++tcnt] = 0;
60          for(int i = 0; i < sigma_size; ++i) ch[tcnt][i] = 0;
61          return tcnt;
62      }
63      inline void init() {
64          tcnt = -1;
65          root = newnode();
66      }
67      inline int idx(char c) { return c - 'a'; }
68      void extend(char *s, int sn) {
69          int cur = root;
70          for(int i = 0, c; i < sn; i++) {
71              if(!ch[cur][c = idx(s[i])])
72                  ch[cur][c] = newnode();
73              cur = ch[cur][c];
74          }
75      }
76      int q[MAXN], qh, qt;
77      void get_fail() {
78          qh = 1; qt = 0;
79          fa[root] = 0;
80          for(int c = 0, now; c < sigma_size; c++)
81              if((now = ch[root][c]) != 0)
82                  q[++qt] = now;
83          while(qh <= qt) {
84              int cur = q[qh++];
85              for(int c = 0, now; c < sigma_size; c++)
86                  if((now = ch[cur][c]) != 0) {
87                      fa[now] = ch[fa[cur]][c];
88                      q[++qt] = now;
89                  }else
90                      ch[cur][c] = ch[fa[cur]][c];
91          }
92      }
93  //统计模板串出现次数，每个模板串只计算一次
94  //      int cur = root, ans = 0;
95  //      for(int i = 0; i < sn; ++i) {
96  //          cur = ch[cur][idx(s[i])];
97  //          for(int j = cur; j && cnt[j] != -1; j = fa[j]) {
98  //              ans += cnt[j];
99  //              cnt[j] = -1;
100 //          }
101 //      }
102 }
```

3.3 Suffix Related

3.3.1 Suffix Array

```cpp
namespace SA {
    char s[MAXN];
    int sa[MAXN], rank[MAXN], height[MAXN];
    int t[MAXN], t2[MAXN], c[MAXN], n;
    void clear() { n = 0; memset(sa, 0, sizeof(sa)); }
    void build(int m) {
        int *x = t, *y = t2;
        for(int i = 0; i < m; i++) c[i] = 0;
        for(int i = 0; i < n; i++) c[x[i] = s[i]]++;
        for(int i = 1; i < m; i++) c[i] += c[i - 1];
        for(int i = n - 1; i >= 0; i--) sa[--c[x[i]]] = i;
        for(int k = 1; k <= n; k <<= 1) {
            int p = 0;
            for(int i = n - k; i < n; i++) y[p++] = i;
            for(int i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i] - k;
            for(int i = 0; i < m; i++) c[i] = 0;
            for(int i = 0; i < n; i++) c[x[y[i]]]++;
            for(int i = 1; i < m; i++) c[i] += c[i - 1];
            for(int i = n - 1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
            swap(x, y);
            p = 1; x[sa[0]] = 0;
            for(int i = 1; i < n; i++)
                x[sa[i]] = y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k]
    ? p - 1 : p++;
            if(p >= n) break;
            m = p;
        }
    }
    void buildHeight() {
        int k = 0;
        for(int i = 0; i < n; i++) rank[sa[i]] = i;
        for(int i = 0; i < n; i++) {
            if(k) k--;
            int j = sa[rank[i] - 1];
            while(s[i + k] == s[j + k]) k++;
            height[rank[i]] = k;
        }
    }
    void init() {
        n = strlen(s) + 1;
        build('z' + 1);
        buildHeight();
    }
}
```

3.3.2 Suffix Automaton

```cpp
//root = 1
//len[i]: i表示的状态集中最长的字符串长度
namespace SAM{
    int scnt, root, last;
    int fa[MAXN<<1], len[MAXN<<1], ch[MAXN<<1][26];
    int sc[MAXN<<1], tmpl[MAXN<<1], minl[MAXN<<1];

    int newnode(int _len, int q = 0) {
```

```
 9            fa[++scnt] = fa[q]; len[scnt] = _len;
10            sc[scnt] = 0;tmpl[scnt] = 0; minl[scnt] = INF;
11            for(int i = 0; i < 26; i++) ch[scnt][i] = ch[q][i];
12            return scnt;
13        }
14        void init() {
15            scnt = 0;
16            root = last = newnode(0);
17        }
18        void extend(int c) {
19            int p = last, np = newnode(len[p] + 1);
20            for(;p && ch[p][c] == 0; p = fa[p]) ch[p][c] = np;
21            if(!p) fa[np] = root;
22            else{
23                int q = ch[p][c];
24                if(len[p] + 1 == len[q]) fa[np] = q;
25                else{
26                    int nq = newnode(len[p] + 1, q);
27                    fa[np] = fa[q] = nq;
28                    for(; p && ch[p][c] == q; p = fa[p]) ch[p][c] = nq;
29                }
30            }
31            last = np;
32        }
33        int c[MAXN], rs[MAXN << 1];
34        void radix_sort(int n){
35            for(int i = 0; i <= n; i++) c[i] = 0;
36            for(int i = 1; i <= scnt; i++) c[len[i]]++;
37            for(int i = 1; i <= n; i++) c[i] += c[i-1];
38            for(int i = scnt; i >= 1; i--) rs[c[len[i]]--] = i;
39        }
40        void go(){
41            scanf("%s",s);
42            int n = strlen(s);
43            for(int i = 0; i < n; ++i)
44                extend(s[i] - 'a');
45            radix_sort(n);
46            //以下sc集合意义不同
47            {//预处理从每个节点出发，还有多少本质不同的子串可以到达，注意sc要开 long long
48                for(int i = scnt; i >= 1; i--) {
49                    LL S = 0;
50                    for(int j = 0; j < 26; j++)
51                        S += sc[ ch[rs[i]][j] ];
52                    sc[rs[i]] = S + 1;
53                }
54            }
55            {//right集合大小
56                int cur = root;
57                for(int i = 0; i < n; ++i) {
58                    cur = ch[cur][s[i]-'a'];
59                    sc[cur]++;
60                }
61                for(int i = scnt; i >= 1; --i) {
62                    sc[ fa[rs[i]] ] += sc[rs[i]];
63                }
64            }
65            //公共子串
66            //tmpl,当前字符串:在状态cur，与模板串的最长公共后缀
67            //minl，多个字符串:在状态cur,与模板串的最长公共后缀
68            //注意:在状态cur匹配成功时，cur的祖先状态与字符串的最长公共后缀
69            for(; ~scanf("%s",s);) {
```

```
70              int cur = root, Blen = 0;
71              for(int i = 0; i <= scnt; i++)
72                  tmpl[i] = 0;
73              n = strlen(s);
74              for(int i = 0, x; i < n; i++) {
75                  x = s[i] - 'a';
76                  if(ch[cur][x]) {
77                      ++Blen;
78                      cur = ch[cur][x];
79                  }else{
80                      for(;cur && ch[cur][x] == 0; cur = fa[cur]);
81                      if(cur) {
82                          Blen = len[cur] + 1;
83                          cur = ch[cur][x];
84                      }else{
85                          cur = root; Blen = 0;
86                      }
87                  }
88                  tmpl[cur] = max(tmpl[cur], Blen);
89              }
90              for(int i = scnt; i ; --i) {
91                  if( tmpl[ fa[rs[i]] ] < tmpl[ rs[i] ])
92                      tmpl[ fa[rs[i]] ] = len[ fa[rs[i]] ];
93                  minl[ rs[i] ] = min(minl[ rs[i] ], tmpl[ rs[i] ]);
94              }
95          }
96      }
97  }
98  namespace exSAM{
99      int scnt, root;
100     int fa[MAXN<<1], len[MAXN<<1], ch[MAXN<<1][26];
101     int sc[MAXN<<1], tmpl[MAXN<<1], minl[MAXN<<1];
102
103     int newnode(int _len, int q = 0) {
104         fa[++scnt] = fa[q]; len[scnt] = _len;
105         sc[scnt] = 0;tmpl[scnt] = 0; minl[scnt] = INF;
106         for(int i = 0; i < 26; i++) ch[scnt][i] = ch[q][i];
107         return scnt;
108     }
109     void init() {
110         scnt = 0;
111         root = newnode(0);
112     }
113     int work(int p,int c){
114         int q = ch[p][c];
115         int nq = newnode(len[p] + 1, q);
116         fa[q] = nq;
117         for(; p && ch[p][c] == q; p = fa[p]) ch[p][c] = nq;
118         return nq;
119     }
120     int extend(int p, int c) {
121         if (ch[p][c]){
122             int q = ch[p][c];
123             if (len[p] + 1 == len[q]) return q;
124             return work(p, c);
125         }
126         int np = newnode(len[p] + 1);
127         for(;p && ch[p][c] == 0; p = fa[p]) ch[p][c] = np;
128         if (!p) fa[np] = root;
129         else{
130             int q = ch[p][c];
```

```
131              if (len[p] + 1 == len[q]) fa[np] = q;
132              else fa[np] = work(p, c);
133          }
134          return np;
135      }
136      void solve() {
137          int n; scanf("%d",&n);
138          for(int i = 1; i <= n; i++) {
139              scanf("%s", s);
140              int sn = strlen(s);
141              int last = root;
142              for(int j = 0; j < sn; ++j)
143                  last = extend(last, s[j] - 'a');
144          }
145      }
146 }
```

3.4 Palindrome Related

3.4.1 Manacher

```
 1 namespace Manachar {
 2     char S[MAXN << 1];
 3     int scnt, ans;
 4     int p[MAXN << 1]; //p[i] − 1
 5     void init(char *s0, int sn0) {
 6         S[0] = '$'; S[1] = '#';
 7         for(int i = 0; i < sn0; i++) {
 8             S[2 * i + 2] = s0[i];
 9             S[2 * i + 3] = '#';
10         }
11         scnt = sn0 * 2 + 2;
12         S[scnt] = '&';
13     }
14     void manachar() {
15         int id = 0, mx = 0;
16         for(int i = 1; i < scnt; i++) {
17             p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
18             while(S[i + p[i]] == S[i - p[i]]) p[i]++;
19             if(i + p[i] > mx) {
20                 mx = i + p[i];
21                 id = i;
22             }
23         }
24     }
25 }
```

3.4.2 Palindromic Automaton

pcnt 本质不同的回文串的个数 len[u] 状态 u 代表的串的长度 fail[u] 状态 u 所代表的回文串的最长回文后缀 trans[u] 小于等于当前节点长度一半的最长回文后缀 cnt[u] 包含状态 u 表示的回文串的出现的个数 (调用 count()) num[u] 以状态 u 表示的回文串的右端点为回文串结尾的回文串个数 [..[....[] diff[u] 表示节点 u 和 fail[u] 所代表的回文串的长度差，即 len[u] - len[fail[u] slink[u] 表示 u 一直沿着 fail 向上跳到第一个节点 v, 使得 diff[v]!=diff[u]，也就是 u 所在等差数列中长度最小的那个节点

```
 1 namespace PAM {
 2     int scnt, S[MAXN];
```

```
3      int pcnt, last, len[MAXN], fail[MAXN], ch[MAXN][26];
4      int cnt[MAXN], num[MAXN], trans[MAXN], diff[MAXN], slink[MAXN];
5      int newnode(int _len) {
6          len[pcnt] = _len;
7          cnt[pcnt] = num[pcnt] = 0;
8          for(int i = 0; i < 26; i++) ch[pcnt][i] = 0;
9          return pcnt++;
10      }
11      inline void init() {
12          S[scnt = 0] = -1;
13          pcnt = 0;newnode(0);newnode(-1);
14          fail[0] = 1;last = 0;
15      }
16      int getfail(int x) {
17          while(S[scnt - len[x] - 1] != S[scnt]) x = fail[x];
18          return x;
19      }
20      void extend(int c) {
21          S[++scnt] = c;
22          int cur = getfail(last);
23          if(!ch[cur][c]) {
24              int now = newnode(len[cur] + 2);
25              fail[now] = ch[getfail(fail[cur])][c];
26              ch[cur][c] = now;
27              num[now] = num[fail[now]] + 1;
28              if (len[now] <= 2) trans[now] = fail[now];
29              else{
30                  int x = trans[cur];
31                  while(S[scnt - len[x] - 1] != S[scnt] || (len[x]+2) * 2 > len[now]) x =
    fail[x];
32                  trans[now] = ch[x][c];
33              }
34              diff[now] = len[now] - len[fail[now]];
35              slink[now] = (diff[now] == diff[fail[now]]) ? slink[fail[now]] : fail[now];
36          }
37          last = ch[cur][c];
38          cnt[last]++;
39      }
40      void count() {
41          for(int i = pcnt - 1; i >= 0; i--) cnt[fail[i]] += cnt[i];
42      }
43  };
```

支持前后插入不基于势能分析的构造法可以实现持久化，比如在 Trie 上实现

```
1  namespace PAM {
2      int sL, sR, S[MAXN<<1];
3      int pcnt, lastL, lastR;
4      int len[MAXN<<1], fa[MAXN<<1], quick[MAXN<<1][26], ch[MAXN<<1][26];
5      int cnt[MAXN<<1], num[MAXN<<1];
6      int newnode(int _len) {
7          len[pcnt] = _len;
8          cnt[pcnt] = num[pcnt] = 0;
9          for(int i = 0; i < 26; i++) ch[pcnt][i] = 0;
10          return pcnt++;
11      }
12      inline void init() {
13          pcnt = 0;newnode(0);newnode(-1);
14          fa[0] = 1;
15          for(int i = 0; i < 26; i++) quick[0][i] = quick[1][i] = 1;
16          lastL = lastR = 0;
17          sL = MAXN; sR = MAXN-1;
```

```
18          S[sL] = S[sR] = -1;
19      }
20      void push_front(int c) {
21          S[--sL] = c; S[sL-1]=-1;
22          int p = lastL;
23          if (S[sL+len[p]+1] ^ c) p = quick[p][c];
24          if (!ch[p][c]) {
25              int np = newnode(len[p]+2), q = fa[p];
26              if (S[sL+len[q]+1] ^ c) q = quick[q][c];
27              fa[np] = ch[q][c];
28              memcpy(quick[np], quick[fa[np]], sizeof(quick[np]));
29              quick[np][S[sL+len[fa[np]]]] = fa[np];
30              ch[p][c] = np;
31              num[np] = num[fa[np]] + 1;
32          }
33          lastL = ch[p][c];
34          cnt[lastL]++;
35          if (len[lastL] == sR-sL+1) lastR = lastL;
36      }
37      void push_back(int c) {
38          S[++sR] = c; S[sR+1]=-1;
39          int p = lastR;
40          if (S[sR-len[p]-1] ^ c) p = quick[p][c];
41          if (!ch[p][c]) {
42              int np = newnode(len[p]+2), q = fa[p];
43              if (S[sR-len[q]-1] ^ c) q = quick[q][c];
44              fa[np] = ch[q][c];
45              memcpy(quick[np],quick[fa[np]], sizeof(quick[np]));
46              quick[np][S[sR-len[fa[np]]]] = fa[np];
47              ch[p][c] = np;
48              num[np] = num[fa[np]] + 1;
49          }
50          lastR = ch[p][c];
51          cnt[lastR]++;
52          if (len[lastR] == sR-sL+1) lastL = lastR;
53      }
54      int c[MAXN<<1], rs[MAXN<<2];
55      void count() {
56          for (int i = 0; i < pcnt; i++) c[i] = 0;
57          for (int i = 2; i < pcnt; i++) c[len[i]]++;
58          for (int i = 1; i < pcnt; i++) c[i] += c[i-1];
59          for (int i = 2; i < pcnt; i++) rs[c[len[i]]--] = i;
60          for (int i = pcnt-1; i; i--)  cnt[fa[rs[i]]]+=cnt[rs[i]];
61      }
62  };
```

3.5   Substring Automaton

```
1  for(int j = 0; j < 26; j++)
2      ch[n][j] = ch[n+1][j] = n + 1; //或者 −1
3  for(int i = n; i >= 1; i--) {
4      for(int j = 0; j < 26; j++)
5          ch[i-1][j] = ch[i][j];
6      ch[i-1][s[i]-'a'] = i;
7  }
```

当字符集过大时使用主席树维护 ch

4　Math

4.1　Algebra

4.1.1　FFT

```
1   //不预处理精度
2   const double pi = acos(-1.0);
3   const int MAXN = 300003;
4   struct comp {
5       double x, y;
6       comp operator + (const comp& a) const { return (comp) {x + a.x, y + a.y}; }
7       comp operator - (const comp& a) const { return (comp) {x - a.x, y - a.y}; }
8       comp operator * (const comp& a) const { return (comp) {x * a.x - y * a.y, x * a.y +
    y * a.x}; }
9   };
10  int rev[MAXN], T;
11  comp tmp;
12  void fft(comp *a, int r) {
13      if(r == -1) for(int i = 0; i < T; i++) a[i] = a[i] * a[i];
14      for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
15      for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
16          comp step = (comp) {cos(pi / mid), r * sin(pi / mid)};
17          for(int j = 0; j < T; j += i) {
18              comp cur = (comp) {1, 0};
19              for(int k = j; k < j + mid; k++, cur = cur * step) {
20                  tmp = a[k + mid] * cur;
21                  a[k + mid] = a[k] - tmp;
22                  a[k] = a[k] + tmp;
23              }
24          }
25      }
26      if(r == -1) for(int i = 0; i < T; i++) a[i].y = (int)(a[i].y / T / 2 + 0.5);
27  }
28  comp A[MAXN];
29  void init(int n) {
30      //A[0] = (comp) {0, 0};
31      for(T = 1; T <= n; T <<= 1);
32      for(int i = 1; i < T; i++) {
33          if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
34          else rev[i] = rev[i >> 1] >> 1;
35          //A[i] = (comp) {0, 0};
36      }
37  }
38  //预处理精度
39  int rev[MAXN], T;
40  comp Sin[MAXN], tmp;
41  void fft(comp *a, int r) {
42      if(r == -1) {
43          for(int i = 0; i < (T >> 1); i++) Sin[i].y = -Sin[i].y;
44          for(int i = 0; i < T; i++) a[i] = a[i] * a[i];
45      }
46      for(int i = 1; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
47      for(int i = 2, mid = 1, s = (T >> 1); i <= T; mid = i, i <<= 1, s >>= 1) {
48          for(int j = 0; j < T; j += i) {
49              for(int k = j, cur = 0; k < j + mid; k++, cur += s) {
50                  tmp = a[k + mid] * Sin[cur];
51                  a[k + mid] = a[k] - tmp;
52                  a[k] = a[k] + tmp;
```

