



*Zhejiang Normal University*

*School of Mathematical Sciences*

---

# ACM-ICPC Code Template

---

*Author:*  
Jerry Black

*Date:*  
2024 年 11 月 10 日

Contest

1	Graph	1
1.1	Dijkstra	1
1.2	匈牙利算法	1
1.3	Hopcroft_Karp(匈牙利算法优化)	2
1.4	二分图染色 + 拓扑排序	3
1.5	网络流	4
1.5.1	最大流	4
1.5.2	KM	5
1.5.3	最小费用最大流	6
1.5.4	最大费用可行流	6
1.6	连通分量	7
1.6.1	强连通分量	7
1.6.2	边双连通分量	7
1.6.3	点双连通分量	8
1.6.4	割点	9
1.6.5	割边	9
1.6.6	圆方树	10
1.7	2-SAT	10
1.8	虚树	11
1.9	树分治	11
1.9.1	点分治	11
1.9.2	点分树	12
1.10	异或最小生成树	13
2	Data Structure	14
2.1	种类并查集	14
2.2	二维树状数组	14
2.2.1	单点修改, 区间查询	14
2.2.2	区间修改, 单点查询	15
2.2.3	区间修改, 区间查询	15
2.3	线段树	15
2.3.1	维护等差数列	15
2.3.2	有加有乘	17
2.3.3	最大子段和	18
2.4	非递归版本线段树	19
2.4.1	用法 1:	21
2.4.2	用法 2:	22
2.5	动态开点线段树	22
2.6	Segment Tree Beats	23
2.6.1	区间取 min、max $O(n\log n)$	23
2.6.2	区间取 min、max + 区间加减 $O(n\log^2 n)$	24
2.7	可持续化线段树	26
2.7.1	静态版本, 权值线段树	26
2.7.2	动态版本, 支持修改	27
2.8	线段树合并 & 分裂	27
2.9	分块	29
2.10	珂朵莉树	30
2.11	莫队	31
2.11.1	普通莫队	31
2.11.2	带修莫队	31
2.11.3	树上莫队	32
2.11.4	回滚莫队	33
2.12	CDQ 分治	34
2.12.1	二维	34
2.12.2	三维	35
2.13	链式前向星	36
2.14	Kruskal 重构树	36
2.15	树链剖分	37
2.16	ST 表	38
2.17	DSU on Tree	39
2.18	线性基	39
2.19	平衡树	40
2.19.1	替罪羊树	40
2.19.2	FHQ Treap	42
2.19.3	Splay	45
2.20	LCT	47
3	String	49
3.1	KMP	49
3.2	马拉车	49
3.3	哈希	49
3.3.1	一维	49
3.3.2	二维	50
3.4	Trie 树	51
3.5	AC 自动机	51
3.6	回文自动机	52
3.7	广义回文自动机	53
3.8	后缀数组 (SA)	54
3.9	后缀数组 (SA-IS)	55
3.10	后缀自动机	58
3.11	广义后缀自动机	60
4	Mathematics	62
4.1	快速乘	62
4.2	欧几里得算法	62
4.3	拓展欧几里得算法	62
4.4	BSGS	62
4.5	埃氏筛	63
4.6	欧拉筛	63
4.7	米勒罗宾素性测试 & Pollard rho 算法	63
4.8	Stein 算法 (大整数的 GCD)	65
4.9	逆元	65
4.10	欧拉函数	65
4.11	欧拉降幂	66

4.12	拓展中国剩余定理	66	7.9	gp_hash_table	100
4.13	中国剩余定理	66	7.10	关同步流	100
4.14	二次剩余	66	7.11	计时器	100
4.15	拓展欧拉定理	66			
4.16	组合数	67			
4.17	卢卡斯定理	67			
4.18	拓展卢卡斯定理	67			
4.19	积性函数	68			
4.20	杜教筛	68			
4.21	Min_25 筛	69			
4.22	莫比乌斯函数	73			
4.23	快速数论变换	73			
4.24	FWT	79			
4.25	BM	80			
4.26	矩阵求逆	81			
4.27	矩阵乘法 & 快速幂	82			
4.28	整除分块	83			
4.29	自适应辛普森积分	83			
4.30	最小二乘法	83			
4.31	高精度	83			
5	Dynamic Programming	85			
5.1	最长公共子序列	85			
5.2	背包	85			
5.2.1	01 背包	85			
5.2.2	完全背包	85			
5.2.3	多重背包	86			
5.2.4	多种背包混合问题	86			
5.2.5	二维费用的背包问题	87			
5.2.6	分组的背包问题	87			
5.2.7	有依赖的背包问题	88			
5.3	状压 dp	89			
5.4	数位 dp	90			
5.5	带障碍的路径计数	90			
6	Geometry	91			
6.1	二维	91			
6.2	三维	93			
7	Others	95			
7.1	头文件	95			
7.2	吸氧大法	95			
7.3	迭代器	96			
7.4	lambda 函数	96			
7.5	结构体	96			
7.6	快读快写	96			
7.7	取模	98			
7.8	Int128	100			

# 1 Graph

## 1.1 Dijkstra

```

1 struct Edge
2 {
3     int to,dis;
4 };
5 vector<Edge>edge[200005];
6 struct city
7 {
8     int id,dis;
9     bool operator<(const city& c)const
10    {
11        return dis>c.dis;
12    }
13 };
14 priority_queue<city>q;
15 int dis[200005];
16 int vis[200005];
17 int n,m;
18 int s,t;
19 void init()
20 {
21     for(int i=1;i<=n;i++)
22     {
23         dis[i]=INF;
24         vis[i]=0;
25     }
26     dis[s]=0;
27     q.push({s,0});
28 }
29 void dijkstra()
30 {
31     while(!q.empty())
32     {
33         auto [id,d]=q.top();
34         q.pop();
35         if(vis[id])continue;
36         else vis[id]=1;
37         for(auto [it,dd]:edge[id])
38         {
39             if(dis[it]>dis[id]+dd)
40             {
41                 dis[it]=dis[id]+dd;
42                 q.push({it,dis[it]});
43             }
44         }
45     }
46 }

```

```

44     }
45 }
46 }
47 int main()
48 {
49     while(scanf("%d%d",&n,&m)!=EOF)
50     {
51         for(int i=0;i<n;i++)edge[i].clear();
52         while(!q.empty())q.pop();
53         for(int i=1;i<=m;i++)
54         {
55             int from,to,d;
56             scanf("%d%d%d",&from,&to,&d);
57             edge[from].push_back({to,d});
58             edge[to].push_back({from,d});
59         }
60         scanf("%d%d",&s,&t);
61         init();
62         dijkstra();
63         if(dis[t]==1e9)printf("-1\n");
64         else printf("%d\n",dis[t]);
65     }
66     return 0;
67 }

```

## 1.2 匈牙利算法

```

1 const int MAXN=505;
2 int k,n,m;
3 int Map[MAXN][MAXN];
4 int p[MAXN];
5 bool vis[MAXN];
6 bool match(int i)
7 {
8     for (int j=1;j<=m;j++)
9     {
10         if(Map[i][j]&&!vis[j])
11         {
12             vis[j]=1;
13             if(p[j]==0||match(p[j]))
14             {
15                 p[j]=i;
16                 return 1;
17             }
18         }
19     }
20 }

```

```

20     return 0;
21 }
22 int main()
23 {
24     while(scanf("%d",&k) != EOF)
25     {
26         memset(Map,0,sizeof(Map));
27         memset(p,0,sizeof(p));
28         if(k==0) return 0;
29         scanf("%d%d",&n,&m);
30         for (int i=1;i<=k;i++)
31         {
32             int x,y;
33             scanf("%d%d",&x,&y);
34             Map[x][y]=1;
35         }
36         int cnt=0;
37         for (int i=1;i<=n;i++)
38         {
39             memset(vis,0,sizeof(vis));
40             if(match(i))cnt++;
41         }
42         printf("%d\n",cnt);
43     }
44     return 0;
45 }

```

### 1.3 Hopcroft\_Karp(匈牙利算法优化)

```

1 struct Hopcroft_Karp
2 {
3     vector<vector<int>>>es;
4     vector<int>d,match;
5     vector<bool>used,used2;
6     const int n,m;
7     Hopcroft_Karp(int n,int m):es(n),d(n),match(m),used(n),used2(n),n(n),m(m){}
8     void add_edge(int u,int v){es[u].push_back(v);}
9     void _bfs()
10    {
11        fill(begin(d),end(d),-1);
12        queue<int>que;
13        for(int i=0;i<n;i++)
14        {
15            if (!used[i]){que.push(i);d[i] = 0;}
16        }

```

```

17     while (!que.empty())
18     {
19         int i=que.front();
20         que.pop();
21         for(auto &e:es[i])
22         {
23             int j=match[e];
24             if(j!=-1&&d[j]==-1){que.push(j);d[j]=d[i]+1;}
25         }
26     }
27 }
28 bool _dfs(int now)
29 {
30     used2[now]=true;
31     for (auto &e:es[now])
32     {
33         int u=match[e];
34         if(u==-1||(!used2[u]&&d[u]==d[now]+1&& _dfs(u)))
35         {
36             match[e]=now,used[now]=true;
37             return true;
38         }
39     }
40     return false;
41 }
42 int max_matching() // 右边的i与左边的match[i]匹配。
43 {
44     fill(begin(match),end(match),-1);
45     fill(begin(used),end(used),false);
46     int ret=0;
47     while(1)
48     {
49         _bfs();
50         fill(begin(used2),end(used2),false);
51         int flow=0;
52         for(int i=0;i<n;i++){if(!used[i]&&_dfs(i))flow++;}
53         if(flow==0)break;
54         ret+=flow;
55     }
56     return ret;
57 }
58 };
59
60 string s[305];
61 int x[305][305],y[305][305];
62
63 void solve()

```

```

64 {
65     int n,m;
66     cin>>n>>m;
67     for(int i=0;i<n;i++)cin>>s[i];
68     int l=0,r=0;
69     memset(x,-1,sizeof(x));
70     memset(y,-1,sizeof(y));
71     for(int i=0;i<n;i++)
72     {
73         for(int j=0;j<m;j++)
74         {
75             if(s[i][j]=='#')continue;
76             x[i][j]=l;
77             while(j+1<m&&s[i][j+1]=='.') {x[i][j+1]=l;j++;}
78             l++;
79         }
80     }
81     for(int j=0;j<m;j++)
82     {
83         for(int i=0;i<n;i++)
84         {
85             if(s[i][j]=='#')continue;
86             y[i][j]=r;
87             while(i+1<n&&s[i+1][j]=='.') {y[i+1][j]=r;i++;}
88             r++;
89         }
90     }
91     if(l==0&&r==0)
92     {
93         cout<<0;
94         return;
95     }
96     Hopcroft_Karp BM(l,r);
97     for(int i=0;i<n;i++)
98     {
99         for(int j=0;j<m;j++)
100         {
101             if(x[i][j]!=-1&&y[i][j]!=-1)BM.add_edge(x[i][j],y[i][j]);
102         }
103     }
104     cout<<BM.max_matching();
105 }

```

#### 1.4 二分图染色 + 拓扑排序

```
1 #include<bits/stdc++.h>
```

```

2 using namespace std;
3 struct edge
4 {
5     int type,u,v;
6 };
7 vector<int>adj[200005];
8 int col[200005],topo[200005];
9 void dfs(int v)
10 {
11     for(int u:adj[v])if(col[u]==-1)
12     {
13         col[u]=col[v]^1;
14         dfs(u);
15     }
16 }
17 bool BipartiteColoring(int n)
18 {
19     for(int i=1;i<=n;i++)col[i]=-1;
20     for(int i=1;i<=n;i++)if(col[i]==-1)
21     {
22         col[i]=0;
23         dfs(i);
24     }
25     for(int i=1;i<=n;i++)for(int j:adj[i])if(col[i]==col[j])return 0;
26     return 1;
27 }
28 bool TopologicalSort(int n)
29 {
30     vector<int>in(n+1,0);
31     for(int i=1;i<=n;i++)for(int j:adj[i])in[j]++;
32     queue<int>q;
33     for(int i=1;i<=n;i++)if(in[i]==0)q.push(i);
34     int ord=0;
35     while(!q.empty())
36     {
37         int v=q.front();
38         q.pop();
39         topo[v]=ord++;
40         for(int u:adj[v])
41         {
42             in[u]--;
43             if(in[u]==0)q.push(u);
44         }
45     }
46     return ord==n;
47 }
48 int main()

```

```

49 {
50     int n,m;
51     scanf("%d%d",&n,&m);
52     vector<edge>a(m);
53     for(int i=0;i<m;i++)
54     {
55         scanf("%d%d%d",&a[i].type,&a[i].u,&a[i].v);
56         adj[a[i].u].push_back(a[i].v);
57         adj[a[i].v].push_back(a[i].u);
58     }
59     if(!BipartiteColoring(n)){printf("NO\n");return 0;}
60     // col = 0 -> orient left, col = 1 -> orient right
61     for(int i=1;i<=n;i++)adj[i].clear();
62     for(edge e:a)
63     {
64         if(col[e.u]==1)swap(e.u,e.v);
65         if(e.type==1)adj[e.u].push_back(e.v);
66         else adj[e.v].push_back(e.u);
67     }
68     if(!TopologicalSort(n)){printf("NO\n");return 0;}
69     printf("YES\n");
70     for(int i=1;i<=n;i++)
71     {
72         col[i]==1?printf("R %d\n",topo[i]):printf("L %d\n",topo[i]);
73     }
74     return 0;
75 }

```

## 1.5 网络流

### 1.5.1 最大流

```

1  const int MAXN=1000005;
2  const int inf=1000000000;
3  int S,T;
4  struct EDGE
5  {
6      int v,flow,nxt;
7  }edge[MAXN<<2];
8  int head[MAXN],cur[MAXN],num=0;
9  int dep[MAXN],q[MAXN];
10 void add_edge(int u,int v,int w)
11 {
12     edge[num].v=v;
13     edge[num].flow=w;
14     edge[num].nxt=head[u];

```

```

15     head[u]=num++;
16 }
17 void Add_edge(int u,int v,int w)
18 {
19     add_edge(u,v,w);
20     add_edge(v,u,0);
21 }
22 bool bfs()
23 {
24     memset(dep,0,sizeof(dep));
25     dep[S]=1;
26     int l=0,r=1;
27     q[++l]=S;
28     while(l<=r)
29     {
30         int p=q[l++];
31         if(p==T)break;
32         for(int i=head[p];~i;i=edge[i].nxt)
33         {
34             if(!dep[edge[i].v]&&edge[i].flow)
35             {
36                 dep[edge[i].v]=dep[p]+1;
37                 q[++r]=edge[i].v;
38                 if(edge[i].v==T)return 1;
39             }
40         }
41     }
42     return dep[T];
43 }
44 int dfs(int now,int nowflow)
45 {
46     if(now==T)return nowflow;
47     int totflow=0;
48     for(int& i=cur[now];~i;i=edge[i].nxt)
49     {
50         if(dep[edge[i].v]==dep[now]+1&&edge[i].flow)
51         {
52             int canflow=dfs(edge[i].v,min(nowflow,edge[i].flow));
53             edge[i].flow-=canflow;
54             edge[i^1].flow+=canflow;
55             totflow+=canflow;
56             nowflow-=canflow;
57             if(nowflow<=0)break;
58         }
59     }
60     return totflow;
61 }

```

```

62 int dinic()
63 {
64     int ans=0;
65     while(bfs())
66     {
67         memcpy(cur,head,sizeof(head));
68         ans+=dfs(S,inf);
69     }
70     return ans;
71 }
72 void init()
73 {
74     memset(q,0,sizeof(q));
75     memset(cur,0,sizeof(cur));
76     memset(dep,0,sizeof(dep));
77     memset(head,-1,sizeof(head));
78     for(int i=0;i<=num;i++)edge[i].v=edge[i].flow=edge[i].nxt=0;
79     num=0;
80 }

```

### 1.5.2 KM

```

1 //KM算法适用于求解完美匹配下的最大权匹配,时间复杂度 $O(n^3)$ 
2 const int INF=0x3f3f3f3f;
3 const int N=305;
4 int n,m,match[N],pre[N];
5 bool vis[N];
6 int favor[N][N];
7 int val1[N],val2[N],slack[N];
8 void bfs(int p)
9 {
10     memset(pre,0,sizeof pre);
11     memset(slack,INF,sizeof slack);
12     match[0]=p;
13     int x=0,nex=0;
14     do{
15         vis[x]=true;
16         int y=match[x],d=INF;
17         // 对于当前节点y, bfs有连边的下一点
18         for(int i=1;i<=m;i++)
19         {
20             if(!vis[i])
21             {
22                 if(slack[i]>val1[y]+val2[i]-favor[y][i])
23                 {
24                     slack[i]=val1[y]+val2[i]-favor[y][i];

```

```

25         pre[i]=x;
26     }
27     if(slack[i]<d)
28     {
29         d=slack[i];
30         nex=i;
31     }
32 }
33 }
34 for(int i=0;i<=m;i++)
35 {
36     if(vis[i])
37         val1[match[i]]-=d,val2[i]+=d;
38     else
39         slack[i]-=d;
40 }
41 x=nex;
42 }while(match[x]);
43
44 // pre数组对bfs访问路径进行记录,在最后一并改变match
45 while(x)
46 {
47     match[x]=match[pre[x]];
48     x=pre[x];
49 }
50 }
51 int KM()
52 {
53     memset(match,0,sizeof match);
54     memset(val1,0,sizeof val1);
55     memset(val2,0,sizeof val2);
56     for(int i=1;i<=n;i++)
57     {
58         memset(vis,false,sizeof vis);
59         bfs(i);
60     }
61     int res=0;
62     for(int i=1;i<=m;i++)
63         res+=favor[match[i]][i];
64     return res;
65 }
66 int main()
67 {
68     // Input favor[i][j]
69
70     printf("%d",KM());
71 }

```



```

72     return 0;
73 }

```

### 1.5.3 最小费用最大流

```

1  const int maxn=1050;
2
3  struct MCMF {
4      struct E {
5          int from, to, cap, v;
6          E() {}
7          E(int f, int t, int cap, int v) : from(f), to(t), cap(cap), v(v)
8              {}
9      };
10     int n, m, s, t;
11     vector<E> edges;
12     vector<int> G[maxn];
13     bool inq[maxn];
14     int dis[maxn], pre[maxn], a[maxn];
15     void init(int _n, int _s, int _t) {
16         n = _n; s = _s; t = _t;
17         for (int i = 0; i <= n; i++)
18             G[i].clear();
19         edges.clear();
20         m = 0;
21     }
22     void add(int from, int to, int cap, int cost) {
23         edges.emplace_back(from, to, cap, cost);
24         edges.emplace_back(to, from, 0, -cost);
25         G[from].push_back(m++);
26         G[to].push_back(m++);
27     }
28     bool spfa() {
29         for (int i = 0; i <= n; i++) {
30             dis[i] = 1e9;
31             pre[i] = -1;
32             inq[i] = false;
33         }
34         dis[s] = 0, a[s] = 1e9, inq[s] = true;
35         queue<int> Q; Q.push(s);
36         while (!Q.empty()) {
37             int u = Q.front(); Q.pop();
38             inq[u] = false;
39             for (int& idx: G[u]) {
40                 E& e = edges[idx];
41                 if (e.cap && dis[e.to] > dis[u] + e.v) {

```

```

41                 dis[e.to] = dis[u] + e.v;
42                 pre[e.to] = idx;
43                 a[e.to] = min(a[u], e.cap);
44                 if (!inq[e.to]) {
45                     inq[e.to] = true;
46                     Q.push(e.to);
47                 }
48             }
49         }
50     }
51     return pre[t] != -1;
52 }
53 int solve() {
54     int flow = 0, cost = 0;
55     while (spfa()) {
56         flow += a[t];
57         cost += a[t] * dis[t];
58         int u = t;
59         while (u != s) {
60             edges[pre[u]].cap -= a[t];
61             edges[pre[u] ^ 1].cap += a[t];
62             u = edges[pre[u]].from;
63         }
64     }
65     return cost;
66 }
67 }f;

```

### 1.5.4 最大费用可行流

```

1  struct MCFF{ // Maximum cost feasible flow 最大费用可行流
2      int n, s, t, cost, dis[N], st[N], vis[N];
3      int head[N], nxt[M], v[M], w[M], flow[M], ecnt;
4
5      inline void ade(int a, int b, int c, int dis){ v[++ecnt]=b; nxt[
6          ecnt]=head[a]; w[ecnt]=dis; flow[ecnt]=c; head[a]=ecnt;}
7
8      inline void addedge(int a, int b, int c, int dis){ ade(a,b,c,dis);
9          ade(b,a,0,-dis);}
10
11     inline void init(int _n, int _s, int _t){
12         n=_n; s=_s; t=_t; ecnt=1; cost=0;
13         memset(head, 0, sizeof head);
14     }
15
16     inline int spfa(){

```

```

15 queue<int> q; q.push(s);
16 memset(st,0,sizeof st);
17 memset(dis,0x3f,sizeof dis); dis[s]=0;
18 while(q.size()){
19     int u=q.front(); q.pop(); vis[u]=0;
20     for(int i=head[u];i;i=nxt[i]){
21         if(flow[i] and dis[v[i]]>dis[u]+w[i]){
22             dis[v[i]]=dis[u]+w[i];
23             if(!vis[v[i]]) q.push(v[i]), vis[v[i]]=1;
24         }
25     }
26 } return dis[t]<0;
27 }
28
29 int dfs(int x, int f){
30     if(x==t) return cost+=dis[t]*f,f;
31     st[x]=1; int fl=0;
32     for(int i=head[x];i;i=nxt[i]){
33         if(flow[i] and !st[v[i]] and dis[v[i]]==dis[x]+w[i]){
34             int mi=dfs(v[i], min(f, flow[i]));
35             flow[i]-=mi; flow[i^1]+=mi; fl+=mi; f-=mi;
36         }
37     } return fl;
38 }
39
40 inline int work(){
41     while(spfa()) dfs(s, INF);
42     return cost;
43 }
44 }f;

```

## 1.6 连通分量

### 1.6.1 强连通分量

```

1 int dfn[N], low[N], dfncnt, s[N], in_stack[N], tp;
2 int scc[N], sc; // 结点 i 所在 SCC 的编号
3 int sz[N]; // 强连通 i 的大小
4
5 void tarjan(int u) {
6     low[u] = dfn[u] = ++dfncnt, s[++tp] = u, in_stack[u] = 1;
7     for (int i = h[u]; i; i = e[i].nex) {
8         const int &v = e[i].t;
9         if (!dfn[v]) {
10             tarjan(v);
11             low[u] = min(low[u], low[v]);

```

```

12         } else if (in_stack[v]) {
13             low[u] = min(low[u], dfn[v]);
14         }
15     }
16     if (dfn[u] == low[u]) {
17         ++sc;
18         while (s[tp] != u) {
19             scc[s[tp]] = sc;
20             sz[sc]++;
21             in_stack[s[tp]] = 0;
22             --tp;
23         }
24         scc[s[tp]] = sc;
25         sz[sc]++;
26         in_stack[s[tp]] = 0;
27         --tp;
28     }
29 }

```

### 1.6.2 边双连通分量

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <stack>
4 #include <vector>
5
6 using namespace std;
7 const int N = 5e5 + 5, M = 2e6 + 5;
8 int n, m;
9
10 struct edge {
11     int to, nt;
12 } e[M << 2];
13
14 int hd[N << 1], tot = 1;
15
16 void add(int u, int v) { e[++tot] = (edge){v, hd[u]}, hd[u] = tot; }
17
18 void uadd(int u, int v) { add(u, v), add(v, u); }
19
20 int bcc_cnt, sum;
21 int dfn[N], low[N];
22 bool vis[N];
23 vector<vector<int>> ans;
24 stack<int> st;
25

```

```

26 void tarjan(int u, int in) {
27     low[u] = dfn[u] = ++bcc_cnt;
28     st.push(u), vis[u] = 1;
29     for (int i = hd[u]; i; i = e[i].nt) {
30         int v = e[i].to;
31         if (i == (in ^ 1)) continue;
32         if (!dfn[v])
33             tarjan(v, i), low[u] = min(low[u], low[v]);
34         else if (vis[v])
35             low[u] = min(low[u], dfn[v]);
36     }
37     if (dfn[u] == low[u]) {
38         vector<int> t;
39         t.push_back(u);
40         while (st.top() != u) t.push_back(st.top()), vis[st.top()] = 0,
41             st.pop();
42         st.pop(), ans.push_back(t);
43     }
44 }
45 int main() {
46     scanf("%d%d", &n, &m);
47     int u, v;
48     for (int i = 1; i <= m; i++) {
49         scanf("%d%d", &u, &v);
50         if (u != v) uadd(u, v);
51     }
52     for (int i = 1; i <= n; i++)
53         if (!dfn[i]) tarjan(i, 0);
54     printf("%llu\n", ans.size());
55     for (int i = 0; i < ans.size(); i++) {
56         printf("%llu ", ans[i].size());
57         for (int j = 0; j < ans[i].size(); j++) printf("%d ", ans[i][j]);
58         printf("\n");
59     }
60     return 0;
61 }

```

### 1.6.3 点双连通分量

```

1 #include <cstdio>
2 #include <vector>
3 using namespace std;
4 const int N = 5e5 + 5, M = 2e6 + 5;
5 int n, m;

```

```

6 struct edge {
7     int to, nt;
8 } e[M << 1];
9
10 int hd[N], tot = 1;
11
12 void add(int u, int v) { e[++tot] = (edge){v, hd[u]}, hd[u] = tot; }
13
14 void uadd(int u, int v) { add(u, v), add(v, u); }
15
16 int ans;
17 int dfn[N], low[N], bcc_cnt;
18 int sta[N], top, cnt;
19 bool cut[N];
20 vector<int> dcc[N];
21 int root;
22
23 void tarjan(int u) {
24     dfn[u] = low[u] = ++bcc_cnt, sta[++top] = u;
25     if (u == root && hd[u] == 0) {
26         dcc[++cnt].push_back(u);
27         return;
28     }
29     int f = 0;
30     for (int i = hd[u]; i; i = e[i].nt) {
31         int v = e[i].to;
32         if (!dfn[v]) {
33             tarjan(v);
34             low[u] = min(low[u], low[v]);
35             if (low[v] >= dfn[u]) {
36                 if (++f > 1 || u != root) cut[u] = true;
37                 cnt++;
38                 do dcc[cnt].push_back(sta[top--]);
39                 while (sta[top + 1] != v);
40                 dcc[cnt].push_back(u);
41             }
42         } else
43             low[u] = min(low[u], dfn[v]);
44     }
45 }
46
47 int main() {
48     scanf("%d%d", &n, &m);
49     int u, v;
50     for (int i = 1; i <= m; i++) {
51         scanf("%d%d", &u, &v);
52     }

```

```

53     if (u != v) uadd(u, v);
54 }
55 for (int i = 1; i <= n; i++)
56     if (!dfn[i]) root = i, tarjan(i);
57 printf("%d\n", cnt);
58 for (int i = 1; i <= cnt; i++) {
59     printf("%llu ", dcc[i].size());
60     for (int j = 0; j < dcc[i].size(); j++) printf("%d ", dcc[i][j]);
61     printf("\n");
62 }
63 return 0;
64 }

```

#### 1.6.4 割点

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int n, m; // n: 点数 m: 边数
4 int dfn[100001], low[100001], inde, res;
5 // dfn: 记录每个点的时间戳
6 // low: 能经过父亲到达最小的编号, inde: 时间戳, res: 答案数量
7 bool vis[100001], flag[100001]; // flag: 答案 vis: 标记是否重复
8 vector<int> edge[100001]; // 存图用的
9
10 void Tarjan(int u, int father) { // u 当前点的编号, father 自己爸爸的编号
11     vis[u] = true; // 标记
12     low[u] = dfn[u] = ++inde; // 打上时间戳
13     int child = 0; // 每一个点儿子数量
14     for (auto v : edge[u]) { // 访问这个点的所有邻居 (C++11)
15         if (!vis[v]) {
16             child++; // 多了一个儿子
17             Tarjan(v, u); // 继续
18             low[u] = min(low[u], low[v]); // 更新能到的最小节点编号
19             if (father != u && low[v] >= dfn[u] && !flag[u]) { // 主要代码
20                 // 如果不是自己, 且不通过父亲返回的最小点符合割点的要求, 并且没有被标记过
21                 // 要求即为: 删了父亲连不上去了, 即为最多连到父亲
22                 flag[u] = true;
23                 res++; // 记录答案
24             }
25         } else if (v != father) {
26             // 如果这个点不是自己的父亲, 更新能到的最小节点编号
27             low[u] = min(low[u], dfn[v]);
28         }
29     }
30 }

```

```

30 // 主要代码, 自己的话需要 2 个儿子才可以
31 if (father == u && child >= 2 && !flag[u]) {
32     flag[u] = true;
33     res++; // 记录答案
34 }
35 }
36
37 int main() {
38     cin >> n >> m; // 读入数据
39     for (int i = 1; i <= m; i++) { // 注意点是从 1 开始的
40         int x, y;
41         cin >> x >> y;
42         edge[x].push_back(y);
43         edge[y].push_back(x);
44     } // 使用 vector 存图
45     for (int i = 1; i <= n; i++) // 因为 Tarjan 图不一定连通
46         if (!vis[i]) {
47             inde = 0; // 时间戳初始为 0
48             Tarjan(i, i); // 从第 i 个点开始, 父亲为自己
49         }
50     cout << res << endl;
51     for (int i = 1; i <= n; i++)
52         if (flag[i]) cout << i << " "; // 输出结果
53     return 0;
54 }

```

#### 1.6.5 割边

```

1 int low[MAXN], dfn[MAXN], dfs_clock;
2 bool isbridge[MAXN];
3 vector<int> G[MAXN];
4 int cnt_bridge;
5 int father[MAXN];
6
7 void tarjan(int u, int fa) {
8     father[u] = fa;
9     low[u] = dfn[u] = ++dfs_clock;
10    for (int i = 0; i < G[u].size(); i++) {
11        int v = G[u][i];
12        if (!dfn[v]) {
13            tarjan(v, u);
14            low[u] = min(low[u], low[v]);
15            if (low[v] > dfn[u]) {
16                isbridge[v] = true;
17                ++cnt_bridge;
18            }
19        }
20    }
21 }

```

```

19     } else if (dfn[v] < dfn[u] && v != fa) {
20         low[u] = min(low[u], dfn[v]);
21     }
22 }
23 }

```

### 1.6.6 圆方树

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <vector>
4
5 const int MN = 100005;
6
7 int N, M, cnt;
8 std::vector<int> G[MN], T[MN * 2];
9
10 int dfn[MN], low[MN], dfc;
11 int stk[MN], tp;
12
13 void Tarjan(int u) {
14     printf(" Enter : %d\n", u);
15     low[u] = dfn[u] = ++dfc; // low 初始化为当前节点 dfn
16     stk[++tp] = u; // 加入栈中
17     for (int v : G[u]) { // 遍历 u 的相邻节点
18         if (!dfn[v]) { // 如果未访问过
19             Tarjan(v); // 递归
20             low[u] = std::min(low[u], low[v]); // 未访问的和 low 取 min
21             if (low[v] == dfn[u]) { // 标志着找到一个以 u 为根的点双连通分量
22                 ++cnt; // 增加方点个数
23                 printf(" Found a New BCC %d.\n", cnt - N);
24                 // 将点双中除了 u 的点退栈, 并在圆方树中连边
25                 for (int x = 0; x != v; --tp) {
26                     x = stk[tp];
27                     T[cnt].push_back(x);
28                     T[x].push_back(cnt);
29                     printf(" BCC %d has vertex %d\n", cnt - N, x);
30                 }
31                 // 注意 u 自身也要连边 (但不退栈)
32                 T[cnt].push_back(u);
33                 T[u].push_back(cnt);
34                 printf(" BCC %d has vertex %d\n", cnt - N, u);
35             }
36         } else
37             low[u] = std::min(low[u], dfn[v]); // 已访问的和 dfn 取 min
38     }
39 }

```

```

39     printf(" Exit : %d : low = %d\n", u, low[u]);
40     printf(" Stack:\n ");
41     for (int i = 1; i <= tp; ++i) printf("%d, ", stk[i]);
42     puts("");
43 }
44
45 int main() {
46     scanf("%d%d", &N, &M);
47     cnt = N; // 点双 / 方点标号从 N 开始
48     for (int i = 1; i <= M; ++i) {
49         int u, v;
50         scanf("%d%d", &u, &v);
51         G[u].push_back(v); // 加双向边
52         G[v].push_back(u);
53     }
54     // 处理非连通图
55     for (int u = 1; u <= N; ++u)
56         if (!dfn[u]) Tarjan(u), --tp;
57     // 注意到退出 Tarjan 时栈中还有一个元素即根, 将其退栈
58     return 0;
59 }

```

### 1.7 2-SAT

```

1 struct Twosat {
2     int n;
3     vector<int> g[maxn * 2];
4     bool mark[maxn * 2];
5     int s[maxn * 2], c;
6
7     bool dfs(int x) {
8         if (mark[x ^ 1]) return false;
9         if (mark[x]) return true;
10        mark[x] = true;
11        s[c++] = x;
12        for (int i = 0; i < (int)g[x].size(); i++)
13            if (!dfs(g[x][i])) return false;
14        return true;
15    }
16
17    void init(int n) {
18        this->n = n;
19        for (int i = 0; i < n * 2; i++) g[i].clear();
20        memset(mark, 0, sizeof(mark));
21    }
22 }

```

```

23 void add_clause(int x, int y) { // 这个函数随题意变化
24     g[x].push_back(y ^ 1); // 选了 x 就必须选 y^1
25     g[y].push_back(x ^ 1);
26 }
27
28 bool solve() {
29     for (int i = 0; i < n * 2; i += 2)
30         if (!mark[i] && !mark[i + 1]) {
31             c = 0;
32             if (!dfs(i)) {
33                 while (c > 0) mark[s[--c]] = false;
34                 if (!dfs(i + 1)) return false;
35             }
36         }
37     return true;
38 }
39 };

```

## 1.8 虚树

## 1.9 树分治

### 1.9.1 点分治

```

1 // 给定一棵有 n 个点的带边权树, m 次询问, 每次询问给出 k, 询问树上距离为 k 的点对
  // 是否存在。
2 // n <= 10000, m <= 100, k <= 10000000
3
4 #include <cmath>
5 #include <queue>
6 #include <cstdio>
7 #include <bitset>
8 #include <cstring>
9 #include <iostream>
10 #include <algorithm>
11
12 #define endl '\n'
13
14 using namespace std;
15 using i64 = long long;
16
17 const int si = 3e5 + 10;
18 const int inf = 1e9 + 7;
19
20 int n, m, q[si];
21 int tot = 0, head[si];
22 struct Edge { int ver, Next, w; } e[si << 1];

```

```

23 inline void add(int u, int v, int w) { e[tot] = (Edge){v, head[u], w},
  head[u] = tot++; }
24
25 std::queue<int> rec;
26 bool tf[10000010], can[si], vis[si];
27 // tf: 当前子树的可行性。
28
29 int cnt = 0, sum = 0;
30 int maxv[si], rt = 0;
31 int d[si], dis[si], siz[si];
32 // d: 当前子树的 节点-根 距离。
33 void calcsiz(int u, int fa) {
34     siz[u] = 1, maxv[u] = 0;
35     for (int i = head[u]; ~i; i = e[i].Next) {
36         int v = e[i].ver;
37         if (v == fa || vis[v]) continue;
38         calcsiz(v, u);
39         maxv[u] = max(maxv[u], siz[v]), siz[u] += siz[v];
40     }
41     maxv[u] = max(maxv[u], sum - siz[u]); // 注意这里是当前子树的节点个数。
42     if (maxv[rt] > maxv[u]) rt = u;
43 }
44 void calcdis(int u, int fa) {
45     d[++cnt] = dis[u]; // 这里复制是为了枚举的时候不全部枚举, 保证复杂度。
46     for (int i = head[u]; ~i; i = e[i].Next) {
47         int v = e[i].ver, w = e[i].w;
48         if (v == fa || vis[v]) continue;
49         dis[v] = dis[u] + w, calcdis(v, u);
50     }
51 }
52 void dfs(int u, int fa) {
53     tf[0] = true, rec.push(0), vis[u] = true; // 打 vis 是为了确保在子树中
  // 进行操作, 不会递归出去。
54     // 或者不妨说, 我们是利用 vis, 将树划分成了一个联通块来处理, 因为它每次都只会
  // 标记到重心嘛。
55     for (int i = head[u]; ~i; i = e[i].Next) {
56         int v = e[i].ver, w = e[i].w;
57         if (v == fa || vis[v]) continue;
58         dis[v] = w, calcdis(v, u);
59         for (int j = 1; j <= cnt; ++j) {
60             for (int k = 1; k <= m; ++k) {
61                 if (q[k] >= d[j]) can[q[k]] |= tf[q[k] - d[j]];
62             }
63         } // 先判断再添加, 不然算的不是除了自己子树的情况, 这样会多算。
64         for (int j = 1; j <= cnt; ++j) {
65             if (d[j] < 10000010) rec.push(d[j]), tf[d[j]] = true;
66         }

```

```

67     cnt = 0;
68 }
69
70 while(!rec.empty()) tf[rec.front()] = false, rec.pop();
71 for(int i = head[u]; ~i; i = e[i].Next) {
72     int v = e[i].ver;
73     if(v == fa || vis[v]) continue;
74     rt = 0, maxv[rt] = inf, sum = siz[v];
75     calcsiz(v, u), calcsiz(rt, -1), dfs(rt, u); // 先找重心再递归。
76 }
77 }
78
79 int main() {
80
81     cin.tie(0) -> sync_with_stdio(false);
82     cin.exceptions(cin.failbit | cin.badbit);
83
84     memset(tf, false, sizeof tf);
85     memset(head, -1, sizeof head);
86     memset(vis, false, sizeof vis);
87     memset(can, false, sizeof can);
88
89     cin >> n >> m;
90     for(int i = 1; i < n; ++i) {
91         int u, v, w;
92         cin >> u >> v >> w;
93         add(u, v, w), add(v, u, w);
94     }
95     for(int nw = 1; nw <= m; ++nw) {
96         cin >> q[nw];
97     }
98
99     rt = 0, maxv[rt] = inf, sum = n;
100    calcsiz(1, -1), calcsiz(rt, -1), dfs(rt, -1); // 因为本题需要用到 tf
101    (0) 所以 fa 就用 -1 了。
102
103    for(int nw = 1; nw <= m; ++nw) {
104        if(can[q[nw]]) cout << "AYE" << endl;
105        else cout << "NAY" << endl;
106    }
107
108    return 0;
109 }

```

## 1.9.2 点分树

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  typedef vector<int>::iterator IT;
6
7  struct Edge {
8      int to, nxt, val;
9
10     Edge() {}
11
12     Edge(int to, int nxt, int val) : to(to), nxt(nxt), val(val) {}
13 } e[300010];
14
15 int head[150010], cnt;
16
17 void addedge(int u, int v, int val) {
18     e[++cnt] = Edge(v, head[u], val);
19     head[u] = cnt;
20 }
21
22 int siz[150010], son[150010];
23 bool vis[150010];
24
25 int tot, lasttot;
26 int maxp, root;
27
28 void getG(int now, int fa) {
29     siz[now] = 1;
30     son[now] = 0;
31     for (int i = head[now]; i; i = e[i].nxt) {
32         int vs = e[i].to;
33         if (vs == fa || vis[vs]) continue;
34         getG(vs, now);
35         siz[now] += siz[vs];
36         son[now] = max(son[now], siz[vs]);
37     }
38     son[now] = max(son[now], tot - siz[now]);
39     if (son[now] < maxp) {
40         maxp = son[now];
41         root = now;
42     }
43 }
44
45 struct Node {
46     int fa;
47     vector<int> anc;

```



```

48 vector<int> child;
49 } nd[150010];
50
51 int build(int now, int ntot) {
52     tot = ntot;
53     maxp = 0x7f7f7f7f;
54     getG(now, 0);
55     int g = root;
56     vis[g] = 1;
57     for (int i = head[g]; i; i = e[i].nxt) {
58         int vs = e[i].to;
59         if (vis[vs]) continue;
60         int tmp = build(vs, ntot - son[vs]);
61         nd[tmp].fa = now;
62         nd[now].child.push_back(tmp);
63     }
64     return g;
65 }
66
67 int virtroot;
68
69 int main() {
70     int n;
71     cin >> n;
72     for (int i = 1; i < n; i++) {
73         int u, v, val;
74         cin >> u >> v >> val;
75         addedge(u, v, val);
76         addedge(v, u, val);
77     }
78     virtroot = build(1, n);
79 }

```

```

1 int k, a[si];
2 int stk[si], top = 0;
3 bool cmp(int x, int y) { return dfn[x] < dfn[y]; }
4 inline void ADD(int u, int v, int w) { E[Tot] = (Edge){v, Head[u], w},
    Head[u] = Tot++; }
5 inline void Add(int u, int v) { int w = dist(u, v); ADD(u, v, w), ADD(
    v, u, w); }
6 void build() {
7     sort(a + 1, a + 1 + k, cmp);
8     stk[top = 1] = 1, Tot = 0, Head[1] = -1; // 这样清空复杂度才是对的。
9     for(int i = 1, Lca; i <= k; ++i) {
10         if(a[i] == 1) continue;
11         Lca = lca(a[i], stk[top]);
12         if(Lca != stk[top]) {

```

```

13         while(dfn[Lca] < dfn[stk[top - 1]])
14             Add(stk[top - 1], stk[top]), --top;
15         if(dfn[Lca] > dfn[stk[top - 1]])
16             Head[Lca] = -1, Add(Lca, stk[top]), stk[top] = Lca;
17         else Add(Lca, stk[top--]); // Lca = stk[top - 1].
18     }
19     Head[a[i]] = -1, stk[++top] = a[i];
20 }
21 for(int i = 1; i < top; ++i)
22     Add(stk[i], stk[i + 1]);
23 return;
24 }

```

## 1.10 异或最小生成树

```

1 /*
2 给定n个点的无向完全图,
3 给出每个点的点权ai,
4 两点之间的边权是两点权异或值,
5 求最小生成树
6 ai<2^30
7 复杂度o(nlogn)
8 */
9 #include <bits/stdc++.h>
10 using namespace std;
11 typedef long long ll;
12 const ll mod=998244353;
13
14 struct
15 {
16     int nxt[2];
17 }trie[8000005];
18 int cnt;
19
20 void Insert(int x)
21 {
22     int now=0;
23     for(int i=29;i>=0;i--)
24     {
25         int t=(x>>i)&1;
26         if(!trie[now].nxt[t])trie[now].nxt[t]=++cnt;
27         now=trie[now].nxt[t];
28     }
29 }
30 ll Find_MIN(int x)
31 {

```



```

32 ll ans=0;
33 int now=0;
34 for(int i=29;i>=0;i--)
35 {
36     int t=(x>>i)&1;
37     if(trie[now].nxt[t])
38     {
39         now=trie[now].nxt[t];
40     }
41     else
42     {
43         now=trie[now].nxt[t^1];
44         ans|=1<<i;
45     }
46 }
47 return ans;
48 }
49
50 int a[200005];
51 ll fenzhi(int *a,int l,int r,int id)
52 {
53     if(l>=r||id==-1)return 0;
54     int mid=r;
55     while(mid>=l&&((a[mid]>>id)&1))mid--;
56     ll ans=0;
57     ans+=fenzhi(a,l,mid,id-1);
58     ans+=fenzhi(a,mid+1,r,id-1);
59     if(mid==l-1||mid==r)return ans;
60     for(int i=l;i<=mid;i++)Insert(a[i]);
61     ll minn=1e18;
62     for(int i=mid+1;i<=r;i++)minn=min(minn,Find_MIN(a[i]));
63     ans+=minn;
64     for(int i=0;i<=cnt;i++)trie[i].nxt[0]=trie[i].nxt[1]=0;
65     cnt=0;
66     return ans;
67 }
68
69 int main()
70 {
71     int n;
72     scanf("%d",&n);
73     for(int i=1;i<=n;i++)scanf("%d",&a[i]);
74     sort(a+1,a+n+1);
75     ll ans=fenzhi(a,1,n,29); //29是位数
76     printf("%lld\n",ans);
77     return 0;
78 }

```

## 2 Data Structure

### 2.1 种类并查集

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int fa[150005];
4 int findd(int x)
5 {
6     return fa[x]==x?x:fa[x]=findd(fa[x]);
7 }
8 int main()
9 {
10     int n,k;
11     scanf("%d%d",&n,&k);
12     for(int i=1;i<=3*n;i++)fa[i]=i;
13     int ans=0;
14     for(int i=1;i<=k;i++)
15     {
16         int d,x,y;
17         scanf("%d%d%d",&d,&x,&y);
18         if(x>n||y>n){ans++;continue;}
19         if(d==1)
20         {
21             if(findd(x)==findd(y+n)||findd(x)==findd(y+2*n)){ans++;
22                 continue;}
23             fa[findd(y)]=findd(x);
24             fa[findd(y+n)]=findd(x+n);
25             fa[findd(y+2*n)]=findd(x+2*n);
26         }
27         else
28         {
29             if(findd(x)==findd(y+2*n)||findd(x)==findd(y)){ans++;continue;}
30             fa[findd(y+n)]=findd(x);
31             fa[findd(y+2*n)]=findd(x+n);
32             fa[findd(y)]=findd(x+2*n);
33         }
34     }
35     printf("%d\n",ans);
36     return 0;
37 }

```

### 2.2 二维树状数组

#### 2.2.1 单点修改, 区间查询

```

1 int tree[maxn][maxn];
2 int n,m;
3 int lowbit(int x){
4     return x&(-x);
5 }
6 void add(int x,int y,int w){
7     while(x<=n){
8         for(int i=y;i<=m;i+=lowbit(i))tree[x][i]+=w;
9         x+=lowbit(x);
10    }
11 }
12 int sum(int x,int y){
13     int res=0;
14     while(x>0){
15         for(int i=y;i>0;i-=lowbit(i)){
16             res+=tree[x][i];
17         }
18         x-=lowbit(x);
19     }
20     return res;
21 }
22 int query(int x1,int y1,int x2,int y2){ //左上右下
23     return sum(x2,y2)+sum(x1-1,y1-1)-sum(x2,y1-1)-sum(x1-1,y2);
24 }

```

### 2.2.2 区间修改，单点查询

```

1 void update(int x1,int y1,int x2,int y2,int w){ //左上右下
2     add(x1,y1,w);
3     add(x2+1,y1,-w);
4     add(x1,y2+1,-w);
5     add(x2+1,y2+1,w);
6 }
7 int query(int x,int y){
8     return sum(x,y);
9 }

```

### 2.2.3 区间修改，区间查询

```

1 int n,m,q;
2 ll t1[maxn][maxn],t2[maxn][maxn],t3[maxn][maxn],t4[maxn][maxn];
3 int lowbit(int x){
4     return x&(-x);
5 }

```

```

6 void add(int x,int y,int w){
7     for(int i=x;i<=n;i+=lowbit(i)){
8         for(int j=y;j<=m;j+=lowbit(j)){
9             t1[i][j]+=w;
10            t2[i][j]+=x*w;
11            t3[i][j]+=w*y;
12            t4[i][j]+=w*x*y;
13        }
14    }
15 }
16 void update(int x1,int y1,int x2,int y2,int w){ //左上右下
17     add(x1,y1,w);
18     add(x1,y2+1,-w);
19     add(x2+1,y1,-w);
20     add(x2+1,y2+1,w);
21 }
22 ll ask(int x,int y){
23     ll res=0;
24     for(int i=x;i>0;i-=lowbit(i)){
25         for(int j=y;j>0;j-=lowbit(j)){
26             res+=(x+1)*(y+1)*t1[i][j]-(y+1)*t2[i][j]
27             -(x+1)*t3[i][j]+t4[i][j];
28         }
29     }
30     return res;
31 }
32 ll query(int x1,int y1,int x2,int y2){ //左上右下
33     return ask(x2,y2)-ask(x2,y1-1)-ask(x1-1,y2)+ask(x1-1,y1-1);
34 }

```

## 2.3 线段树

### 2.3.1 维护等差数列

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 namespace Segtree
4 {
5     int n,m;
6     double tree[1005<<2],laz1[1005<<2],laz2[1005<<2];
7     void pushdown(int p,int l,int r)
8     {
9         int mid=(l+r)>>1;
10        double x=laz1[p],y=laz2[p];
11        laz1[p<<1]+=x;
12        laz2[p<<1]+=y;

```

```

13     laz1[p<<1|1]+=x+y*(mid-l+1);
14     laz2[p<<1|1]+=y;
15     tree[p<<1]+=(x+x+(mid-l)*y)*(mid-l+1)/2;
16     tree[p<<1|1]+=(x+y*(mid-l+1)+x+y*(mid-l+1)+(r-mid-1)*y)*(r-mid)
17         /2;
18     laz1[p]=laz2[p]=0;
19 }
20 void build(int p,int l,int r)
21 {
22     laz1[p]=laz2[p]=0;
23     if(l==r)tree[p]=n-1;
24     else
25     {
26         int mid=(l+r)>>1;
27         build(p<<1,l,mid);
28         build(p<<1|1,mid+1,r);
29         tree[p]=tree[p<<1]+tree[p<<1|1];
30     }
31 }
32 void update(int p,int l,int r,double v,double d,int x,int y)
33 {
34     if(x<=l&&r<=y)
35     {
36         double vv=v+(l-x)*d;
37         laz1[p]+=vv;
38         laz2[p]+=d;
39         tree[p]+=(vv+vv+d*(r-l))*(r-l+1)/2;
40     }
41     else
42     {
43         int mid=(l+r)>>1;
44         pushdown(p,l,r);
45         if(x<=mid)update(p<<1,l,mid,v,d,x,y);
46         if(mid<y)update(p<<1|1,mid+1,r,v,d,x,y);
47         tree[p]=tree[p<<1]+tree[p<<1|1];
48     }
49 }
50 double query(int p,int l,int r,int x,int y)
51 {
52     if(x<=l&&r<=y)return tree[p];
53     else
54     {
55         pushdown(p,l,r);
56         int mid=(l+r)>>1;
57         double ans=0;
58         if(x<=mid)ans+=query(p<<1,l,mid,x,y);
59         if(mid<y)ans+=query(p<<1|1,mid+1,r,x,y);

```

```

59         tree[p]=tree[p<<1]+tree[p<<1|1];
60         return ans;
61     }
62 }
63 int h[100005];
64 }
65 using namespace Segtree;
66 int main()
67 {
68     scanf("%d%d",&n,&m);
69     for(int i=0;i<n;i++)scanf("%d",&h[i]);
70     build(1,1,1000);
71     for(int i=0;i<n-1;i++)
72     {
73         int minn=min(h[i],h[i+1]);
74         int maxx=max(h[i],h[i+1]);
75         if(minn>=1)update(1,1,1000,-1,0,1,minn);
76         if(maxx>minn)update(1,1,1000,-(1.0-1.0/(maxx-minn)/2),1.0/(maxx-
77             minn),minn+1,maxx);
78     }
79     while(m--)
80     {
81         char op[10];
82         scanf("%s",op+1);
83         if(op[1]=='Q')
84         {
85             int H;
86             scanf("%d",&H);
87             if(H==0)printf("0.000\n");
88             else printf("%.3f\n",query(1,1,1000,1,H));
89         }
90         else
91         {
92             int id,H;
93             scanf("%d%d",&id,&H);
94             if(id>0)
95             {
96                 if(h[id]>h[id-1])
97                 {
98                     update(1,1,1000,1.0-1.0/(h[id]-h[id-1])/2,-1.0/(h[id]-h[
99                         id-1]),h[id-1]+1,h[id]);
100                 }
101                 else if(h[id]<h[id-1])
102                 {

```

```

103     if(H>h[id-1])
104     {
105         update(1,1,1000,-(1.0-1.0/(H-h[id-1])/2),1.0/(H-h[id
106             -1]),h[id-1]+1,H);
107     }
108     else if(H<h[id-1])
109     {
110         update(1,1,1000,1.0/(h[id-1]-H)/2,1.0/(h[id-1]-H),H+1,h
111             [id-1]);
112     }
113     if(id<n-1)
114     {
115         if(h[id]>h[id+1])
116         {
117             update(1,1,1000,1.0-1.0/(h[id]-h[id+1])/2,-1.0/(h[id]-h
118                 [id+1]),h[id+1]+1,h[id]);
119         }
120         else if(h[id]<h[id+1])
121         {
122             update(1,1,1000,-1.0/(h[id+1]-h[id])/2,-1.0/(h[id+1]-h[
123                 id]),h[id]+1,h[id+1]);
124         }
125         if(H>h[id+1])
126         {
127             update(1,1,1000,-(1.0-1.0/(H-h[id+1])/2),1.0/(H-h[id
128                 +1]),h[id+1]+1,H);
129         }
130         else if(H<h[id+1])
131         {
132             update(1,1,1000,1.0/(h[id+1]-H)/2,1.0/(h[id+1]-H),H+1,h
133                 [id+1]);
134         }
135     }
136     h[id]=H;
137 }
138 return 0;
139 }

```

### 2.3.2 有加有乘

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 long long mod;
4 namespace Segtree

```

```

5 {
6     int a[100005];
7     long long tree[100005<<2],laz1[100005<<2],laz2[100005<<2];
8     void build(int p,int l,int r)
9     {
10         laz1[p]=0,laz2[p]=1;
11         if(l==r)tree[p]=a[l]%mod;
12         else
13         {
14             int mid=(l+r)>>1;
15             build(p<<1,l,mid);
16             build(p<<1|1,mid+1,r);
17             tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
18         }
19     }
20     void pushdown(int p,int l,int r)
21     {
22         int mid=(l+r)>>1;
23         long long x=laz1[p],y=laz2[p];
24         tree[p<<1]=(tree[p<<1]*y%mod+x*(mid-l+1)%mod)%mod;
25         tree[p<<1|1]=(tree[p<<1|1]*y%mod+x*(r-mid)%mod)%mod;
26         laz1[p<<1]=(laz1[p<<1]*y%mod+x)%mod;
27         laz1[p<<1|1]=(laz1[p<<1|1]*y%mod+x)%mod;
28         laz2[p<<1]=laz2[p<<1]*y%mod;
29         laz2[p<<1|1]=laz2[p<<1|1]*y%mod;
30         laz1[p]=0,laz2[p]=1;
31     }
32     void update(int p,int l,int r,long long w,int x,int y,int op)
33     {
34         if(op==2)//加
35         {
36             if(x<=l&&r<=y)
37             {
38                 tree[p]=(tree[p]+w*(r-l+1)%mod)%mod;
39                 laz1[p]=(laz1[p]+w)%mod;
40             }
41             else
42             {
43                 pushdown(p,l,r);
44                 int mid=(l+r)>>1;
45                 if(x<=mid)update(p<<1,l,mid,w,x,y,op);
46                 if(mid<y)update(p<<1|1,mid+1,r,w,x,y,op);
47                 tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
48             }
49         }
50         else//乘
51         {

```