```
53                  }
54              }
55          }
56      if(r == -1) for(int i = 0; i < T; i++) a[i].y = (int)(a[i].y / T / 2 + 0.5);
57  }
58  comp A[MAXN];
59  void init(int n) {
60      for(T = 1; T <= n; T <<= 1);
61      for(int i = 0; i < T; i++) {
62          if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
63          else rev[i] = rev[i >> 1] >> 1;
64          //A[i] = (comp) {0, 0};
65      }
66      for(int i = 0; i < (T >> 1); i++) {
67          Sin[i] = (comp) {cos(2 * pi * i / T), sin(2 * pi * i / T)};
68      }
69  }
70  int main() {
71      scanf("%d%d", &n, &m);
72      init(n + m);
73      for(int i = 0; i <= n; i++) scanf("%lf", &A[i].x);
74      for(int i = 0; i <= m; i++) scanf("%lf", &A[i].y);
75      fft(A, 1);
76      fft(A, -1);
77      for(int i = 0; i <= n + m; i++) printf("%d%c", (int)(A[i].y), i == n + m ? '\n' : '
        ');
78      return 0;
79  }
```

4.1.2 NTT

4.常用NTT模数:

以下模数的共同$g = 3189$

$p = r \times 2^k + 1$	k	g
104857601	22	3
167772161	25	3
469762049	26	3
950009857	21	7
998244353	23	3
1004535809	21	3
2013265921	27	31
2281701377	27	3
3221225473	30	5

```
1   const int MAXN = 300005, G = 3, mod = 998244353; //or (479LL<<21) + 1
2   int rev[MAXN], T;
3   LL qpow(LL x, LL y) {
4       LL res = 1;
5       while(y) {
6           if(y & 1) res = res * x % mod;
7           x = x * x % mod;
8           y >>= 1;
9       }
10      return res;
11  }
12  LL A[MAXN], B[MAXN];
```

```
13  void ntt(LL *a, int r) {
14      if(r == -1) for(int i = 0; i < T; i++) A[i] = A[i] * B[i] % mod;
15      for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
16      for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
17          LL gn = qpow(G, (mod - 1) / i);
18          if(r == -1) gn = qpow(gn, mod - 2);
19          for(int j = 0; j < T; j += i) {
20              LL cur = 1, tmp;
21              for(int k = j; k < j + mid; k++, cur = cur * gn % mod) {
22                  tmp = a[k + mid] * cur % mod;
23                  a[k + mid] = ((a[k] - tmp) % mod + mod) % mod;
24                  a[k] = (a[k] + tmp) % mod;
25              }
26          }
27      }
28      if(r == -1) {
29          LL inv = qpow(T, mod - 2);
30          for(int i = 0; i < T; i++) a[i] = a[i] * inv % mod;
31      }
32  }
33  void init(int n) {
34      for(T = 1; T <= n; T <<= 1);
35      for(int i = 0; i < T; i++) {
36          if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
37          else rev[i] = rev[i >> 1] >> 1;
38      }
39  }
```

4.1.3 MTT

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   typedef long long LL;
4   const double pi = acos(-1.0);
5   const int MAXN = 300003;
6   struct comp {
7       double x, y;
8       comp operator + (const comp& a) const { return (comp) {x + a.x, y + a.y}; }
9       comp operator - (const comp& a) const { return (comp) {x - a.x, y - a.y}; }
10      comp operator * (const comp& a) const { return (comp) {x * a.x - y * a.y, x * a.y +
    y * a.x}; }
11  };
12  #define conj(a) ((comp){a.x, -a.y})
13  int rev[MAXN], T;
14  comp Sin[MAXN], tmp;
15  void fft(comp *a, int r) {
16      for(int i = 1; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
17      for(int i = 2, mid = 1, s = (T >> 1); i <= T; mid = i, i <<= 1, s >>= 1) {
18          for(int j = 0; j < T; j += i) {
19              for(int k = j, cur = 0; k < j + mid; k++, cur += s) {
20                  tmp = a[k + mid] * Sin[cur];
21                  a[k + mid] = a[k] - tmp;
22                  a[k] = a[k] + tmp;
23              }
24          }
25      }
26  }
27  void init(int n) {
28      for(T = 1; T <= n; T <<= 1);
29      for(int i = 0; i < T; i++) {
```

```
30              if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
31              else rev[i] = rev[i >> 1] >> 1;
32          }
33          for(int i = 0; i < (T >> 1); i++) {
34              Sin[i] = (comp) {cos(2 * pi * i / T), sin(2 * pi * i / T)};
35          }
36  }
37  int n, m, mod;
38  void mtt(int *x, int *y) {
39      for(int i = 0; i < T; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
40      static comp a[MAXN], b[MAXN];
41      static comp dfta[MAXN], dftb[MAXN], dftc[MAXN], dftd[MAXN];
42      for(int i = 0; i < T; i++) {
43          a[i] = {x[i] & 0x7fff, x[i] >> 15};
44          b[i] = {y[i] & 0x7fff, y[i] >> 15};
45      }
46      fft(a, 1); fft(b, 1);
47      for(int i = 0; i < T; i++) {
48          int j = (T - i) & (T - 1);
49          static comp da, db, dc, dd;
50          da = (a[i] + conj(a[j])) * (comp){0.5, 0};
51          db = (a[i] - conj(a[j])) * (comp){0, -0.5};
52          dc = (b[i] + conj(b[j])) * (comp){0.5, 0};
53          dd = (b[i] - conj(b[j])) * (comp){0, -0.5};
54          dfta[j] = da * dc;
55          dftb[j] = da * dd;
56          dftc[j] = db * dc;
57          dftd[j] = db * dd;
58      }
59      for(int i = 0; i < T; i++) {
60          a[i] = dfta[i] + dftb[i] * (comp) {0, 1};
61          b[i] = dftc[i] + dftd[i] * (comp) {0, 1};
62      }
63      //for(int i = 0; i < (T >> 1); i++) Sin[i].y = -Sin[i].y;
64      fft(a, -1); fft(b, -1);
65      for(int i = 0; i < T; i++) {
66          static int da, db, dc, dd;
67          da = (LL)(a[i].x / T + 0.5) % mod;
68          db = (LL)(a[i].y / T + 0.5) % mod;
69          dc = (LL)(b[i].x / T + 0.5) % mod;
70          dd = (LL)(b[i].y / T + 0.5) % mod;
71          x[i] = ((da + ((LL)(db + dc) << 15) + ((LL)dd << 30)) % mod + mod) % mod;
72      }
73  }
74  int main() {
75      static int a[MAXN], b[MAXN];
76      scanf("%d%d%d", &n, &m, &mod);
77      for(int i = 0; i <= n; i++) scanf("%d", a + i);
78      for(int i = 0; i <= m; i++) scanf("%d", b + i);
79      init(n + m);
80      mtt(a, b);
81      for(int i = 0; i <= n + m; i++) printf("%d%c", a[i], i == n + m ? '\n' : ' ');
82      return 0;
83  }
```

4.1.4 FWT

Hint: $n = 2^k$

```
1  void FWT(LL *a,int n) {
2      for(int i = 2;i <= n; i <<= 1) {
```

```
3            for(int j = 0; j < n; j += i) {
4                for(int d = 0, w = i >> 1; d < w; d++){
5                    LL u = a[j + d], v = a[j + d + w];
6                    //xor: a[j + d] = u + v, a[j + d + w] = u - v;
7                    //and: a[j + d] = u + v;
8                    //or : a[j + d + w] = u + v;
9                }
10           }
11       }
12   }
13   void UFWT(LL *a, int n) {
14       for(int i = 2; i <= n; i <<= 1) {
15           for(int j = 0; j < n; j += i) {
16               for(int d = 0, w = i >> 1; d < w; d++) {
17                   LL u = a[j + d], v = a[j + d + w];
18                   //xor: a[j + d] = (u + v) / 2, a[j + d + w] = (u - v) / 2;
19                   //and: a[j + d] = u - v;
20                   //or : a[j + d + w] = v - u;
21               }
22           }
23       }
24   }
25   void solve(int n) {
26       FWT(a, n); FWT(b, n);
27       for(int i = 0; i < n; i++) a[i] = a[i] * b[i];
28       UFWT(a, n);
29   }
```

4.1.5 FFT Divide and Conquer

$$f_i = \sum_{j=1}^{i-1} f_j \cdot g_{i-j}$$

```
1    #include <bits/stdc++.h>
2    using namespace std;
3    typedef long long LL;
4    const int MAXN = 300005, G = 3, mod = 998244353;
5    namespace NTT {
6        LL A[MAXN], B[MAXN]
7        int rev[MAXN], T;
8        LL qpow(LL x, LL y) {
9            LL res = 1;
10           while(y) {
11               if(y & 1) res = res * x % mod;
12               x = x * x % mod;
13               y >>= 1;
14           }
15           return res;
16       }
17       void ntt(LL *a, int r) {
18           for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
19           for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
20               LL gn = qpow(G, (mod - 1) / i);
21               if(r == -1) gn = qpow(gn, mod - 2);
22               for(int j = 0; j < T; j += i) {
23                   LL cur = 1, tmp;
24                   for(int k = j; k < j + mid; k++, cur = cur * gn % mod) {
25                       tmp = a[k + mid] * cur % mod;
```

```
26                        a[k + mid] = ((a[k] - tmp) % mod + mod) % mod;
27                        a[k] = (a[k] + tmp) % mod;
28                    }
29                }
30            }
31            if(r == -1) {
32                LL inv = qpow(T, mod - 2);
33                for(int i = 0; i < T; i++) a[i] = a[i] * inv % mod;
34            }
35        }
36        void init(int n) {
37            for(T = 1; T <= n; T <<= 1);
38            for(int i = 0; i < T; i++) {
39                if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
40                else rev[i] = rev[i >> 1] >> 1;
41            }
42        }
43    }
44    LL f[MAXN], g[MAXN];
45    using namespace NTT;
46    void solve(int l, int r) {
47        if(l == r) return;
48        int mid = (l + r) >> 1;
49        solve(l, mid);
50        init(r - l);
51        for(int i = 0; i < T; i++) A[i] = B[i] = 0;
52        for(int i = 0; i <= mid - l; i++) A[i] = f[i + l];
53        for(int i = 0; i <= r - l; i++) B[i] = g[i];
54        ntt(A, 1); ntt(B, 1);
55        for(int i = 0; i < T; i++) A[i] = A[i] * B[i] % mod;
56        ntt(A, -1);
57        for(int i = mid + 1; i <= r; i++) f[i] =(f[i] + A[i - l]) % mod;
58        solve(mid + 1, r);
59    }
60    int main() {
61        int n; scanf("%d", &n);
62        for(int i = 1; i < n; i++) scanf("%lld", g + i);
63        f[0] = 1;
64        solve(0, n - 1);
65        for(int i = 0; i < n; i++) printf("%lld%c", f[i], i == n - 1 ? '\n' : ' ');
66        return 0;
67    }
```

4.1.6 Linear Basis

```
1    //dynamic
2    const int D = 60;
3    struct Basis {
4        vector<int> ind;
5        vector<LL> base;
6        Basis() {
7            ind.resize(D, -1);
8            base.resize(D);
9        }
10       bool update(LL x, int id) {
11           for(int i = 0; i < D; i++) if(~ind[i] && x >> i & 1) {
12               x ^= base[i];
13           }
14           if(!x) return 1;
15           int pos = __builtin_ctzll(x);
```

```
16              ind[pos] = id;
17              base[pos] = x;
18              return 0;
19          }
20  };
21  //array
22  int Gauss(int n, int m) {
23      int num = 1;
24      for(int x = 1; x <= n && x <= m; x++) {
25          int t = 0;
26          for(int j = x; j <= m; j++) if(g[j][x]) { t = j; break; }
27          if(t) {
28              swap(g[x], g[t]);
29              for(int i = x + 1; i <= n; i++) {
30                  if(g[i][x]) {
31                      for(int k = 1; k <= m; k++) g[i][k] ^= g[x][k];
32                  }
33              }
34              num++;
35          }
36      }
37      return --num;
38  }
39  //long long
40  int Gauss() {
41      int num = 1;
42      for(int k = 61; k >= 0; k--) {
43          int t = 0;
44          for(int j = num; j <= cnt; j++) if((A[j] >> k) & 1) { t = j; break; }
45          if(t) {
46              swap(A[t], A[num]);
47              for(int j = num + 1; j <= cnt; j++) if((A[j] >> k) & 1) A[j] ^= A[num];
48              num++;
49          }
50      }
51      return --num;
52  }
53  //det
54  LL det(int n){
55      LL ret = 1;
56      for(int i = 1; i < n; i++){
57          for(int j = i + 1; j < n; j++)
58              while(a[j][i]){
59                  LL t = (LL)a[i][i] / a[j][i];
60                  for(int k = i; k < n; k++) {
61                      a[i][k] = (a[i][k]-a[j][k] * t);
62                      swap(a[i][k], a[j][k]);
63                  }
64                  ret = -ret;
65              }
66          if(a[i][i] == 0)return 0;
67          ret = ret * a[i][i];
68      }
69      if(ret<0)ret = -ret;
70      return ret;
71  }
```

4.1.7 Polynomial

Inverse:

$$A_x * B'_x \equiv 1 \qquad\qquad (\mathrm{mod}\ x^{\frac{n}{2}})$$
$$A_x * B_x \equiv 1 \qquad\qquad (\mathrm{mod}\ x^n)$$
$$B_x \equiv 2 \cdot B'_x - A_x \cdot {B'}_x^2 \qquad (\mathrm{mod}\ x^n)$$

Division:

$$A_r(x) = x^n A(\frac{1}{x}) \Rightarrow A_r(x)[i] = A(x)[n-i]$$

$$A(x) = B(x) * Q(x) + R(x)$$
$$Q_r(x) = A_r(x) * B_r^{-1}(x)$$
$$R(x) = A(x) - B(x) * Q(x)$$

```
1   //NTT模数
2   #include <bits/stdc++.h>
3   using namespace std;
4   typedef long long LL;
5   const int MAXN = 300005, G = 3, mod = 998244353;
6   namespace NTT {
7       int rev[MAXN], T;
8       LL qpow(LL x, LL y) {
9           LL res = 1;
10          while(y) {
11              if(y & 1) res = res * x % mod;
12              x = x * x % mod;
13              y >>= 1;
14          }
15          return res;
16      }
17      void ntt(LL *a, int r) {
18          for(int i = 0; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
19          for(int i = 2, mid = 1; i <= T; mid = i, i <<= 1) {
20              LL gn = qpow(G, (mod - 1) / i);
21              if(r == -1) gn = qpow(gn, mod - 2);
22              for(int j = 0; j < T; j += i) {
23                  LL cur = 1, tmp;
24                  for(int k = j; k < j + mid; k++, cur = cur * gn % mod) {
25                      tmp = a[k + mid] * cur % mod;
26                      a[k + mid] = ((a[k] - tmp) % mod + mod) % mod;
27                      a[k] = (a[k] + tmp) % mod;
28                  }
29              }
30          }
31          if(r == -1) {
32              LL inv = qpow(T, mod - 2);
33              for(int i = 0; i < T; i++) a[i] = a[i] * inv % mod;
34          }
35      }
36      void init(int n) {
37          for(T = 1; T <= n; T <<= 1);
38          for(int i = 0; i < T; i++) {
39              if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
40              else rev[i] = rev[i >> 1] >> 1;
```

```
41            }
42        }
43 }
44 namespace poly {
45 using namespace NTT;
46     void mul(LL *a, LL *b, LL *c, int n, int m) {
47         init(n + m);
48         static LL x[MAXN], y[MAXN];
49         for(int i = 0; i < T; i++) {
50             x[i] = i <= n ? a[i] : 0;
51             y[i] = i <= m ? b[i] : 0;
52         }
53         ntt(x, 1); ntt(y, 1);
54         for(int i = 0; i < T; i++) c[i] = x[i] * y[i] % mod;
55         ntt(c, -1);
56     }
57     void poly_inv(LL *a, LL *b, int n) {
58         if(n == 1) {
59             b[0] = qpow(a[0], mod - 2);
60             return;
61         }
62         static LL c[MAXN], d[MAXN];
63         memset(c, 0, n * sizeof(LL));
64         poly_inv(a, c, n >> 1);
65         for(int i = 0; i < n; i++) {
66             d[i] = a[i];
67         }
68         init(n);
69         ntt(c, 1); ntt(d, 1);
70         for(int i = 0; i < T; i++) b[i] = c[i] * (2 + mod - d[i] * c[i] % mod) % mod;
71         ntt(b, -1);
72         for(int i = n; i < T; i++) b[i] = 0;
73     }
74     void inv(LL *a, LL *b, int n) {//A must be different from B
75         init(n);
76         poly_inv(a, b, T);
77     }
78     //A_x = B_x * Q_x + R_x;
79     void div(LL *A, LL *B, LL *Q, LL *R, int n, int m) {
80         static LL f[MAXN], g[MAXN], inv_g[MAXN];
81         for(int i = 0; i <= n; i++) f[n - i] = A[i];
82         for(int i = 0; i <= m; i++) g[m - i] = B[i];
83         //inv(G_r)
84         for(int i = n - m + 1; i <= m; i++) g[i] = 0;
85         inv(g, inv_g, n - m);
86         //Q
87         mul(f, inv_g, f, n, n - m);
88         for(int i = 0; i <= n - m; i++) Q[i] = f[n - m - i];
89         //R
90         mul(Q, B, f, n - m, m);
91         for(int i = 0; i < m; i++) R[i] = (A[i] + mod - f[i]) % mod;
92     }
93 }
94 LL A[MAXN], B[MAXN];
95 LL Q[MAXN], R[MAXN];
96 int main() {
97     ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
98     int n, m; cin >> n >> m;
99     for(int i = 0; i <= n; i++) cin >> A[i];
100     for(int i = 0; i <= m; i++) cin >> B[i];
101     poly::div(A, B, Q, R, n, m);
```

```
102        for(int i = 0; i <= n - m; i++) cout << Q[i] << " "; cout << endl;
103        for(int i = 0; i < m; i++) cout << R[i] << " "; cout << endl;
104        return 0;
105    }
106    //非NTT模数求逆
107    #include <bits/stdc++.h>
108    using namespace std;
109    typedef long long LL;
110    const double pi = acos(-1.0);
111    const int MAXN = 300003;
112    struct comp {
113        double x, y;
114        comp operator + (const comp& a) const { return (comp) {x + a.x, y + a.y}; }
115        comp operator - (const comp& a) const { return (comp) {x - a.x, y - a.y}; }
116        comp operator * (const comp& a) const { return (comp) {x * a.x - y * a.y, x * a.y +
       y * a.x}; }
117    };
118    #define conj(a) ((comp){a.x, -a.y})
119    int rev[MAXN], T;
120    comp Sin[MAXN], tmp;
121    void fft(comp *a, int r) {
122        for(int i = 1; i < T; i++) if(rev[i] > i) swap(a[rev[i]], a[i]);
123        for(int i = 2, mid = 1, s = (T >> 1); i <= T; mid = i, i <<= 1, s >>= 1) {
124            for(int j = 0; j < T; j += i) {
125                for(int k = j, cur = 0; k < j + mid; k++, cur += s) {
126                    tmp = a[k + mid] * Sin[cur];
127                    a[k + mid] = a[k] - tmp;
128                    a[k] = a[k] + tmp;
129                }
130            }
131        }
132    }
133    void init(int n) {
134        //for(T = 1; T <= n; T <<= 1);
135        T = n << 1;
136        for(int i = 0; i < T; i++) {
137            if(i & 1) rev[i] = (rev[i >> 1] >> 1) ^ (T >> 1);
138            else rev[i] = rev[i >> 1] >> 1;
139        }
140        for(int i = 0; i < (T >> 1); i++) {
141            Sin[i] = (comp) {cos(2 * pi * i / T), sin(2 * pi * i / T)};
142        }
143    }
144    const int mod = 1e9 + 7;
145    void mtt(int *x, int *y) {
146        for(int i = 0; i < T; i++) (x[i] += mod) %= mod, (y[i] += mod) %= mod;
147        static comp a[MAXN], b[MAXN];
148        static comp dfta[MAXN], dftb[MAXN], dftc[MAXN], dftd[MAXN];
149        for(int i = 0; i < T; i++) {
150            a[i] = {x[i] & 0x7fff, x[i] >> 15};
151            b[i] = {y[i] & 0x7fff, y[i] >> 15};
152        }
153        fft(a, 1); fft(b, 1);
154        for(int i = 0; i < T; i++) {
155            int j = (T - i) & (T - 1);
156            static comp da, db, dc, dd;
157            da = (a[i] + conj(a[j])) * (comp){0.5, 0};
158            db = (a[i] - conj(a[j])) * (comp){0, -0.5};
159            dc = (b[i] + conj(b[j])) * (comp){0.5, 0};
160            dd = (b[i] - conj(b[j])) * (comp){0, -0.5};
161            dfta[j] = da * dc;
```

```
162              dftb[j] = da * dd;
163              dftc[j] = db * dc;
164              dftd[j] = db * dd;
165          }
166          for(int i = 0; i < T; i++) {
167              a[i] = dfta[i] + dftb[i] * (comp) {0, 1};
168              b[i] = dftc[i] + dftd[i] * (comp) {0, 1};
169          }
170          //for(int i = 0; i < (T >> 1); i++) Sin[i].y = -Sin[i].y;
171          fft(a, -1); fft(b, -1);
172          for(int i = 0; i < T; i++) {
173              static int da, db, dc, dd;
174              da = (LL)(a[i].x / T + 0.5) % mod;
175              db = (LL)(a[i].y / T + 0.5) % mod;
176              dc = (LL)(b[i].x / T + 0.5) % mod;
177              dd = (LL)(b[i].y / T + 0.5) % mod;
178              x[i] = ((da + ((LL)(db + dc) << 15) + ((LL)dd << 30)) % mod + mod) % mod;
179          }
180  }
181  LL qpow(LL x, LL y) {
182      LL res = 1;
183      while(y) {
184          if(y & 1) res = res * x % mod;
185          x = x * x % mod;
186          y >>= 1;
187      }
188      return res;
189  }
190  void poly_inv(int *a, int *b, int n) {
191      if(n == 1) {
192          b[0] = qpow(a[0], mod - 2);
193          return;
194      }
195      static int c[MAXN], cc[MAXN], d[MAXN];
196      memset(c, 0, n * sizeof(int)); memset(cc, 0, n * sizeof(int)); memset(d, 0, n *
           sizeof(int));
197      poly_inv(a, c, n >> 1);
198      for(int i = 0; i < n; i++) cc[i] = c[i], d[i] = a[i];
199      init(n);
200      mtt(cc, c);
201      mtt(d, cc);
202      for(int i = 0; i < T; i++) b[i] = (2 * c[i] % mod + mod - d[i]) % mod;
203      for(int i = 0; i < n; i++) b[n + i] = 0;
204  }
205  int A[MAXN], B[MAXN];
206  int main() {
207      ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
208      int n; cin >> n; n--;
209      for(T = 1; T <= n; T <<= 1);
210      for(int i = 0; i <= n; i++) cin >> A[i];
211      poly_inv(A, B, T);
212      for(int i = 0; i <= n; i++) cout << B[i] << " ";
213      cout << endl;
214      return 0;
215  }
216
217  //LOJ挑战多项式(备用)
218  #include <bits/stdc++.h>
219  #define rep(i, a, b) for (int i = (a); i <= (b); i++)
220  #define per(i, a, b) for (int i = (a); i >= (b); i--)
221  #define REP(i, n) for (int i = (0); i < (n); i++)
```

```
222  #define fi first
223  #define se second
224  #define mp make_pair
225  #define pb push_back
226  using namespace std;
227  typedef unsigned long long ull;
228  typedef pair<int, int> pii;
229  typedef vector<int> poly;
230  typedef long long ll;
231  struct ano {
232      char a[1 << 25], *s;
233      char b[1 << 25], *t;
234      ano() : s(a), t(b) { a[fread(a, 1, sizeof a, stdin)] = 0; }
235      ~ano() { fwrite(b, 1, t - b, stdout); }
236      operator int() {
237          int x = 0;
238          while (*s < 48) ++s;
239          while (*s > 32) x = x * 10 + *s++ - 48;
240          return x;
241      }
242      ll in() {
243          ll x = 0;
244          while (*s < 48) ++s;
245          while (*s > 32) x = x * 10 + *s++ - 48;
246          return x;
247      }
248      void out(int x, char e = ' ') {
249          static char c[12];
250          char *i = c;
251          if (!x)
252              *t++ = 48;
253          else {
254              while (x) {
255                  int y = x / 10;
256                  *i++ = x - y * 10 + 48, x = y;
257              }
258              while (i != c) *t++ = *--i;
259          }
260          *t++ = e;
261      }
262  } buf;
263  const int mod = 998244353;
264  namespace Poly {
265  const int N = (1 << 20) + 5, g = 3;
266  inline int power(int x, int p) {
267      int res = 1;
268      for (; p; p >>= 1, x = (ll)x * x % mod)
269          if (p & 1)
270              res = (ll)res * x % mod;
271      return res;
272  }
273  inline int fix(const int x) { return x >= mod ? x - mod : x; }
274  void dft(poly &A, int n) {
275      static ull W[N << 1], *H[30], *las = W, mx = 0;
276      for (; mx < n; mx++) {
277          H[mx] = las;
278          ull w = 1, wn = power(g, (mod - 1) >> (mx + 1));
279          REP(i, 1 << n) *las++ = w, w = w * wn % mod;
280      }
281      if (A.size() != (1 << n))
282          A.resize(1 << n);
```

```
283        static ull a[N];
284        for (int i = 0, j = 0; i < (1 << n); ++i) {
285            a[i] = A[j];
286            for (int k = 1 << (n - 1); (j ^= k) < k; k >>= 1)
287                ;
288        }
289        for (int k = 0, d = 1; k < n; k++, d <<= 1)
290            for (int i = 0; i < (1 << n); i += (d << 1)) {
291                ull *l = a + i, *r = a + i + d, *w = H[k], t;
292                for (int j = 0; j < d; j++, l++, r++) {
293                    t = (*r) * (*w++) % mod;
294                    *r = *l + mod - t, *l += t;
295                }
296            }
297        REP(i, 1 << n) A[i] = a[i] % mod;
298    }
299    void idft(poly &a, int n) {
300        a.resize(1 << n), reverse(a.begin() + 1, a.end());
301        dft(a, n);
302        int inv = power(1 << n, mod - 2);
303        REP(i, 1 << n) a[i] = (ll)a[i] * inv % mod;
304    }
305    poly FIX(poly a) {
306        while (!a.empty() && !a.back()) a.pop_back();
307        return a;
308    }
309    poly add(poly a, poly b, int op = 0) {
310        a.resize(max(a.size(), b.size()));
311        REP(i, b.size()) a[i] = fix(op ? a[i] + mod - b[i] : a[i] + b[i]);
312        return FIX(a);
313    }
314    poly mul(poly a, poly b, int t = 1) {
315        if (t == 1 && a.size() + b.size() <= 24) {
316            poly c(a.size() + b.size(), 0);
317            REP(i, a.size()) REP(j, b.size()) c[i + j] = (c[i + j] + (ll)a[i] * b[j]) % mod;
318            return FIX(c);
319        }
320        int n = 1, aim = a.size() * t + b.size();
321        while ((1 << n) <= aim) n++;
322        dft(a, n), dft(b, n);
323        if (t == 1)
324            REP(i, 1 << n) a[i] = (ll)a[i] * b[i] % mod;
325        else
326            REP(i, 1 << n) a[i] = (ll)a[i] * a[i] % mod * b[i] % mod;
327        idft(a, n), a.resize(aim);
328        return FIX(a);
329    }
330    poly mul(poly a, int b) {
331        REP(i, a.size()) a[i] = (ll)a[i] * b % mod;
332        return FIX(a);
333    }
334
335    poly inv(poly a, int n) {  // a[0] != 0
336        a.resize(n);
337        poly b;
338        if (n == 1) {
339            b.pb(power(a[0], mod - 2));
340            return b;
341        }
342        b = inv(a, n + 1 >> 1);
343        b = add(mul(b, 2), mul(b, a, 2), 1);
```

```
344        return b.resize(n), b;
345 }
346
347 poly Der(poly a) {
348        REP(i, a.size() - 1) a[i] = (ll)(i + 1) * a[i + 1] % mod;
349        return a.pop_back(), a;
350 }
351 poly Int(poly a) {
352        static int inv[N];
353        inv[1] = 1;
354        a.pb(0);
355        rep(i, 2, a.size()) inv[i] = (ll)(mod - mod / i) * inv[mod % i] % mod;
356        per(i, a.size() - 1, 1) a[i] = (ll)a[i - 1] * inv[i] % mod;
357        return a[0] = 0, a;
358 }
359 poly Ln(poly a, int n) {   // a[0] = 1
360        a = mul(Der(a), inv(a, n)), a.resize(n - 1);
361        return FIX(Int(a));
362 }
363 poly Exp(poly a, int n) {   // a[0] = 0
364        a.resize(n);
365        poly b, one(1, 1);
366        if (n == 1)
367            return one;
368        b = Exp(a, n + 1 >> 1);
369        b = mul(b, add(add(a, Ln(b, n), 1), one));
370        return b.resize(n), b;
371 }
372
373 poly Div(poly a, poly b) {
374        poly c;
375        int n = a.size() - 1, m = b.size() - 1;
376        if (n < m)
377            return c;
378        reverse(a.begin(), a.end());
379        a.resize(n - m + 1);
380        reverse(b.begin(), b.end());
381        b.resize(n - m + 1);
382        c = mul(a, inv(b, n - m + 1));
383        c.resize(n - m + 1);
384        return reverse(c.begin(), c.end()), c;
385 }
386 poly Mod(poly a, poly b) { return FIX(add(a, mul(Div(a, b), b), 1)); }
387 inline int chk(int x) { return power(x, (mod - 1) / 2) == 1; }
388 inline int R() { return rand() % mod; }
389 inline pii mul(pii a, pii b, int w) {
390        return mp(((ll)a.fi * b.fi + (ll)a.se * b.se % mod * w) % mod, ((ll)a.fi * b.se + (
           ll)a.se * b.fi) % mod);
391 }
392 inline int Sqrt(int x) {
393        if (!chk(x))
394            return -1;
395        int a = R();
396        while (chk(((ll)a * a - x + mod) % mod)) a = R();
397        int w = ((ll)a * a - x + mod) % mod, p = (mod + 1) / 2;
398        pii res = mp(1, 0), t = mp(a, 1);
399        for (; p; p >>= 1, t = mul(t, t, w))
400            if (p & 1)
401                res = mul(res, t, w);
402        assert(!res.se);
403        return min(res.fi, mod - res.fi);
```

```
404  }
405  poly Sqrt(poly a, int n) {
406      if (n == 1) {
407          poly b(1, Sqrt(a[0]));
408          return b;
409      }
410      a.resize(n);
411      poly b = Sqrt(a, n + 1 >> 1);
412      b = mul(add(b, mul(a, inv(b, n))), (mod + 1) / 2);
413      return b.resize(n), b;
414  }
415  poly fastpow(poly a, ll k, int n) {
416      a.resize(n), a = FIX(a);
417      if (!a.size())
418          return a;
419      int st = 0, base = 0;
420      while (!a[st]) ++st;
421      if (st * k >= n)
422          return a.resize(0), a;
423      REP(i, a.size() - st) a[i] = a[i + st];
424      if (st)
425          a.resize(a.size() - st);
426      base = a[0];
427      ll inv = power(base, mod - 2);
428      REP(i, a.size()) a[i] = a[i] * inv % mod;
429      a = FIX(Exp(mul(Ln(a, n), k % mod), n));
430      ;
431      if (st) {
432          reverse(a.begin(), a.end());
433          a.resize(a.size() + st * k);
434          reverse(a.begin(), a.end());
435          a.resize(n), a = FIX(a);
436      }
437      base = power(base, k);
438      REP(i, a.size()) a[i] = (ll)a[i] * base % mod;
439      return FIX(a);
440  }
441  }  // namespace Poly
442  using namespace Poly;
443  int main() {
444      int n = buf + 1, K = buf;
445      poly a(n, 0), b;
446      REP(i, n) a[i] = buf;
447      b = add(a, Exp(Int(inv(Sqrt(a, n), n)), n), 1);
448      b[0] = (b[0] + 2 - a[0] + mod) % mod;
449      a = Ln(b, n);
450      a[0]++;
451      b = Der(fastpow(a, K, n));
452      b.resize(n - 1);
453      REP(i, n - 1) buf.out(b[i]);
454      return 0;
455  }
```