```

52     if(x<=l&&r<=y)
53     {
54         tree[p]=tree[p]*w%mod;
55         laz1[p]=laz1[p]*w%mod;
56         laz2[p]=laz2[p]*w%mod;
57     }
58     else
59     {
60         pushdown(p,l,r);
61         int mid=(l+r)>>1;
62         if(x<=mid)update(p<<1,l,mid,w,x,y,op);
63         if(mid<y)update(p<<1|1,mid+1,r,w,x,y,op);
64         tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
65     }
66 }
67 }
68 long long query(int p,int l,int r,int x,int y)
69 {
70     if(x<=l&&r<=y)return tree[p]%mod;
71     else
72     {
73         pushdown(p,l,r);
74         long long ans=0;
75         int mid=(l+r)>>1;
76         if(x<=mid)ans=(ans+query(p<<1,l,mid,x,y))%mod;
77         if(mid<y)ans=(ans+query(p<<1|1,mid+1,r,x,y))%mod;
78         tree[p]=(tree[p<<1]+tree[p<<1|1])%mod;
79         return ans;
80     }
81 }
82 }
83 using namespace Segtree;
84 int main()
85 {
86     int n,m;
87     scanf("%d%d%lld",&n,&m,&mod);
88     for(int i=1;i<=n;i++)scanf("%d",&a[i]);
89     build(1,1,n);
90     while(m--)
91     {
92         int op;
93         scanf("%d",&op);
94         if(op==1||op==2)
95         {
96             int l,r;long long w;
97             scanf("%d%d%lld",&l,&r,&w);
98             update(1,1,n,w,l,r,op);

```

```

99     }
100     else
101     {
102         int l,r;
103         scanf("%d%d",&l,&r);
104         printf("%lld\n",query(1,1,n,l,r));
105     }
106 }
107 return 0;
108 }

```

### 2.3.3 最大子段和

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  namespace Segtree
4  {
5      int n,m;
6      int a[500005];
7      struct node
8      {
9          int sum,max_l,max_r,maxx;
10         node operator+(const node& y)const
11         {
12             return {sum+y.sum,
13                     max(max_l,sum+y.max_l),
14                     max(max_r+y.sum,y.max_r),
15                     max({maxx,y.maxx,max_r+y.max_l})};
16         }
17     }tree[500005<<2];
18     void build(int p,int l,int r)
19     {
20         if(l==r)tree[p]={a[l],a[l],a[l],a[l]};
21         else
22         {
23             int mid=(l+r)>>1;
24             build(p<<1,l,mid);
25             build(p<<1|1,mid+1,r);
26             tree[p]=tree[p<<1]+tree[p<<1|1];
27         }
28     }
29     void update(int p,int l,int r,int w,int x)
30     {
31         if(l==r)tree[p]={w,w,w,w};
32         else
33         {

```

```

34     int mid=(l+r)>>1;
35     if(x<=mid)update(p<<1,l,mid,w,x);
36     else update(p<<1|1,mid+1,r,w,x);
37     tree[p]=tree[p<<1]+tree[p<<1|1];
38 }
39 }
40 node query(int p,int l,int r,int x,int y)
41 {
42     if(x<=l&&r<=y)return tree[p];
43     else
44     {
45         node ans={0,(int)-1e9,(int)-1e9,(int)-1e9};
46         int mid=(l+r)>>1;
47         if(x<=mid)ans=ans+query(p<<1,l,mid,x,y);
48         if(mid<y)ans=ans+query(p<<1|1,mid+1,r,x,y);
49         return ans;
50     }
51 }
52 }
53 using namespace Segtree;
54 int main()
55 {
56     scanf("%d%d",&n,&m);
57     for(int i=1;i<=n;i++)
58         scanf("%d",&a[i]);
59     build(1,1,n);
60     while(m--)
61     {
62         int op,l,r;
63         scanf("%d%d%d",&op,&l,&r);
64         if(op==1)
65         {
66             if(l>r)swap(l,r);
67             printf("%d\n",query(1,1,n,l,r).maxx);
68         }
69         else update(1,1,n,r,l);
70     }
71     return 0;
72 }

```

## 2.4 非递归版本线段树

```

1 unsigned int bit_ceil(unsigned int n) {
2     unsigned int x = 1;
3     while (x < (unsigned int)(n)) x *= 2;
4     return x;

```

```

5 }
6
7 int countr_zero(unsigned int n) {
8     return __builtin_ctz(n);
9 }
10
11 template <class S,
12           S (*op)(S, S),
13           S (*e)(),
14           class F,
15           S (*mapping)(F, S),
16           F (*composition)(F, F),
17           F (*id)()>
18 struct lazy_segtree {
19     public:
20         lazy_segtree() : lazy_segtree(0) {}
21         explicit lazy_segtree(int n) : lazy_segtree(std::vector<S>(n, e())) {}
22         explicit lazy_segtree(const std::vector<S>& v) : _n(int(v.size()))
23         {
24             size = (int)bit_ceil((unsigned int)(_n));
25             log = countr_zero((unsigned int)size);
26             d = std::vector<S>(2 * size, e());
27             lz = std::vector<F>(size, id());
28             for (int i = 0; i < _n; i++) d[size + i] = v[i];
29             for (int i = size - 1; i >= 1; i--) {
30                 update(i);
31             }
32
33         void set(int p, S x) {
34             assert(0 <= p && p < _n);
35             p += size;
36             for (int i = log; i >= 1; i--) push(p >> i);
37             d[p] = x;
38             for (int i = 1; i <= log; i++) update(p >> i);
39         }
40
41         S get(int p) {
42             assert(0 <= p && p < _n);
43             p += size;
44             for (int i = log; i >= 1; i--) push(p >> i);
45             return d[p];
46         }
47
48         S prod(int l, int r) {
49             assert(0 <= l && l <= r && r <= _n);

```

```

50     if (l == r) return e();
51
52     l += size;
53     r += size;
54
55     for (int i = log; i >= 1; i--) {
56         if (((l >> i) << i) != l) push(l >> i);
57         if (((r >> i) << i) != r) push((r - 1) >> i);
58     }
59
60     S sm1 = e(), smr = e();
61     while (l < r) {
62         if (l & 1) sm1 = op(sm1, d[l++]);
63         if (r & 1) smr = op(d[--r], smr);
64         l >>= 1;
65         r >>= 1;
66     }
67
68     return op(sm1, smr);
69 }
70
71 S all_prod() { return d[1]; }
72
73 void apply(int p, F f) {
74     assert(0 <= p && p < _n);
75     p += size;
76     for (int i = log; i >= 1; i--) push(p >> i);
77     d[p] = mapping(f, d[p]);
78     for (int i = 1; i <= log; i++) update(p >> i);
79 }
80 void apply(int l, int r, F f) {
81     assert(0 <= l && l <= r && r <= _n);
82     if (l == r) return;
83
84     l += size;
85     r += size;
86
87     for (int i = log; i >= 1; i--) {
88         if (((l >> i) << i) != l) push(l >> i);
89         if (((r >> i) << i) != r) push((r - 1) >> i);
90     }
91
92     {
93         int l2 = l, r2 = r;
94         while (l < r) {
95             if (l & 1) all_apply(l++, f);
96             if (r & 1) all_apply(--r, f);

```

```

97         l >>= 1;
98         r >>= 1;
99     }
100     l = l2;
101     r = r2;
102 }
103
104 for (int i = 1; i <= log; i++) {
105     if (((l >> i) << i) != l) update(l >> i);
106     if (((r >> i) << i) != r) update((r - 1) >> i);
107 }
108
109 template <bool (*g)(S)> int max_right(int l) {
110     return max_right(l, [](S x) { return g(x); });
111 }
112 template <class G> int max_right(int l, G g) {
113     assert(0 <= l && l <= _n);
114     assert(g(e()));
115     if (l == _n) return _n;
116     l += size;
117     for (int i = log; i >= 1; i--) push(l >> i);
118     S sm = e();
119     do {
120         while (l % 2 == 0) l >>= 1;
121         if (!g(op(sm, d[l]))) {
122             while (l < size) {
123                 push(l);
124                 l = (2 * l);
125                 if (g(op(sm, d[l]))) {
126                     sm = op(sm, d[l]);
127                     l++;
128                 }
129             }
130             return l - size;
131         }
132         sm = op(sm, d[l]);
133         l++;
134     } while ((l & -l) != l);
135     return _n;
136 }
137
138 template <bool (*g)(S)> int min_left(int r) {
139     return min_left(r, [](S x) { return g(x); });
140 }
141 template <class G> int min_left(int r, G g) {
142     assert(0 <= r && r <= _n);

```

```

144 assert(g(e()));
145 if (r == 0) return 0;
146 r += size;
147 for (int i = log; i >= 1; i--) push((r - 1) >> i);
148 S sm = e();
149 do {
150     r--;
151     while (r > 1 && (r % 2)) r >>= 1;
152     if (!g(op(d[r], sm))) {
153         while (r < size) {
154             push(r);
155             r = (2 * r + 1);
156             if (g(op(d[r], sm))) {
157                 sm = op(d[r], sm);
158                 r--;
159             }
160         }
161         return r + 1 - size;
162     }
163     sm = op(d[r], sm);
164 } while ((r & -r) != r);
165 return 0;
166 }
167
168 private:
169 int _n, size, log;
170 std::vector<S> d;
171 std::vector<F> lz;
172
173 void update(int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
174 void all_apply(int k, F f) {
175     d[k] = mapping(f, d[k]);
176     if (k < size) lz[k] = composition(f, lz[k]);
177 }
178 void push(int k) {
179     all_apply(2 * k, lz[k]);
180     all_apply(2 * k + 1, lz[k]);
181     lz[k] = id();
182 }
183 };

```

#### 2.4.1 用法 1:

```

1 // operation 1: a[i] -> b*a[i]+c
2 // operation 2: \sum_{l<=i<r} a[i]
3

```

```

4 struct S
5 {
6     mint a;
7     int siz;
8 };
9
10 struct F
11 {
12     mint a,b;
13 };
14
15 S op(S l,S r){return S{l.a+r.a,l.siz+r.siz};}
16
17 S e(){return S{0,0};}
18
19 S mapping(F l,S r){return S{r.a*l.a+r.siz*l.b,r.siz};}
20
21 F composition(F l,F r){return F{r.a*l.a,r.b*l.a+l.b};}
22
23 F id(){return F{1,0};}
24
25 void solve()
26 {
27     int n,q;
28     cin>>n>>q;
29     vector<S>a(n);
30     for(int i=0;i<n;i++)
31     {
32         mint x;
33         cin>>x;
34         a[i]=S{x,1};
35     }
36
37     lazy_segtree<S,op,e,F,mapping,composition,id>seg(a);
38
39     for(int i=0;i<q;i++)
40     {
41         int t,l,r;
42         cin>>t>>l>>r;
43         if(t==0)
44         {
45             mint b,c;
46             cin>>b>>c;
47             seg.apply(l,r,F{b,c});
48         }
49         else
50         {

```



```

51     cout<<seg.prod(l,r).a<<endl;
52 }
53 }
54 }

```

#### 2.4.2 用法 2:

```

1  // a[1],a[2],... ,a[n] in [0,1]
2  // operation 1: forall i in [l,r], 0 -> 1, 1 -> 0
3  // operation 2: calc the inversion of [l,r]
4
5  struct S
6  {
7      ll zero,one,inversion;
8  };
9
10 struct F
11 {
12     bool f;
13 };
14
15 S op(S l,S r)
16 {
17     return S{l.zero+r.zero,l.one+r.one,l.inversion+r.inversion+l.one*r.
18         zero};
19 }
20 S e(){return S{0,0,0};}
21
22 S mapping(F l,S r)
23 {
24     if(!l.f)return r;
25     return S{r.one,r.zero,r.zero*r.one-r.inversion};
26 }
27
28 F composition(F l,F r){return F{(!l.f&& r.f)|| (l.f&&!r.f)};}
29
30 F id(){return F{false};}
31
32 void solve()
33 {
34     int n,q;
35     cin>>n>>q;
36     vector<S>a(n);
37     for(int i=0;i<n;i++)
38     {

```

```

39     int x;
40     cin>>x;
41     a[i]=S{x==0,x==1,0};
42 }
43
44 lazy_segtree<S,op,e,F,mapping,composition,id>seg(a);
45
46 for(int i=0;i<q;i++)
47 {
48     int t,l,r;
49     cin>>t>>l>>r;
50     if(t==1)
51     {
52         seg.apply(l-1,r,F{1});
53     }
54     else
55     {
56         cout<<seg.prod(l-1,r).inversion<<endl;
57     }
58 }
59 }

```

#### 2.5 动态开点线段树

```

1  int root=1;
2  template<typename T,int Sz,int Rng>
3  struct DynamicSegTree
4  {
5      int cnt=1;
6      T val[Sz];
7      int ls[Sz];
8      int rs[Sz];
9      void clear(){cnt=1;}
10     int newnode()
11     {
12         cnt++;
13         ls[cnt]=rs[cnt]=0;
14         val[cnt]=0;
15         return cnt;
16     }
17     void update(T w,int x,int y,int&p=root,int l=1,int r=Rng)
18     {
19         if(!p)p=newnode();
20         if(x<=l&&r<=y)
21         {
22             val[p]+=w;

```

```

23     }
24     else
25     {
26         int mid=(l+r)>>1;
27         if(x<=mid)update(w,x,y,ls[p],l,mid);
28         if(mid<y)update(w,x,y,rs[p],mid+1,r);
29     }
30 }
31 T query(int x,int p=root,int l=1,int r=Rng,T ans=0)
32 {
33     if(!p)return ans;
34     ans+=val[p];
35     if(l==r)return ans;
36     else
37     {
38         int mid=(l+r)>>1;
39         if(x<=mid)return query(x,ls[p],l,mid,ans);
40         else return query(x,rs[p],mid+1,r,ans);
41     }
42 }
43 };
44
45 DynamicSegTree<int,40000005,1000000000>seg;

```

## 2.6 Segment Tree Beats

### 2.6.1 区间取 min 、 max $O(n\log n)$

```

1  /*
2  1s
3  长度 n, 询问 m 都为 1e6
4  两种操作:
5  1 x y——将第 x 个数变为 y;
6  2 y——将所有小于 y 的数修改为 y;
7  */
8  #include<bits/stdc++.h>
9  using namespace std;
10
11 inline int qread(){
12     int s=0,w=1;char ch=getchar();
13     for(;!isdigit(ch);ch=getchar())if(ch=='-')w=-1;
14     for (;ch>='0' && ch<='9';ch=getchar())s=(s<<1)+(s<<3)+(ch^48);
15     return (w==-1?-s:s);}
16
17 int n,m;
18 int a[200050];

```

```

19 struct node{
20     int mi,smi,cnt,tag;
21 }tr[800050];
22 void pushup(int p){
23     if(tr[2*p].mi<tr[2*p+1].mi){
24         tr[p].mi=tr[2*p].mi;tr[p].cnt=tr[2*p].cnt;
25         tr[p].smi=min(tr[2*p+1].mi,tr[2*p].smi);
26     }else if(tr[2*p].mi>tr[2*p+1].mi){
27         tr[p].mi=tr[2*p+1].mi;tr[p].cnt=tr[2*p+1].cnt;
28         tr[p].smi=min(tr[2*p].mi,tr[2*p+1].smi);
29     }else {
30         tr[p].mi=tr[2*p].mi;tr[p].cnt=tr[2*p].cnt+tr[2*p+1].cnt;
31         tr[p].smi=min(tr[2*p].smi,tr[2*p+1].smi);
32     }
33 }
34 void build(int p,int l,int r){
35     tr[p].tag=-1;
36     if(l==r){
37         tr[p].mi=a[l];
38         tr[p].smi=1e9+7;
39         tr[p].cnt=1;
40         return;
41     }
42     int mid=l+r>>1;
43     build(2*p,l,mid);
44     build(2*p+1,mid+1,r);
45     pushup(p);
46 }
47 void pushdown(int p){
48     if(tr[p].tag==-1)return ;
49     if(tr[p].tag>tr[2*p].mi){
50         tr[2*p].mi=tr[2*p].tag=tr[p].tag;
51     }
52     if(tr[p].tag>tr[2*p+1].mi){
53         tr[2*p+1].mi=tr[2*p+1].tag=tr[p].tag;
54     }
55     tr[p].tag=-1;
56 }
57 void update(int p,int l,int r,int w){
58     if(tr[p].mi>=w)return;
59     if(tr[p].smi>w){
60         tr[p].mi=tr[p].tag=w;
61         return;
62     }
63     pushdown(p);
64     int mid=l+r>>1;
65     update(2*p,l,mid,w);update(2*p+1,mid+1,r,w);

```

```

66     pushup(p);
67 }
68 void update1(int p,int l,int r,int x,int w){
69     if(l==r){
70         tr[p].mi=w;
71         return;
72     }
73     int mid=l+r>>1;pushdown(p);
74     if(x<=mid)update1(2*p,l,mid,x,w);
75     else update1(2*p+1,mid+1,r,x,w);
76     pushup(p);
77 }
78 int query(int p,int l,int r,int x){
79     if(l==r)return tr[p].mi;
80     int mid=l+r>>1;
81     pushdown(p);
82     if(x<=mid)return query(2*p,l,mid,x);
83     else return query(2*p+1,mid+1,r,x);
84 }
85 int main()
86 {
87     n=qread();
88     for(int i=1;i<=n;i++){
89         a[i]=qread();
90     }
91     build(1,1,n);
92     m=qread();
93     for(int i=1;i<=m;i++){
94         int op;
95         op=qread();
96         if(op==1){
97             int a,b;
98             a=qread();b=qread();
99             update1(1,1,n,a,b);
100         }else {
101             int a;a=qread();
102             update(1,1,n,a);
103         }
104     }
105     for(int i=1;i<=n;i++){
106         printf("%d ",query(1,1,n,i));
107     }
108     return 0;
109 }

```

## 2.6.2 区间取 min 、 max + 区间加减 $O(n\log^2 n)$

```

1  /*
2  1s
3  数组长度n, 操作数m都为5e5;
4  五种操作:
5  1. 给一个区间 [L,R] 加上一个数 x
6  2. 把一个区间 [L,R] 里小于 x 的数变成 x
7  3. 把一个区间 [L,R] 里大于 x 的数变成 x
8  4. 求区间 [L,R] 的和
9  5. 求区间 [L,R] 的最大值
10 6. 求区间 [L,R] 的最小值
11
12 分别维护最小值、最大值、其他值的加标记;
13 区间加操作 就是把三个标记都加; 区间取 min 操作就是在最大值的加标记上修改; 取 max 同理。
14
15 两个注意点:
16 以最大值上的加减标记为例。下传这个标记时要判断子区间内是否包含最大值, 如果不包含则应下传其他值的加减标记;
17 如果一个区间的值域很小(只有 1 或 2 个数), 可能会发生一个值既是最大值又是次小值这种情况, 也就是发生了数域的重叠。这种情况要特判, 分辨到底该被哪个标记作用。
18 */
19 #include<bits/stdc++.h>
20 using namespace std;
21 #define ll long long
22 inline int qread(){
23     int s=0,w=1;char ch=getchar();
24     for(;!isdigit(ch);ch=getchar())if(ch=='-')w=-1;
25     for (;ch>='0'&&ch<='9';ch=getchar())s=(s<<1)+(s<<3)+(ch^48);
26     return (w==-1?-s:s);}
27
28 const int INF=2e9;
29 int n,m;
30 int s[500050];
31 struct node{
32     //最大值、次大值、最大值个数、最小值、次小值、最小值个数
33     int mx,smx,cmx,mi,smi,cmi;
34     ll sum;
35     //最小值、最大值、其他值的修改标记
36     int add1,add2,add3;
37 }tr[2000050];
38 #define ls (p<<1)
39 #define rs (p<<1|1)
40 void pushup(int p){
41     tr[p].sum=tr[ls].sum+tr[rs].sum;
42     if(tr[ls].mi==tr[rs].mi){

```

```

43     tr[p].mi=tr[ls].mi;
44     tr[p].smi=min(tr[ls].smi,tr[rs].smi);
45     tr[p].cmi=tr[ls].cmi+tr[rs].cmi;
46 }else if(tr[ls].mi<tr[rs].mi){
47     tr[p].mi=tr[ls].mi;
48     tr[p].smi=min(tr[ls].smi,tr[rs].mi);
49     tr[p].cmi=tr[ls].cmi;
50 }else {
51     tr[p].mi=tr[rs].mi;
52     tr[p].smi=min(tr[ls].mi,tr[rs].smi);
53     tr[p].cmi=tr[rs].cmi;
54 }
55 if(tr[ls].mx==tr[rs].mx){
56     tr[p].mx=tr[ls].mx;
57     tr[p].smx=max(tr[ls].smx,tr[rs].smx);
58     tr[p].cmx=tr[ls].cmx+tr[rs].cmx;
59 }else if(tr[ls].mx>tr[rs].mx){
60     tr[p].mx=tr[ls].mx;
61     tr[p].smx=max(tr[ls].smx,tr[rs].mx);
62     tr[p].cmx=tr[ls].cmx;
63 }else {
64     tr[p].mx=tr[rs].mx;
65     tr[p].smx=max(tr[ls].mx,tr[rs].smx);
66     tr[p].cmx=tr[rs].cmx;
67 }
68 }
69 //对tr[p]的修改, k1、k2、k3 分别为最小值、最大值、其他值的标记
70 void update(int p,int l,int r,int k1,int k2,int k3){
71     if(tr[p].mi==tr[p].mx){
72         //如果区间只有一种值, 应该只作用最大值、最小值的加减标记
73         if(k1==k3)k1=k2;
74         else k2=k1;
75         tr[p].sum+=1LL*k1*tr[p].cmi;
76     }else{
77         tr[p].sum+=1LL*k1*tr[p].cmi+1LL*k2*tr[p].cmx+1LL*k3*(r-l+1-tr[p].cmi-tr[p].cmx);
78     }
79     //次小值为最大值, 应被最大值标记作用
80     if(tr[p].smi==tr[p].mx)tr[p].smi+=k2;
81     else if(tr[p].smi!=INF)tr[p].smi+=k3; //否则被其他值的标记作用
82
83     if(tr[p].smx==tr[p].mi)tr[p].smx+=k1;
84     else if(tr[p].smx!=INF)tr[p].smx+=k3;
85
86     tr[p].mi+=k1;tr[p].mx+=k2;
87     tr[p].add1+=k1;tr[p].add2+=k2;tr[p].add3+=k3;
88 }

```

```

89 void pushdown(int p,int l,int r){
90     int mi=min(tr[ls].mi,tr[rs].mi);
91     int mx=max(tr[ls].mx,tr[rs].mx);
92     int mid=l+r>>1;
93     update(ls,l,mid,tr[ls].mi==mi?tr[p].add1:tr[p].add3,
94           tr[ls].mx==mx?tr[p].add2:tr[p].add3,tr[p].add3);
95     update(rs,mid+1,r,tr[rs].mi==mi?tr[p].add1:tr[p].add3,
96           tr[rs].mx==mx?tr[p].add2:tr[p].add3,tr[p].add3);
97
98     tr[p].add1=tr[p].add2=tr[p].add3=0;
99 }
100 void build(int p,int l,int r){
101     tr[p].add1=tr[p].add2=tr[p].add3=0;
102     if(l==r){
103         tr[p].sum=tr[p].mi=tr[p].mx=s[l];
104         tr[p].smi=INF;tr[p].smx=-INF;
105         tr[p].cmi=tr[p].cmx=1;
106         return;
107     }
108     int mid=l+r>>1;
109     build(ls,l,mid);
110     build(rs,mid+1,r);
111     pushup(p);
112 }
113 void update1(int p,int l,int r,int x,int y,int w){
114     if(x<=l&&r<=y){
115         update(p,l,r,w,w,w);
116         return;
117     }
118     int mid=l+r>>1;
119     pushdown(p,l,r);
120     if(x<=mid)update1(2*p,l,mid,x,y,w);
121     if(mid<y)update1(2*p+1,mid+1,r,x,y,w);
122     pushup(p);
123 }
124 void update2(int p,int l,int r,int x,int y,int w){
125     if(tr[p].mi>=w)return;
126     if(x<=l&&r<=y&&tr[p].smi>w){
127         update(p,l,r,w-tr[p].mi,0,0);
128         return;
129     }
130     pushdown(p,l,r);
131     int mid=l+r>>1;
132     if(x<=mid)update2(ls,l,mid,x,y,w);
133     if(mid<y)update2(rs,mid+1,r,x,y,w);
134     pushup(p);
135 }

```

```

136 void update3(int p,int l,int r,int x,int y,int w){
137     if(tr[p].mx<=w)return;
138     if(x<=l&&r<=y&&tr[p].smx<w){
139         update(p,l,r,0,w-tr[p].mx,0);
140         return;
141     }
142     int mid=l+r>>1;
143     pushdown(p,l,r);
144     if(x<=mid)update3(ls,l,mid,x,y,w);
145     if(mid<y)update3(rs,mid+1,r,x,y,w);
146     pushup(p);
147 }
148 ll query1(int p,int l,int r,int x,int y){
149     if(x<=l&&r<=y)return tr[p].sum;
150     int mid=l+r>>1;
151     pushdown(p,l,r);
152     ll ans=0;
153     if(x<=mid)ans+=query1(ls,l,mid,x,y);
154     if(mid<y)ans+=query1(rs,mid+1,r,x,y);
155     return ans;
156 }
157 int query2(int p,int l,int r,int x,int y){
158     if(x<=l&&r<=y)return tr[p].mx;
159     int mid=l+r>>1,ans=-INF;
160     pushdown(p,l,r);
161     if(x<=mid)ans=max(ans,query2(ls,l,mid,x,y));
162     if(mid<y)ans=max(ans,query2(rs,mid+1,r,x,y));
163     return ans;
164 }
165 int query3(int p,int l,int r,int x,int y){
166     if(x<=l&&r<=y)return tr[p].mi;
167     int mid=l+r>>1,ans=INF;
168     pushdown(p,l,r);
169     if(x<=mid)ans=min(ans,query3(ls,l,mid,x,y));
170     if(mid<y)ans=min(ans,query3(rs,mid+1,r,x,y));
171     return ans;
172 }
173 #undef ls
174 #undef rs
175 int main(){
176     n=qread();
177     for(int i=1;i<=n;i++)s[i]=qread();
178     build(1,1,n);
179     m=qread();
180     while(m--){
181         int op=qread(),l=qread(),r=qread();
182         int x;

```

```

183         if(op<=3)x=qread();
184         if(op==1)update1(1,1,n,l,r,x);
185         else if(op==2)update2(1,1,n,l,r,x);
186         else if(op==3)update3(1,1,n,l,r,x);
187         else if(op==4)printf("%lld\n",query1(1,1,n,l,r));
188         else if(op==5)printf("%d\n",query2(1,1,n,l,r));
189         else printf("%d\n",query3(1,1,n,l,r));
190     }
191     return 0;
192 }

```

## 2.7 可持久化线段树

### 2.7.1 静态版本，权值线段树

```

1 ll cnt=0,n,m,len; //cnt记录当前节点数目，len代表离散化后的区间大小；
2 ll a[200050],b[200050],root[200050];
3 struct node{
4     ll l,r,num;
5 }z[maxn];
6 ll clone(ll x){
7     cnt++;
8     z[cnt]=z[x];
9     return cnt;
10 }
11 void update(ll id1,ll &id2,ll l,ll r,ll x){
12     id2=clone(id1);z[id2].num++;
13     if(l>=r)return;
14     ll mid=(l+r)>>1;
15     if(x<=mid)update(z[id1].l,z[id2].l,l,mid,x);
16     else update(z[id1].r,z[id2].r,mid+1,r,x);
17 }
18 ll query(ll id1,ll id2,ll l,ll r,ll k){
19     if(l>=r)return l;
20     ll mid=(l+r)>>1;
21     ll num=z[z[id2].l].num-z[z[id1].l].num;
22     if(k<=num)return query(z[id1].l,z[id2].l,l,mid,k);
23     //这里的判断条件小心，容易出错！
24     //k<=num才往左子树走
25     else return query(z[id1].r,z[id2].r,mid+1,r,k-num);
26 }
27 int main(){
28     n=qread(),m=qread();
29     for(int i=1;i<=n;i++){
30         a[i]=qread();b[i]=a[i];
31     }

```

```

32 sort(b+1,b+1+n);
33 len=unique(b+1,b+1+n)-b-1;
34 for(int i=1;i<=n;i++){
35     ll x=lower_bound(b+1,b+1+len,a[i])-b;
36     update(root[i-1],root[i],1,len,x);
37 }
38 for(int i=1;i<=m;i++){
39     ll l=qread(),r=qread(),k=qread();
40     ll w=query(root[l-1],root[r],1,len,k);
41     printf("%lld\n",b[w]);
42 }
43 return 0;
44 }

```

### 2.7.2 动态版本，支持修改

```

1  ll n,m,cnt=0;
2  ll a[1000060],root[maxn];
3  struct node{
4      ll l,r,num;
5  }z[maxn];
6  ll clone(ll x){
7      cnt++;z[cnt]=z[x];
8      return cnt;
9  }
10 ll build(ll id,ll l,ll r){
11     id=++cnt;
12     if(l==r){
13         z[cnt].num=a[l];return cnt;
14     }
15     ll mid=(l+r)>>1;
16     z[id].l=build(z[id].l,l,mid);
17     z[id].r=build(z[id].r,mid+1,r);
18     return id;
19 }
20 void update(ll id1,ll &id2,ll l,ll r,ll x,ll w){
21     id2=clone(id1);
22     if(l>=r){
23         z[id2].num=w; return;
24     }
25     ll mid=(l+r)>>1;
26     if(x<=mid)update(z[id1].l,z[id2].l,l,mid,x,w);
27     else update(z[id1].r,z[id2].r,mid+1,r,x,w);
28 }
29 ll query(ll id,ll l,ll r,ll x){
30     if(l>=r)return z[id].num; //这里要return z[id].num;

```

```

31     ll mid=(l+r)>>1;
32     if(x<=mid)return query(z[id].l,l,mid,x);
33     else return query(z[id].r,mid+1,r,x);
34 }
35 int main(){
36     n=qread(),m=qread();
37     for(int i=1;i<=n;i++){
38         a[i]=qread();
39     }
40     root[0]=build(root[0],1,n);
41     for(int i=1;i<=m;i++){
42         ll q=qread(),p=qread(),w=qread();
43         if(p==1){
44             ll k=qread();
45             update(root[q],root[i],1,n,w,k);
46         }else{
47             root[i]=clone(root[q]);
48             printf("%lld\n",query(root[q],1,n,w));
49         }
50     }
51     return 0;
52 }

```

## 2.8 线段树合并 & 分裂

```

1  // 给出一个可重集 a (编号为 1)，它支持以下操作：
2  // 0 p x y: 将可重集 p 中大于等于 x 且小于等于 y 的值移动到一个新的可重集中（新
   // 可重集编号为从 2 开始的正整数，是上一次产生的新可重集的编号+1）。
3  // 1 p t: 将可重集 t 中的数放入可重集 p，且清空可重集 t（数据保证在此后的操作中不
   // 会出现可重集 t）。
4  // 2 p x q: 在 p 这个可重集中加入 x 个数字 q。
5  // 3 p x y: 查询可重集 p 中大于等于 x 且小于等于 y 的值的个数。
6  // 4 p k: 查询在 p 这个可重集中第 k 小的数，不存在时输出 -1。
7
8  namespace Segtree
9  {
10     int n,m;
11     int ls[(n+1)*40],rs[(n+1)*40],root[n+1];
12     ll tree[(n+1)*40];
13     int cnt=0,top=1;
14     void update(int& p,int l,int r,ll w,int x)
15     {
16         if(!p)p=++cnt;
17         if(l==r)tree[p]+=w;
18         else
19             {

```

```

20     int mid=(l+r)>>1;
21     if(x<=mid)update(ls[p],l,mid,w,x);
22     else update(rs[p],mid+1,r,w,x);
23     tree[p]=tree[ls[p]]+tree[rs[p]];
24 }
25 }
26 int merge(int a,int b,int l,int r)
27 {
28     if(!a||!b)return a+b;
29     else if(l==r)
30     {
31         tree[a]+=tree[b];
32         return a;
33     }
34     else
35     {
36         int mid=(l+r)>>1;
37         ls[a]=merge(ls[a],ls[b],l,mid);
38         rs[a]=merge(rs[a],rs[b],mid+1,r);
39         tree[a]=tree[ls[a]]+tree[rs[a]];
40         return a;
41     }
42 }
43 void split(int& p,int& q,int l,int r,int x,int y)
44 {
45     if(x<=l&&r<=y)
46     {
47         p=q;
48         q=0;
49     }
50     else
51     {
52         if(!p)p=++cnt;
53         int mid=(l+r)>>1;
54         if(x<=mid)split(ls[p],ls[q],l,mid,x,y);
55         if(mid<y)split(rs[p],rs[q],mid+1,r,x,y);
56         tree[p]=tree[ls[p]]+tree[rs[p]];
57         tree[q]=tree[ls[q]]+tree[rs[q]];
58     }
59 }
60 ll query1(int p,int l,int r,int x,int y)
61 {
62     if(x<=l&&r<=y)return tree[p];
63     else
64     {
65         int mid=(l+r)>>1;
66         ll ans=0;

```

```

67         if(x<=mid)ans+=query1(ls[p],l,mid,x,y);
68         if(mid<y)ans+=query1(rs[p],mid+1,r,x,y);
69         return ans;
70     }
71 }
72 int query2(int p,int l,int r,ll x)
73 {
74     if(l==r)return l;
75     int mid=(l+r)>>1;
76     if(tree[ls[p]]>=x)return query2(ls[p],l,mid,x);
77     else if(tree[rs[p]]>=x-tree[ls[p]])return query2(rs[p],mid+1,r,x-tree[ls[p]]);
78     else return -1;
79 }
80 }
81 using namespace Segtree;
82 void solve()
83 {
84     scanf("%d%d",&n,&m);
85     for(int i=1;i<=n;i++)
86     {
87         int x;
88         scanf("%d",&x);
89         update(root[top],1,n,x,i);
90     }
91     for(int i=1;i<=m;i++)
92     {
93         int op;
94         scanf("%d",&op);
95         if(op==0)
96         {
97             int p,x,y;
98             scanf("%d%d%d",&p,&x,&y);
99             split(root[++top],root[p],1,n,x,y);
100         }
101         else if(op==1)
102         {
103             int p,t;
104             scanf("%d%d",&p,&t);
105             root[p]=merge(root[p],root[t],1,n);
106             root[t]=0;
107         }
108         else if(op==2)
109         {
110             int p,x,q;
111             scanf("%d%d%d",&p,&x,&q);
112             update(root[p],1,n,x,q);

```



```

113 }
114 else if(op==3)
115 {
116     int p,x,y;
117     scanf("%d%d%d",&p,&x,&y);
118     printf("%lld\n",query1(root[p],1,n,x,y));
119 }
120 else if(op==4)
121 {
122     int p,k;
123     scanf("%d%d",&p,&k);
124     printf("%d\n",query2(root[p],1,n,k));
125 }
126 // printf("query%d:\n",i);
127 // for(int j=1;j<=top;j++)
128 // {
129 //     printf("%d:",j);
130 //     for(int k=1;k<=n;k++)printf("%d%c",query1(root[j],1,n,k,
131 //         k)," \n"[k==n]);
132 // }
133 }

```

## 2.9 分块

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 const int N=1000005;
5 const int sqN=1005;
6
7 int n,sq;
8 int st[sqN]; //第i号块的第一个元素的下标
9 int ed[sqN]; //第i号块最后一个元素的下标
10 int id[N]; //元素i的分块编号
11
12 ll arr[N]; //零散数组
13 ll sum[N]; //分块和数组
14 ll laz[N]; //分块公共加数组
15
16 void init()
17 {
18     sq=(int)sqrt(n);
19     for(int i=1;i<=sq;i++)
20     {
21         st[i]=n/sq*(i-1)+1;

```

```

22         ed[i]=n/sq*i;
23     }
24     ed[sq]=n;
25
26     for(int i=1;i<=sq;i++)
27     {
28         for(int j=st[i];j<=ed[i];j++)
29         {
30             id[j]=i;
31             sum[i]+=arr[j];
32         }
33     }
34 }
35
36 void update(int l,int r,ll x)
37 {
38     if(id[l]==id[r])
39     {
40         for(int i=l;i<=r;i++)arr[i]+=x;
41         sum[id[l]]+=x*(r-l+1);
42     }
43     else
44     {
45         for(int i=l;i<=ed[id[l]];i++)arr[i]+=x;
46         sum[id[l]]+=x*(ed[id[l]]-l+1);
47         for(int i=st[id[r]];i<=r;i++)arr[i]+=x;
48         sum[id[r]]+=x*(r-st[id[r]]+1);
49         for(int i=id[l]+1;i<id[r];i++)laz[i]+=x;
50     }
51 }
52
53 ll query(int l,int r)
54 {
55     ll ans=0;
56     if(id[l]==id[r])
57     {
58         for(int i=l;i<=r;i++)ans+=arr[i];
59         ans+=laz[id[l]]*(r-l+1);
60     }
61     else
62     {
63         for(int i=l;i<=ed[id[l]];i++)ans+=arr[i];
64         ans+=laz[id[l]]*(ed[id[l]]-l+1);
65         for(int i=st[id[r]];i<=r;i++)ans+=arr[i];
66         ans+=laz[id[r]]*(r-st[id[r]]+1);
67         for(int i=id[l]+1;i<id[r];i++)
68             ans+=sum[i]+laz[i]*(ed[i]-st[i]+1);

```



```

69     }
70     return ans;
71 }
72
73 int main()
74 {
75     scanf("%d",&n);
76     for(int i=1;i<=n;i++)scanf("%d",&arr[i]);
77     init();
78     for(int i=1;i<=n;i++)
79     {
80         int op;
81         scanf("%d",&op);
82         if(op==0)
83         {
84             int l,r,c;
85             scanf("%d%d%d",&l,&r,&c);
86             update(l,r,c);
87         }
88         else
89         {
90             int l,r,c;
91             scanf("%d%d%d",&l,&r,&c);
92             printf("%lld\n",query(l,r)%(c+1));
93         }
94     }
95     return 0;
96 }

```

## 2.10 珂朵莉树

```

1  /*
2  例题:
3  操作1: 区间加x
4  操作2: 区间赋值x
5  操作3: 求区间第K大
6  操作4: 求区间的x幂和 模y;
7  */
8  #include<bits/stdc++.h>
9  using namespace std;
10 #define ll long long
11
12 struct node{
13     int l,r;
14     mutable ll val;
15     bool operator < (const node &ths)const{

```

```

16         return l<ths.l;
17     }
18     node(int L,int R,ll Val):l(L),r(R),val(Val){}
19     node(int L):l(L){}
20 };
21
22 set<node>s;
23 using si=set<node>::iterator;
24
25 si split(int pos){
26     si it=s.lower_bound(node(pos));
27     if(it!=s.end()&&it->l==pos)return it;
28     --it;
29     int l=it->l,r=it->r;
30     ll val=it->val;
31     s.erase(it);
32     s.insert(node(l,pos-1,val));
33     return s.insert(node(pos,r,val)).first;
34 }
35 void assign(int l,int r,int val){
36     si itr=split(r+1),itl=split(l);
37     s.erase(itl,itr);
38     s.insert(node(l,r,val));
39 }
40 void add(int l,int r,ll val){
41     si itr=split(r+1),itl=split(l);
42     for(si it=itl;it!=itr;++it)
43         it->val+=val;
44 }
45 ll kth(int l,int r,int k){
46     si itr=split(r+1),itl=split(l);
47     vector<pair<ll,int>>v;
48     v.clear();
49     for(si it=itl;it!=itr;++it)
50         v.push_back(pair<ll,int>(it->val,it->r-it->l+1));
51     sort(v.begin(),v.end());
52     for(int i=0;i<v.size();i++){
53         k-=v[i].second;
54         if(k<=0)return v[i].first;
55     }
56     return -1;
57 }
58 ll query(int l,int r,int x,int y){
59     si itr=split(r+1),itl=split(l);
60     ll res=0;
61     for(si it=itl;it!=itr;++it)
62         res=(res+(it->r-it->l+1)*qp(it->val,x,y)%y)%y;

```

```

63     return res;
64 }
65 int n,m,vmax;
66 ll seed;
67 int rnd(){
68     int ret=(int)seed;
69     seed=(seed*7+13)%1000000007;
70     return ret;
71 }
72 int main(){
73     scanf("%d%d%lld%d",&n,&m,&seed,&vmax);
74     for(int i=1;i<=n;i++){
75         int a=rnd()%vmax+1;
76         s.insert(node(i,i,a));
77     }
78     s.insert(node(n+1,n+1,0));
79     for(int i=1;i<=m;i++){
80         int l,r,x,y;
81         int op=rnd()%4+1;
82         l=rnd()%n+1,r=rnd()%n+1;
83         if(l>r)swap(l,r);
84         if(op==3)x=rnd()%(r-l+1)+1;
85         else x=rnd()%vmax+1;
86         if(op==4)y=rnd()%vmax+1;
87         if(op==1)add(l,r,x);
88         else if(op==2)assign(l,r,x);
89         else if(op==3)printf("%lld\n",kth(l,r,x));
90         else printf("%lld\n",query(l,r,x,y));
91     }
92 }

```

## 2.11 莫队

### 2.11.1 普通莫队

```

1 struct Query
2 {
3     int l,r,id,pos;
4     bool operator<(const Query &x)const
5     {
6         return (pos^x.pos)?pos<x.pos:(pos&1)?r<x.r:r>x.r;
7     }
8 }a[100005];
9 int b[100005],n,m;
10 ll cnt[100005],Ans[100005];
11

```

```

12 void solve()
13 {
14     read(n,m);
15     int siz=n/sqrt(m);
16     if(!siz)siz++;
17     for(int i=1;i<=n;i++)read(b[i]);
18     for(int i=1;i<=m;i++)
19     {
20         read(a[i].l,a[i].r);
21         a[i].pos=(a[i].l-1)/siz+1;
22         a[i].id=i;
23     }
24     sort(a+1,a+m+1);
25     int l=1,r=0;
26     int ans=0;
27     for(int i=1;i<=m;i++)
28     {
29         while(l>a[i].l)ans+=(++cnt[b[--l]]==2);
30         while(r<a[i].r)ans+=(++cnt[b[++r]]==2);
31         while(l<a[i].l)ans--(--cnt[b[l++]]=1);
32         while(r>a[i].r)ans--(--cnt[b[r--]]=1);
33         Ans[a[i].id]=ans;
34     }
35     for(int i=1;i<=m;i++)
36     {
37         if(Ans[i])println("No");
38         else println("Yes");
39     }
40 }

```

### 2.11.2 带修莫队

```

1 struct Query
2 {
3     int l,r,t,id,posl,posr;
4     bool operator<(const Query &x)const
5     {
6         return (posl^x.posl)?posl<x.posl:((posr^x.posr)?posr<x.posr:t<x.
7             t);
8     }
9 }a[200005];
10 struct Replace
11 {
12     int p,col;
13 }c[200005];
14 int b[200005],n,m;

```

```

14 int cnta,cntc;
15 ll cnt[1000005],Ans[200005];
16
17 void solve()
18 {
19     read(n,m);
20     int siz=pow(n,2.0/3.0);
21     for(int i=1;i<=n;i++)read(b[i]);
22     for(int i=1;i<=m;i++)
23     {
24         string s;int x,y;
25         read(s,x,y);
26         if(s=="Q")
27         {
28             a[++cnta].l=x;a[cnta].r=y;
29             a[cnta].t=cntc;a[cnta].id=cnta;
30             a[cnta].posl=(a[cnta].l-1)/siz+1;
31             a[cnta].posr=(a[cnta].r-1)/siz+1;
32         }
33         else{c[++cntc].p=x;c[cntc].col=y;}
34     }
35     sort(a+1,a+cnta+1);
36     int l=1,r=0,t=0;ll ans=0;
37     for(int i=1;i<=m;i++)
38     {
39         while(l<a[i].l)ans-=!cnt[b[l++]];
40         while(l>a[i].l)ans+=!cnt[b[--l]]++;
41         while(r<a[i].r)ans+=!cnt[b[++r]]++;
42         while(r>a[i].r)ans-=!cnt[b[r--]];
43         while(t<a[i].t)
44         {
45             t++;
46             if(a[i].l<=c[t].p&& c[t].p<=a[i].r)
47                 ans-=!cnt[b[c[t].p]]-!cnt[c[t].col]++;
48             swap(b[c[t].p],c[t].col);
49         }
50         while(t>a[i].t)
51         {
52             if(a[i].l<=c[t].p&& c[t].p<=a[i].r)
53                 ans-=!cnt[b[c[t].p]]-!cnt[c[t].col]++;
54             swap(b[c[t].p],c[t].col);
55             t--;
56         }
57         Ans[a[i].id]=ans;
58     }
59     for(int i=1;i<=cnta;i++)println(Ans[i]);
60 }

```

## 2.11.3 树上莫队

```

1 struct Query
2 {
3     int s,lca,l,r,id,pos;
4     bool operator<(const Query &x)const
5     {
6         return (pos^x.pos)?pos<x.pos:(pos&1)?r<x.r:r>x.r;
7     }
8 }a[100005];
9 int b[40005],d[40005],cntd,n,m;
10 int cnt[40005],vis[40005],Ans[100005];
11 vector<int>edge[40005];
12 int fa[40005][20],dep[40005];
13 int c[80005],tot,fir[40005],las[40005];
14 void dfs(int x,int father)
15 {
16     fa[x][0]=father;dep[x]=dep[father]+1;
17     for(int i=1;i<20;i++)fa[x][i]=fa[fa[x][i-1]][i-1];
18     c[++tot]=x;fir[x]=tot;
19     for(auto it:edge[x])if(it!=father)dfs(it,x);
20     c[++tot]=x;las[x]=tot;
21 }
22 int cal_lca(int x,int y)
23 {
24     if(dep[x]<dep[y])swap(x,y);
25     for(int i=19;i>=0;i--)if(dep[fa[x][i]]>=dep[y])x=fa[x][i];
26     if(x==y)return x;
27     for(int i=19;i>=0;i--)if(fa[x][i]!=fa[y][i])x=fa[x][i],y=fa[y][i];
28     return fa[x][0];
29 }
30
31 void solve()
32 {
33     read(n,m);
34     int siz=(int)sqrt(2*n);
35     for(int i=1;i<=n;i++)
36     {
37         read(b[i]);
38         d[++cntd]=b[i];
39     }
40     sort(d+1,d+cntd+1);
41     cntd=unique(d+1,d+cntd+1)-d-1;
42     for(int i=1;i<=n;i++)
43         b[i]=lower_bound(d+1,d+cntd+1,b[i])-d;
44     for(int i=1;i<=n;i++)
45     {

```

```

46     int x,y;read(x,y);
47     edge[x].push_back(y);
48     edge[y].push_back(x);
49 }
50 dfs(1,1);
51 for(int i=1;i<=m;i++)
52 {
53     int x,y;read(x,y);
54     if(fir[x]>fir[y])swap(x,y);a[i].s=x;
55     ((a[i].lca=cal_lca(x,y))==x)?a[i].l=fir[x]:a[i].l=las[x];
56     a[i].r=fir[y];a[i].pos=(a[i].l-1)/siz+1;a[i].id=i;
57 }
58 sort(a+1,a+m+1);
59 int l=1,r=0,ans=0;
60 for(int i=1;i<=m;i++)
61 {
62     while(l>a[i].l)(vis[c[--l]]?ans-=!cnt[b[c[l]]]:
63     ans+=!cnt[b[c[l]]]++),vis[c[l]]^=1;
64     while(l<a[i].l)(vis[c[l]]?ans-=!cnt[b[c[l]]]:
65     ans+=!cnt[b[c[l]]]++),vis[c[l]]^=1;
66     while(r<a[i].r)(vis[c[++r]]?ans-=!cnt[b[c[r]]]:
67     ans+=!cnt[b[c[r]]]++),vis[c[r]]^=1;
68     while(r>a[i].r)(vis[c[r]]?ans-=!cnt[b[c[r]]]:
69     ans+=!cnt[b[c[r]]]++),vis[c[r]]^=1;
70     Ans[a[i].id]=((a[i].lca==a[i].s)?ans:ans+
71     (vis[a[i].lca]?((cnt[b[a[i].lca]]==1)?-1:0):((cnt[b[a[i].lca]]==0)?1:0)));
72 }
73 for(int i=1;i<=m;i++)println(Ans[i]);
74 }

```

#### 2.11.4 回滚莫队

```

1 struct Query
2 {
3     int l,r,posl,posr,id;
4     bool operator<(const Query&x)const
5     {
6         return (posl^x.posl)?posl<x.posl:r<x.r;
7     }
8 }a[100005];
9 int b[100005],d[100005],cntd,n,m;
10 ll cnt[100005],_cnt[100005],Ans[100005];
11 int L[100005],R[100005];
12 void solve()

```

```

14 {
15     read(n,m);
16     int siz=(int)sqrt(n),T=n/siz;
17     for(int i=1;i<=n;i++)
18     {
19         read(b[i]);
20         d[++cntd]=b[i];
21     }
22     sort(d+1,d+cntd+1);
23     cntd=unique(d+1,d+cntd+1)-d-1;
24     for(int i=1;i<=n;i++)
25     b[i]=lower_bound(d+1,d+cntd+1,b[i])-d;
26     for(int i=1;i<=T;i++)
27     {
28         if(i*siz>n)break;
29         L[i]=(i-1)*siz+1;
30         R[i]=i*siz;
31     }
32     if(R[T]<n)T++,L[T]=R[T-1]+1,R[T]=n;
33     for(int i=1;i<=m;i++)
34     {
35         read(a[i].l,a[i].r);
36         a[i].posl=(a[i].l-1)/siz+1;
37         a[i].posr=(a[i].r-1)/siz+1;
38         a[i].id=i;
39     }
40     sort(a+1,a+m+1);
41     int l=1,r=0;
42     ll ans=0;
43     int laspos=0;
44     for(int i=1;i<=m;i++)
45     {
46         if(a[i].posl==a[i].posr)
47         {
48             for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]++;
49             ll tmp=0;
50             for(int j=a[i].l;j<=a[i].r;j++)
51             tmp=max(tmp,_cnt[b[j]]*d[b[j]]);
52             for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]--;
53             Ans[a[i].id]=tmp;
54         }
55         else
56         {
57             if(laspos^a[i].posl)
58             {
59                 while(l<R[a[i].posl]+1)cnt[b[l++]]--;
60                 while(l>R[a[i].posl]+1)cnt[b[--l]]++;

```

```

61         while(r>R[a[i].posl])cnt[b[r--]]--;
62         while(r<R[a[i].posl])
63             ans=0,laspos=a[i].posl;
64     }
65     while(r<a[i].r)ans=max(ans,++cnt[b[++r]]*d[b[r]]);
66     ll tmp=ans;
67     while(l>a[i].l)tmp=max(tmp,++cnt[b[--l]]*d[b[l]]);
68     while(l<R[a[i].posl]+1)cnt[b[l++]]--;
69     Ans[a[i].id]=tmp;
70 }
71 }
72 for(int i=1;i<=m;i++)println(Ans[i]);
73 }

```

```

1 struct Query
2 {
3     int l,r,posl,posr,id;
4     bool operator<(const Query&x)const
5     {
6         return (posl^x.posl)?posl<x.posl:r>x.r;
7     }
8 }a[200005];
9 int b[200005],n,m;
10 int cnt[200005],_cnt[200005],Ans[200005];
11 int L[200005],R[200005];
12
13 void solve()
14 {
15     read(n,m);
16     int siz=(int)sqrt(n),T=n/siz;
17     for(int i=1;i<=n;i++)read(b[i]);
18     for(int i=1;i<=T;i++)
19     {
20         if(i*siz>n)break;
21         L[i]=(i-1)*siz+1;
22         R[i]=i*siz;
23     }
24     if(R[T]<n)T++,L[T]=R[T-1]+1,R[T]=n;
25     for(int i=1;i<=m;i++)
26     {
27         read(a[i].l,a[i].r);
28         a[i].posl=(a[i].l-1)/siz+1;
29         a[i].posr=(a[i].r-1)/siz+1;
30         a[i].id=i;
31     }
32     sort(a+1,a+m+1);
33     int l=1,r=0;

```

```

34     int ans=0;
35     int laspos=0;
36     for(int i=1;i<=m;i++)
37     {
38         if(a[i].posl==a[i].posr)
39         {
40             for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]++;
41             ll tmp=0;
42             while(_cnt[tmp])tmp++;
43             for(int j=a[i].l;j<=a[i].r;j++)_cnt[b[j]]--;
44             Ans[a[i].id]=tmp;
45         }
46         else
47         {
48             if(laspos^a[i].posl)
49             {
50                 while(r<n)cnt[b[++r]]++;
51                 while(l>L[a[i].posl])cnt[b[--l]]++;
52                 ans=0;
53                 while(cnt[ans])ans++;
54                 while(l<L[a[i].posl])!--cnt[b[l]]?ans=min(ans,b[l++]):l++;
55                 laspos=a[i].posl;
56             }
57             while(r>a[i].r)!--cnt[b[r]]?ans=min(ans,b[r--]):r--;
58             int tmp=ans;
59             while(l<a[i].l)!--cnt[b[l]]?tmp=min(tmp,b[l++]):l++;
60             while(l>L[a[i].posl])cnt[b[--l]]++;
61             Ans[a[i].id]=tmp;
62         }
63     }
64     for(int i=1;i<=m;i++)println(Ans[i]);
65 }

```

## 2.12 CDQ 分治

### 2.12.1 二维

```

1 //求p[i].x<=p[j].x&& p[i].y<=p[j].y&&i!=j的二元组的个数(此题y已有序)
2 struct Point
3 {
4     int x,id;
5 }p[15005],tmp[15005];
6 int ans[15005],cnt[15005];
7
8 void CDQ(int l,int r)
9 {

```

```

10 if(l==r)return;
11 int mid=(l+r)>>1;
12 CDQ(l,mid);CDQ(mid+1,r);
13 int i=l,j=mid+1,pos=l;
14 while(i<=mid&&j<=r)
15 {
16     if(p[i].x<=p[j].x)tmp[pos++]=p[i++];
17     else{ans[p[j].id]+=i-l;tmp[pos++]=p[j++];}
18 }
19 while(i<=mid)tmp[pos++]=p[i++];
20 while(j<=r){ans[p[j].id]+=i-l;tmp[pos++]=p[j++];}
21 for(int k=l;k<=r;k++)p[k]=tmp[k];
22 }
23
24 void solve()
25 {
26     int n;
27     while(read(n))
28     {
29         for(int i=1;i<=n;i++)
30         {
31             ans[i]=cnt[i-1]=0;
32             int b;read(p[i].x,b);
33             p[i].id=i;
34         }
35         CDQ(1,n);
36         for(int i=1;i<=n;i++)cnt[ans[p[i].id]]++;
37         for(int i=0;i<=n;i++)println(cnt[i]);
38     }
39 }

```

### 2.12.2 三维