4.1.8   Lagrange Polynomial

$$L(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

```
1   //O(n^2)
2   #include <bits/stdc++.h>
3   using namespace std;
4   typedef long long LL;
5   typedef pair<int, int> P;
6   const int MAXN = 3005, mod = 998244353;
7   int exgcd(int a, int b, int &x, int &y) {
8       int d = a;
9       if(b != 0) {
10          d = exgcd(b, a % b, y, x);
11          y -= (a / b) * x;
12      }
13      else {
14          x = 1; y = 0;
15      }
16      return d;
17  }
18  int inv(int a) {
19      int x, y;
20      exgcd(a, mod, x, y);
21      return (x % mod + mod) % mod;
22  }
23  struct Lagrange {
24      int n, a[MAXN][2];
25      void init() {
26          for(int i = 0; i <= n; i++) a[i][0] = a[i][1] = 0;
27          n = 0;
28          a[0][1] = 1;
29      }
30      int query(int x, int q = 0) {
31          int res = 0;
32          for(int i = n; i >= 0; i--) res = ((LL)res * x + a[i][q]) % mod;
33          return res;
34      }
35      void update(int x, int y) {
36          a[n][0] = 0;
37          int v = (LL)(y - query(x) + mod) % mod * inv(query(x, 1)) % mod;
38          for(int i = 0; i <= n; i++) a[i][0] = (a[i][0] + (LL)a[i][1] * v) % mod;
39          a[++n][1] = 0;
40          for(int i = n; i; i--) a[i][1] = (a[i - 1][1] + (LL)a[i][1] * (mod - x)) % mod;
41          a[0][1] = (LL)a[0][1] * (mod - x) % mod;
42      }
43  }p;
44  int main() {
45      ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
46      int Q;
47      cin >> Q;
48      int op, x, y;
49      p.n = 0;
50      p.init();
51      while(Q--) {
52          cin >> op >> x;
53          if(op == 1) {
54              cin >> y;
55              p.update(x, y);
56          }
57          else cout << p.query(x) << endl;
58      }
59      return 0;
60  }
61  //f(x) = y, 求 f(z_i)
```

```cpp
62  #include <bits/stdc++.h>
63  using namespace std;
64  #define ll long long
65  int n, m;
66  vector<int> x, y, z;
67  namespace Poly{
68      const int P = 998244353;
69      vector<int> ans;// for Evaluate()
70      vector<vector<int>> p;// for Evaluate() & Interpolate()
71      inline int Pow(ll x, int y=P-2){ // x^y
72          int ans=1;
73          for(; y; y>>=1, x=x*x%P) if(y&1) ans=ans*x%P;
74          return ans;
75      }
76      inline int Ge(int x){ int n=1; while(n<=x) n<<=1; return n;}
77      inline int Mod(int x){ return x<P?x:x-P;}
78      inline void NTT(vector<int> &f, int g, int n){
79          f.resize(n);
80          for(int i=0, j=0; i<n; ++i){
81              if(i>j) swap(f[i], f[j]);
82              for(int k=n>>1; (j^=k)<k; k>>=1);
83          }
84          vector<int> w(n>>1);
85          for(int i=1; i<n; i<<=1){
86              for(int j=w[0]=1, w0=(g==1?Pow(3, (P-1)/i/2):Pow(Pow(3, (P-1)/i/2))); j<i; ++j) w[j]=(ll)w[j-1]*w0%P;
87              for(int j=0; j<n; j+=i<<1){
88                  for(int k=j; k<j+i; ++k){
89                      int t=(ll)f[k+i]*w[k-j]%P;
90                      f[k+i]=Mod(f[k]-t+P);
91                      f[k]=Mod(f[k]+t);
92                  }
93              }
94          }
95          if(g==-1) for(int i=0, I=Pow(n); i<n; ++i) f[i]=(ll)f[i]*I%P;
96      }
97      inline vector<int> Add(const vector<int> &f, const vector<int> &g){
98          vector<int> ans=f;
99          for(unsigned i=0; i<f.size(); ++i) (ans[i]+=g[i])%=P;
100         return ans;
101     }
102     inline vector<int> Mul(const vector<int> &f, const vector<int> &g){// f*g
103         vector<int> F=f, G=g;
104         int p=Ge(f.size()+g.size()-2);
105         NTT(F, 1, p), NTT(G, 1, p);
106         for(int i=0; i<p; ++i) F[i]=(ll)F[i]*G[i]%P;
107         NTT(F, -1, p);
108         return F.resize(f.size()+g.size()-1), F;
109     }
110     inline vector<int> PolyInv(const vector<int> &f, int n=-1){// 1/f
111         if(n==-1) n=f.size();
112         vector<int> ans;
113         if(n==1) return ans.push_back(Pow(f[0])), ans;
114         ans=PolyInv(f, (n+1)/2);
115         vector<int> tmp(&f[0], &f[0]+n);
116         int p=Ge(n*2-2);
117         NTT(tmp, 1, p), NTT(ans, 1, p);
118         for(int i=0; i<p; ++i) ans[i]=(2-(ll)ans[i]*tmp[i]%P+P)*ans[i]%P;
119         NTT(ans, -1, p);
120         return ans.resize(n), ans;
121     }
```

```
122    inline void PolyDiv(const vector<int> &a, const vector<int> &b, vector<int> &d,
       vector<int> &r){//a=d*b+r
123        if(b.size()>a.size()) return (void)(d.clear(), r=a);
124
125        vector<int> A=a, B=b, iB;
126        int n=a.size(), m=b.size();
127        reverse(A.begin(), A.end()), reverse(B.begin(), B.end());
128        B.resize(n-m+1), iB=PolyInv(B, n-m+1);
129        d=Mul(A, iB);
130        d.resize(n-m+1), reverse(d.begin(), d.end());
131
132        r=Mul(b, d);
133        for(int i=0; i<m-1; ++i) r[i]=(P+a[i]-r[i])%P;
134        r.resize(m-1);
135    }
136    inline vector<int> Derivative(const vector<int> &a){ // a'
137        vector<int> ans;
138        ans.resize(a.size()-1);
139        for(unsigned i=1; i<a.size(); ++i) ans[i-1]=(ll)a[i]*i%P;
140        return ans;
141    }
142    void Evaluate_Interpolate_Init(int l, int r, int t, const vector<int> &a){
143        if(l==r) return p[t].clear(), p[t].push_back(P-a[l]), p[t].push_back(1);
144        int mid=(l+r)/2, k=t<<1;
145        Evaluate_Interpolate_Init(l, mid, k, a), Evaluate_Interpolate_Init(mid+1, r, k
       |1, a);
146        p[t]=Mul(p[k], p[k|1]);
147    }
148    inline void Evaluate(int l, int r, int t, const vector<int> &f, const vector<int> &a
       ){
149        if(r-l+1<=512){
150            for(int i=l; i<=r; ++i){
151                int x=0, j=f.size(), a1=a[i], a2=(ll)a[i]*a[i]%P, a3=(ll)a[i]*a2%P, a4=(
       ll)a[i]*a3%P, a5=(ll)a[i]*a4%P, a6=(ll)a[i]*a5%P, a7=(ll)a[i]*a6%P, a8=(ll)a[i]*a7%P
       ;
152                while(j>=8)
153                x=((ll)x*a8+(ll)f[j-1]*a7+(ll)f[j-2]*a6+(ll)f[j-3]*a5+(ll)f[j-4]*a4+(ll)
       f[j-5]*a3+(ll)f[j-6]*a2+(ll)f[j-7]*a1+f[j-8])%P, j-=8;
154                while(j--) x=((ll)x*a[i]+f[j])%P;
155                ans.push_back(x);
156            }
157            return;
158        }
159        vector<int> tmp;
160        PolyDiv(f, p[t], tmp, tmp);
161        Evaluate(l, (l+r)/2, t<<1, tmp, a), Evaluate((l+r)/2+1, r, t<<1|1, tmp, a);
162    }
163    inline vector<int> Evaluate(const vector<int> &f, const vector<int> &a, int flag=-1)
       {// f(a_i)
164        if(flag==-1) p.resize(a.size()<<2), Evaluate_Interpolate_Init(0, a.size()-1, 1,
       a);
165        ans.clear(), Evaluate(0, a.size()-1, 1, f, a);
166        return ans;
167    }
168    vector<int> Interpolate(int l, int r, int t, const vector<int> &x, const vector<int>
        &f){
169        if(l==r){
170            vector<int> ans;
171            return ans.push_back(f[l]), ans;
172        }
173        int mid=(l+r)/2, k=t<<1;
```

```
174          return Add(Mul(Interpolate(l, mid, k, x, f), p[k|1]), Mul(Interpolate(mid+1, r,
        k|1, x, f), p[k]));
175        }
176        inline vector<int> Interpolate(const vector<int> &x, const vector<int> &y){// (x_i,
        y_i)
177            int n=x.size();
178            p.resize(n<<2), Evaluate_Interpolate_Init(0, n-1, 1, x);
179            vector<int> f=Evaluate(Derivative(p[1]), x, 0);
180            for(int i=0; i<n; ++i) f[i]=(ll)y[i]*Pow(f[i])%P;
181            return Interpolate(0, n-1, 1, x, f);
182        }
183 }
184 using namespace Poly;
185 int main() {
186     cin >> n; x.resize(n), y.resize(n);
187     for(int i=0; i<n; ++i) cin >> x[i], cin >> y[i];
188     cin >> m, z.resize(m);
189     for(int i=0; i<m; ++i) cin >> z[i];
190     x=Evaluate(Interpolate(x, y), z);
191     for(int i:x) cout << i << " ";
192     return 0;
193 }
```

4.1.9 BM Alogrithm

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define rep(i,a,n) for (int i=a;i<n;i++)
4  #define per(i,a,n) for (int i=n-1;i>=a;i--)
5  #define pb push_back
6  #define mp make_pair
7  #define all(x) (x).begin(),(x).end()
8  #define fi first
9  #define se second
10 #define SZ(x) ((int)(x).size())
11 typedef vector<int> VI;
12 typedef long long ll;
13 typedef pair<int,int> PII;
14 const ll mod=1000000007;
15 ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%mod;
       a=a*a%mod;}return res;}
16 // head
17 namespace linear_seq {
18     const int N=10010;
19     ll res[N],base[N],_c[N],_md[N];
20
21     vector<int> Md;
22     void mul(ll *a,ll *b,int k) {
23         rep(i,0,k+k) _c[i]=0;
24         rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
25         for (int i=k+k-1;i>=k;i--) if (_c[i])
26             rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
27         rep(i,0,k) a[i]=_c[i];
28     }
29     int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
30 //         printf("%d\n",SZ(b));
31         ll ans=0,pnt=0;
32         int k=SZ(a);
33         assert(SZ(a)==SZ(b));
34         rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
```

```
35          Md.clear();
36          rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
37          rep(i,0,k) res[i]=base[i]=0;
38          res[0]=1;
39          while ((1ll<<pnt)<=n) pnt++;
40          for (int p=pnt;p>=0;p--) {
41              mul(res,res,k);
42              if ((n>>p)&1) {
43                  for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
44                  rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
45              }
46          }
47          rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
48          if (ans<0) ans+=mod;
49          return ans;
50      }
51      VI BM(VI s) {
52          VI C(1,1),B(1,1);
53          int L=0,m=1,b=1;
54          rep(n,0,SZ(s)) {
55              ll d=0;
56              rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
57              if (d==0) ++m;
58              else if (2*L<=n) {
59                  VI T=C;
60                  ll c=mod-d*powmod(b,mod-2)%mod;
61                  while (SZ(C)<SZ(B)+m) C.pb(0);
62                  rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
63                  L=n+1-L; B=T; b=d; m=1;
64              } else {
65                  ll c=mod-d*powmod(b,mod-2)%mod;
66                  while (SZ(C)<SZ(B)+m) C.pb(0);
67                  rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
68                  ++m;
69              }
70          }
71          return C;
72      }
73      int gao(VI a,ll n) {
74          VI c=BM(a);
75          c.erase(c.begin());
76          rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
77          return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
78      }
79  };
80
81  int main() {
82      while (~scanf("%d",&n)) {
83          vector<int>v;
84          v.push_back(1);
85          v.push_back(2);
86          v.push_back(4);
87          v.push_back(7);
88          v.push_back(13);
89          v.push_back(24);
90          //VI{1,2,4,7,13,24}
91          printf("%d\n",linear_seq::gao(v,n-1));
92      }
93  }
```

4.2 Math Theory

4.2.1 Inverse

```
1   //O(logn)求n的逆元
2   const int mod = 1e6 + 3;
3   int exgcd(int a, int b, int &x, int &y) {
4       int d = a;
5       if(b != 0) {
6           d = exgcd(b, a % b, y, x);
7           y -= (a / b) * x;
8       }
9       else {
10          x = 1; y = 0;
11      }
12      return d;
13  }
14  int inverse(int a) {
15      int x, y;
16      exgcd(a, mod, x, y);
17      return (x % mod + mod) % mod;
18  }
19  int inverse(int a) { return qpow(a, mod - 2); }
20  //O(n)求1~n的逆元
21  int inv[MAXN];
22  void init() {
23      inv[0] = inv[1] = 1;
24      for(int i = 2; i < MAXN; i++) inv[i] = (long long)(mod - mod / i) * inv[mod % i] %
        mod;
25  }
```

4.2.2 Lucas

```
1   //mod很小可以预处理逆元的情况
2   void init() {
3       fac[0] = 1;
4       for(int i = 1; i < mod; i++) fac[i] = (long long)fac[i - 1] * i % mod;
5       inv[0] = inv[1] = 1;
6       for(int i = 2; i < mod; i++) inv[i] = (long long)(mod - mod / i) * inv[mod % i] %
        mod;
7       for(int i = 1; i < mod; i++) inv[i] = (long long)inv[i] * inv[i - 1] % mod;
8   }
9   int C(int a, int b) {
10      if(b > a) return 0;
11      if(a < mod) return (long long)fac[a] * inv[b] % mod * inv[a - b] % mod;
12      return (long long)C(a / mod, b / mod) * C(a % mod, b % mod) % mod;
13  }
14  //mod过大不能预处理逆元的情况
15  LL qpow(LL x, LL y) {
16      LL res = 1;
17      while(y) {
18          if(y & 1) res = res * x % mod;
19          x = x * x % mod;
20          y >>= 1;
21      }
22      return res;
23  }
24  LL C(LL a, LL b) {
25      if(b > a) return 0;
```

```
26        if(b > a - b) b = a - b;
27        LL s1 = 1, s2 = 1;
28        for(LL i = 0; i < b; i++) {
29            s1 = s1 * (a - i) % mod;
30            s2 = s2 * (i + 1) % mod;
31        }
32        return s1 * qpow(s2, mod - 2) % mod;
33    }
34  LL lucas(LL a, LL b) {
35        if(a < mod) return C(a, b);
36        return lucas(a / mod, b / mod) * C(a % mod, b % mod);
37    }
```

4.2.3   CRT && exCRT

$$x \equiv a_i \pmod{m_i}$$

```
1   namespace CRT {
2       LL m[MAXN], a[MAXN];
3       LL exgcd(LL _a, LL _b, LL &x, LL &y) {
4           if(!_b) {
5               x = 1; y = 0;
6               return _a;
7           }
8           LL d = exgcd(_b, _a % _b, y, x);
9           y -= (_a / _b) * x;
10          return d;
11      }
12      LL crt(int n) {
13          LL M = 1, tmp, res = 0, x, y;
14          for(int i = 1; i <= n; i++) M *= m[i];
15          for(int i = 1; i <= n; i++) {
16              tmp = M / m[i];
17              exgcd(tmp, m[i], x, y);
18              x = (x + m[i]) % m[i];
19              res = (a[i] * x % M * tmp % M + res) % M;
20          }
21          return res;
22      }
23  }
24  namespace EXCRT {
25      LL m[MAXN], a[MAXN];
26      LL exgcd(LL _a, LL _b, LL &x, LL &y) {
27          if(!_b) {
28              x = 1; y = 0;
29              return _a;
30          }
31          LL d = exgcd(_b, _a % _b, y, x);
32          y -= (_a / _b) * x;
33          return d;
34      }
35      LL excrt(int n) {
36          LL M = m[1], A = a[1], x, y, d, tmp;
37          for(int i = 2; i <= n; i++) {
38              d = exgcd(M, m[i], x, y);
39              if((A - a[i]) % d) return -1; //No solution
40              tmp = M / d; M *= m[i] / d;
41              y = (A - a[i]) / d % M * y % M;
```

```
42              y = (y + tmp) % tmp;
43              A = (m[i] % M * y % M + a[i]) % M;
44              A = (A + M) % M;
45          }
46          return A;
47      }
48      LL inv(LL _a, LL _b) {
49          LL x, y;
50          exgcd(_a, _b, x, y);
51          return (x % _b + _b) % _b;
52      }
53      LL excrt(int n) {
54          LL M = m[1], A = a[1], x, y, d, c, tmp;
55          for(int i = 2; i <= n; i++) {
56              d = exgcd(M, m[i], x, y);
57              c = a[i] - A;
58              if(c % d) return -1;
59              c = (c % m[i] + m[i]) % m[i];
60              M /= d; m[i] /= d;
61              c = c / d * inv(M % m[i], m[i]) % m[i];
62              tmp = M;
63              M *= m[i] * d;
64              A = (c * tmp % M * d % M + A) % M;
65          }
66          return A;
67      }
68      //当a[i] 一开始就是负数时，转成正数a[i]=(a[i]%m[i]+m[i])%n[i];
69  }
```

4.2.4 BSGS

```
1   const int MOD = 76543;
2   int hs[MOD + 5], head[MOD + 5], nxt[MOD + 5], id[MOD + 5], ecnt;
3   void insert(int x, int y) {
4       int k = x % MOD;
5       hs[ecnt] = x, id[ecnt] = y, nxt[ecnt] = head[k], head[k] = ecnt++;
6   }
7   int find(int x) {
8       int k = x % MOD;
9       for(int i = head[k]; i; i = nxt[i])
10          if(hs[i] == x)
11              return id[i];
12      return -1;
13  }
14  int BSGS(int a, int b, int c){
15      memset(head, 0, sizeof head); ecnt = 1;
16      if(b == 1) return 0;
17      int m = sqrt(c * 1.0), j;
18      LL x = 1, p = 1;
19      for(int i = 0; i < m; i++, p = p * a % c)
20          insert(p * b % c, i);
21      for(LL i = m; ;i += m){
22          if((j = find(x = x * p % c)) != -1) return i - j;
23          if(i > c) break;
24      }
25      return -1;
26  }
```

4.2.5   Miller-Rabin && PollardRho

```
1  LL ksc(LL a,LL n,LL mod){
2      LL ret=0;
3      for(;n;n>>=1){
4          if(n&1){ret+=a;if(ret>=mod)ret-=mod;}
5          a<<=1;if(a>=mod)a-=mod;
6      }
7      return ret;
8  }
9  LL ksm(LL a,LL n,LL mod){
10     LL ret = 1;
11     for(;n;n>>=1){
12         if(n&1)ret=ksc(ret,a,mod);
13         a=ksc(a,a,mod);
14     }
15     return ret;
16 }
17 int millerRabin(LL n){
18     if(n<2 || (n!=2 && !(n&1)))return 0;
19     LL d=n-1;for(;!(d&1);d>>=1);
20     for(int i=0;i<20;++i){
21         LL a=rand()%(n-1)+1;
22         LL t=d,m=ksm(a,d,n);
23         for(;t!=n-1 && m!=1 && m!=n-1;m=ksc(m,m,n),t<<=1);
24         if(m!=n-1 && !(t&1)) return 0;
25     }
26     return 1;
27 }
28 LL cnt,fact[100];
29 LL gcd(LL a,LL b){return !b?a:gcd(b,a%b);}
30 LL pollardRho(LL n, int a){
31     LL x=rand()%n,y=x,d=1,k=0,i=1;
32     while(d==1){
33         ++k;
34         x=ksc(x,x,n)+a;if(x>=n)x-=n;
35         d=gcd(x>y?x-y:y-x,n);
36         if(k==i){y=x;i<<=1;}
37     }
38     if(d==n)return pollardRho(n,a+1);
39     return d;
40 }
41 void findfac(LL n){
42     if(millerRabin(n)){fact[++cnt]=n;return;}
43     LL p=pollardRho(n,rand()%(n-1)+1);
44     findfac(p);
45     findfac(n/p);
46 }
```

4.2.6 $\varphi(n)$

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