```

1 //求p[i].x<=p[j].x&&p[i].y<=p[j].y&&p[i].z<=p[j].z&&i!=j的二元组的个数
2 struct Point
3 {
4     int x,y,z,w,id;
5     bool operator<(const Point&p)const
6     {
7         if(x!=p.x)return x<p.x;
8         if(y!=p.y)return y<p.y;
9         return z<p.z;
10    }
11 }p[100005],tmp[100005];
12 int ans[100005],cnt[100005];
13 int tree[200005];

```

```

14 int lowbit(int x){return x&(-x);}
15 void update(int x,int w)
16 {
17     while(x<=200000)
18     {
19         tree[x]+=w;
20         x+=lowbit(x);
21     }
22 }
23 int query(int x)
24 {
25     int sum=0;
26     while(x)
27     {
28         sum+=tree[x];
29         x-=lowbit(x);
30     }
31     return sum;
32 }
33
34 void CDQ(int l,int r)
35 {
36     if(l==r)return;
37     int mid=(l+r)>>1;
38     CDQ(l,mid);CDQ(mid+1,r);
39     int i=l,j=mid+1,pos=l;
40     while(i<=mid&&j<=r)
41     {
42         if(p[i].y<=p[j].y)
43         {
44             update(p[i].z,p[i].w);
45             tmp[pos++]=p[i++];
46         }
47         else
48         {
49             ans[p[j].id]+=query(p[j].z);
50             tmp[pos++]=p[j++];
51         }
52     }
53     while(i<=mid)
54     {
55         update(p[i].z,p[i].w);
56         tmp[pos++]=p[i++];
57     }
58     while(j<=r)
59     {
60         ans[p[j].id]+=query(p[j].z);

```

```

61     tmp[pos++] = p[j++];
62 }
63 for(int k=l; k<=mid; k++) update(p[k].z, -p[k].w);
64 for(int k=l; k<=r; k++) p[k] = tmp[k];
65 }
66
67 void solve()
68 {
69     int n, k;
70     read(n, k);
71     for(int i=1; i<=n; i++)
72     {
73         read(p[i].x, p[i].y, p[i].z);
74         p[i].w = 1; p[i].id = i;
75     }
76     sort(p+1, p+n+1);
77     int tot = 1;
78     for(int i=2; i<=n; i++)
79     {
80         if(p[tot].x == p[i].x && p[tot].y == p[i].y && p[tot].z == p[i].z) p[tot].w
81             ++;
82         else p[++tot] = p[i];
83     }
84     CDQ(1, tot);
85     for(int i=1; i<=tot; i++) cnt[ans[p[i].id] + p[i].w - 1] += p[i].w;
86     for(int i=0; i<=n; i++) println(cnt[i]);
87 }

```

## 2.13 链式前向星

```

1  const int maxn = 200005;
2  struct EDGE
3  {
4      int next, to, w;
5  } edge[maxn << 1];
6  int head[maxn];
7  int cnt;
8  void add(int u, int v, int w)
9  {
10     edge[cnt].w = w;
11     edge[cnt].to = v;
12     edge[cnt].next = head[u];
13     head[u] = cnt++;
14 }
15
16 for(int i=head[x]; ~i; i=edge[i].next) //遍历

```

```

17 {
18
19 }

```

## 2.14 Kruskal 重构树

```

1  namespace Kruskal_Rebuild_Tree
2  {
3      struct EDGE
4      {
5          int a, b, c;
6          bool operator<(const EDGE& e) const { return c < e.c; }
7      };
8      vector<EDGE> e(m+1);
9      vector<int> edge[n+m+1];
10     int st[n+m+1], fa[n+m+1], dep[n+m+1];
11     int f[n+m+1][20];
12     int findd(int a)
13     {
14         return fa[a] == a ? a : fa[a] = findd(fa[a]);
15     };
16     void kruskal()
17     {
18         sort(e.begin()+1, e.end());
19         for(int i=1; i<=m; i++)
20         {
21             int u = findd(e[i].a), v = findd(e[i].b), w = e[i].c;
22             if(u != v)
23             {
24                 st[++n] = w;
25                 fa[n] = fa[u] = fa[v] = n;
26                 edge[n].push_back(u);
27                 edge[n].push_back(v);
28                 edge[u].push_back(n);
29                 edge[v].push_back(n);
30             }
31         }
32     }
33     void lca_init(int u, int father)
34     {
35         f[u][0] = father; dep[u] = dep[father] + 1;
36         for(int i=1; i<20; i++) f[u][i] = f[f[u][i-1]][i-1];
37         for(auto v: edge[u])
38         {
39             if(v == father) continue;
40             lca_init(v, u);

```

```

41     }
42 }
43 int cal_lca(int x,int y)
44 {
45     if(dep[x]<dep[y])swap(x,y);
46     for(int i=19;i>=0;i--)if(dep[f[x][i]]>=dep[y])x=f[x][i];
47     if(x==y)return x;
48     for(int i=19;i>=0;i--)if(f[x][i]!=f[y][i])x=f[x][i],y=f[y][i];
49     return f[x][0];
50 }
51 }
52 using namespace Kurskal_Rebuild_Tree;
53 void solve()
54 {
55     int n,m,q;
56     read(n,m,q);
57     for(int i=1;i<=m;i++)read(e[i].a,e[i].b,e[i].c);
58     for(int i=1;i<=n;i++)fa[i]=i;
59
60     kruskal();
61     lca_init(n,0);
62     for(int i=1;i<=q;i++)
63     {
64         int u,v,w;
65         read(u,v,w);
66         int lca=cal_lca(u,v);
67         if(st[lca]>w)println("Yes");
68         else println("No");
69     }
70 }

```

## 2.15 树链剖分

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int n,q;
4 long long a[100005];
5 vector<int>edge[100005];
6 int dep[100005],siz[100005],son[100005],fa[100005];
7 int id[100005],top[100005],tot=0;
8 void dfs(int x,int father)
9 {
10     siz[x]=1;
11     dep[x]=dep[father]+1;
12     int maxsize=0;
13     for(auto it:edge[x])if(it!=father)

```

```

14     {
15         fa[it]=x;
16         dfs(it,x);
17         siz[x]+=siz[it];
18         if(siz[it]>maxsize)
19         {
20             maxsize=siz[it];
21             son[x]=it;
22         }
23     }
24 }
25 long long k[100005];
26 void dfs1(int x,int father)
27 {
28     id[x]=++tot;
29     k[id[x]]=a[x];
30     if(son[x])
31     {
32         top[son[x]]=top[x];
33         dfs1(son[x],x);
34     }
35     for(auto it:edge[x])if(it!=father&&it!=son[x])
36     {
37         top[it]=it;
38         dfs1(it,x);
39     }
40 }
41 long long tree[100005<<2];
42 void build(int p,int l,int r)
43 {
44     if(l==r)
45     {
46         tree[p]=k[l];
47         return;
48     }
49     int mid=(l+r)>>1;
50     build(p<<1,l,mid);
51     build(p<<1|1,mid+1,r);
52     tree[p]=tree[p<<1]^tree[p<<1|1];
53 }
54 void update(int p,int l,int r,long long w,int x)
55 {
56     if(l==r)
57     {
58         tree[p]=w;
59         return;
60     }

```



```

61     int mid=(l+r)>>1;
62     if(x<=mid)update(p<<1,l,mid,w,x);
63     else update(p<<1|1,mid+1,r,w,x);
64     tree[p]=tree[p<<1]^tree[p<<1|1];
65 }
66 long long query(int p,int l,int r,int x,int y)
67 {
68     if(x<=l&&r<=y)return tree[p];
69     long long ans=0;
70     int mid=(l+r)>>1;
71     if(x<=mid)ans^=query(p<<1,l,mid,x,y);
72     if(mid<y)ans^=query(p<<1|1,mid+1,r,x,y);
73     return ans;
74 }
75 long long Query(int x,int y)
76 {
77     long long ans=0;
78     while(top[x]!=top[y])
79     {
80         if(dep[top[x]]<dep[top[y]])swap(x,y);
81         ans^=query(1,1,n,id[top[x]],id[x]);
82         x=fa[top[x]];
83     }
84     if(dep[x]<dep[y])swap(x,y);
85     ans^=query(1,1,n,id[y],id[x]);
86     return ans;
87 }
88 int main()
89 {
90     scanf("%d%d",&n,&q);
91     for(int i=1;i<=n;i++)
92         scanf("%lld",&a[i]);
93     for(int i=1;i<n;i++)
94     {
95         int u,v;
96         scanf("%d%d",&u,&v);
97         edge[u].push_back(v);
98         edge[v].push_back(u);
99     }
100     dep[1]=1;
101     dfs(1,0);
102     top[1]=1;
103     dfs1(1,0);
104     build(1,1,n);
105     while(q--)
106     {
107         long long op,l,r;

```

```

108         scanf("%lld%lld%lld",&op,&l,&r);
109         if(op==1)
110         {
111             update(1,1,n,r,id[l]);
112         }
113         else
114         {
115             printf("%lld\n",Query(l,r));
116         }
117     }
118     return 0;
119 }

```

## 2.16 ST 表

```

1  int LOG[2000005];
2  void init()
3  {
4      LOG[0]=-1;
5      for(int i=1;i<=2000000;i++)
6          LOG[i]=LOG[i/2]+1;
7  }
8
9  template<typename T,int N,int K,T(*F)(T&,T&)>
10 struct Sparse_Table
11 {
12     T st[K][N];
13     template<typename Iterator>
14     void build(Iterator bg,Iterator ed)
15     {
16         int now=0;
17         for(auto i=bg;i!=ed;i++) // 下标从 1 开始
18         {
19             st[0][++now]=(*i);
20         }
21         for(int k=1;k<K;k++)
22         {
23             for(int i=1;i+(1<<k)-1<=now;i++)
24             {
25                 st[k][i]=F(st[k-1][i],st[k-1][i+(1<<(k-1))]);
26             }
27         }
28     }
29     T query(int l,int r)
30     {
31         int k=LOG[r-l+1];

```

```

32     return F(st[k][l],st[k][r-(1<<k)+1]);
33 }
34 // 记得 init !!!
35 };
36
37 int Max(int &a,int &b)
38 {
39     if(a>=b)return a;
40     return b;
41 }
42
43 Sparse_Table<int,2000005,25,Max>st; // [(int &a,int &b){return max(a,
44     b);}
45
46 void solve()
47 {
48     init();
49     vector<int>a(20);
50     for(int i=0;i<20;i++)a[i]=i+1;
51     st.build(a.begin(),a.end()); // st.build(a+1,a+n+1);
52     cout<<st.query(3,10)<<endl;

```

## 2.17 DSU on Tree

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int n;
4 int a[100005];
5 vector<int>edge[100005];
6 int siz[100005],son[100005],rev[100005],dfn[100005],num;
7 void dfs(int u,int fa)
8 {
9     siz[u]=1;rev[dfn[u]=++num]=u;
10    for(int v:edge[u])if(v!=fa)
11    {
12        dfs(v,u);siz[u]+=siz[v];
13        if(siz[son[u]]<siz[v])son[u]=v;
14    }
15 }
16 long long ans[100005],res;
17 int cnt[100005],maxx;
18 inline void ins(int x)
19 {
20     cnt[x]++;
21     if(maxx<cnt[x])maxx=cnt[x],res=0;

```

```

22     if(maxx==cnt[x])res+=x;
23 }
24 void dsu(int u,int fa,bool kp)
25 {
26     for(int v:edge[u])if(v!=fa&&v!=son[u])dsu(v,u,0);
27     if(son[u])dsu(son[u],u,1);
28     ins(a[u]);
29     for(auto v:edge[u])if(v!=fa&&v!=son[u])
30     for(int j=dfn[v];j<=dfn[v]+siz[v]-1;j++)
31     ins(a[rev[j]]);
32     ans[u]=res;
33     if(!kp)
34     {
35         maxx=res=0;
36         for(int i=dfn[u];i<=dfn[u]+siz[u]-1;i++)
37             cnt[a[rev[i]]]=0;
38     }
39 }
40 int main()
41 {
42     read(n);
43     for(int i=1;i<=n;i++)read(a[i]);
44     for(int i=1;i<=n;i++)
45     {
46         int x,y;
47         read(x);read(y);
48         edge[x].push_back(y);
49         edge[y].push_back(x);
50     }
51     dfs(1,0);
52     dsu(1,0,1);
53     for(int i=1;i<=n;i++){print(ans[i]);putchar(' ');}
54     return 0;
55 }

```

## 2.18 线性基

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 const int N = 65;
6
7 int n;
8 ll b[N];
9 ll tmp[N];

```

```

10 bool flag;
11
12 void ins(ll x) //插入数字, 获取线性基
13 {
14     for(int i = 62; i >= 0; i --) {
15         if(x >> i & 1) {
16             if(!b[i]) {
17                 b[i] = x;
18                 return;
19             }
20             else x ^= b[i];
21         }
22     }
23     flag = true;
24 }
25
26 ll find_max() //找到能被表示出来的最大值
27 {
28     ll ans = 0;
29     for(int i = 62; i >= 0; i --) {
30         ans = max(ans, ans ^ b[i]);
31     }
32     return ans;
33 }
34
35 ll find_min() //找到能被表示出来的最小值
36 {
37     for(int i = 0; i <= 62; i ++){
38         if(b[i]) {
39             return b[i];
40         }
41     }
42 }
43
44 ll get_kth(ll k) //找到第k小的能被表示出来的数
45 {
46     ll res = 0;
47     int cnt = 0;
48     k -= flag;
49     if(!k) return 0;
50     for(int i = 0; i <= 62; i ++){
51         for(int j = i - 1; j >= 0; j --){
52             if(b[i] >> j & 1) b[i] ^= b[j];
53         }
54         if(b[i]) tmp[cnt++] = b[i];
55     }
56     if(k >= (1ll << cnt)) return -1;

```

```

57     for(int i = 0; i < cnt; i ++){
58         if(k >> i & 1) {
59             res ^= tmp[i];
60         }
61     }
62     return res;
63 }
64
65 bool find_x(ll x) //x是否能被线性基表示出来
66 {
67     for(int i = 62; i >= 0; i --){
68         if(x >> i & 1) {
69             if(!b[i]) return false;
70             x ^= b[i];
71         }
72     }
73     return true;
74 }
75
76 int main()
77 {
78     scanf("%d", &n);
79     for(int i = 1; i <= n; i ++){
80         ll x;
81         scanf("%lld", &x);
82         ins(x);
83     }
84     ll ans = find_max();
85     printf("%lld\n", ans);
86     return 0;
87 }

```

## 2.19 平衡树

### 2.19.1 替罪羊树

```

1 namespace Scapegoat_Tree
2 {
3     const double alpha=0.75;
4     int ls[100005],rs[100005];
5     int siz[100005],val[100005];
6     int fac[100005]/*实际大小*/,cnt[100005]/*存在标记*/;
7     int tot,root;
8     void newnode(int&now,int w)
9     {
10         now=++tot;

```

```

11     val[now]=w;
12     siz[now]=fac[now]=1;
13     cnt[now]=1;
14 }
15 int imbalance(int now)//判断是否平衡
16 {
17     if(max(siz[ls[now]],siz[rs[now]])>siz[now]*alpha
18     ||siz[now]-fac[now]>siz[now]*0.3)return 1;
19     return 0;
20 }
21 vector<int>v;
22 void ldr(int now)
23 {
24     if(!now)return;
25     ldr(ls[now]);
26     if(cnt[now])v.push_back(now);
27     ldr(rs[now]);
28 }
29 void lift(int l,int r,int &now)
30 {
31     if(l==r)
32     {
33         now=v[l];
34         ls[now]=rs[now]=0;
35         siz[now]=fac[now]=cnt[now];
36         return;
37     }
38     int mid=(l+r)>>1;
39     now=v[mid];
40     if(l<mid)lift(l,mid-1,ls[now]);
41     else ls[now]=0;
42     lift(mid+1,r,rs[now]);
43     siz[now]=siz[ls[now]]+siz[rs[now]]+cnt[now];
44     fac[now]=fac[ls[now]]+fac[rs[now]]+cnt[now];
45 }
46 void rebuild(int &now)//中序遍历之后拎起来
47 {
48     v.clear();
49     ldr(now);
50     if(v.empty())
51     {
52         now=0;
53         return;
54     }
55     lift(0,v.size()-1,now);
56 }
57 void update(int now,int end)

```

```

58 {
59     if(!now)return;
60     if(val[end]<val[now])update(ls[now],end);
61     else update(rs[now],end);
62     siz[now]=siz[ls[now]]+siz[rs[now]]+cnt[now];
63 }
64 void check(int &now,int end)
65 {
66     if(now==end)return;
67     if(imbalance(now))
68     {
69         rebuild(now);
70         update(root,now);
71         return;
72     }
73     if(val[end]<val[now])check(ls[now],end);
74     else check(rs[now],end);
75 }
76 void insert(int &now,int w)
77 {
78     if(!now)
79     {
80         newnode(now,w);
81         check(root,now);
82         return;
83     }
84     siz[now]++;
85     fac[now]++;
86     if(w<val[now])insert(ls[now],w);
87     else if(w==val[now])cnt[now]++;
88     else insert(rs[now],w);
89 }
90 void del(int now,int w)
91 {
92     if(cnt[now]&&val[now]==w)
93     {
94         cnt[now]--;
95         fac[now]--;
96         check(root,now);
97         return;
98     }
99     fac[now]--;
100     if(w<val[now])del(ls[now],w);
101     else if(w==val[now])cnt[now]--;
102     else del(rs[now],w);
103 }
104 int queryrnk(int w)

```

```

105 {
106     int now=root,rnk=1;
107     while(now)
108     {
109         if(w<val[now])now=ls[now];
110         else if(w==val[now])
111         {
112             rnk+=fac[ls[now]];
113             break;
114         }
115         else
116         {
117             rnk+=cnt[now]+fac[ls[now]];
118             now=rs[now];
119         }
120     }
121     return rnk;
122 }
123 int querynum(int rnk)
124 {
125     int now=root;
126     while(now)
127     {
128         if(cnt[now]&&fac[ls[now]]+1<=rnk&&fac[ls[now]]+cnt[now]+1>rnk
129             )break;
130         else if(fac[ls[now]]>=rnk)now=ls[now];
131         else
132         {
133             rnk-=fac[ls[now]]+cnt[now];
134             now=rs[now];
135         }
136     }
137     return val[now];
138 }
139 int pre(int w)
140 {
141     return querynum(queryrnk(w)-1);
142 }
143 int sub(int w)
144 {
145     return querynum(queryrnk(w+1));
146 }
147 using namespace Scapegoat_Tree;
148 void solve()
149 {
150

```

```

151 int n;
152 scanf("%d",&n);
153 for(int i=1;i<=n;i++)
154 {
155     int op,x;
156     scanf("%d%d",&op,&x);
157     if(op==1)insert(root,x);
158     else if(op==2)del(root,x);
159     else if(op==3)printf("%d\n",queryrnk(x));
160     else if(op==4)printf("%d\n",querynum(x));
161     else if(op==5)printf("%d\n",pre(x));
162     else if(op==6)printf("%d\n",sub(x));
163 }
164 }

```

### 2.19.2 FHQ\_Treap

```

1 //普通版本
2 namespace FHQ_Treap
3 {
4     int tot,root;
5     int ls[100005],rs[100005];
6     int siz[100005],val[100005],rnd[100005];
7     #include<random>
8     mt19937 rand(233);
9     inline int newnode(int w)
10     {
11         val[++tot]=w;
12         rnd[tot]=rand();
13         siz[tot]=1;
14         return tot;
15     }
16     inline void pushup(int now)
17     {
18         siz[now]=siz[ls[now]]+siz[rs[now]]+1;
19     }
20     void split(int now,int w,int &x,int &y)//按值分裂
21     {
22         if(!now)x=y=0;
23         else
24         {
25             if(val[now]<=w)
26             {
27                 x=now;
28                 split(rs[now],w,rs[now],y);
29             }

```

```

30     else
31     {
32         y=now;
33         split(ls[now],w,x,ls[now]);
34     }
35     pushup(now);
36 }
37
38 int merge(int x,int y)
39 {
40     if(!x||!y)return x+y;
41     if(rnd[x]>rnd[y])//>、>=、<、<=都可以
42     {
43         rs[x]=merge(rs[x],y);
44         pushup(x);
45         return x;
46     }
47     else
48     {
49         ls[y]=merge(x,ls[y]);
50         pushup(y);
51         return y;
52     }
53 }
54 int x,y,z;
55 inline void insert(int w)
56 {
57     split(root,w,x,y);
58     root=merge(merge(x,newnode(w)),y);
59 }
60 inline void del(int w)
61 {
62     split(root,w,x,z);
63     split(x,w-1,x,y);
64     y=merge(ls[y],rs[y]);
65     root=merge(merge(x,y),z);
66 }
67 inline int queryrnk(int w)
68 {
69     split(root,w-1,x,y);
70     int ans=siz[x]+1;
71     root=merge(x,y);
72     return ans;
73 }
74 inline int querynum(int rnk)
75 {
76     int now=root;

```

```

77     while(now)
78     {
79         if(siz[ls[now]]+1==rnk)break;
80         else if(siz[ls[now]]>=rnk)now=ls[now];
81         else
82         {
83             rnk-=siz[ls[now]]+1;
84             now=rs[now];
85         }
86     }
87     return val[now];
88 }
89 inline int pre(int w)
90 {
91     split(root,w-1,x,y);
92     int now=x;
93     while(rs[now])now=rs[now];
94     root=merge(x,y);
95     return val[now];
96 }
97 inline int sub(int w)
98 {
99     split(root,w,x,y);
100    int now=y;
101    while(ls[now])now=ls[now];
102    root=merge(x,y);
103    return val[now];
104 }
105 }
106 using namespace FHQ_Treap;
107
108 void solve()
109 {
110     int n;
111     scanf("%d",&n);
112     for(int i=1;i<=n;i++)
113     {
114         int op,w;
115         scanf("%d%d",&op,&w);
116         if(op==1)insert(w);
117         else if(op==2)del(w);
118         else if(op==3)printf("%d\n",queryrnk(w));
119         else if(op==4)printf("%d\n",querynum(w));
120         else if(op==5)printf("%d\n",pre(w));
121         else printf("%d\n",sub(w));
122     }
123 }

```

```

1 //区间翻转
2 namespace FHQ_Treap
3 {
4     int tot,root;
5     int ls[100005],rs[100005],laz[100005];
6     int siz[100005],val[100005],rnd[100005];
7     #include<random>
8     mt19937 rand(233);
9     inline int newnode(int w)
10    {
11        val[++tot]=w;
12        rnd[tot]=rand();
13        siz[tot]=1;
14        return tot;
15    }
16    inline void pushup(int now)
17    {
18        siz[now]=siz[ls[now]]+siz[rs[now]]+1;
19    }
20    inline void pushdown(int now)
21    {
22        swap(ls[now],rs[now]);
23        laz[ls[now]]^=1;
24        laz[rs[now]]^=1;
25        laz[now]=0;
26    }
27    void split(int now,int size,int &x,int &y)//按大小分裂
28    {
29        if(!now)x=y=0;
30        else
31        {
32            if(laz[now])pushdown(now);
33            if(siz[ls[now]]<size)
34            {
35                x=now;
36                split(rs[now],size-siz[ls[now]]-1,rs[now],y);
37            }
38            else
39            {
40                y=now;
41                split(ls[now],size,x,ls[now]);
42            }
43            pushup(now);
44        }
45    }
46    int merge(int x,int y)
47    {

```

```

48        if(!x||!y)return x+y;
49        if(rnd[x]>rnd[y])
50        {
51            if(laz[x])pushdown(x);
52            rs[x]=merge(rs[x],y);
53            pushup(x);
54            return x;
55        }
56        else
57        {
58            if(laz[y])pushdown(y);
59            ls[y]=merge(x,ls[y]);
60            pushup(y);
61            return y;
62        }
63    }
64    void reverse(int l,int r)
65    {
66        int x,y,z;
67        split(root,l-1,x,y);
68        split(y,r-l+1,y,z);
69        laz[y]^=1;
70        merge(merge(x,y),z);
71    }
72    void ldr(int now)//中序遍历
73    {
74        if(!now)return;
75        if(laz[now])pushdown(now);
76        ldr(ls[now]);
77        printf("%d ",val[now]);
78        ldr(rs[now]);
79    }
80 }
81 using namespace FHQ_Treap;
82
83 void solve()
84 {
85     int n,m;
86     scanf("%d%d",&n,&m);
87     for(int i=1;i<=n;i++)
88         root=merge(root,newnode(i));
89     for(int i=1;i<=m;i++)
90     {
91         int l,r;
92         scanf("%d%d",&l,&r);
93         reverse(l,r);
94     }

```

```

95   ldr(root);
96 }

```

### 2.19.3 Splay

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  #define ll long long
5  // #define int ll
6  constexpr int inf=0x3f3f3f3f;
7  constexpr ll INF=0x3f3f3f3f3f3f3f3f;
8
9  const int N = 100005;
10
11 struct Splay
12 {
13     int rt,tot,fa[N],ch[N][2],val[N],cnt[N],sz[N];
14     void maintain(int x){sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+cnt[x];}
15     bool get(int x){return x==ch[fa[x]][1];}
16     void clear(int x){ch[x][0]=ch[x][1]=fa[x]=val[x]=sz[x]=cnt[x]=0;}
17     void rotate(int x)
18     {
19         int y=fa[x],z=fa[y],chk=get(x);
20         ch[y][chk]=ch[x][chk^1];
21         if(ch[x][chk^1])fa[ch[x][chk^1]]=y;
22         ch[x][chk^1]=y;
23         fa[y]=x;
24         fa[x]=z;
25         if(z)ch[z][y==ch[z][1]]=x;
26         maintain(x);
27         maintain(y);
28     }
29     void splay(int x)
30     {
31         for(int f=fa[x];f=fa[x],f; rotate(x))
32             if(fa[f])rotate(get(x)==get(f)?f:x);
33         rt=x;
34     }
35     void ins(int k)//插入操作
36     {
37         if(!rt)
38         {
39             val[++tot]=k;
40             cnt[tot]++;
41             rt=tot;

```

```

42         maintain(rt);
43         return;
44     }
45     int cur=rt,f=0;
46     while(1)
47     {
48         if(val[cur]==k)
49         {
50             cnt[cur]++;
51             maintain(cur);
52             maintain(f);
53             splay(cur);
54             break;
55         }
56         f=cur;
57         cur=ch[cur][val[cur]<k];
58         if(!cur)
59         {
60             val[++tot]=k;
61             cnt[tot]++;
62             fa[tot]=f;
63             ch[f][val[f]<k]=tot;
64             maintain(tot);
65             maintain(f);
66             splay(tot);
67             break;
68         }
69     }
70 }
71 int rk(int k)//查询x的排名
72 {
73     int res=0,cur=rt;
74     while(1)
75     {
76         if(k<val[cur])
77         {
78             cur=ch[cur][0];
79         }
80         else
81         {
82             res+=sz[ch[cur][0]];
83             if(k==val[cur])
84             {
85                 splay(cur);
86                 return res+1;
87             }
88             res+=cnt[cur];

```