```
9        if(x > 1) res = res / x * (x - 1);
10       return res;
11   }
```

4.2.7   Euler Sieve

```
1    int prime[MAXN], cnt, phi[MAXN], mu[MAXN];
2    bool isp[MAXN];
3
4    int min_pow[MAXN];    //最小质因子最高次幂
5    int min_sum[MAXN];    //1+p+p^2+...+p^k
6    int div_sum[MAXN];    //约数和
7
8    int min_index[MAXN]; //最小质因子的指数
9    int div_num[MAXN];    //约数个数
10   void Euler(int n) {
11       mu[1] = phi[1] = div_num[1] = div_sum[1] = 1;
12       for(int i = 2; i <= n; i++) {
13           if(!isp[i]) {
14               prime[++cnt] = min_pow[i] = i;
15               phi[i] = i - 1;
16               mu[i] = -1;
17               min_index[i] = 1; div_num[i] = 2;
18               div_sum[i] = min_sum[i] = i + 1;
19           }
20           for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
21               isp[i * prime[j]] = 1;
22               if(i % prime[j] == 0) {
23                   phi[i * prime[j]] = phi[i] * prime[j];
24                   mu[i * prime[j]] = 0;
25
26                   min_index[i * prime[j]] = min_index[i] + 1;
27                   div_num[i * prime[j]] = div_num[i] / (min_index[i] + 1) * (min_index[i *
     prime[j]] + 1);
28
29                   min_sum[i * prime[j]] = min_sum[i] + min_pow[i] * prime[j];
30                   div_sum[i * prime[j]] = div_sum[i] / min_sum[i] * min_sum[i * prime[j]];
31                   min_pow[i * prime[j]] = min_pow[i] * prime[j];
32                   break;
33               }
34               phi[i * prime[j]] = phi[i] * (prime[j] - 1);
35               mu[i * prime[j]] = -mu[i];
36
37               div_num[i * prime[j]] = div_num[i] << 1;
38               min_index[i * prime[j]] = 1;
39
40               div_sum[i * prime[j]] = div_sum[i] * (prime[j] + 1);
41               min_pow[i * prime[j]] = prime[j];
42               min_sum[i * prime[j]] = prime[j] + 1;
43           }
44       }
45   }
```

4.2.8   DuJiao Sieve

$$\sum_{i=1}^{n} \varphi(i)$$

```
1   vector<int> prime;
2   int phi[MAXN], P[MAXN];
3   bool isp[MAXN];
4   unordered_map<LL, int> mp;
5   void Euler(int n) {
6       phi[1] = 1;
7       for(int i = 2; i <= n; i++) {
8           if(!isp[i]) {
9               prime.push_back(i);
10              phi[i] = i - 1;
11          }
12          for(auto x : prime) {
13              if(i * x > n) break;
14              isp[i * x] = 1;
15              if(i % x == 0) {
16                  phi[i * x] = phi[i] * x;
17                  break;
18              }
19              phi[i * x] = phi[i] * (x - 1);
20          }
21      }
22      for(int i = 1; i <= n; i++) P[i] = (P[i - 1] + phi[i]) % mod;
23  }
24  LL cal(LL n) {
25      if(n < MAXN) return P[n];
26      if(mp.count(n)) return mp[n];
27      LL res = 0;
28      for(LL i = 2, last; i <= n; i = last + 1) {
29          last = n / (n / i);
30          res += (last - i + 1) % mod * cal(n / i) % mod;
31          res %= mod;
32      }
33      mp[n] = ((__int128)n * (n + 1) / 2 % mod + mod - res) % mod;
34      return mp[n];
35  }
```

$$\sum_{i=1}^{n} \mu(i)$$

```
1   LL cal(LL n) {
2       if(n < MAXN) return M[n];
3       if(mp.count(n)) return mp[n];
4       LL res = 0;
5       for(LL i = 2, last; i <= n; i = last + 1) {
6           last = n / (n / i);
7           res += (last - i + 1) * cal(n / i);
8       }
9       mp[n] = 1 - res;
10      return 1 - res;
11  }
```

4.2.9 Min_25 Sieve

思路为把结果分为 i 为质数的和，i 为合数的和，i=1 的和

g(n, j) 表示从 1 累加到 n 的 f(i)，其中的 i 满足要么 i 自己是质数，要么 i 的最小质因子大于第 j 个质数

要求 $f(p)(p\ is\ prime)$ 可被多项式表示，$f(p^k)$ 可快速计算

分别计算多项式 x^0, x^1, x^2 的 g(n,j) 和 h(n,j)

h 由欧拉筛计算，递归起点 g(n, 0)

公式

$$\sum_i^n f(i)$$

$$g(n, j) = \begin{cases} g(n, j-1) & p_j^2 > n \\ g(n, j-1) - f(p_j)[g(\frac{n}{p_j}, j-1) - g(p_j - 1, j-1)] & p_j^2 \leq n \end{cases}$$

其中

$$g(p_j - 1, j - 1) = \sum_i^{j-1} f(p_i)$$

可用 h 表示

S(n,j) 表示从 1 累加到 n 的 f(i)，同样的，i 满足要么 i 自己是质数，要么 i 的最小质因子大于第 j 个质数
和 g 顺序相反，最后答案为 S(n, 1)，递推式为

$$S(n, j) = g(n, |P|) - \sum_{i=1}^{j} f(p_i) + \sum_{k \geq j} \sum_{p_k^{e+1} < n} (f(p_k^e) S(\frac{n}{p_k^e}, k+1) + f(p_k^{e+1}))$$

前一部分 $g(n, |P|) - \sum_{i=1}^{j} f(p_i)$ 是相应的质数部分和。

例题：欧拉函数前缀和

$$\sum_{i=1}^{n} \varphi(i)$$

$$g_{k,n} \ and \ h_{k,n} \ Count$$

$$\sum_{i=1}^{n} i^k$$

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 1e6 + 5, mod = 1e9 + 7;
const int inv2 = (mod + 1) / 2, inv6 = (mod + 1) / 6;
int prime[MAXN], isp[MAXN], cnt;
LL g[3][MAXN << 1], h[3][MAXN << 1];
LL w[MAXN << 1];
int id1[MAXN], id2[MAXN];
inline int MOD(LL x) { return x >= mod ? x - mod : x; }
//inline int MOD(LL x) { return x % mod; }
inline int add(LL x, LL y) { return MOD(MOD(x) + MOD(y)); }
void Euler(int n) {
    for(int i = 2; i <= n; i++) {
        if(!isp[i]) {
            prime[++cnt] = i;
            h[0][cnt] = h[0][cnt - 1] + 1;
            h[1][cnt] = add(h[1][cnt - 1], i);
            h[2][cnt] = add(h[2][cnt - 1], (LL)i * i % mod);
        }
        for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
            isp[i * prime[j]] = 1;
            if(i % prime[j] == 0) {
                break;
            }
        }
    }
```

```
27        }
28    }
29    LL n;
30    int sz, m;
31    inline int id(LL x) {
32        return x <= sz ? id1[x] : id2[n / x];
33    }
34    //f(p ^ k)
35    inline int f(int p, LL pk) {
36        return pk / p * (p - 1) % mod;
37    }
38    LL S(LL x, int y) {
39        if(x <= 1 || prime[y] > x) return 0;
40        //G(x) - H(j - 1) (first part)
41        LL res = add(add(g[1][id(x)], mod - g[0][id(x)]), mod - add(h[1][y - 1], mod - h[0][
      y - 1]));
42        for(int j = y, k = 1; j <= cnt && (LL)prime[j] * prime[j] <= x; j++, k = 1) {
43            for(LL pk = prime[j]; pk * prime[j] <= x; pk *= prime[j], k++) {
44                res = add(res, S(x / pk, j + 1) * f(prime[j], pk) % mod + f(prime[j], pk *
      prime[j]));
45            }
46        }
47        return res;
48    }
49    int main() {
50        ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
51        cin >> n;
52        sz = sqrt(n);
53        Euler(sz);
54        for(LL i = 1, last, t; i <= n; i = last + 1) {
55            last = n / (n / i);
56            w[++m] = n / i, t = n / i % mod;
57            w[m] <= sz ? id1[w[m]] = m : id2[last] = m;
58            g[0][m] = MOD(t + mod - 1);
59            g[1][m] = add(t * (t + 1) % mod * inv2 % mod, mod - 1);
60            g[2][m] = add((2 * t + 1) % mod * t * (t + 1) % mod * inv6 % mod, mod - 1);
61        }
62        for(int j = 1; j <= cnt; j++) {
63            for(int i = 1; i <= m && (LL)prime[j] * prime[j] <= w[i]; i++) {
64                g[0][i] = MOD(g[0][i] + mod - (g[0][id(w[i] / prime[j])] - h[0][j - 1]));
65                g[1][i] = MOD(g[1][i] + mod - ((LL)prime[j] * MOD(g[1][id(w[i] / prime[j])]
      + mod - h[1][j - 1]) % mod));
66                g[2][i] = MOD(g[2][i] + mod - ((LL)prime[j] * prime[j] % mod * MOD(g[2][id(w
      [i] / prime[j])] + mod - h[2][j - 1]) % mod));
67            }
68        }
69        //S(n, 1) + F(1);
70        LL ans = MOD(S(n, 1) + 1);
71        cout << ans << endl;
72        return 0;
73    }
```

$$\sum_{i=1}^{n} \mu(i)$$

```
1    #include <bits/stdc++.h>
2    using namespace std;
3    typedef long long LL;
4    const int MAXN = 1e6 + 5;
5    int prime[MAXN], isp[MAXN], cnt;
6    LL g[3][MAXN << 1], h[3][MAXN << 1];
```

```
 7  LL w[MAXN << 1];
 8  int id1[MAXN], id2[MAXN];
 9  void Euler(int n) {
10      for(int i = 2; i <= n; i++) {
11          if(!isp[i]) {
12              prime[++cnt] = i;
13              h[0][cnt] = h[0][cnt - 1] + 1;
14          }
15          for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
16              isp[i * prime[j]] = 1;
17              if(i % prime[j] == 0) {
18                  break;
19              }
20          }
21      }
22  }
23  LL a, b;
24  LL n;
25  int sz, m;
26  inline int id(LL x) {
27      return x <= sz ? id1[x] : id2[n / x];
28  }
29  //f(p ^ k)
30  inline int f(int p, int k) {
31      return k == 1 ? -1 : 0;
32  }
33  LL S(LL x, int y) {
34      if(x <= 1 || prime[y] > x) return 0;
35      //g(x) - h(j - 1)
36      LL res = - g[0][id(x)] + h[0][y - 1];
37      for(int j = y, k = 1; j <= cnt && (LL)prime[j] * prime[j] <= x; j++, k = 1) {
38          for(LL pk = prime[j]; pk * prime[j] <= x; pk *= prime[j], k++) {
39              res += S(x / pk, j + 1) * f(prime[j], k) + f(prime[j], k + 1);
40          }
41      }
42      return res;
43  }
44  LL cal(LL x) {
45      n = x;
46      m = 0;
47      sz = sqrt(n);
48      for(LL i = 1, last, t; i <= n; i = last + 1) {
49          last = n / (n / i);
50          w[++m] = n / i, t = n / i;
51          w[m] <= sz ? id1[w[m]] = m : id2[last] = m;
52          g[0][m] = t - 1;
53      }
54      for(int j = 1; j <= cnt; j++) {
55          for(int i = 1; i <= m && (LL)prime[j] * prime[j] <= w[i]; i++) {
56              g[0][i] = g[0][i] - (g[0][id(w[i] / prime[j])] - h[0][j - 1]);
57          }
58      }
59      return S(x, 1) + 1;
60  }
61  int main() {
62      ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
63      cin >> a >> b;
64      Euler(sqrt(b));
65
66      //S(n, 1) + F(1);
67      cout << cal(b) - cal(a - 1) << endl;
```

```
68        return 0;
69    }
```

4.2.10   Möbius Inversion

$$\sum_{i}^{n} \sum_{j}^{m} lcm(i,j) \pmod{p}$$

```
1    int mu[MAXN], prime[MAXN], sum[MAXN], cnt;
2    bool isp[MAXN];
3    void getmu(int n) {
4        mu[1] = 1;
5        for(int i = 2; i <= n; i++) {
6            if(!isp[i]) {
7                mu[i] = -1;
8                prime[++cnt] = i;
9            }
10           for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
11               isp[i * prime[j]] = 1;
12               if(i % prime[j] == 0) {
13                   mu[i * prime[j]] = 0;
14                   break;
15               }
16               mu[i * prime[j]] = -mu[i];
17           }
18       }
19   }
20   ll n, m, ans;
21   ll query(ll x, ll y) { return (x * (x + 1) / 2 % mod) * (y * (y + 1) / 2 % mod) % mod; }
22   ll F(ll x, ll y) {
23       ll res = 0, last;
24       for(ll i = 1; i <= min(x, y); i = last + 1) {
25           last = min(x / (x / i), y / (y / i));
26           res = (res + (sum[last] - sum[i - 1]) * query(x / i, y / i) % mod) % mod;
27       }
28       return res;
29   }
30   int main() {
31       cin>>n>>m;
32       getmu(min(n, m));
33       for(ll i = 1; i <= min(n, m); i++) sum[i] = (sum[i - 1] + (i * i * mu[i]) % mod) %
         mod;
34       ll last;
35       for(ll d = 1; d <= min(n, m); d = last + 1) {
36           last = min(n / (n / d), m / (m / d));
37           ans = (ans + (last - d + 1) * (d + last) / 2 % mod * F(n / d, m / d) % mod) %
         mod;
38       }
39       ans = (ans + mod) % mod;
40       cout<<ans<<endl;
41       return 0;
42   }
```

4.2.11   Primitive Root

简易版如果模 m 有原根, 则有 $\varphi(\varphi(m))$ 个原根当 d 遍历 $\varphi(m)$ 的简化剩余系时, g^d 遍历完 m 的全部原根

```
1   const int MAXN = ;
2   LL qpow(LL a, LL b, LL mod) {
3       LL res = 1;
4       for (;b; b >>= 1) {
5           if(b&1) res = res * a % mod;
6           a = a * a % mod;
7       }
8       return res;
9   }
10  LL phi(LL x) {
11      LL res = x;
12      for(int i = 2; i <= x / i; i++) {
13          if(x % i == 0) {
14              res = res / i * (i - 1);
15              while(x % i == 0) x /= i;
16          }
17      }
18      if(x > 1) res = res / x * (x - 1);
19      return res;
20  }
21  int has_primitive_root(int p) {
22      if (p == 4) return 1;
23      if (p % 2 == 0) p /= 2;
24      if (p % 2 == 0) return false;
25      for (int i = 2; i <= p/i; i++)
26      if (p % i == 0) {
27          while(p % i == 0) p /= i;
28          return p == 1 ? i : 0;
29      }
30      return p;
31  }
32  //int indg[MAXN];
33  int get_g_init(LL p) {
34      //p : 2 or 4 or (p**n) or (2 * p**n); else return −1;
35      if(p == 2) {puts("1");return 1;}
36      if(p == 4) {puts("3"); return 3;}
37      if(!has_primitive_root(p)) {puts("-1"); return -1;}
38      vector<int> p_fact;
39      LL p_phi = phi(p), tp = p_phi ;
40      for(int i = 2, in = sqrt(p) + 0.5; i <= in; i++)
41      if(tp % i == 0) {
42          p_fact.push_back(p_phi / i);
43          while(tp % i == 0) tp /= i;
44      }
45      if(tp != 1) p_fact.push_back(p_phi / tp);
46      int g = 1;
47      for (bool fg = 0; !fg ;) {
48          fg = 1; g++;
49          if(qpow(g, p_phi, p) != 1) {fg = 0; continue;}
50          for(auto it : p_fact)
51              if(qpow(g, it, p) == 1) {fg = 0; break;}
52      }
53      //for(int i = 0, tg = 1; i < p_phi; i++, tg = tg * g % p) indg[tg] = i;
54      /*vector<int> fac;
55      for(int d = 1, tg = g; d < p_phi; d++, tg = tg * g % p)
56          if(__gcd((LL)d, p_phi) == 1) fac.push_back(tg);
57      sort(fac.begin(), fac.end());
58      int ed = fac.back();
59      for(auto it : fac)
60          printf("%d%c", it, it != ed ? ' ':'\n');*/
61      return g;
```

```
62  }
```

5   Geometry

5.1   Commonly Definition and Functions

5.1.1   Const and Functions

```
1   namespace CG{
2       #define Point Vector
3       const double pi=acos(-1.0);
4       const double inf=1e100;
5       const double eps=1e-9;
6       template <typename T> inline T Abs(T x){return x>0?x:-x;}
7       template <typename T> inline bool operator == (T x,T y){return Abs(x-y)<eps;}
8       int sgn(double x){
9           if (Abs(x)<eps) return 0;
10          if (x>0) return 1;
11          else return -1;
12      }
13  }
```

5.1.2   Point Definition

```
1   namespace CG{
2       struct Point{
3           double x,y;
4           Point(double x=0,double y=0):x(x),y(y){}
5       };
6       Vector operator + (const Vector a,const Vector b){return Vector(a.x+b.x,a.y+b.y);}
7       Vector operator - (const Vector a,const Vector b){return Vector(a.x-b.x,a.y-b.y);}
8       Vector operator * (const Vector a,const double k){return Vector(a.x*k,a.y*k);}
9       Vector operator / (const Vector a,const double k){return Vector(a.x/k,a.y/k);}
10      bool operator < (const Vector a,const Vector b) {return a.x==b.x?a.y<b.y:a.x<b.x;}
11      bool operator == (const Vector a,const Vector b) {return a.x==b.x && a.y==b.y;}
12      double Dot(const Vector a,const Vector b){return a.x*b.x+a.y*b.y;}
13      double Cross(const Vector a,const Vector b){return a.x*b.y-a.y*b.x;}
14      double mult_Cross(const Vector a,const Vector b,const Vector c){return (a.x-c.x)*(b.
        y-c.y)-(b.x-c.x)*(a.y-c.y);}
15      double mult_Dot(const Vector a,const Vector b,const Vector c){return (a.x-c.x)*(b.x-
        c.x)+(a.y-c.y)*(b.y-c.y);}
16      double Norm(const Vector a){return sqrt(Dot(a,a));}
17      double Angle(const Vector a,const Vector b){return acos(Dot(a,b)/Norm(a)/Norm(b));}
18      Vector Rotate(const Vector a,const double theta){return Vector(a.x*cos(theta)-a.y*
        sin(theta),a.x*sin(theta)+a.y*cos(theta));}
19      bool ToLeftTest(const Vector a,const Vector b){return Cross(a,b)<0;}
20      double DisPP(const Vector a,const Vector b){return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y
        )*(a.y-b.y));}
21  }
```

5.1.3   Line Definition

```
1   namespace CG{
2       struct Line{
3           Point p0,v,p1;
4           double t,theta;
5           Line(Point _p0=0,Point _v=0,double _t=1):p0(_p0),v(_v),t(_t){p1=p0+v*t; theta=
        atan2(v.y,v.x);}
```

```
6           // Line(Point _p0=0,Point _v=0,double _t=1):p0(_p0),p1(_v){v=(p1-p0)/t; theta=
      atan2(v.y,v.x);}
7        };
8        bool operator < (const Line n,const Line m) {return n.theta<m.theta;}
9        Point GetIntersection(const Line n,const Line m){return n.p0+n.v*Cross(m.v,(n.p0-m.
      p0))/Cross(n.v,m.v);}
10       bool OnLine(const Vector a,const Line l){return Cross(l.p0-a,l.p1-a)==0;}
11       bool OnSegment(const Point a,const Line l){return sgn(Cross(l.p0-a,l.p1-a))==0 &&
      sgn(Dot(l.p0-a,l.p1-a))<0;}
12       double DisPL(const Point a,const Line l){return Abs(Cross(l.p1-l.p0,a-l.p0)/Norm(l.
      p1-l.p0));}
13       double DisPS(const Point a,const Line l){
14           if (l.p0==l.p1) return Norm(a-l.p0);
15           Vector v1=l.p1-l.p0,v2=a-l.p0,v3=a-l.p1;
16           if (sgn(Dot(v1,v2))<0) return Norm(v2);
17           if (sgn(Dot(v1,v3))>0) return Norm(v3);
18           return DisPL(a,l);
19       }
20       Point GetProjection(const Point a,const Line l){
21           Vector v=l.p1-l.p0;
22           return l.p0+v*(Dot(v,a-l.p0)/Dot(v,v));
23       }
24       bool SegmentIntersection(const Line n,const Line m,bool p){
25           double c1=Cross(n.p1-n.p0,m.p0-n.p0);
26           double c2=Cross(n.p1-n.p0,m.p1-n.p0);
27           double c3=Cross(m.p1-m.p0,n.p0-m.p0);
28           double c4=Cross(m.p1-m.p0,n.p1-m.p0);
29           if (p){
30               if (!sgn(c1) || !sgn(c2) || !sgn(c3) || !sgn(c4)){
31                   return OnSegment(n.p0,m) || OnSegment(n.p1,m) || OnSegment(m.p0,n) ||
      OnSegment(m.p0,m);

32
33               }
34           }
35           return (sgn(c1)*sgn(c2)<0 && sgn(c3)*sgn(c4)<0);
36       }
37   }
```

5.1.4   Get Area

```
1   namespace CG{
2       double GetArea(Point *p,int n){
3           double area=Cross(p[n],p[1]);
4           for (int i=2;i<=n;i++) area+=0.5*Cross(p[i-1],p[i]);
5           return Abs(area);
6       }
7   }
```

5.1.5   Get Circumference

```
1   namespace CG{
2       double GetCircumference(Point *p,int n){
3           double Circumference=DisPP(p[n],p[1]);
4           for (int i=2;i<=n;i++) Circumference+=DisPP(p[i-1],p[i]);
5           return Circumference;
6       }
7   }
```

5.1.6 Anticlockwise Sort

```
1  namespace CG{
2      \\p为一个凸包，只是不知其点集是否为逆时针
3      void clockwise_sort(Point *p,int n){
4          for(int i=0;i<n-2;i++){
5              double tmp = mult_Cross(p[i+1],p[i+2],p[i]);
6              if(tmp>0) return;
7              else if(tmp<0){
8                  reverse(p,p+n);
9                  return;
10             }
11         }
12     }
13 }
```

5.2 Convex Hull

5.2.1 Get Convex Hull

```
1  namespace CG{
2      Point p[MAXN],s[MAXN]; // both based from 0
3      int ConvexHull(Point *p,int n,Point *s){
4          sort(p,p+n,cmp); //x从小到大,y从小到大;
5          int m=0;
6          for (int i=0;i<n;i++){
7              for (;m>=2 && Cross(s[m-1]-s[m-2],p[i]-s[m-1])<=0;m--);
8              s[m++]=p[i];
9          }
10         int k=m;
11         for (int i=n-2;i;i--){
12             for (;m>=k+1 && Cross(s[m-1]-s[m-2],p[i]-s[m-1])<=0;m--);
13             s[m++]=p[i];
14         }
15         return m-1;
16     }
17 }
```

5.2.2 Point in Convex Hull

```
1  namespace CG{
2      bool PointInConvexHull(Point A){
3          int l=1,r=tot-2,mid;
4          while(l<=r){
5              mid=(l+r)>>1;
6              double a1=Cross(p[mid]-p[0],A-p[0]);
7              double a2=Cross(p[mid+1]-p[0],A-p[0]);
8              if(a1>=0 && a2<=0){
9                  if(Cross(p[mid+1]-p[mid],A-p[mid])>=0) return true;
10                 return false;
11             }
12             else if(a1<0) r=mid-1;
13             else l=mid+1;
14         }
15         return false;
16     }
17 }
```

5.3 Minkowski Sum

```
1  namespace CG{
2      void Minkowski(Point *C1,int n,Point *C2,int m){
3          for(int i=1;i<=n;i++) s1[i]=C1[i]-C1[i-1];
4          for(int i=1;i<=m;i++) s2[i]=C2[i]-C2[i-1];
5          A[tot=1]=C1[1]+C2[1];
6          int p1=1,p2=1;
7          while (p1<=n && p2<=m) ++tot,A[tot]=A[tot-1]+(s1[p1]*s2[p2]>=0?s1[p1++]:s2[p2
   ++]);
8          while (p1<=n) ++tot,A[tot]=A[tot-1]+s1[p1++];
9          while (p2<=m) ++tot,A[tot]=A[tot-1]+s2[p2++];
10         tot=ConvexHull(A,tot);
11     }
12 }
```

5.4 Rotating Calipers

5.4.1 The Diameter of Convex Hull

```
1  namespace CG{
2      double RotatingCalipers(Point *p,int n){
3          double dis=0;
4          for(int i=0,j=2;i<n;++i){
5              while (abs(Cross(p[i+1]-p[i],p[j]-p[i]))<abs(Cross(p[i+1]-p[i],p[j+1]-p[i]))
   ) j=(j+1)%n;
6              dis=max(dis,max(DisPP(p[j],p[i]),DisPP(p[j],p[i+1])));
7          }
8          return dis;
9      }
10 }
```

5.4.2 The Min Distance Bewteen two Convex Hull

```
1  namespace CG{
2      ///点c到线段ab的最短距离
3      double GetDist(Point a,Point b,Point c){
4          if(dis(a,b)<esp) return dis(b,c); ///a,b是同一个点
5          if(mult_Dot(b,c,a)<-esp) return dis(a,c); ///投影
6          if(mult_Dot(a,c,b)<-esp) return dis(b,c);
7          return fabs(mult_Cross(b,c,a)/dis(a,b));
8
9      }
10     ///求一条线段ab的两端点到另外一条线段bc的距离，反过来一样，共4种情况
11     double MinDist(Point a,Point b,Point c,Point d){
12         return min(min(GetDist(a,b,c),GetDist(a,b,d)),min(GetDist(c,d,a),GetDist(c,d,b))
   );
13     }
14     double RotatingCalipers(Point *p,int n,Point *q,int m){
15         int yminP = 0,ymaxQ=0;
16         for(int i=1;i<n;i++){  ///找到点集p组成的凸包的左下角
17             if(p[i].y<p[yminP].y||(p[i].y==p[yminP].y)&&(p[i].x<p[yminP].x)) yminP = i;
18         }
19         for(int i=1;i<m;i++){  ///找到点集q组成的凸包的右上角
20             if(q[i].y>q[ymaxQ].y||(q[i].y==q[ymaxQ].y)&&(q[i].x>q[ymaxQ].x)) ymaxQ = i;
21         }
22         double ans = DisPP(p[yminP],q[ymaxQ]); ///距离(yminP,ymaxQ)维护为当前最小值。
```

```
23          for(int i=0;i<n;i++){
24              double tmp;
25              while(tmp=(mult_Cross(q[ymaxQ+1],p[yminP],p[yminP+1])-mult_Cross(q[ymaxQ],p[
     yminP],p[yminP+1]))>esp)
26                  ymaxQ = (ymaxQ+1)%m;
27              if(tmp<-esp) ans = min(ans,GetDist(p[yminP],p[yminP+1],q[ymaxQ]));
28              else ans=min(ans,MinDist(p[yminP],p[yminP+1],q[ymaxQ],q[ymaxQ+1]));
29              yminP = (yminP+1)%n;
30          }
31          return ans;
32      }
33  }
```

5.5   Half Plane Intersection

```
1   namespace CG{
2       void HalfPlaneIntersection(Line l[],int n){
3           deque <Point> p;
4           sort(l+1,l+1+n);
5           deque <Line> q;
6           q.push_back(l[1]);
7           for (int i=2;i<=n;i++){
8               for (;!p.empty() && !ToLeftTest(p.back()-l[i].p0,l[i].v);q.pop_back(),p.
     pop_back());
9               for (;!p.empty() && !ToLeftTest(p.front()-l[i].p0,l[i].v);q.pop_front(),p.
     pop_front());
10              if (sgn(Cross(l[i].v,q.back().v))==0)
11                  if (ToLeftTest(l[i].p0-q.back().p0),q.back().v){
12                      q.pop_back();
13                      if (!p.empty()) p.pop_back();
14                  }
15              if (!q.empty()) p.push_back(GetIntersection(q.back(),l[i]));
16              q.push_back(l[i]);
17          }
18          for (;!p.empty() && !ToLeftTest(p.back()-q.front().p0,q.front().v);q.pop_back(),
     p.pop_back());
19          p.push_back(GetIntersection(q.back(),q.front()));
20          if (p.size() < 3) printf("0\n");
21          else{
22              cerr << "!" << endl;
23              double area = 0.5 * Cross(p.back(), p.front());
24              Point last = p.front();
25              for (p.pop_front(); !p.empty(); last = p.front(), p.pop_front())
26                  area += 0.5 * Cross(last, p.front());
27              printf("%.8lf\n", fabs(area));
28          }
29      }
30  }
```

5.6   Min Circle Cover

```
1   namespace CG{
2       Point GetCircleCenter(const Point a,const Point b,const Point c){
3           Point p=(a+b)/2.0,q=(a+c)/2.0;
4           Vector v=Rotate(b-a,pi/2.0),w=Rotate(c-a,pi/2.0);
5           if (sgn(Norm(Cross(v,w)))==0){
6               if (sgn(Norm(a-b)+Norm(b-c)-Norm(a-c))==0) return (a+c)/2;
7               if (sgn(Norm(b-a)+Norm(a-c)-Norm(b-c))==0) return (b+c)/2;
```

```
 8              if (sgn(Norm(a-c)+Norm(c-b)-Norm(a-b))==0) return (a+c)/2;
 9          }
10          return GetIntersection(Line(p,v),Line(q,w));
11      }
12      void MinCircleCover(Point p[],int n){
13          random_shuffle(p+1,p+1+n);
14          Point c=p[1];
15          double r=0;
16          for (int i=2;i<=n;i++)
17              if (sgn(Norm(c-p[i])-r)>0){
18                  c=p[i],r=0;
19                  for (int j=1;j<i;j++)
20                      if (sgn(Norm(c-p[j])-r)>0){
21                          c=(p[i]+p[j])/2.0;
22                          r=Norm(c-p[i]);
23                          for (int k=1;k<j;k++)
24                              if (sgn(Norm(c-p[k])-r)>0){
25                                  c=GetCircleCenter(p[i],p[j],p[k]);
26                                  r=Norm(c-p[i]);
27                              }
28                      }
29              }
30          printf("%.10f\n%.10f %.10f",r,c.x,c.y);
31      }
32  }
```

5.7 Circle Union Area

```
 1  //k次覆盖
 2  //圆并去重后 s[0]
 3  typedef pair<double, int> P;
 4  const double pi = acos(-1.0);
 5  const int MAXN = 10003;
 6  P arc[MAXN << 1];
 7  int acnt, cnt;
 8  double s[1003];
 9  bool del[1003];
10  void add(double st, double en) {
11      if(st < -pi) {
12          add(st + 2 * pi, pi);
13          add(-pi, en);
14          return;
15      }
16      if(en > pi) {
17          add(st, pi);
18          add(-pi, en - 2 * pi);
19          return;
20      }
21      arc[++acnt] = P(st, 1);
22      arc[++acnt] = P(en, -1);
23  }
24  double F(double x) {
25      return (x - sin(x)) / 2;
26  }
27  struct Node {
28      int x, y, r;
29      Node(int _x = 0, int _y = 0, int _r = 0):x(_x), y(_y), r(_r) {}
30      bool operator == (const Node& t) {
31          return x == t.x && y == t.y && r == t.r;
```

```
32          }
33      inline void read() {
34          scanf("%d%d%d", &x, &y, &r);
35      }
36  }a[1003];
37  int main() {
38      int n;
39      scanf("%d", &n);
40      for(int i = 1; i <= n; i++) a[i].read();
41      /*
42      //去重
43      int nn = 0;
44      for(int i = 1; i <= n; i++) {
45          bool same = 0;
46          for(int j = 1; j < i; j++) {
47              if(a[i] == a[j]) {
48                  same = 1; break;
49              }
50          }
51          if(!same) a[++nn] = a[i];
52      }
53      n = nn;
54      //去包含
55      for(int i = 1; i <= n; i++) {
56          for(int j = 1; j <= n; j++) if(i != j) {
57              if(hypot(a[i].x - a[j].x, a[i].y - a[j].y) < (double)(a[i].r - a[j].r)) del[
    j] = 1;
58          }
59      }
60      nn = 0;
61      for(int i = 1; i <= n; i++) if(!del[i]) {
62          a[++nn] = a[i];
63      }
64      n = nn;
65      */
66      for(int i = 1; i <= n; i++) {
67          acnt = 0;
68          for(int j = 1; j <= n; j++) if(i != j) {
69              int dis = (a[i].x - a[j].x) * (a[i].x - a[j].x) + (a[i].y - a[j].y) * (a[i].
    y - a[j].y);
70              if(a[j].r > a[i].r && dis <= (a[j].r - a[i].r) * (a[j].r - a[i].r)) add(-pi,
     pi);
71              else if(dis > (a[i].r - a[j].r) * (a[i].r - a[j].r) && dis < (a[i].r + a[j].
    r) * (a[i].r + a[j].r)){
72                  double c = sqrt(dis);
73                  double angle = acos((a[i].r * a[i].r + c * c - a[j].r * a[j].r) / (2 * a
    [i].r * c));
74                  double k = atan2(a[j].y - a[i].y, a[j].x - a[i].x);
75                  add(k - angle, k + angle);
76              }
77          }
78          arc[++acnt] = P(pi, -1);
79          sort(arc + 1, arc + acnt + 1);
80          cnt = 0;
81          double last = -pi;
82          for(int j = 1; j <= acnt; j++) {
83              s[cnt] += F(arc[j].first - last) * a[i].r * a[i].r; //扇形 - 三角形
84              double xa = a[i].x + a[i].r * cos(last);
85              double ya = a[i].y + a[i].r * sin(last);
86              last = arc[j].first;
87              double xb = a[i].x + a[i].r * cos(last);
```

```
88              double yb = a[i].y + a[i].r * sin(last);
89              s[cnt] += (xa * yb - xb * ya) / 2; //到圆心的三角形面积
90              cnt += arc[j].second;
91          }
92      }
93      //printf("%.3f\n", s[0]);
94      for (int i = 0; i < n; i++) {
95          printf("[%d] = %.3f\n", i + 1, s[i] - s[i + 1]);
96      }
97      return 0;
98  }
```

5.8 Simpson Integrate

```
1   double Simpson(double l,double r){
2       return (r-l)*(F(l)+4*F((l+r)/2)+F(r))/6;
3   }
4   double Integrate(double l,double r,double S){
5       double mid=(l+r)/2;
6       double A=Simpson(l,mid);
7       double B=Simpson(mid,r);
8       if(A+B-S<eps)return S;
9       return Integrate(l,mid,A)+Integrate(mid,r,B);
10  }
```

5.9 Closest Point

```
1   vector <Point> P;
2   DB CP(int l, int r) {
3       if (l == r) return DB_INF;
4       if (l + 1 == r) return DisPP(P[l], P[r]);
5       int mid = (l + r) >> 1;
6       DB d = min(CP(l, mid), CP(mid + 1, r));
7       vector <Point> tmp;
8       for (int i = l; i <= r; i++)
9           if (fabs(P[mid].x - P[i].x) < d) tmp.push_back(P[i]);
10      sort(tmp.begin(), tmp.end(), cmpy);
11      for (int i = 0; i < tmp.size(); i++)
12          for (int j = i + 1; j < tmp.size() && tmp[j].y - tmp[i].y < d; j++)
13              d = min(d, DisPP(tmp[i], tmp[j]));
14      return d;
15  }
```