```

89         cur=ch[cur][1];
90     }
91 }
92 }
93
94 int kth(int k) // 查询排名 x 的数
95 {
96     int cur=rt;
97     while(1)
98     {
99         if(ch[cur][0]&&k<=sz[ch[cur][0]])
100         {
101             cur=ch[cur][0];
102         }
103         else
104         {
105             k-=cnt[cur]+sz[ch[cur][0]];
106             if(k<=0)
107             {
108                 splay(cur);
109                 return val[cur];
110             }
111             cur=ch[cur][1];
112         }
113     }
114 }
115
116 int pre() // 查询前驱
117 {
118     int cur=ch[rt][0];
119     if(!cur) return cur;
120     while(ch[cur][1]) cur=ch[cur][1];
121     splay(cur);
122     return cur;
123 }
124
125 int nxt() // 查询后继
126 {
127     int cur=ch[rt][1];
128     if(!cur) return cur;
129     while(ch[cur][0]) cur=ch[cur][0];
130     splay(cur);
131     return cur;
132 }
133
134 void del(int k) // 删除操作
135 {
136     rk(k);
137     if(cnt[rt]>1)
138     {

```

```

136         cnt[rt]--;
137         maintain(rt);
138         return;
139     }
140     if(!ch[rt][0]&&!ch[rt][1])
141     {
142         clear(rt);
143         rt=0;
144         return;
145     }
146     if(!ch[rt][0])
147     {
148         int cur=rt;
149         rt=ch[rt][1];
150         fa[rt]=0;
151         clear(cur);
152         return;
153     }
154     if(!ch[rt][1])
155     {
156         int cur=rt;
157         rt=ch[rt][0];
158         fa[rt]=0;
159         clear(cur);
160         return;
161     }
162     int cur=rt;
163     int x=pre();
164     fa[ch[cur][1]]=x;
165     ch[x][1]=ch[cur][1];
166     clear(cur);
167     maintain(rt);
168 }
169 }tree;
170
171
172 void solve()
173 {
174     int n;
175     scanf("%d",&n);
176     for(int i=1;i<=n;i++)
177     {
178         int op,x;
179         scanf("%d%d",&op,&x);
180         if(op==1) tree.ins(x);
181         else if(op==2) tree.del(x);
182         else if(op==3) printf("%d\n",tree.rk(x));

```

```

183     else if(op==4)printf("%d\n",tree.kth(x));
184     else if(op==5)
185     {
186         tree.ins(x);
187         printf("%d\n",tree.val[tree.pre()]);
188         tree.del(x);
189     }
190     else if(op==6)
191     {
192         tree.ins(x);
193         printf("%d\n",tree.val[tree.nxt()]);
194         tree.del(x);
195     }
196 }
197 }
198
199 signed main()
200 {
201     // ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
202
203     // int _;scanf("%d",&_);while(--)
204
205     solve();
206     return 0;
207 }
208

```

## 2.20 LCT

```

1 //0 x y 代表询问从x到y的路径上的点的权值的xor和。保证x到y是联通的。
2 //1 x y 代表连接x到y, 若x到y已经联通则无需连接。
3 //2 x y 代表删除边(x,y), 不保证边(x,y)存在。
4 //3 x y 代表将点x上的权值变成y。
5 #include<bits/stdc++.h>
6 using namespace std;
7
8 #define ll long long
9 // #define int ll
10 constexpr int inf=0x3f3f3f3f;
11 constexpr ll INF=0x3f3f3f3f3f3f3f3f;
12
13 constexpr int N=500005;
14
15 struct Link_Cut_Tree
16 {
17     #define get(x) (son[1][fa[x]]==x) //获取x是父亲的那个儿子

```

```

18     int son[2][N],fa[N],siz[N],val[N];
19     inline void pushup(int x)
20     {
21         siz[x]=siz[son[0][x]]^siz[son[1][x]]^val[x];
22     }
23     bool rev[N];
24     inline void pushdown(int x)
25     {
26         if(!rev[x])return;
27         rev[son[0][x]]^=1;rev[son[1][x]]^=1;
28         swap(son[0][x],son[1][x]);rev[x]=0;
29     }
30     inline bool isroot(int x) //判断x是否是所在树的根
31     {
32         return !(son[0][fa[x]]==x||son[1][fa[x]]==x);
33     }
34     inline void rotate(int x) //将x向上旋转一层的操作
35     {
36         int y=fa[x],z=fa[y];
37         if(!isroot(y))son[y==son[1][z]][z]=x;
38         bool is=(son[1][y]==x);
39         son[is][y]=son[!is][x];fa[son[!is][x]]=y;
40         son[!is][x]=y;fa[y]=x;fa[x]=z;pushup(y);pushup(x);
41     }
42     int stk[N],top;
43     inline void splay(int x) //通过和rotate操作联动实现把x转移到当前splay的根
44     {
45         stk[++top]=x;
46         for(int i=x;!isroot(i);i=fa[i])stk[++top]=fa[i];
47         while(top)pushdown(stk[top--]);
48         while(!isroot(x))
49         {
50             int y=fa[x],z=fa[y]; //cout<<x<<" "<<y<<" "<<z<<endl;
51             if(!isroot(y))
52             {
53                 if((son[1][y]==x)^(son[1][z]==y))rotate(x);
54                 else rotate(y);
55             }
56             rotate(x);
57         }
58     }
59     inline int access(int x) //把从根到x的所有点放在一条实链里,使根到x成为一条
60     { //实路径,并且在同一棵splay里
61         int p;
62         for(p=0;x;p=x,x=fa[x])
63             splay(x),son[1][x]=p,pushup(x);

```

```

64     return p;
65     //连续两次access操作时,第二次access操作的返回值等于这两个节点的LCA
66     //表示x到根的链所在的splay树的根.这个节点一定已经被旋转到了根节点,且父亲一定为空
67 }
68 void update(int x)//在access操作之后,递归地从上到下pushdown信息
69 {
70     if(!isroot(x))update(fa[x]);
71     pushdown(x);
72 }
73 inline void makeroot(int x)//使x点成为其所在树的根
74 {
75     access(x);splay(x);rev[x]^=1;
76 }
77 inline int find(int x)//找到x所在树的根节点编号
78 {
79     access(x);splay(x);
80     while(son[0][x])x=son[0][x];
81     return x;
82 }
83 inline void link(int x,int y)//把x,y两点间连一条边
84 {
85     if(find(x)==find(y))return;
86     makeroot(x);fa[x]=y;
87 }
88 inline void split(int x,int y)//提取出x,y间的路径,方便做区间操作,并以y为根
89 {
90     makeroot(x);access(y);splay(y);
91 }
92 inline void cut(int x,int y)//把x,y两点间边删掉
93 {
94     split(x,y);
95     if(son[0][y]==x&&son[1][x]==0)son[0][y]=fa[x]=0;
96 }
97 inline int lca(int x,int y)//求两点LCA
98 {
99     access(x);splay(x);
100    return access(y);
101 }
102 }T;
103
104 void solve()
105 {
106     int n,m;
107     scanf("%d%d",&n,&m);
108     for(int i=1;i<=n;i++)

```

```

109 {
110     scanf("%d",&T.val[i]);
111     T.siz[i]=T.val[i];
112 }
113 for(int i=1;i<=m;i++)
114 {
115     int op,x,y;
116     scanf("%d%d%d",&op,&x,&y);
117     if(op==0)T.split(x,y),printf("%d\n",T.siz[y]);
118     else if(op==1)T.link(x,y);
119     else if(op==2)T.cut(x,y);
120     else T.makeroot(x),T.val[x]=y,T.pushup(x);
121 }
122 }
123
124 signed main()
125 {
126     // ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
127
128
129     // int _;scanf("%d",&_);while(--)
130     solve();
131     return 0;
132 }
133

```

### 3 String

#### 3.1 KMP

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const ll mod = 223092870;
4 const int MAXN = 1000005;
5 char s1[MAXN],s2[MAXN];
6 int nex[MAXN];
7 int main()
8 {
9     scanf("%s%s",s1+1,s2+1);
10    int n=strlen(s1+1),m=strlen(s2+1);
11    nex[1]=0;
12    for(int i=2,j=0;i<=m;i++)
13    {
14        while(j>0&&s2[i]!=s2[j+1])j=nex[j];
15        if(s2[i]==s2[j+1])j++;
16        nex[i]=j;
17    }
18    for(int i=1,j=0;i<=n;i++)
19    {
20        while(j>0&&s1[i]!=s2[j+1])j=nex[j];
21        if(s1[i]==s2[j+1])j++;
22        if(j==m)printf("%d\n",i-m+1),j=nex[j];
23    }
24    for(int i=1;i<=m;i++)printf("%d ",nex[i]);
25    return 0;
26 }
```

#### 3.2 马拉车

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 const int N=100005;
5 int len[2*N+5];
6
7 void manacher(string& s)
8 {
9     int siz=s.size();
10    string str="#";
11    for(int i=0;i<siz;i++)
12        str=str+s[i]+"#";
13    cout<<str<<'\n';
```

```

14    siz=str.size();
15    for(int i=0,l=0,r=-1;i<siz;i++)
16    {
17        int k=(i>r)?1:min(len[l+r-i],r-i+1);
18        while(0<=i-k&&i+k<siz&&str[i-k]==str[i+k])
19        {
20            k++;
21        }
22        len[i]=k--;
23        if(i+k>r)
24        {
25            l=i-k;
26            r=i+k;
27        }
28    }
29 }
30
31 int main()
32 {
33     ios::sync_with_stdio(false);cin.tie(nullptr);
34
35     int n;
36     cin>>n;
37     string s;
38     cin>>s;
39     manacher(s);
40     for(int i=0;i<2*n+1;i++)
41         cout<<len[i]<<" \n"[i==2*n];
42
43     return 0;
44 }
```

#### 3.3 哈希

##### 3.3.1 一维

```

1 #include<bits/stdc++.h>
2 typedef unsigned long long ull;
3 using namespace std;
4 char t[1000005];
5 char s[1000005];
6 ull h[1000005],p[1000005];
7 ull h1[1000005];
8 ull get(int l,int r)
9 {
10     return h[r]-h[l-1]*p[r-l+1];
```

```

11 }
12 ull get1(int l, int r)
13 {
14     return h1[r]-h1[l-1]*p[r-l+1];
15 }
16 unordered_map<ull, bool> mp;
17 int main()
18 {
19     scanf("%s", t+1);
20     int len=strlen(t+1);
21     p[0]=1;
22     h[0]=0;
23     for(int i=1; i<=1000000; i++)
24         p[i]=p[i-1]*131;
25     for(int i=1; i<=len; i++)
26         h[i]=h[i-1]*131+t[i]-'a'+1;
27     mp[get1(1, len)]=1;
28     for(int j=1; j<len; j++)
29         mp[get1(1, j)+get1(j+1, len)*p[j]]=1;
30     int _;
31     scanf("%d", &_);
32     while (_--)
33     {
34         scanf("%s", s+1);
35         int l=strlen(s+1);
36         h1[0]=0;
37         for(int i=1; i<=l; i++)
38             h1[i]=h1[i-1]*131+s[i]-'a'+1;
39         int num=0;
40         for(int i=1; i<=l-len+1; i++)
41             if(mp[get1(i, i+len-1)]) num++;
42         printf("%d\n", num);
43     }
44     return 0;
45 }

```

### 3.3.2 二维

```

1 #include<bits/stdc++.h>
2 typedef unsigned int ui;
3 using namespace std;
4 ui h[1005][1005];
5 ui p1[1005], p2[1005];
6 ui get(int x1, int y1, int x2, int y2)
7 {

```

```

8     return h[x2][y2]-h[x2][y1-1]*p1[y2-y1+1]-h[x1-1][y2]*p2[x2-x1+1]+h[
        x1-1][y1-1]*p1[y2-y1+1]*p2[x2-x1+1];
9 }
10 unordered_map<ui, bool> mp;
11 int main()
12 {
13     char x;
14     p1[0]=1; p2[0]=1;
15     for(int i=1; i<=1000; i++)
16     {
17         p1[i]=p1[i-1]*131;
18         p2[i]=p2[i-1]*233;
19     }
20     int n, m, a, b;
21     scanf("%d%d%d%d", &n, &m, &a, &b);
22     getchar();
23     for(int j=0; j<=m; j++) h[0][j]=0;
24     for(int i=1; i<=n; i++, getchar())
25     {
26         h[i][0] = 0;
27         for(int j=1; j<=m; j++)
28         {
29             x=getchar();
30             h[i][j]=h[i][j-1]*131+x;
31         }
32     }
33     for(int i=1; i<=n; i++)
34         for(int j=1; j<=m; j++)
35             h[i][j]=h[i-1][j]*233;
36     for(int i=1; i<=n-a+1; i++)
37         for(int j=1; j<=m-b+1; j++)
38         {
39             ui z=get(i, j, i+a-1, j+b-1);
40             if(!mp.count(z)) mp[z]=1;
41         }
42     int _;
43     scanf("%d", &_); getchar();
44     while(_--)
45     {
46         for(int i=1; i<=a; i++, getchar())
47         {
48             h[i][0]=0;
49             for(int j=1; j<=b; j++)
50             {
51                 x=getchar();
52                 h[i][j]=h[i][j-1]*131+x;
53             }

```

```

54     }
55     for(int i=1;i<=a;i++)
56         h[i][b]+=h[i-1][b]*233;
57     if(mp.count(h[a][b]))printf("1\n");
58     else printf("0\n");
59 }
60 return 0;
61 }

```

### 3.4 Trie 树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int trie[500005][30];
4  int color[500005];
5  int k=1;
6  void insert(char *w)
7  {
8      int len=strlen(w);
9      int p=0;
10     for(int i=0;i<len;i++)
11     {
12         int c=w[i]-'a';
13         if(!trie[p][c])
14             trie[p][c]=k++;
15         p=trie[p][c];
16     }
17     color[p]=1;
18 }
19 int search(char *s)
20 {
21     int len=strlen(s);
22     int p=0;
23     for(int i=0;i<len;i++)
24     {
25         int c=s[i]-'a';
26         if(!trie[p][c])return 0;
27         p=trie[p][c];
28     }
29     if(color[p]==1){color[p]=2;return 1;}
30     else if(color[p]==2)return 2;
31     else return 0;
32 }
33 char str[55];
34 int main()
35 {

```

```

36     int n;
37     scanf("%d",&n);
38     for(int i=1;i<=n;i++)
39     {
40         scanf("%s",str);
41         insert(str);
42     }
43     int m;
44     scanf("%d",&m);
45     for(int i=1;i<=m;i++)
46     {
47         scanf("%s",str);
48         int res=search(str);
49         if(!res)printf("WRONG\n");
50         else if(res==1)printf("OK\n");
51         else printf("REPEAT\n");
52     }
53     return 0;
54 }

```

### 3.5 AC 自动机

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXN=2000005;
4  int trie[MAXN][26];
5  int ed[MAXN];
6  int fail[MAXN];
7  int mp[200005];
8  int tot=0;
9  int ans[200005];
10 int vis[200005];
11 int in[200005];
12 void insert(char *s,int num)
13 {
14     int len=strlen(s),p=0;
15     for(int i=0;i<len;i++)
16     {
17         int ch=s[i]-'a';
18         if(trie[p][ch]==0)trie[p][ch]=++tot;
19         p=trie[p][ch];
20     }
21     if(!ed[p])ed[p]=num;
22     mp[num]=ed[p];
23 }
24 void build()

```

```

25 {
26     queue<int>q;
27     memset(fail,0,sizeof(fail));
28     for(int i=0;i<26;i++)if(trie[0][i])q.push(trie[0][i]);
29     while(!q.empty())
30     {
31         int fa=q.front();
32         q.pop();
33         for(int i=0;i<26;i++)
34         {
35             if(trie[fa][i])
36             {
37                 fail[trie[fa][i]]=trie[fail[fa]][i];
38                 in[fail[trie[fa][i]]]++;
39                 q.push(trie[fa][i]);
40             }
41             else
42             {
43                 trie[fa][i]=trie[fail[fa]][i];
44             }
45         }
46     }
47 }
48 void TopologicalSort()
49 {
50     queue<int>q;
51     for(int i=1;i<=tot;i++)if(!in[i])q.push(i);
52     while(!q.empty())
53     {
54         int u=q.front();
55         q.pop();
56         vis[ed[u]]=ans[u];
57         int v=fail[u];
58         in[v]--;
59         ans[v]+=ans[u];
60         if(!in[v])q.push(v);
61     }
62 }
63 void query(char *s)
64 {
65     int len=strlen(s);
66     int p=0;
67     for(int i=0;i<len;i++)
68         p=trie[p][s[i]-'a'],ans[p]++;
69 }
70 char s[200005];
71 char str[2000005];

```

```

72 int main()
73 {
74     int n;
75     scanf("%d",&n);
76     tot=0;
77     for(int i=1;i<=n;i++)
78     {
79         scanf("%s",s);
80         insert(s,i);
81     }
82     build();
83     scanf("%s",str);
84     query(str);
85     TopologicalSort();
86     for(int i=1;i<=n;i++)printf("%d\n",vis[mp[i]]);
87     return 0;
88 }

```

### 3.6 回文自动机

```

1 // BZOJ 3676
2 // calc max(len(t)*cnt(t)) t为s回文子串, cnt(t)=t出现次数
3 #include<bits/stdc++.h>
4 using namespace std;
5 const int maxn = 3e5+100;
6 struct Palindromic_AutoMaton{
7     //basic
8     int s[maxn][{}],now[{}];
9     int nxt[maxn][26][{}],fail[maxn][{}],len[maxn][{}],last[{}],tot[{}];
10    // extension
11    int num[maxn][{}]; /*节点代表的所有回文串出现次数*/
12    void clear(){
13        //1节点: 奇数长度root 0节点: 偶数长度root
14        s[0]=len[1]=-1;
15        fail[0] = tot = now =1;
16        last = len[0]=0;
17        memset(nxt[0],0,sizeof nxt[0]);
18        memset(nxt[1],0,sizeof nxt[1]);
19    }
20    Palindromic_AutoMaton(){clear();}
21    int newnode(int ll){
22        tot++;
23        memset(nxt[tot],0,sizeof nxt[tot]);
24        fail[tot]=num[tot]=0;
25        len[tot]=ll;
26        return tot;

```

```

27 }
28 int get_fail(int x){
29     while (s[now-len[x]-2]!=s[now-1])x = fail[x];
30     return x;
31 }
32 void add(int ch){
33     s[now++] = ch;
34     int cur = get_fail(last);
35     if(!nxt[cur][ch]){
36         int tt = newnode(len[cur]+2);
37         fail[tt] = nxt[get_fail(fail[cur])][ch];
38         nxt[cur][ch] = tt;
39     }
40     last = nxt[cur][ch];num[last]++;
41 }
42 void build(){
43     //fail[i]<i, 拓扑更新可以单调扫描。
44     for (int i=tot;i>=2;i--){
45         num[fail[i]]+=num[i];
46     }
47     num[0]=num[1]=0;
48 }
49 void init(char* ss){
50     while (*ss){
51         add(*ss-'a');ss++;
52     }
53 }
54 void init(const string& str){
55     for (char i : str){
56         add(i-'a');
57     }
58 }
59 long long query();
60 }pam;
61 long long Palindromic_AutoMaton::query(){
62     long long ret =1;
63     for (int i=2;i<=tot;i++){
64         ret = max(ret,1LL*len[i]*num[i]);
65     }
66     return ret;
67 }
68 char s[maxn];
69 int main(){
70     scanf("%s",s);
71     pam.init(s);
72     pam.build();
73     printf("%lld\n",pam.query());

```

```

74     return 0;
75 }

```

### 3.7 广义回文自动机

```

1 //树上回文自动机，每个点到根的路径代表一个字符串。
2 //强制在线增加叶子，求最大回文后缀。
3 #include<bits/stdc++.h>
4 using namespace std;
5 const int mod = 998244353;
6 int lastans = 0;
7 const int maxn = 5e5+100;
8 int st[maxn][20];
9 int pos[maxn];
10 map<char,int> score;
11 char a[maxn];
12 int sum[maxn];
13 int get_anc(int x,int len){
14     while (len){
15         int bit = __builtin_ctz(len);
16         x = st[x][bit];
17         len ^= (1 << bit);
18     }
19     return x;
20 }
21 int Damage[maxn];
22 struct Palindromic_AutoMaton{
23     //basic
24     int now;
25     int nxt[maxn][26],fail[maxn],len[maxn],tot;
26     int damage[maxn];
27     int damage_sum[maxn];
28     int total[maxn];
29     int Link[maxn][26];
30     // extension
31     int num[maxn];/*节点代表的所有回文串出现次数*/
32     void clear(){
33         //1节点: 奇数长度root 0节点: 偶数长度root
34         len[1]=-1;
35         fail[0] = tot = now =1;
36         len[0]=0;
37         memset(nxt[0],0,sizeof nxt[0]);
38         memset(nxt[1],0,sizeof nxt[1]);
39         for (int i = 0;i < 26;i ++ )Link[1][i]=1,Link[0][i] = 1;
40     }
41     Palindromic_AutoMaton(){clear();}

```



```

42 int newnode(int ll){
43     tot++;
44     memset(nxt[tot],0,sizeof nxt[tot]);
45     fail[tot]=num[tot]=0;
46     len[tot]=ll;
47     return tot;
48 }
49 int add(int i,int F,int last,int ch){
50     int cur = Link[last][ch];
51     if (a[get_anc(i, len[last] + 1)] - 'A' == ch){
52         cur = last;
53     }
54     if(!nxt[cur][ch]){
55         int tt = newnode(len[cur] + 2);
56         fail[tt] = nxt[Link[cur][ch]][ch];
57         nxt[cur][ch] = tt;
58         memcpy(Link[tt],Link[fail[tt]],sizeof Link[tt]);
59
60         int u = get_anc(i, len[fail[tt]]);
61
62         Link[tt][a[u] - 'A'] = fail[tt];
63
64         damage[tt] = sum[i] - sum[get_anc(i, len[tt])];
65         damage_sum[tt] = (damage_sum[fail[tt]] + damage[tt])% mod;
66     }
67     last = nxt[cur][ch];num[last]++;
68     return last;
69 }
70 }pam;
71 char buf[10];
72 int main(){
73     int _;
74     scanf("%d",&_);
75     score['A'] = _;
76     scanf("%d",&_);
77     score['C'] = _;
78     scanf("%d",&_);
79     score['G'] = _;
80     scanf("%d",&_);
81     score['U'] = _;
82     scanf("%s",buf);
83     a[1] = buf[0];
84     a[0] = 'Z';
85     sum[1] = score[buf[0]];
86     pos[0] = 1;
87     pos[1] = pam.add(1,0,1,buf[0] - 'A');
88     lastans = pam.damage[pos[1]];

```

```

89     printf("%d\n",lastans);
90     int total = lastans;
91     for (int i = 2;i ++){
92         int F;
93         scanf("%d",&F);
94         F ^= lastans;
95         if (!F)break;
96         scanf("%s",buf);
97         a[i] = buf[0];
98         sum[i] = sum[F] + score[buf[0]];
99         st[i][0] = F;
100         for (int step = 1;step < 20 and st[i][step-1];step ++){
101             st[i][step] = st[st[i][step-1]][step-1];
102         }
103         pos[i] = pam.add(i,F,pos[F],buf[0] - 'A');
104         Damage[i] = pam.damage_sum[pos[i]] % mod;
105         (total += Damage[i]) %= mod;
106         printf("%d\n",lastans = pam.damage[pos[i]]);
107     }
108     printf("%d\n",total);
109     return 0;
110 }

```

### 3.8 后缀数组 (SA)

```

1  template<int N>
2  struct suffixarray
3  {
4      char s[N];
5      int rk[N],sa[N],ht[N],tmp[N],cnt[N];
6      // n:串长 m:字符集大小 s[0..n-1]:字符串 sa[1..n]:字典序第 i 小的是哪个后缀
7      // rk[0..n-1]:后缀 i 的排名 ht[i]:lcp(sa[i],sa[i-1])
8      void build(int n,int m)
9      {
10         int i,j,k;n++;
11         for(i=0;i<n*2+5;i++)rk[i]=sa[i]=ht[i]=tmp[i]=0;
12         for(i=0;i<m;i++)cnt[i]=0;
13         for(int i=0;i<n;i++)cnt[rk[i]=s[i]]++;
14         for(int i=1;i<m;i++)cnt[i]+=cnt[i-1];
15         for(i=0;i<n;i++)sa[--cnt[rk[i]]]=i;
16         for(k=1;k<=n;k<=1)
17         {
18             for(i=0;i<n;i++)
19             {
20                 j=sa[i]-k;
21                 if(j<0)j+=n;

```

```

22     tmp[cnt[rk[j]]++] = j;
23 }
24 sa[tmp[cnt[0]=0]] = j = 0;
25 for(i=1; i<n; i++)
26 {
27     if(rk[tmp[i]] != rk[tmp[i-1]] || rk[tmp[i]+k] != rk[tmp[i-1]+k])
28         cnt[++j] = i;
29     sa[tmp[i]] = j;
30 }
31 memcpy(rk, sa, n*sizeof(int));
32 memcpy(sa, tmp, n*sizeof(int));
33 if(j>=n-1) break;
34 }
35 for(j=rk[ht[i=k=0]]=0; i<n-1; i++, k++)
36     while(~k&&s[i] != s[sa[j-1]+k]) ht[j] = k--, j=rk[sa[j]+1];
37 }
38 void debug(int n)
39 {
40     cout<<"sa:"; for(int i=1; i<=n; i++) cout<<sa[i]<<" \n"[i==n];
41     cout<<"rk:"; for(int i=0; i<n; i++) cout<<rk[i]<<" \n"[i==n-1];
42     cout<<"ht:"; for(int i=1; i<=n; i++) cout<<ht[i]<<" \n"[i==n];
43 }
44 int mi[__lg(N)+1][N];
45 void rmqinit(int n)
46 {
47     for(int i=1; i<=n; i++)
48     {
49         mi[0][i] = ht[i];
50     }
51     for(int i=1; i<=__lg(n); i++)
52     {
53         for(int j=1; j+(1<<i)-1<=n; j++)
54         {
55             mi[i][j] = min(mi[i-1][j], mi[i-1][j+(1<<(i-1))]);
56         }
57     }
58 }
59 int query(int l, int r)
60 {
61     int k = __lg(r-l+1);
62     return min(mi[k][l], mi[k][r-(1<<k)+1]);
63 }
64 int Query(int L, int R)
65 {
66     int l = min(rk[L], rk[R]);
67     int r = max(rk[L], rk[R]);
68     return query(l+1, r);

```

```

69 }
70 int cmp(int l1, int r1, int l2, int r2)
71 {
72     int len = Query(l1, l2);
73     if(r1-l1+1<=len&&r2-l2+1<=len) return 0; // ==
74     else if(r1-l1+1<=len) return -1; // <
75     else if(r2-l2+1<=len) return 1; // >
76     else
77     {
78         if(s[l1+len]>s[l2+len]) return 1; // >
79         else return -1; // <
80     }
81 }
82 };
83
84 suffixarray<200005>sa;

```

### 3.9 后缀数组 (SA-IS)

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5 #define fi first
6 #define se second
7 #define lowbit(x) (x&(-x))
8 const int mod=998244353;
9 const double eps=1e-12;
10 const int inf=0x3f3f3f3f;
11 const ll INF=0x3f3f3f3f3f3f3f3f;
12 int dcmp(double x){if(fabs(x)<eps)return 0;return x>0?1:-1;}
13
14 // #define int ll
15
16 const int N=400050;
17 int nxt[N];
18 string s;
19 const int MAXLEN = 200005;
20 struct SAIS
21 {
22     // 后缀类型
23     #define L_TYPE 0
24     #define S_TYPE 1
25
26     int st[MAXLEN];
27     int rl[MAXLEN], rk[MAXLEN], lcp[MAXLEN];