5.10 K-D Tree

```
1   #include <iostream>
2   #include <algorithm>
3   #include <stack>
4   #include <math.h>
5   using namespace std;
6   /*function of this program: build a 2d tree using the input trainingdata
7    the input is exm_set which contains a list of tuples (x,y)
8    the output is a 2d tree pointer*/
9
10
11  struct data
```

```
12  {
13      double x = 0;
14      double y = 0;
15  };
16
17  struct Tnode
18  {
19      struct data dom_elt;
20      int split;
21      struct Tnode * left;
22      struct Tnode * right;
23  };
24
25  bool cmp1(data a, data b){
26      return a.x < b.x;
27  }
28
29  bool cmp2(data a, data b){
30      return a.y < b.y;
31  }
32
33  bool equal(data a, data b){
34      if (a.x == b.x && a.y == b.y)
35      {
36          return true;
37      }
38      else{
39          return false;
40      }
41  }
42
43  void ChooseSplit(data exm_set[], int size, int &split, data &SplitChoice{
44      /*compute the variance on every dimension. Set split as thedismension that have the
         biggest
45       variance. Then choose the instance which is the median on thissplit dimension.*/
46      /*compute variance on the x,y dimension. DX=EX^2-(EX)^2*/
47      double tmp1,tmp2;
48      tmp1 = tmp2 = 0;
49      for (int i = 0; i < size; ++i)
50      {
51          tmp1 += 1.0 / (double)size * exm_set[i].x * exm_set[i].x;
52          tmp2 += 1.0 / (double)size * exm_set[i].x;
53      }
54      double v1 = tmp1 - tmp2 * tmp2;  //compute variance on the xdimension
55
56      tmp1 = tmp2 = 0;
57      for (int i = 0; i < size; ++i)
58      {
59          tmp1 += 1.0 / (double)size * exm_set[i].y * exm_set[i].y;
60          tmp2 += 1.0 / (double)size * exm_set[i].y;
61      }
62      double v2 = tmp1 - tmp2 * tmp2;  //compute variance on the ydimension
63
64      split = v1 > v2 ? 0:1; //set the split dimension
65
66      if (split == 0)
67      {
68          sort(exm_set,exm_set + size, cmp1);
69      }
70      else{
71          sort(exm_set,exm_set + size, cmp2);
```

```
72        }
73
74        //set the split point value
75        SplitChoice.x = exm_set[size / 2].x;
76        SplitChoice.y = exm_set[size / 2].y;
77
78   }
79
80   Tnode* build_kdtree(data exm_set[], int size, Tnode* T){
81        //call function ChooseSplit to choose the split dimension and splitpoint
82        if (size == 0){
83            return NULL;
84        }
85        else{
86            int split;
87            data dom_elt;
88            ChooseSplit(exm_set, size, split, dom_elt);
89            data exm_set_right [100];
90            data exm_set_left [100];
91            int sizeleft ,sizeright;
92            sizeleft = sizeright = 0;
93
94            if (split == 0)
95            {
96                for (int i = 0; i < size; ++i)
97                {
98
99                    if (!equal(exm_set[i],dom_elt) && exm_set[i].x <=dom_elt.x)
100                    {
101                        exm_set_left[sizeleft].x = exm_set[i].x;
102                        exm_set_left[sizeleft].y = exm_set[i].y;
103                        sizeleft++;
104                    }
105                    else if (!equal(exm_set[i],dom_elt) && exm_set[i].x >dom_elt.x)
106                    {
107                        exm_set_right[sizeright].x = exm_set[i].x;
108                        exm_set_right[sizeright].y = exm_set[i].y;
109                        sizeright++;
110                    }
111                }
112            }
113            else{
114                for (int i = 0; i < size; ++i)
115                {
116
117                    if (!equal(exm_set[i],dom_elt) && exm_set[i].y <=dom_elt.y)
118                    {
119                        exm_set_left[sizeleft].x = exm_set[i].x;
120                        exm_set_left[sizeleft].y = exm_set[i].y;
121                        sizeleft++;
122                    }
123                    else if (!equal(exm_set[i],dom_elt) && exm_set[i].y >dom_elt.y)
124                    {
125                        exm_set_right[sizeright].x = exm_set[i].x;
126                        exm_set_right[sizeright].y = exm_set[i].y;
127                        sizeright++;
128                    }
129                }
130            }
131            T = new Tnode;
132            T->dom_elt.x = dom_elt.x;
```

```
133        T->dom_elt.y = dom_elt.y;
134        T->split = split;
135        T->left = build_kdtree(exm_set_left, sizeleft, T->left);
136        T->right = build_kdtree(exm_set_right, sizeright, T->right);
137        return T;
138
139    }
140  }
141
142
143  double Distance(data a, data b){
144      double tmp = (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
145      return sqrt(tmp);
146  }
147
148
149  void searchNearest(Tnode * Kd, data target, data &nearestpoint, double &distance){
150
151      //1. 如果Kd是空的，则设 dist 为无穷大返回
152
153      //2. 向下搜索直到叶子结点
154
155      stack<Tnode*> search_path;
156      Tnode* pSearch = Kd;
157      data nearest;
158      double dist;
159
160      while(pSearch != NULL)
161      {
162          //pSearch加入到 search_path 中 ;
163          search_path.push(pSearch);
164
165          if (pSearch->split == 0)
166          {
167              if(target.x <= pSearch->dom_elt.x) /* 如果小于就进入左子树 */
168              {
169                  pSearch = pSearch->left;
170              }
171              else
172              {
173                  pSearch = pSearch->right;
174              }
175          }
176          else{
177              if(target.y <= pSearch->dom_elt.y) /* 如果小于就进入左子树 */
178              {
179                  pSearch = pSearch->left;
180              }
181              else
182              {
183                  pSearch = pSearch->right;
184              }
185          }
186      }
187      //取出 search_path 最后一个赋给 nearest
188      nearest.x = search_path.top()->dom_elt.x;
189      nearest.y = search_path.top()->dom_elt.y;
190      search_path.pop();
191
192
193      dist = Distance(nearest, target);
```

```
194        //3. 回溯搜索路径
195
196    Tnode* pBack;
197
198    while(search_path.size() != 0)
199    {
200            //取出 search_path 最后一个结点赋给 pBack
201        pBack = search_path.top();
202        search_path.pop();
203
204        if(pBack->left == NULL && pBack->right == NULL) /* 如果 pBack 为子结点 */
205
206        {
207
208            if( Distance(nearest, target) > Distance(pBack->dom_elt,target) )
209            {
210                nearest = pBack->dom_elt;
211                dist = Distance(pBack->dom_elt, target);
212            }
213
214        }
215
216        else
217
218        {
219
220            int s = pBack->split;
221            if (s == 0)
222            {
223                if( fabs(pBack->dom_elt.x - target.x) < dist) /* 如果 target 为中心的圆
    （球或超球），半径为 dist 的圆与分割超平面交，那么就要跳到另一边的子空间去搜索 */
224                {
225                    if( Distance(nearest, target) > Distanc(pBack->dom_elt, target) )
226                    {
227                        nearest = pBack->dom_elt;
228                        dist = Distance(pBack->dom_elt, target);
229                    }
230                    if(target.x <= pBack->dom_elt.x) /* 如果 target 位 pBack 的左子空间，那
    么就要跳到右子空间去搜索 */
231                        pSearch = pBack->right;
232                    else
233                        pSearch = pBack->left; /* 如果 target 位于 pBack 的子空间，那么就要
    跳到左子空间去搜索 */
234                    if(pSearch != NULL)
235                        //pSearch 加入到 search_path 中
236                        search_path.push(pSearch);
237                }
238            }
239            else {
240                if( fabs(pBack->dom_elt.y - target.y) < dist) /* 如果 target 为中心的圆
    （球或超球），半径为 dist 的圆与分割超平面交，那么就要跳到另一边的子空间去搜索 */
241                {
242                    if( Distance(nearest, target) > Distanc(pBack->dom_elt, target) )
243                    {
244                        nearest = pBack->dom_elt;
245                        dist = Distance(pBack->dom_elt, target);
246                    }
247                    if(target.y <= pBack->dom_elt.y) /* 如果 target 位 pBack 的左子空间，那
    么就要跳到右子空间去搜索 */
248                        pSearch = pBack->right;
249                    else
```

```
250                              pSearch = pBack->left; /* 如果target位于pBack的子空间，那么就要
        跳到左子空间去搜索 */
251                      if(pSearch != NULL)
252                          // pSearch加入到search_path中
253                          search_path.push(pSearch);
254                  }
255              }

256
257          }
258      }

259
260      nearestpoint.x = nearest.x;
261      nearestpoint.y = nearest.y;
262      distance = dist;

263
264  }

265
266  int main(){
267      data exm_set[100]; //assume the max training set size is 100
268      double x,y;
269      int id = 0;
270      cout<<"Please input the training data in the form x y. One instanceper line. Enter
        -1 -1 to stop."<<endl;
271      while (cin>>x>>y){
272          if (x == -1)
273          {
274              break;
275          }
276          else{
277              exm_set[id].x = x;
278              exm_set[id].y = y;
279              id++;
280          }
281      }
282      struct Tnode * root = NULL;
283      root = build_kdtree(exm_set, id, root);

284
285      data nearestpoint;
286      double distance;
287      data target;
288      cout <<"Enter search point"<<endl;
289      while (cin>>target.x>>target.y)
290      {
291          searchNearest(root, target, nearestpoint, distance);
292          cout<<"The nearest distance is "<<distance<<",and the nearestpoint is "<<
        nearestpoint.x<<","<<nearestpoint.y<<endl;
293          cout <<"Enter search point"<<endl;

294
295      }
296  }
```

6 Others

6.1 Offline Algorithm

6.1.1 CDQ Divide and Conquer

```cpp
struct Node {
    int x, y, z, ans;
    Node() {}
    Node(int _x, int _y, int _z):x(_x), y(_y), z(_z) {}
    bool operator < (const Node &b) const {
        if(y == b.y) {
            if(z == b.z) return x < b.x;
            return z < b.z;
        }
        return y < b.y;
    }
}A[MAXN], B[MAXN], C[MAXN];
int bit[MAXN];
void add(int k, int v) {
    for(; k <= m; k += k & -k) bit[k] = max(bit[k], v);
}
void clear(int k) {
    for(; k <= m; k += k & -k) bit[k] = 0;
}
int sum(int k) {
    int res = 0;
    for(; k; k -= k & -k) res = max(res, bit[k]);
    return res;
}
void solve(int l, int r) {
    if(l == r) {
        B[l] = A[l];
        return;
    }
    int mid = (l + r) >> 1;
    solve(l, mid);
    for(int i = mid + 1; i <= r; i++) B[i] = A[i];
    //sort(B + l, B + mid + 1);
    sort(B + mid + 1, B + r + 1);
    int L = l;
    for(int R = mid + 1; R <= r; R++) {
        while(L <= mid && B[L].y < B[R].y) add(B[L].z, B[L].ans), L++;
        A[B[R].x].ans = max(A[B[R].x].ans, sum(B[R].z - 1) + 1);
        B[R].ans = A[B[R].x].ans;
    }
    for(int i = l; i <= L; i++) clear(B[i].z);
    solve(mid + 1, r);
    L = l;
    int p = l, q = mid + 1;
    while(p <= mid || q <= r) {
        if(q > r || (p <= mid && B[p].y <= B[q].y)) C[L++] = B[p++];
        else C[L++] = B[q++];
    }
    for(int i = l; i <= r; i++) B[i] = C[i];
}
```

6.1.2 Mo' s Algorithm

```
1  struct Node{
2      int l, r, t, id;
3      bool operator < (const Node& a) const {
4          if(l /sz == a.l / sz) {
5              if(r == a.r) return t < a.t;
6              return r < a.r;
7          }
8          return l / sz < a.l / sz;
9      }
10 }q[MAXN];
11 void solve() {
12     while (t < q[i].t) addTime(t++, 1);
13     while (t > q[i].t) addTime(--t, -1);
14     while(L < q[i].l) add(L++, -1);
15     while(L > q[i].l) add(--L, 1);
16     while(R < q[i].r) add(++R, 1);
17     while(R > q[i].r) add(R--, -1);
18 }
```

6.1.3 Mo's Algorithm On Tree

```
1  struct Edge {
2      int to, nxt;
3  }e[MAXN << 1];
4  int head[MAXN], ecnt;
5  int stack[MAXN], top, belong[MAXN], cnt, sz;
6  struct Node {
7      int l, r, id, ti;
8      bool operator < (const Node &x) const {
9          return belong[l] < belong[x.l] || (belong[l] == belong[x.l] && belong[r] <
       belong[x.r]) || (belong[l] == belong[x.l] && belong[r] == belong[x.r] && ti < x.ti);
10     }
11 }q[MAXN];
12 struct Node2 {
13     int l, r, ti;
14 }qq[MAXN];
15 int n, m, Q, Q0, Q1;
16 int V[MAXN], W[MAXN], C[MAXN];
17 int fa[MAXN][S + 3], dep[MAXN];
18 long long ans[MAXN], tans;
19 int vis[MAXN], cur[MAXN];
20 long long sum[MAXN];
21 int l, r, tm;
22 inline int read() {
23     int x = 0; char ch = getchar(); bool fg = 0;
24     while(ch < '0' || ch > '9') { if(ch == '-') fg = 1; ch = getchar(); }
25     while(ch >= '0' && ch <= '9') { x = x * 10 + ch - '0'; ch = getchar(); }
26     return fg ? -x : x;
27 }
28 inline void add_edge(int u, int v) {
29     e[++ecnt] = (Edge) {v, head[u]}; head[u] = ecnt;
30     e[++ecnt] = (Edge) {u, head[v]}; head[v] = ecnt;
31 }
32 void dfs(int u, int f) {
33     fa[u][0] = f;
34     dep[u] = dep[f] + 1;
35     int bot = top;
```

```
36        for(int i = head[u]; i; i = e[i].nxt) {
37            int v = e[i].to;
38            if(v == f) continue;
39            dfs(v, u);
40            if(top - bot >= sz) {
41                cnt++;
42                while(top != bot) belong[stack[top--]] = cnt;
43            }
44        }
45        stack[++top] = u;
46    }
47    void G(int &u, int step) {
48        for(int i = 0; i < S; i++) if((1 << i) & step) u = fa[u][i];
49    }
50    int lca(int u, int v) {
51        if(dep[u] > dep[v]) swap(u, v);
52        G(v, dep[v] - dep[u]);
53        if(u == v) return u;
54        for(int i = S; i >= 0; i--) if(fa[u][i] != fa[v][i]) {
55            u = fa[u][i]; v = fa[v][i];
56        }
57        return fa[u][0];
58    }
59    inline void modify(int u) {
60        tans -= V[C[u]] * sum[cur[C[u]]];
61        cur[C[u]] += vis[u];
62        vis[u] = -vis[u];
63        tans += V[C[u]] * sum[cur[C[u]]];
64    }
65    inline void update(int u, int v) {
66        if(u == v) return;
67        if(dep[u] > dep[v]) swap(u, v);
68        while(dep[v] > dep[u]) {
69            modify(v);
70            v = fa[v][0];
71        }
72        while(u != v) {
73            modify(u); modify(v);
74            u = fa[u][0]; v = fa[v][0];
75        }
76    }
77    inline void upd(int t) {
78        if(vis[qq[t].l] == -1) {
79            modify(qq[t].l);
80            swap(C[qq[t].l], qq[t].r);
81            modify(qq[t].l);
82        }
83        else swap(C[qq[t].l], qq[t].r);
84    }
85    inline void moveto(int u, int v) {
86        update(l, u); update(r, v);
87        l = u; r = v;
88    }
89    int main() {
90        n = read(); m = read(); Q = read();
91        sz = (int)pow(n, 2.0 / 3.0);
92        for(int i = 1; i <= m; i++) V[i] = read();
93        for(int i = 1; i <= n; i++) W[i] = read();
94        for(int i = 1, u, v; i < n; i++) {
95            u = read(); v = read();
96            add_edge(u, v);
```

```
97          }
98          for(int i = 1; i <= n; i++) {
99              C[i] = read();
100             vis[i] = 1;
101             sum[i] = sum[i - 1] + W[i];
102         }
103         for(int i = 1, tp; i <= Q; i++) {
104             tp = read();
105             if(tp) {
106                 ++Q1;
107                 q[Q1].l = read(); q[Q1].r = read();
108                 q[Q1].id = Q1;
109                 q[Q1].ti = i;
110             }
111             else {
112                 ++Q0;
113                 qq[Q0].l = read(); qq[Q0].r = read();
114                 qq[Q0].ti = i;
115             }
116         }
117         dfs(1, 0);
118         while(top) belong[stack[top--]] = cnt;
119         sort(q + 1, q + Q1 + 1);
120         for(int k = 1; k <= S; k++) {
121             for(int i = 1; i <= n; i++) {
122                 fa[i][k] = fa[fa[i][k - 1]][k - 1];
123             }
124         }
125         for(int i = 1; i <= Q1; i++) {
126             if(belong[q[i].l] > belong[q[i].r]) swap(q[i].l, q[i].r);
127             moveto(q[i].l, q[i].r);
128             int lc = lca(l, r);
129             modify(lc);
130             while(qq[tm + 1].ti < q[i].ti && tm < Q0) upd(++tm);
131             while(qq[tm].ti > q[i].ti) upd(tm--);
132             ans[q[i].id] = tans;
133             modify(lc);
134         }
135         for(int i = 1; i <= Q1; i++) printf("%lld\n", ans[i]);
136         return 0;
137 }
```