```

```

28 int n;
29
30 void init(const string &s)
31 {
32     n = s.size();
33     for (int i=1;i<=n;++i) st[i] = s[i-1];
34     // st中的字符必须调成正数!!!!!!!!!!!!!!
35 }
36
37 // 判断一个字符是否为LMS字符
38 inline bool is_lms_char(int *type, int x) {
39     return x > 1 && type[x] == S_TYPE && type[x - 1] == L_TYPE;
40 }
41
42 // 判断两个LMS子串是否相同
43 inline bool equal_substring(int *S, int x, int y, int *type) {
44     do {
45         if (S[x] != S[y])
46             return false;
47         x++, y++;
48     } while (!is_lms_char(type, x) && !is_lms_char(type, y));
49
50     return S[x] == S[y] && type[x] == type[y];
51 }
52
53 // 诱导排序(从*型诱导到L型、从L型诱导到S型)
54 // 调用之前应将*型按要求放入SA中
55 inline void induced_sort(int *S, int *SA, int *type, int *bucket,
56     int *lbucket,
57     int *sbucket, int n, int SIGMA) {
58     for (int i = 1; i <= n; i++)
59         if (SA[i] > 1 && type[SA[i] - 1] == L_TYPE)
60             SA[lbucket[S[SA[i] - 1]]++] = SA[i] - 1;
61     for (int i = 0; i <= SIGMA; i++) // Reset S-type bucket
62         sbucket[i] = bucket[i];
63     for (int i = n; i >= 1; i--)
64         if (SA[i] > 1 && type[SA[i] - 1] == S_TYPE)
65             SA[sbucket[S[SA[i] - 1]]--] = SA[i] - 1;
66 }
67
68 // SA-IS主体
69 // S是输入字符串, length是字符串的长度, SIGMA是字符集的大小
70 int *sais(int *S, int length, int SIGMA) {
71     int n = length;
72     assert(S[n]==0);
73     int *type = new int[n + 5]; // 后缀类型
74     int *position = new int[n + 5]; // 记录LMS子串的起始位置

```

```

74 int *name = new int[n + 5]; // 记录每个LMS子串的新名称
75 int *SA = new int[n + 5]; // SA数组
76 int *bucket = new int[SIGMA + 5]; // 每个字符的桶
77 int *lbucket = new int[SIGMA + 5]; // 每个字符的L型桶的起始位置
78 int *sbucket = new int[SIGMA + 5]; // 每个字符的S型桶的起始位置
79
80 // 初始化每个桶
81 memset(bucket, 0, sizeof(int) * (SIGMA + 5));
82 for (int i = 1; i <= n; i++)
83     bucket[S[i]]++;
84 for (int i = 0; i <= SIGMA; i++) {
85     if (i==0)
86     {
87         bucket[i] = bucket[i];
88         lbucket[i] = 1;
89     } else
90     {
91         bucket[i] += bucket[i - 1];
92         lbucket[i] = bucket[i - 1] + 1;
93     }
94     sbucket[i] = bucket[i];
95 }
96
97 // 确定后缀类型(利用引理2.1)
98 type[n] = S_TYPE;
99 for (int i = n - 1; i >= 1; i--) {
100     if (S[i] < S[i + 1])
101         type[i] = S_TYPE;
102     else if (S[i] > S[i + 1])
103         type[i] = L_TYPE;
104     else
105         type[i] = type[i + 1];
106 }
107 // 寻找每个LMS子串
108 int cnt = 0;
109 for (int i = 1; i <= n; i++)
110     if (is_lms_char(type, i))
111         position[++cnt] = i;
112 // 对LMS子串进行排序
113 fill(SA, SA + n + 3, -1);
114 for (int i = 1; i <= cnt; i++)
115     SA[sbucket[S[position[i]]]--] = position[i];
116 induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);
117
118 // 为每个LMS子串命名
119 fill(name, name + n + 3, -1);
120 int lastx = -1, namecnt = 1; // 上一次处理的LMS子串与名称的计数

```

```

121 bool flag = false; // 这里顺便记录是否有重复的字符
122 for (int i = 2; i <= n; i++) {
123     int x = SA[i];
124
125     if (is_lms_char(type, x)) {
126         if (lastx >= 0 && !equal_substring(S, x, lastx, type))
127             namecnt++;
128         // 因为只有相同的LMS子串才会有同样的名称
129         if (lastx >= 0 && namecnt == name[lastx])
130             flag = true;
131
132         name[x] = namecnt;
133         lastx = x;
134     }
135 } // for
136 name[n] = 0;
137
138 // 生成S1
139 int *S1 = new int[cnt+5];
140 int pos = 0;
141 for (int i = 1; i <= n; i++)
142     if (name[i] >= 0)
143         S1[++pos] = name[i];
144 int *SA1;
145 if (!flag) {
146     // 直接计算SA1
147     SA1 = new int[cnt + 5];
148
149     for (int i = 1; i <= cnt; i++)
150         SA1[S1[i]+1] = i;
151 } else
152     SA1 = sais(S1, cnt, namecnt); // 递归计算SA1
153
154 // 从SA1诱导到SA
155 for (int i = 0; i <= SIGMA; i++) {
156     if (i==0)
157         lbucket[i] = 1;
158     else
159         lbucket[i] = bucket[i - 1] + 1;
160     sbucket[i] = bucket[i];
161 }
162 fill(SA, SA + n + 3, -1);
163 for (int i = cnt; i >= 1; i--) // 这里是逆序扫描SA1, 因为SA中S型桶是
    倒序的
164     SA[sbucket[S[position[SA1[i]]]]--] = position[SA1[i]];
165 induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);
166

```

```

167 delete[] S1;
168 delete[] SA1;
169 delete[] bucket;
170 delete[] lbucket;
171 delete[] sbucket;
172 delete[] position;
173 delete[] type;
174 delete[] name;
175 // 后缀数组计算完毕
176 return SA;
177 }
178
179 void build()
180 {
181     st[0]=st[n+2]=-1;
182     st[n+1]=0;
183     int SIGMA = 0;
184     for (int i=1;i<=n;++i) SIGMA = max(SIGMA,st[i]);
185     int * sa = sais(st,n+1,SIGMA);
186     for (int i=2;i<=n+1;++i) rk[sa[i]]=i-1;
187     delete[] sa;
188     for (int i=1;i<=n;++i) rl[rk[i]]=i;
189     for (int i=1,len=0;i<=n;++i)
190     {
191         if (len) --len;
192         while (i+len<=n && rl[rk[i]-1]+len<=n && st[i+len]==st[rl[rk[
193             i]-1]+len]) ++len;
194         lcp[rk[i]]=len;
195     }
196
197     #undef L_TYPE
198     #undef R_TYPE
199 }sa;
200 int len(int x){int ans=0;while(x)ans++,x>>=1;return ans;}
201 int mi[25][MAXLEN];
202 void build(int n)
203 {
204     for(int i=1;i<=n;i++)mi[0][i]=sa.lcp[i];
205     for(int j=1;j<25;j++)
206     {
207         for(int i=1;i<=n-(1<<j)+1;i++)
208         {
209             mi[j][i]=min(mi[j-1][i],mi[j-1][i+(1<<(j-1))]);
210         }
211     }
212 }

```

```

213 int query(int l, int r)
214 {
215     int k=len(r-l+1)-1;
216     return min(mi[k][l],mi[k][r-(1<<k)+1]);
217 }
218 int Query(int L, int R)
219 {
220     int l=min(sa.rk[L],sa.rk[R]);
221     int r=max(sa.rk[L],sa.rk[R]);
222     return query(l+1,r);
223 }
224 int cmp(int l1,int r1,int l2,int r2)
225 {
226     int len=Query(l1,l2);
227     if(r1-l1+1<=len&& r2-l2+1<=len) return 0; // ==
228     else if(r1-l1+1<=len) return -1; // <
229     else if(r2-l2+1<=len) return 1; // >
230     else
231     {
232         if(s[l1+len-1]>s[l2+len-1]) return 1; // >
233         else return -1; // <
234     }
235 }
236
237 void solve()
238 {
239     int n;
240     cin>>n;
241     cin>>s;
242     sa.init(s);
243     sa.build();
244     build(n);
245     for(int i=1;i<=n;i++)nxt[i]=1;
246     for(int i=n;i>=1;i--)
247     {
248         int j=i+1;
249         while(j<=n)
250         {
251             if(cmp(i,j-1,j,j+nxt[j]-1)<0)
252             {
253                 j=j+nxt[j];
254             }
255             else
256             {
257                 break;
258             }
259         }

```

```

260         nxt[i]=j-i;
261     }
262     for(int i=1;i<=n;i++)
263     {
264         cout<<nxt[i]<<" \n"[i==n];
265     }
266 }
267
268 /*
269 */
270
271 #undef int
272
273 int main()
274 {
275     ios::sync_with_stdio(false);cin.tie(nullptr);
276     // cout<<fixed<<setprecision(15);
277
278     srand(time(0));
279
280     int _;cin>>_;while(--_)
281     {
282         solve();
283     }
284     return 0;
285 }
286

```

### 3.10 后缀自动机

```

1 //SPOJ substring
2 // calc ans_i=长度=i的所有子串，出现次数最多的一种出现了多少次。
3 #define RIGHT
4 //RIGHT: parent树的dfs序上主席树，求每个点的Right集合
5 const int maxn = 25e4+100;
6 #ifdef RIGHT
7 struct Node{int L,R,val;}Tree[maxn*40];
8 struct Chairman_Tree{
9     int cnt = 0;
10    int root[maxn*2];
11    void init(){
12        memset(root,0,sizeof root);
13        cnt =0;
14    }
15    /* 建T0空树 */
16    int buildT0(int l, int r){

```

```

17     int k = cnt++;
18     Tree[k].val = 0;
19     if (l==r) return k;
20     int mid = (l+r) >> 1;
21     Tree[k].L = buildT0(l, mid); Tree[k].R = buildT0(mid + 1, r);
22     return k;
23 }
24 /* 上一个版本节点P, [ppos]+=del 返回新版本节点*/
25 int update (int p, int l, int r, int ppos, int del){
26     assert(cnt < maxn*50);
27     int k = cnt++;
28     Tree[k].val = Tree[p].val + del;
29     if (l==r) return k;
30     int mid = (l+r) >> 1;
31     if (ppos <= mid){
32         Tree[k].L = update(Tree[p].L, l, mid, ppos, del);
33         Tree[k].R = Tree[p].R;
34     }else{
35         Tree[k].L = Tree[p].L;
36         Tree[k].R = update(Tree[p].R, mid+1, r, ppos, del);
37     }
38     return k;
39 }
40 int query(int PL, int PR, int l, int r, int L, int R){
41     if (l>R || L>r) return 0;
42     if (L <= l && r <= R) return Tree[PR].val - Tree[PL].val;
43     int mid = (l + r) >> 1;
44     return query(Tree[PL].L, Tree[PR].L, l, mid, L, R) + query(Tree[PL].R,
45         Tree[PR].R, mid+1, r, L, R);
46 }
47 }tree;
48 #endif
49 char s[maxn]; int n, ans[maxn];
50 /*注意需要按l将节点基数排序来拓扑更新parent树*/
51 struct Suffix_Automaton{
52     //basic
53     int nxt[maxn*2][26], fa[maxn*2], len[maxn*2];
54     int last, cnt;
55     //nxt[x][c]表示从x点, 通过字符c可以到达的点。
56     //len[x]表示x点所能表示的最长子串长度。
57     //fa[x]表示x在后缀链接上的父亲。
58     //extension
59     int cntA[maxn*2], A[maxn*2]; /*辅助拓扑更新*/
60     int num[maxn*2]; /*每个节点代表的所有串的出现次数*/
61 #ifdef RIGHT
62     vector<int> E[maxn*2];
63     int dfs1[maxn*2], dfsr[maxn*2], dfn;

```

```

63     int pos[maxn*2];
64     int end_pos[maxn*2]; //1基
65 #endif
66 Suffix_Automaton(){ clear(); }
67 void clear(){
68     last = cnt=1;
69     fa[1]=len[1]=0;
70     memset(nxt[1], 0, sizeof nxt[1]);
71 }
72 void init(char *str){
73     while (*str){
74         add(*str-'a'); str++;
75     }
76 }
77 void add(int c){
78     int p = last;
79     int np = ++cnt;
80     memset(nxt[cnt], 0, sizeof nxt[cnt]);
81     len[np] = len[p]+1; last = np;
82     while (p && !nxt[p][c]) nxt[p][c] = np, p = fa[p];
83     if (!p) fa[np]=1;
84     else{
85         int q = nxt[p][c];
86         if (len[q]==len[p]+1) fa[np] = q;
87         else{
88             int nq = ++cnt;
89             len[nq] = len[p]+1;
90             memcpy(nxt[nq], nxt[q], sizeof (nxt[q]));
91             fa[nq] = fa[q]; fa[np] = fa[q] = nq;
92             while (nxt[p][c]==q) nxt[p][c] = nq, p = fa[p];
93         }
94     }
95 }
96 void build(){
97     memset(cntA, 0, sizeof cntA);
98     memset(num, 0, sizeof num);
99     for (int i=1; i<=cnt; i++) cntA[len[i]]++;
100     for (int i=1; i<=cnt; i++) cntA[i] += cntA[i-1];
101     for (int i=cnt; i>=1; i--) A[cntA[len[i]]--] = i;
102     /*更新主串节点*/
103     int temp=1;
104     for (int i=0; i<n; i++){
105         num[temp = nxt[temp][s[i]-'a']] = 1;
106     }
107     /*拓扑更新*/
108     for (int i=cnt; i>=1; i--){
109         //basic

```

```

110     int x = A[i];
111     num[fa[x]] += num[x];
112     //special
113     ans[len[x]] = max(ans[len[x]], num[x]);
114 }
115 //special
116 for (int i = len[last]; i > 1; i--) {
117     ans[i-1] = max(ans[i-1], ans[i]);
118 }
119 }
120
121 #ifdef RIGHT
122 int get_right_between(int u, int l, int r) {
123     return tree.query(tree.root[dfsl[u] - 1], tree.root[dfsr[u]], 1, ::
        n, l, r);
124 }
125 void dfs(int u) {
126     dfsl[u] = ++ dfn;
127     pos[dfn] = u;
128     for (int v : E[u]) {
129         dfs(v);
130     }
131     dfsr[u] = dfn;
132 }
133 void extract_right() {
134     int temp = 1;
135     for (int i = 0; i < n; i++) {
136         temp = nxt[temp][s[i] - 'a'];
137         end_pos[temp] = i + 1;
138     }
139     for (int i = 2; i <= cnt; i++) {
140         E[fa[i]].push_back(i);
141     }
142     dfn = 0;
143     dfs(1);
144     tree.root[0] = tree.buildT0(1, n);
145     for (int i = 1; i <= cnt; i++) {
146         int u = pos[i];
147         if (end_pos[u]) {
148             int idx = end_pos[u];
149             tree.root[i] = tree.update(tree.root[i-1], 1, n, idx, 1);
150         } else {
151             tree.root[i] = tree.root[i-1];
152         }
153     }
154 }
155 #endif

```

```

156 void debug() {
157     for (int i = cnt; i >= 1; i--) {
158         cout << "num[" << i << "] = " << num[i] << " len[" << i << "] = " << len[i] << " fa
            [" << i << "] = " << fa[i] << '\n';
159     }
160 }
161 } sam;
162
163 void solve()
164 {
165     scanf("%s", s);
166     /* calc n must before sam.init() */
167     n = strlen(s);
168     sam.init(s);
169     sam.build();
170     for (int i = 1; i <= n; i++) {
171         cout << ans[i] << '\n';
172     }
173 }

```

### 3.11 广义后缀自动机

```

1 //build sam using trie
2 #include <bits/stdc++.h>
3 using namespace std;
4 const int maxn = 1e6 + 100;
5 typedef long long ll;
6 struct Suffix_Automaton {
7     int nxt[maxn * 2][26], fa[maxn * 2], l[maxn * 2];
8     int last, cnt;
9     vector<int> E[maxn * 2];
10    int Num[maxn * 2];
11    Suffix_Automaton() { clear(); }
12    void clear() {
13        last = cnt = 1;
14        fa[1] = l[1] = 0;
15        memset(nxt[1], 0, sizeof nxt[1]);
16    }
17    int add(int pre, int c, int num) {
18        last = pre;
19        int p = last;
20        int np = ++cnt;
21        Num[np] = num;
22        memset(nxt[cnt], 0, sizeof nxt[cnt]);
23        l[np] = l[p] + 1; last = np;
24        while (p && !nxt[p][c]) nxt[p][c] = np, p = fa[p];

```



```

25     if (!p)fa[np]=1;
26     else{
27         int q = nxt[p][c];
28         if (l[q]==l[p]+1)fa[np] =q;
29         else{
30             int nq = ++ cnt;
31             l[nq] = l[p]+1;
32             memcpy(nxt[nq],nxt[q],sizeof (nxt[q]));
33             fa[nq] =fa[q];fa[np] = fa[q] =nq;
34             while (nxt[p][c]==q)nxt[p][c] =nq,p = fa[p];
35         }
36     }
37     return np;
38 }
39 int dfs1[maxn*2],dfsr[maxn*2];
40 int dfn = 0;
41 ll sum[maxn*2];
42 void dfs(int u){
43     dfs1[u] = ++dfn;
44     sum[dfn] = Num[u];
45     for (int v : E[u]){
46         dfs(v);
47     }
48     dfsr[u] = dfn;
49 }
50 void build(){
51     for (int i=2;i<=cnt;i++){
52         E[fa[i]].push_back(i);
53     }
54     dfs(1);
55     for (int i=1;i<=cnt;i++){
56         sum[i] += sum[i-1];
57     }
58 }
59 void query(char * s){
60     int temp = 1;
61     while (*s){
62         int ch = *s - 'A';
63         if (!nxt[temp][ch]){
64             printf("0\n");
65             return;
66         }
67         temp = nxt[temp][ch];
68         s++;
69     }
70     ll ans = sum[dfsr[temp]] - sum[dfs1[temp] - 1];
71     printf("%lld\n",ans);

```

```

72     }
73 }sam;
74 struct Trie{
75     int Root = 1;
76     int cnt = 2;
77     int nxt[maxn][26];
78     int num[maxn];
79     int sam_pos[maxn];
80     int add(int p,int ch){
81         if (!nxt[p][ch]){
82             nxt[p][ch] = cnt++;
83         }
84         int now = nxt[p][ch];
85         num[now] ++;
86         return now;
87     }
88     void bfs(){
89         queue<int> Q;
90         Q.push(1);
91         sam_pos[1] = 1;
92         while (!Q.empty()){
93             int head = Q.front();
94             Q.pop();
95             for (int i=0;i<26;i++){
96                 if (!nxt[head][i])continue;
97                 int now = nxt[head][i];
98                 sam_pos[now] = sam.add(sam_pos[head],i,num[now]);
99                 Q.push(now);
100             }
101         }
102     }
103 }trie;
104 int trie_pos[maxn];
105 int main(){
106     int n,k;
107     scanf("%d%d",&n,&k);
108     trie_pos[0] = 1;
109     for (int i=1;i<=n;i++){
110         static char s[5];
111         int p;
112         scanf("%s%d",s,&p);
113         int ch = s[0] - 'A';
114         trie_pos[i] = trie.add(trie_pos[p],ch);
115     }
116     trie.bfs();
117     sam.build();
118     for (int i=0;i<k;i++){

```



```

119 static char t[maxn];
120 scanf("%s",t);
121 int N = strlen(t);
122 reverse(t,t+N);
123 sam.query(t);
124 }
125 return 0;
126 }

```

## 4 Mathematics

### 4.1 快速乘

```

1 inline ll mull(const ll&a,const ll&b,const ll&mod)
2 {
3     ll tmp=a*b-(ll)((long double)a/mod*b-1e-9)*mod;
4     return tmp<0?tmp+mod:tmp;
5 }

```

### 4.2 欧几里得算法

```

1 int gcd(int a,int b)
2 {
3     return b==0?a:gcd(b,a%b);
4 }

```

### 4.3 拓展欧几里得算法

```

1 int exgcd(int a,int b,int &x,int &y)
2 {
3     if(b==0)return x=1,y=0,a;
4     int r=exgcd(b,a%b,x,y);
5     tie(x,y)=make_tuple(y,x-(a/b)*y);
6     return r;
7 }

```

### 4.4 BSGS

```

1 typedef long long ll;
2 map <ll,ll> f;
3 ll gcd(ll a,ll b) //最大公约数
4 {
5     if (!b)return a;
6     return gcd(b,a%b);
7 }
8 void exgcd(ll a,ll b,ll &x,ll &y) //拓欧
9 {
10     if (!b)x=1,y=0;
11     else
12     {
13         exgcd(b,a%b,x,y);

```

```

14     ll t=x;
15     x=y;
16     y=t-a/b*y;
17 }
18 }
19 ll inv(ll a,ll b/*mod*/) //逆元
20 {
21     ll x,y;
22     exgcd(a,b,x,y);
23     return (x%b+b)%b;
24 }
25 ll poww(ll a,ll n,ll p)
26 {
27     ll ans=1;
28     while(n)
29     {
30         if(n&1)ans=ans*a%p;
31         a=a*a%p;
32         n>>=1;
33     }
34     return ans;
35 }
36 ll bsgs(ll a,ll b,ll p) //BSGS算法,p为质数  $a^x=b(mod\ p)$ 
37 {
38     f.clear();
39     ll m=ceil(sqrt(p));
40     b%=p;
41     for (ll i=1;i<=m;i++)
42     {
43         b=b*a%p;
44         f[b]=i;
45     }
46     ll tmp=poww(a,m,p);
47     b=1;
48     for (ll i=1;i<=m;i++)
49     {
50         b=b*tmp%p;
51         if (f[b])return (i*m-f[b]+p)%p;
52     }
53     return -1;
54 }
55 ll exbsgs(ll a,ll b,ll p)
56 {
57     if (b==1||p==1)return 0; //特殊情况,  $x=0$ 时最小解
58     ll g=gcd(a,p),k=0,na=1;
59     while (g>1)
60     {

```

```

61         if (b%g!=0)return -1; //无法整除则无解
62         k++;b/=g;p/=g;na=na*(a/g)%p;
63         if (na==b)return k; //na=b说明前面的a的次数为0, 只需要返回k
64         g=gcd(a,p);
65     }
66     ll ff=bsgs(a,b*inv(na,p)%p,p);
67     if (ff!=-1)return -1;
68     return ff+k;
69 }

```

#### 4.5 埃氏筛

```

1 const int maxn=10000000;
2 bool not_prime[maxn+5];
3 int prime[maxn+5],tot=0;
4 for(int i=2;i<=maxn;i++)
5 {
6     if(!not_prime[i])prime[++tot]=i;
7     for(int j=2*i;j<=maxn;j+=i)
8     {
9         not_prime[j]=1;
10    }
11 }

```

#### 4.6 欧拉筛

```

1 const int maxn=10000000;
2 bool not_prime[maxn+5];
3 int prime[maxn+5],tot=0;
4 for(int i=2;i<=maxn;i++)
5 {
6     if(!not_prime[i])prime[++tot]=i;
7     for(int j=1;j<=tot&&i*prime[j]<=maxn;j++)
8     {
9         not_prime[i*prime[j]]=1;
10        if(i%prime[j]==0)break;
11    }
12 }

```

#### 4.7 米勒罗宾素性测试 & Pollard rho 算法

```

1 #include <bits/stdc++.h>

```

```

2 using namespace std;
3 typedef long long ll;
4 const int maxn = 105;
5
6 ll x[maxn], ans;
7 queue<ll> aria;
8
9 ll multi(ll a, ll b, ll p) //快速乘
10 {
11     ll ans = 0;
12     while(b)
13     {
14         if(b & 1LL)
15             ans = (ans+a)%p;
16         a = (a+a)%p;
17         b >>= 1;
18     }
19     return ans;
20 }
21
22 ll qpow(ll a, ll b, ll p)
23 {
24     ll ans = 1;
25     while(b)
26     {
27         if(b & 1LL)
28             ans = multi(ans, a, p);
29         a = multi(a, a, p);
30         b >>= 1;
31     }
32     return ans;
33 }
34
35 bool miller_rabin(ll n)
36 {
37     if(n == 2)
38         return true;
39     int s = 20, i, t = 0;
40     ll u = n-1;
41     while(!(u&1))
42     {
43         t++;
44         u >>= 1;
45     }
46     while(s--)
47     {
48         ll a = rand()%(n-2)+2;

```

```

49         x[0] = qpow(a, u, n);
50         for(i = 1; i <= t; i++)
51         {
52             x[i] = multi(x[i-1], x[i-1], n);
53             if(x[i] == 1 && x[i-1] != 1 && x[i-1] != n-1)
54                 return false;
55         }
56         if(x[t] != 1)
57             return false;
58     }
59     return true;
60 }
61
62 ll gcd(ll a, ll b)
63 {
64     if(b == 0)
65         return a;
66     else
67         return gcd(b, a%b);
68 }
69
70 ll Pollard_Rho(ll n, int c)
71 {
72     ll i = 1, k = 2, x = rand()%(n-1)+1, y = x;
73     while(1)
74     {
75         i++;
76         x = (multi(x, x, n)+c)%n;
77         ll p = gcd((y-x+n)%n, n);
78         if(p != 1 && p != n)
79             return p;
80         if(y == x)
81             return n;
82         if(i == k)
83         {
84             y = x;
85             k <<= 1;
86         }
87     }
88 }
89
90 void calc(ll n, int c)
91 {
92     if(n == 1)
93         return;
94     if(miller_rabin(n))
95     {

```

```

96     aria.push(n);
97     return;
98 }
99 ll p = n, k = c;
100 while(p >= n)
101 {
102     p = Pollard_Rho(p, c--);
103 }
104 calc(p, k);
105 calc(n/p, k);
106 }
107
108 int main()
109 {
110     ll n;
111     cin >> n;
112     calc(n, 107);
113     cout << aria.front();
114     aria.pop();
115     while(!aria.empty())
116     {
117         cout << "*" << aria.front();
118         aria.pop();
119     }
120     cout << endl;
121     return 0;
122 }

```

#### 4.8 Stein 算法 (大整数的 GCD)

```

1 //如果a=0或b=0, 则 (0,b)=b, (a,0)=a
2 //如果a,b都是偶数, 则 (a,b)=(a/2,b/2)
3 //如果a是偶数, b是奇数, 则 (a,b)=(a/2,b)
4 //如果a是奇数, b是偶数, 则 (a,b)=(a,b/2)
5 //如果a,b都是奇数, 不妨设a>b, 则 (a,b)=(b,a-b)

```

#### 4.9 逆元

```

1 ll inv(ll a, ll p)
2 {
3     ll x, y;
4     exgcd(a, p, x, y);
5     return (x%p+p)%p;
6 }

```

```

1 const int maxn=10000000;
2 long long inv[maxn+5];
3 inv[1]=1;
4 for(int i=2;i<=maxn;i++)
5 {
6     inv[i]=inv[p%i]*(p-p/i)%p;
7 }