6.2   Randomized Algorithm

6.2.1   Simulated Annealing

```
1   void solve() {
2       while(T > eps) {
3           double alpha = ((rand() % 30001) / 15000.0) * pi;
4           double theta = ((rand() % 10001) / 10000.0) * pi;
5           tmp.x = cur.x + T * sin(theta) * cos(alpha);
6           tmp.y = cur.y + T * sin(theta) * sin(alpha);
7           tmp.z = cur.z + T * cos(theta);
8           tmp.dis = cal(tmp);
9           if(tmp.dis < cur.dis || (tmp.dis * 0.999 < cur.dis && (rand() & 7) == 7)) cur =
    tmp;
10          //if(exp((cur.d - tmp.d)  / T) > ((double)rand() / RAND_MAX)) cur = tmp;
11
12          T *= 0.999;
```

```
13        }
14  }
```

6.3   Other Method

6.3.1   Enumerate Subset

```
1  for(int i = 0; i < (1 << k); i++) {
2      for(int j = i; ; --j &= i) {
3          // work();
4          if(j == 0) break;
5      }
6  }
```

6.3.2   Enumerate $\lfloor\frac{n}{d}\rfloor\lfloor\frac{m}{d}\rfloor$

```
1  int cal(int n, int m) {
2      if(n > m) swap(n, m);
3      int res = 0, last;
4      for(int i = 1; i <= n; i = last + 1) {
5          last = min(n / (n / i), m / (m / i));
6          res += (n / i) * (m / i) * (sum(last) - sum(i - 1));
7      }
8      return res;
9  }
```

6.3.3   Find Primitive Root Modulo N

```
1  for i in range(1,mod):
2      if 3 ** i % mod == 1:
3          if i == mod - 1:
4              print("yes")
5              break
6          print("no")
```

7 Samples

7.1 vimrc

```
1  set cindent
2  set number
3  set mouse=a
4  set tabstop=4
5  set shiftwidth=4
6  syntax on
7  inoremap { {}<left>
8  map <F9> :w<CR> :! g++ % -o %< -Wall --std=c++14 -g && ./%< <CR>
9  "ios::sync_with_stdio(0); cin.tie(0); cout.precision(6); cout << fixed;
```

```
1  set nocompatible
2  source $VIMRUNTIME/vimrc_example.vim
3  source $VIMRUNTIME/mswin.vim
4  se cin nu mouse=a ts=4 sw=4 ww=b,s,<,>,[,]
5  syntax on
6  inoremap { {}<left>
7  map <F9> :w<CR> :! g++ % -o %< -Wall --std=c++14 -g && ./%< <CR>
8  func! AddTitle()
9      call append(0,"// Cease to struggle and you cease to live")
10     call append(1,"// Created by hjj")
11     call append(2,"#include <bits/stdc++.h>")
12     call append(3,"using namespace std;")
13     call append(4,"")
14     call append(5,"int main() {")
15     call append(6,"")
16     call append(7,"    return 0;")
17     call append(8,"}")
18 endfunc
19 map <F8> :call AddTitle()<CR>
```

7.2 Check

Linux

```
1  while true; do
2      ./data > in
3      ./tmp < in > out
4      ./std < in > ans
5      diff out ans
6      if [ $? -ne 0 ] ; then exit; fi
7      echo Passed
8  done
```

windows

```
1  @echo off
2  :loop
3      rand.exe > data.in
4      std.exe < data.in > std.out
5      my.exe < data.in > my.out
6      fc my.out std.out
7  if not errorlevel 1 goto loop
8  pause
9  goto loop
```

7.3 Random

```
1  mt19937_64 mt(chrono::steady_clock::now().time_since_epoch().count());
2  shuffle(per.begin(), per.end(), mt);
3  //random_shuffle(per.begin(),per.end());
```

7.4 FastIO

```
1   //普通情况
2   namespace IO {
3       const int MB = 1048576;
4       const int RMAX = 16 * MB;
5       const int WMAX = 16 * MB;
6       #define getchar() *(rp++)
7       #define putchar(x) (*(wp++) = (x))
8       char rb[RMAX], *rp = rb,  wb[WMAX], *wp = wb;
9       inline void init() {
10          fread(rb, sizeof(char), RMAX, stdin);
11      }
12      template <class _T> inline void read(_T &_a) {
13          _a = 0; bool _f = 0; int _c = getchar();
14          while (_c < '0' || _c > '9') _f |= _c == '-', _c = getchar();
15          while (_c >= '0' && _c <= '9') _a = _a * 10 + (_c ^ '0'), _c = getchar();
16          _a = _f ? -_a : _a;
17      }
18      template <class _T> inline void write(_T _a) {
19          static char buf[20], *top = buf;
20          if (_a) {
21              while (_a) {
22                  _T tm = _a / 10;
23                  *(++top) = char(_a - tm * 10) | '0';
24                  _a = tm;
25              }
26              while (top != buf) putchar(*(top--));
27          }
28          else putchar('0');
29      }
30      void output() {
31          fwrite(wb, sizeof(char), wp - wb, stdout);
32      }
33  }
34  //EOF结尾+分块读入
35  #define likely(x) __builtin_expect(!!(x), 1)
36  #define unlikely(x) __builtin_expect(!!(x), 0)
37  namespace IO {
38      const int MB = 1048576;
39      const int RMAX = 4 * MB;
40      const int WMAX = 4 * MB;
41      unsigned long long filesize;
42      #define putchar(x) (*(wp++) = (x))
43      char rb[RMAX], wb[WMAX], *wp = wb;
44      int rp = 0;
45      inline void init() {
46          filesize = fread(rb, sizeof(char), RMAX, stdin);
47          rp = 0;
48          wp = wb;
49      }
50      void output() {
51          fwrite(wb, sizeof(char), wp - wb, stdout);
```

```cpp
52          }
53      inline char getCHAR(){
54          if(unlikely(rp == filesize)){
55              fwrite(wb, sizeof(char), wp - wb, stdout);
56              init();
57              if(unlikely(filesize == 0)) {
58                  //cerr << 1.0 * (clock() - st) / CLOCKS_PER_SEC << endl;
59                  exit(0);
60              }
61          }
62          return rb[rp++];
63      }
64      template <class _T> inline void read(_T &_a) {
65          _a = 0; static bool _f = 0; static int _c;
66          _f = 0; _c = getCHAR();
67          while (_c < '0' || _c > '9') _f |= _c == '-', _c = getCHAR();
68          while (_c >= '0' && _c <= '9') _a = _a * 10 + (_c ^ '0'), _c = getCHAR();
69          _a = _f ? -_a : _a;
70      }
71      template <class _T> inline void write(_T _a) {
72          static char buf[20], *top = buf;
73          if (_a) {
74              while (_a) {
75                  _T tm = _a / 10;
76                  *(++top) = char(_a - tm * 10) | '0';
77                  _a = tm;
78              }
79              while (top != buf) putchar(*(top--));
80          }
81          else putchar('0');
82          putchar('\n');
83      }
84  }
```

7.5 Java BigNum

```java
1  import java.math.*;
2  import java.util.*;
3  import java.lang.*;
4
5  public class Main{
6      public static void main(String []args){}
7  }
8  //IO
9  Scanner in = new Scanner(System.in);
10 while(in.hasNext()){} //EOF
11 //fast-IO
12 public static void main(String argv[]) throws IOException{}
13 StreamTokenizer cin = new StreamTokenizer(new BufferedReader(new InputStreamReader(
       System.in)));
14 PrintWriter cout = new PrintWriter(new OutputStreamWriter(System.out));
15 while(cin.nextToken() != StreamTokenizer.TT_EOF) ;//EOF
16 cin.nextToken();int n = (int)cin.nval;String s = cin.sval;
17 cout.println( Type );cout.flush();
18 cin.ordinaryChar('/');
19
20 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
21 br.ready()//EOF
22 while ((valueString=bf.readLine())!=null);
```

```java
23   br.close();
24   //true fast-IO
25   static class InputReader {
26       public BufferedReader reader;
27       public StringTokenizer tokenizer;
28
29       public InputReader(InputStream stream) {
30           reader = new BufferedReader(new InputStreamReader(stream), 32768);
31           tokenizer = null;
32       }
33
34       public String next() {
35           while (tokenizer == null || !tokenizer.hasMoreTokens()) {
36               try {
37                   tokenizer = new StringTokenizer(reader.readLine());
38               } catch (IOException e) {
39                   throw new RuntimeException(e);
40               }
41           }
42           return tokenizer.nextToken();
43       }
44
45       public int nextInt() {
46           return Integer.parseInt(next());
47       }
48
49   }
50   //类 Number
51   //doubleValue()
52   //intValue()
53   //longValue()
54   //shortValue()
55   //类 BigDecimal
56   //ROUND_CEILING 接近正无穷大的舍入模式。
57   //ROUND_FLOOR 接近负无穷大的舍入模式。
58   //ROUND_DOWN 接近零的舍入模式
59   //ROUND_HALF_UP 四舍五入 >=0.5向上舍入
60   //ROUND_HALF_DOWN 四舍五入 >0.5向上舍入
61   //BigDecimal(BigInteger val)
62   //BigDecimal(BigInteger unscaledVal, int scale)
63   //BigDecimal(char[] in, int offset, int len, MathContext mc)
64   //BigDecimal(double val, MathContext mc)不建议
65   //BigDecimal(int val, MathContext mc)
66   //BigDecimal(long val, MathContext mc)
67   //BigDecimal(String val, MathContext mc)
68   //abs()
69   //add(BigDecimal augend, MathContext mc)
70   //compareTo(BigDecimal val)
71   //divide(BigDecimal divisor,MathContext mc)
72   //divideToIntegralValue(BigDecimal divisor, MathContext mc)
73   //max(BigDecimal val)
74   //min(BigDecimal val)
75   //multiply(BigDecimal multiplicand, MathContext mc)
76   //negate() 其值为 (-this), 其标度为 this.scale()
77   //pow(int n)
78   //remainder(BigDecimal divisor) 返回其值为 (this % divisor) 的 BigDecimal
79   //round(MathContext mc) 返回根据 MathContext 设置进行舍入后的 BigDecimal。
80   //caleByPowerOfTen(int n) 返回其数值等于 (this * 10^n) 的 BigDecimal。
81   //subtract(BigDecimal subtrahend, MathContext mc)
82   //setScale(int newScale,RoundingMode roundingMode)
83   //toString()
```

```
84   //ulp()返回此 BigDecimal 的 ulp (最后一位的单位) 的大小
85   //String s = b.stripTrailingZeros().toPlainString();让 bigdecimal 不用科学计数法显示
86   //类 BigInteger
87   //parseInt
88   //BigInteger zero = BigInteger.valueOf(0);
89   //BigInteger a = in.nextBigInteger();
90   //abs()
91   //and(BigInteger val) 返回其值为 (this & val)
92   //or(BigInteger val) 返回其值为 (this | val)
93   //andNot(BigInteger val) 返回其值为 (this & ~val)
94   //compareTo(BigInteger val)
95   //add(BigInteger val)
96   //divide(BigInteger val)
97   //BigInteger[] divideAndRemainder(BigInteger val) 返回包含 (this / val) 后跟 (this %
         val) 的两个 BigInteger 的数组。
98   //equals(Object x)
99   //gcd(BigInteger val)
100  //isProbablePrime(int certainty) e.g: a.isProbablePrime(4)
101  //max(BigInteger val) min(BigInteger val)
102  //mod(BigInteger m)
103  //modInverse(BigInteger m) 返回其值为 (this^−1 mod m)
104  //modPow(BigInteger exponent, BigInteger m) 返回其值为 (thisexponent mod m)
105  //multiply(BigInteger val)
106  //not() 返回其值为 (~this)
107  //shiftLeft(int n) 返回其值为 (this << n)
108  //shiftRight(int n) 返回其值为 (this >> n)
109  //toString()
110  //valueOf(long val)
111  //xor(BigInteger val) 返回其值为 (this ^ val)
112  //other
113  //Arrays.sort(array);
```

7.6 pb_ds

```
1    //P.S.:无脑正确使用 pb_ds 代替 std::set/map/priority_queue 不会变慢
2    //可持久化平衡树,不过时间和空间都不太行
3    #include <ext/rope>
4    using namespace __gnu_cxx;
5    int a[1000];
6    rope<int> x;
7    rope<int> x(a,a + n);
8    rope<int> a(x);
9    x->at(10);x[10];
10   x->push_back(x)      // 在末尾添加 x
11   x->insert(pos,x)     // 在 pos 插入 x
12   x->erase(pos,x)      // 从 pos 开始删除 x 个
13   x->replace(pos,x)    // 从 pos 开始换成 x
14   x->substr(pos,x)     // 提取 pos 开始 x 个
15
16   //树
17   //不支持低级操作 (如交换左右子树)
18   #include <ext/pb_ds/assoc_container.hpp>
19   #include <ext/pb_ds/tree_policy.hpp>
20   using namespace __gnu_pbds;
21   定义一颗红黑树
22   tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update>t, other;
23   int  关键字类型
24   null_type
25   无映射(低版本 g++ 为 null_mapped_type) (无映射为类似 set, 有映射类似 map)
```

```
26  less<int>从小到大排序
27  rb_tree_tag 红黑树（splay_tree_tag）
28  tree_order_statistics_node_update结点更新(统计子树size，可自写），不写不支持order_of_key
        以及find_by_order
29  插入:t.insert();
30  删除:t.erase();
31  比x小的个数:t.order_of_key(x);
32  第x+1值:t.find_by_order(x);
33  前驱:t.lower_bound();
34  后继:t.upper_bound();
35  合并:t.join(other); (other和*this值域不能相交)
36  分裂:t.split(x, other); (清空other，将t中比x小的元素移至other)
37
38  //自定义节点更新
39  template <class Node_CItr , class Node_Itr , class Cmp_Fn , class _Alloc >
40  struct my_node_update {
41      virtual Node_CItr node_begin () const = 0;
42      virtual Node_CItr node_end() const = 0;
43      typedef char metadata_type;//节点上记录的额外信息的类型
44      //以上为固定格式
45
46      //operator()的功能是将节点it的信息更新为其左右孩子的信息之和，传入的end_it表示空节点
47      //对Node_Itr可以做的事情有：用get_l_child,get_r_child获取左右孩子，用两个星号获取节
          点信息，用get_metadata获取节点额外信息
48      inline void operator()(Node_Itr it, Node_CItr end_it) {
49          Node_Itr l = it.get_l_child(), r = it.get_r_child();
50          int left = 0, right = 0;
51          if(l != end_it) left = l.get_metadata();
52          if(r != end_it) right = r.get_metadata();
53          const_cast<metadata_type &>(it.get_metadata()) = left + right + (*it)->second;
54          //it是node_Itr，取*后变为iterator，再取—>second变成mapped_value
55      }
56      inline int prefix_sum(int x) {
57          int ans = 0;
58          Node_CItr it = node_begin ();
59          while(it != node_end()) {
60              Node_CItr l = it.get_l_child(), r = it.get_r_child();
61              if(Cmp_Fn()(x, (*it)->first)) it = l;
62              else {
63                  ans += (*it)->second;
64                  if(l != node_end()) ans += l.get_metadata();
65                  it = r;
66              }
67          }
68          return ans;
69      }
70      inline int interval_sum(int l, int r) {
71          return prefix_sum(r) - prefix_sum(l - 1);
72      }
73  };
74
75  tree<int, char, less<int>, rb_tree_tag, my_node_update> T;//map
76  int main() {
77      T[2] = 'a'; T[3] = 'b'; T[4] = 1;
78      cout << (char)T.interval_sum(3, 4) << endl;//c
79      return 0;
80  }
81  //堆
82  #include <ext/pb_ds/priority_queue.hpp>
83  using namespace __gnu_pbds;
84  __gnu_pbds::priority_queue<int, std::less<int>, __gnu_pbds::pairing_heap_tag> q;
```

```
85
86  template <typename Value_Type ,
87  typename Cmp_Fn = std::less<Value_Type>,
88  typename Tag = pairing_heap_tag ,
89  typename Allocator = std::allocator<char> >
90  class priority_queue
91  Tag可以是binary_heap_tag（二叉堆）binomial_heap_tag（二项堆）rc_binomial_heap_tag
        pairing_heap_tag（配对堆）thin_heap_tag
92  用begin()和end()获取迭代器从而遍历
93  删除单个元素 void erase(point_iterator)
94  更改一个元素的值 void modify(point_iterator, const_reference)
95  合并 void join(priority_queue &other)，把other合并到*this，并把other清空
96  push()会返回迭代器
97  五种操作：push、pop、modify、erase、join
98  • pairing_heap_tag：push和joinO(1)，其余均摊O(logn)
99  • binary_heap_tag：只支持push和pop，均为均摊O(logn)
100 • binomial_heap_tag：push为均摊O(1)，其余为O(logn)
101 • rc_binomial_heap_tag：push为O(1)，其余为O(logn)
102 • thin_heap_tag：push为O(1)，不支持join，其余为O(logn)；但是如果只有increase_key，modify
        均摊O(1)
103 • 不支持不是不能用，而是用起来很慢
104 经过实践检测得到的结论：
105 • Dijkstra算法中应用pairing_heap_tag，速度与手写数据结构相当。
106 • binary_heap_tag在绝大多数情况下优于std::priority_queue
107 • pairing_heap_tag在绝大多数情况优于binomial_heap_tag和rc_binomial_heap_tag
108 • 只有push，pop和join操作时，binary_heap_tag速度较快
109 • 有modify操作时，可以考虑thin_heap_tag或者pairing_heap_tag，或手写数据结构。
110
111 //hash_table
112 #include <ext/pb_ds/assoc_container.hpp>
113 #include <ext/pb_ds/hash_policy.hpp>
114 using namespace __gnu_pbds;
115 __gnu_pbds::cc_hash_table <Key, Mapped> mp; //使用链地址法解决哈希冲突
116 __gnu_pbds::gp_hash_table <Key, Mapped> mp; //使用探测法解决哈希冲突
117 //用法和map一样
```