```

#### 4.10 欧拉函数

```

1 long long get_phi(long long n)
2 {
3     long long phi=1;
4     for(int i=2;i<=n/i;i++)
5     {
6         if(n%i==0)
7         {
8             phi*=(i-1);
9             n/=i;
10            while(n%i==0)phi*=i,n/=i;
11        }
12    }
13    if(n>1)phi*=(n-1);
14    return phi;
15 }

```

```

1 const int maxn=10000000;
2 bool not_prime[maxn+5];
3 int prime[maxn+5], tot=0, phi[maxn+5];
4 for(int i=2;i<=n;i++)
5 {
6     if(!not_prime[i])prime[++tot]=i, phi[i]=i-1;
7     for(int j=1;j<=tot&&i*prime[j]<=n;j++)
8     {
9         not_prime[i*prime[j]]=1;
10        if(i%prime[j]==0)
11        {
12            phi[i*prime[j]]=phi[i]*prime[j];
13            break;
14        }
15        phi[i*prime[j]]=phi[i]*(prime[j]-1);
16    }
17 }

```

## 4.11 欧拉降幂

$$a^b \bmod c = \begin{cases} a^b \bmod \varphi(c), & \gcd(a, c) = 1 \\ a^b, & \gcd(a, c) \neq 1, b < \varphi(m) \\ a^{b \bmod \varphi(c) + \varphi(c)}, & \gcd(a, c) \neq 1, b \geq \varphi(m) \end{cases} \bmod c$$

## 4.12 拓展中国剩余定理

```

1 ll exCRT(ll k, ll *a, ll *r) {
2     ll M = r[1], ans = a[1];
3     for (int i = 2; i <= k; i++) {
4         ll x0, y0;
5         ll c = a[i] - ans;
6         ll g = exgcd(M, r[i], x0, y0);
7         if (c % g != 0) return -1;
8         x0 = (__int128)x0 * (c / g) % (r[i] / g);
9         ans = x0 * M + ans;
10        M = lcm(M, r[i]);
11        ans = (ans % M + M) % M;
12    }
13    return ans; // 无解返回 -1
14 }
```

## 4.13 中国剩余定理

```

1 ll CRT(int k, ll* a, ll* r) {
2     ll n = 1, ans = 0;
3     for (int i = 1; i <= k; i++) n = n * r[i];
4     for (int i = 1; i <= k; i++) {
5         ll m = n / r[i], b, y;
6         exgcd(m, r[i], b, y); // b * m mod r[i] = 1
7         ans = (ans + a[i] * m * b % n) % n;
8     }
9     return (ans % n + n) % n;
10 }
```

## 4.14 二次剩余

```

1 struct cipolla
2 {
3     ll n, p, w2, a; // w2 = a*a - n mod p
4
5     struct cx{ll x, y;}; // Fp2 = {x+yw | x, y belong to Fp}
```

```

6     cx mul(cx a, cx b){return (cx){(a.x*b.x+a.y*b.y%p*w2)%p, (a.x*b.y+a.y
7         *b.x)%p};}
8     cx cx_poww(cx a, int n)
9     {
10        cx ans=(cx){1,0};
11        while(n)
12        {
13            if(n&1)ans=mul(ans,a);
14            a=mul(a,a);n>>=1;
15        }
16        return ans;
17    }
18    void getsol(ll _n, ll _p, ll&ans1, ll&ans2)
19    {
20        n=_n, p=_p;
21        ans1=ans2=-1; // 最多两个解, 无解是 -1
22        if(n==0 || p==2){ans1=n; return;}
23
24        if(poww((ll)n, (ll)(p-1)/2, (ll)p)==p-1) return;
25
26        for(a=rand()%p; a==p-1?(a=0):(++a))
27        {
28            if(poww((a*a-n+p)%p, (p-1)/2, p)==p-1)
29            {
30                w2=(a*a-n+p)%p;
31                break;
32            }
33        }
34        cx b=(cx){a,1};
35        b=cx_poww(b, (p+1)/2);
36        ans1=b.x;
37        if(ans1!=p-ans1) ans2=p-ans1;
38        return;
39    }
40 }
```

## 4.15 拓展欧拉定理

```

1 #include <iostream>
2 #include <algorithm>
3 #include <math.h>
4 #include <string.h>
5 #define ll long long
6 using namespace std;
7 ll phi(ll n) { // 欧拉函数
8     ll ans = n;
```

```

9   for (ll i = 2; i * i <= n; i++) {
10       if (n % i == 0) {
11           ans -= ans / i; //遇到质因数,即 $X*1/pi$ 
12           while (n % i == 0) {
13               n /= i;
14           }
15       }
16   }
17   if (n > 1) //若n不为1, 则还剩下一位质因子
18   ans -= ans / n;
19   return ans;
20 }
21 ll fastpow(ll a, ll b, ll m) { //快速幂
22     ll ans = 1;
23     while (b > 0) {
24         if (b & 1) {
25             ans = ans * a % m;
26         }
27         a = a * a % m;
28         b >>= 1;
29     }
30     return ans;
31 }
32 ll eulerDropPow(ll a, char b[], ll c) { //欧拉降幂
33     ll eulerNumbers = phi(c); //模数c降幂
34     ll descendingPower = 0;
35     for (ll i = 0, len = strlen(b); i < len; ++i) {
36         descendingPower = (descendingPower * 10 + b[i] - '0') %
37             eulerNumbers;
38     }
39     descendingPower += eulerNumbers - 1;
40     //descendingPower += eulerNumbers; 应该是这样, 只是这题要求 $a^{(n-1)}$ 
41     return fastpow(a, descendingPower, c);
42 }
43 long long a=2;
44 long long c=1000000007;
45 char b[1000000000];
46 int main() {
47     while (~scanf("%s", b)) { //a是底数, b是幂, c是模
48         cout << eulerDropPow(a, b, c) << endl;
49     }
50 }

```

#### 4.16 组合数

```

1   int MAX=1e6+1;

```

```

2   vector<mint> fac, inv, finv;
3   void binom_init()
4   {
5       fac.resize(MAX);
6       finv.resize(MAX);
7       inv.resize(MAX);
8       fac[0]=fac[1]=1;
9       inv[1]=1;
10      finv[0]=finv[1]=1;
11      for (int i=2; i<MAX; i++)
12      {
13          fac[i]=fac[i-1]*i;
14          inv[i]=-(MOD/i)*inv[MOD%i];
15          finv[i]=finv[i-1]*inv[i];
16      }
17  }
18  mint binom(int n, int r)
19  {
20      if (n<r || n<0 || r<0) return 0;
21      return fac[n]*finv[r]*finv[n-r];
22  }

```

```

1   ll binom(ll n, ll m)
2   {
3       if (n<m || n<0 || m<0) return 0;
4       if (m>n-m) m=n-m;
5       ll ans=1;
6       for (int i=0; i<m; i++)
7           ans=ans*(n-i)/(i+1); // 取模时记得先单独把 (n-i) 取模
8       return ans;
9   }

```

#### 4.17 卢卡斯定理

```

1   long long Lucas(long long n, long long m, long long p) {
2       if (m == 0) return 1;
3       return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
4   }

```

#### 4.18 拓展卢卡斯定理

```

1   LL calc(LL n, LL x, LL P) {
2       if (!n) return 1;
3       LL s = 1;

```

```

4  for (LL i = 1; i <= P; i++)
5  if (i % x) s = s * i % P;
6  s = Pow(s, n / P, P);
7  for (LL i = n / P * P + 1; i <= n; i++)
8  if (i % x) s = i % P * s % P;
9  return s * calc(n / x, x, P) % P;
10 }
11
12 LL multilucas(LL m, LL n, LL x, LL P) {
13     int cnt = 0;
14     for (LL i = m; i; i /= x) cnt += i / x;
15     for (LL i = n; i; i /= x) cnt -= i / x;
16     for (LL i = m - n; i; i /= x) cnt -= i / x;
17     return Pow(x, cnt, P) % P * calc(m, x, P) % P * inv(calc(n, x, P),
18         P) % P * inv(calc(m - n, x, P), P) % P;
19 }
20
21 LL exlucas(LL m, LL n, LL P) {
22     int cnt = 0;
23     LL p[20], a[20];
24     for (LL i = 2; i * i <= P; i++) {
25         if (P % i == 0) {
26             p[++cnt] = i;
27             while (P % i == 0) p[cnt] = p[cnt] * i, P /= i;
28             a[cnt] = multilucas(m, n, i, p[cnt]);
29         }
30     }
31     if (P > 1) p[++cnt] = P, a[cnt] = multilucas(m, n, P, P);
32     return CRT(cnt, a, p);
33 }

```

#### 4.19 积性函数

```

1  ll get_f(ll n)
2  {
3      ll ans=1;
4      for(int i=2;i<=n/i;i++)
5      {
6          int cnt=0;
7          while(n%i==0)cnt++,n/=i;
8          ans*=calc_f(i,cnt); //calc_f(p,k)=f[p^k]
9      }
10     if(n>1)ans*=calc_f(n,1);
11     return ans;
12 }

```

```

1  f[1]=1;
2  for(int i=2;i<=n;i++)
3  {
4      if(!isprime[i])prime[++tot]=i,f[i]=calc_f(i,1);
5      for(int j=1;j<=tot&& i*prime[j]<=n;j++)
6      {
7          if(i%p[j]==0)
8          {
9              cnt[i*p[j]]=cnt[i]+1; //cnt[n]:n的最小质因子的次数
10             f[i*p[j]]=f[i]/calc_f(p[j],cnt[i])*calc_f(p[j],cnt[i]+1);
11             break;
12         }
13         cnt[i*p[j]]=1;
14         f[i*p[j]]=f[i]*calc_f(p[j],1);
15     }
16 }

```

#### 4.20 杜教筛

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

$$\sum_{i=1}^n h(i) = \sum_{i=1}^n (f * g)(i) = \sum_{i=1}^n \sum_{d|i} g(d) f\left(\frac{i}{d}\right) = \sum_{d=1}^n g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) = \sum_{d=1}^n g(d) sum\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$$

$$\Rightarrow g(1)sum(n) = \sum_{i=1}^n h(i) - \sum_{i=2}^n g(i)sum\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  #define ll long long
5  #define fi first
6  #define se second
7  #define lowbit(x) (x&(-x))
8  const double eps=1e-12;
9  const int inf=0x3f3f3f3f;
10 const ll INF=0x3f3f3f3f3f3f3f3f;
11 int dcmp(double x){if(fabs(x)<eps)return 0;return x>0?1:-1;}
12
13 #define int ll
14
15 ll sumf1[3000005];
16 unordered_map<ll,ll>f1;
17
18 inline ll F1(ll n)

```

```

19 {
20     if(n<=3e6)return sumf1[n]; // 预处理出 n 较小时的前缀和
21     if(f1.count(n))return f1[n]; // 记忆化, 如果求过这个值, 就不需要再递归一遍了
22     ll ans=n*(n+1)/2; // 这是 f * g 的 n 项前缀和
23     for(ll l=2,r;l<=n;l=r+1) // 整除分块
24         r=n/(n/l),ans-=(r-l+1)*F1(n/l);
25     // [l,r] 的 F(n/l) 是一样的, 对 g(x) 求个和即可
26     return f1[n]=ans; // 别忘了除上 g(1)
27 }
28
29 ll sumf2[3000005];
30 unordered_map<ll,ll>f2;
31
32 inline ll F2(ll n)
33 {
34     if(n<=3e6)return sumf2[n]; // 预处理出 n 较小时的前缀和
35     if(f2.count(n))return f2[n]; // 记忆化, 如果求过这个值, 就不需要再递归一遍了
36     ll ans=1; // 这是 f * g 的 n 项前缀和
37     for(ll l=2,r;l<=n;l=r+1) // 整除分块
38         r=n/(n/l),ans-=(r-l+1)*F2(n/l);
39     // [l,r] 的 F(n/l) 是一样的, 对 g(x) 求个和即可
40     return f2[n]=ans; // 别忘了除上 g(1)
41 }
42
43 int prime[3000005];
44 int vis[3000005];
45 int len=0;
46
47 void init()
48 {
49     sumf1[1]=sumf2[1]=1;
50     for(int i=2;i<=3000000;i++)
51     {
52         if(!vis[i])
53         {
54             prime[++len]=i;
55             sumf1[i]=i-1;
56             sumf2[i]=-1;
57         }
58         for(int j=1;;j++)
59         {
60             long long t=1ll*prime[j]*i;
61             if(t>3000000)break;
62             vis[t]=1;
63             if(i%prime[j]==0){sumf1[t]=sumf1[i]*prime[j];sumf2[t]=0;break}

```

```

64         }
65         else{sumf1[t]=sumf1[i]*(prime[j]-1);sumf2[t]=-sumf2[i];}
66     }
67     for(int i=2;i<=3000000;i++)
68     {
69         sumf1[i]+=sumf1[i-1];
70         sumf2[i]+=sumf2[i-1];
71     }
72 }
73
74 void solve()
75 {
76     ll n;
77     cin>>n;
78     cout<<F1(n)<<' '<<F2(n)<<'\n';
79 }
80
81 /*
82 */
83
84 #undef int
85
86 int main()
87 {
88     ios::sync_with_stdio(false);cin.tie(nullptr);
89     // cout<<fixed<<setprecision(15);
90
91     init();
92
93     int _;cin>>_;while(--_)
94     {
95         solve();
96     }
97     return 0;
98 }
99

```

## 4.21 Min\_25 筛

To Be Studied...

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 const int maxn = 2000000+100;
5

```



```

6  /******
7
8  f()函数中(31-37行) 填函数在质数幂次处的表达式
9  pow_sum()函数中(38-43行) 填幂和函数(如果需要更高次的话可以在这里添加)
10 202-205行按要求填写
11 f_p[][0/1/2/3/ ... ]分别代表质数个数/质数和/质数平方和/质数三次方和/ ... 根据自己需
   要添加
12 例: 如果该函数在质数处表达式为f(p) = p^2+3*p+1, 则表明需要质数个数/质数和/质数平
   方和, 即f_p[][0],f_p[][1],f_p[][2]
13
14 *****
15
16 ll poww(ll a,ll b){
17     ll res = 1;
18     ll base = a;
19     while(b){
20         if(b&1){
21             res *= base;
22             //res %= mod;
23         }
24         base *= base;
25         //base %= mod;
26         b>>=1;
27     }
28     return res;
29 }
30
31 inline ll f(ll p,int e){
32     if(p==1||e==0) return 1;
33     ///return f(p^e)
34     ll res = poww(p,e);
35     return res*res+3*res+1;
36 }
37
38 ll pow_sum(ll n,int k){
39     ///return sum(i^k), i from 1 to n.
40     if(k==0) return n;
41     if(k==1) return n*(n+1)/2;
42     if(k==2) return n*(n+1)*(2*n+1)/6;
43 }
44 ll f_p[maxn][3]; ///F_prime(id(n/i))
45 ll n;
46 int n_2; ///(int)sqrt(n)
47 int n_3; ///(int)pow(n,1.0/3.0)
48 int n_6; ///(int)pow(n,1.0/6.0)
49 ll val_id[maxn]; ///give the id, return the id-th number like 'n/i' ,(
   val_id[1] = 1)

```

```

50 int val_id_num; ///how many numbers like 'n/i'
51 int val_id_num_3; ///how many numbers like 'n/i' below n/n_3;
52 int p[200000+100];
53 bool isp[maxn];
54 int p_sz_2; ///pi(n_2)
55 int p_sz_3; ///pi(n_3)
56 int p_sz_6; ///pi(n_6)
57 void init(){
58     n_2 = (int)sqrt(n);
59     n_3 = (int)pow(n,1.0/3.0);
60     n_6 = (int)pow(n,1.0/6.0);
61     val_id_num = 0;
62     for(ll i=1;i<=n;){
63         val_id[++val_id_num] = i;
64         if(i==n) break;
65         i = n/(n/(i+1));
66     }
67     memset(isp,1,sizeof isp);
68     isp[1] = 0;
69     for(int i=2;i<=n_2;i++){
70         if(isp[i]){
71             p[++p_sz_2] = i;
72             if(i<=n_3) p_sz_3++;
73             if(i<=n_6) p_sz_6++;
74         }
75         for(int j=1;j<=p_sz_2&& p[j]*i<=n_2;j++){
76             isp[i*p[j]] = 0;
77             if(i%p[j]==0) break;
78         }
79     }
80 }
81 inline int get_id(ll k){ ///give a number like 'n/i', return the id of
   it
82     if(k>n_2) return val_id_num-n/k+1;
83     else return k;
84 }
85 ll c[maxn];
86 int lowbit(int n){return n & (-n);}
87 void add(int x,ll d){
88     while(x<maxn){
89         c[x]+=d;
90         x+=lowbit(x);
91     }
92 }
93 ll sum(int x){
94     ll ans=0;
95     while(x){

```

```

96     ans+=c[x];
97     x-=lowbit(x);
98 }
99 return ans;
100 }
101
102 struct node{
103     int k_max;
104     ll val;
105     ll f_val;
106 };
107 void update_bfs(int k,int type){
108     queue<node> q;
109     while(!q.empty()) q.pop();
110     int e = 1;
111     for(ll i=p[k];i<n/n_3;i*=p[k]){
112         node st;
113         st.k_max = k;
114         st.val = i;
115         if(type==-1)st.f_val = f(p[k],e);
116         else st.f_val = poww(i,type);
117         q.push(st);
118         e++;
119     }
120     while(!q.empty()){
121         node hd = q.front();
122         q.pop();
123         if((hd.val!=p[hd.k_max]&&type>=0)||type==-1) {//if(type==-1)cout
124             << "****" << hd.val << "****" << hd.f_val << endl;
125             ll w = n/hd.val;
126             w = n/w;//cout << hd.val << "[" << w<< " , " << val_id[
127                 val_id_num] << "]" << endl;
128             if(type==-1){
129                 add(get_id(w),hd.f_val);
130                 add(val_id_num+1,-1ll*hd.f_val);
131             }
132             else{
133                 add(get_id(w),-1ll*hd.f_val);
134                 add(val_id_num+1,hd.f_val);
135             }
136         }
137         for(int i=hd.k_max+1;hd.val*p[i]<n/n_3&&i<=p_sz_2;i++){
138             ll res = p[i];
139             for(int e=1;;e++){
140                 if(hd.val*res<n/n_3){
141                     node nxt;
142                     nxt.k_max = i;

```

```

141         nxt.val = hd.val*res;
142         if(type==-1) nxt.f_val = hd.f_val*f(p[i],e);
143         else nxt.f_val = hd.f_val*poww(res,type);
144         q.push(nxt);
145     }
146     else break;
147     res *= p[i];
148 }
149 }
150 }
151 }
152 void get_f_p(ll n,int times){
153     for(int i=1;i<=val_id_num;i++){
154         for(int j=0;j<=times;j++){
155             f_p[i][j] = pow_sum(val_id[i],j)-1;
156         }
157     }
158     int now;
159     //for(now=1;now<=p_sz_2;now++){
160     for(now=1;p[now]<=n_6;now++){
161         for(int j=val_id_num;j>=1;j--){
162             ll w = val_id[j]/p[now];
163             if(w<p[now]) break;
164             ll val=1;
165             for(int k = 0;k<=times;k++){
166                 f_p[j][k] = f_p[j][k] - val*(f_p[get_id(w)][k]-f_p[p[now]
167                     -1][k]);
168                 val *= p[now];
169             }
170         }
171     }
172     int nnow = now;
173     int val = 1;
174     for(int tt = 0;tt<=times;tt++){
175         now = nnow;
176         memset(c,0,sizeof c);
177         add(1,f_p[1][tt]);
178         for(int i=2;val_id[i]<n/n_3;i++){
179             add(i,f_p[i][tt] - f_p[i-1][tt]);
180         }
181         for(;p[now]<=n_3;now++){
182             for(int j=val_id_num;j>=1;j--){
183                 ll w = val_id[j]/p[now];
184                 if(val_id[j]<n/n_3) break;
185                 if(w<p[now]) break;
186                 if(w<n/n_3) f_p[j][tt] = f_p[j][tt] - (sum(get_id(w)) -
187                     sum(p[now-1]))*poww(p[now],tt);

```

```

186         else f_p[j][tt] = f_p[j][tt] - (f_p[get_id(w)][tt]-sum(p[
187             now-1]))*poww(p[now],tt);
188     }
189     update_bfs(now,tt);
190 }
191 for(int i=1;i<=val_id_num&&val_id[i]<n/n_3;i++){
192     f_p[i][tt] = sum(i);
193     for(;now<=p_sz_2;now++){
194         for(int j=val_id_num;j>=1;j--){
195             ll w = val_id[j]/p[now];
196             if(val_id[j]<n/n_3) break;
197             if(w<p[now]) break;
198             f_p[j][tt] -= (f_p[get_id(w)][tt]-f_p[p[now-1]][tt])*poww(
199                 p[now],tt);
200         }
201     }
202     for(int i=1;i<=val_id_num;i++){
203         //if f(p) = p^2+3p+1, then write: f_p[i][0] = f_p[i][2] + 3*f_p[i]
204         // [1] + f_p[i][0];
205         f_p[i][0] = f_p[i][2] + 3*f_p[i][1] + f_p[i][0];
206     }
207 }
208 ll F[2000000+100];
209 void get_f_3(ll n){ //V(F_{pi(n^(1/3))+1},n)
210     ll q = p[p_sz_3+1];
211     for(int now=1;now<=val_id_num;now++){
212         if(val_id[now]<q){
213             F[now] = 1;
214         }
215         else if(val_id[now]<q*q){
216             F[now] = 1+(f_p[now][0]-f_p[q-1][0]);
217         }
218         else{
219             F[now] = 1+(f_p[now][0]-f_p[q-1][0]);
220             for(int pp=p_sz_3+1;p[pp]<=(int)(sqrt(val_id[now]))&&pp<=
221                 p_sz_2;pp++){
222                 F[now] += f(p[pp],2) + (f(p[pp],1))*(f_p[get_id(val_id[now]
223                     ]/p[pp])[0]-f_p[get_id(p[pp])][0]);
224             }
225         }
226     }
227 }
228 void get_f_6(ll n){ //V(F_{pi(n^(1/6))+1},n)
229     memset(c,0,sizeof c);

```

```

228     add(1,F[1]);
229     for(int i=2;val_id[i]<n/n_3;i++){
230         add(i,F[i] - F[i-1]);
231     }
232     for(int k=p_sz_3;k>p_sz_6;k--){
233         int now = val_id_num;
234         for(;val_id[now]>=n/n_3;now--){
235             int e = 1;
236             ll _p = p[k];
237             while(val_id[now]/_p){
238                 if(val_id[now]/_p>=n/n_3){
239                     F[now] += F[get_id(val_id[now]/_p)]*f(p[k],e);
240                 }
241                 else{
242                     F[now] += sum(get_id(val_id[now]/_p))*f(p[k],e);
243                 }
244                 _p *= p[k];
245                 e++;
246             }
247         }
248         if(k==1) break;
249         //cout << "*****" << p[k] << "*****" << n/n_3 << endl;
250         update_bfs(k,-1); //bfs to update [lpf(i)==P[k-1]]f(i)
251     }
252     for(int i=1;i<=val_id_num&&val_id[i]<n/n_3;i++){
253         F[i] = sum(i);
254     }
255 void get_f(ll n){
256     for(int k=p_sz_6;k>=1;k--){
257         for(int now = val_id_num;now>=1;now--){
258             int e = 1;
259             ll _p = p[k];
260             while(val_id[now]/_p){
261                 F[now] += F[get_id(val_id[now]/_p)]*f(p[k],e);
262                 _p *= p[k];
263                 e++;
264             }
265         }
266     }
267 }
268 int main(){//n = 1000000000; //1e10:455052511,0.83s/0.58s 1e12
269     //:37607912018 9.224s/5.105s
270     cin >> n;
271     init();
272     get_f_p(n,2);
273     get_f_3(n);
274     get_f_6(n);

```

```

274 get_f(n);
275 for(int i=1;i<=val_id_num;i++){
276     cout << val_id[i] << " : " << F[i] << endl;
277 }
278 }

```

#### 4.22 莫比乌斯函数

```

1 int prime[10000005];
2 int vis[10000005];
3 int mo[10000005];
4 int len=0;
5 void init()
6 {
7     mo[1]=1;
8     for(int i=2;i<=10000000;i++)
9     {
10         if(!vis[i])
11         {
12             prime[++len]=i;
13             mo[i]=-1;
14         }
15         for(int j=1;;j++)
16         {
17             long long t=1ll*prime[j]*i;
18             if(t>10000000)break;
19             vis[t]=1;
20             if(i%prime[j]==0){mo[t]=0;break;}
21             else mo[t]=-mo[i];
22         }
23     }
24 }

```

#### 4.23 快速数论变换

```

1 #include <bits/stdc++.h>
2
3 using u32 = unsigned;
4 using i64 = long long;
5 using u64 = unsigned long long;
6
7 constexpr int N = 2E5;
8
9 std::vector<int> minp, primes;

```

```

10 template<class T>
11 constexpr T power(T a, i64 b) {
12     T res = 1;
13     for (; b; b /= 2, a *= a) {
14         if (b % 2) {
15             res *= a;
16         }
17     }
18     return res;
19 }
20
21 template<int P>
22 struct MInt {
23     int x;
24     constexpr MInt() : x{} {}
25     constexpr MInt(i64 x) : x{norm(x % getMod())} {}
26
27     static int Mod;
28     constexpr static int getMod() {
29         if (P > 0) {
30             return P;
31         } else {
32             return Mod;
33         }
34     }
35     constexpr static void setMod(int Mod_) {
36         Mod = Mod_;
37     }
38     constexpr int norm(int x) const {
39         if (x < 0) {
40             x += getMod();
41         }
42         if (x >= getMod()) {
43             x -= getMod();
44         }
45         return x;
46     }
47     constexpr int val() const {
48         return x;
49     }
50     explicit constexpr operator int() const {
51         return x;
52     }
53     constexpr MInt operator-() const {
54         MInt res;
55         res.x = norm(getMod() - x);
56         return res;

```

```

57 }
58 constexpr MInt inv() const {
59     assert(x != 0);
60     return power(*this, getMod() - 2);
61 }
62 constexpr MInt &operator*=(MInt rhs) & {
63     x = 1LL * x * rhs.x % getMod();
64     return *this;
65 }
66 constexpr MInt &operator+=(MInt rhs) & {
67     x = norm(x + rhs.x);
68     return *this;
69 }
70 constexpr MInt &operator-=(MInt rhs) & {
71     x = norm(x - rhs.x);
72     return *this;
73 }
74 constexpr MInt &operator/=(MInt rhs) & {
75     return *this *= rhs.inv();
76 }
77 friend constexpr MInt operator*(MInt lhs, MInt rhs) {
78     MInt res = lhs;
79     res *= rhs;
80     return res;
81 }
82 friend constexpr MInt operator+(MInt lhs, MInt rhs) {
83     MInt res = lhs;
84     res += rhs;
85     return res;
86 }
87 friend constexpr MInt operator-(MInt lhs, MInt rhs) {
88     MInt res = lhs;
89     res -= rhs;
90     return res;
91 }
92 friend constexpr MInt operator/(MInt lhs, MInt rhs) {
93     MInt res = lhs;
94     res /= rhs;
95     return res;
96 }
97 friend constexpr std::istream &operator>>(std::istream &is, MInt &a)
98     {
99     i64 v;
100     is >> v;
101     a = MInt(v);
102     return is;
103 }

```

```

103 friend constexpr std::ostream &operator<<(std::ostream &os, const
104     MInt &a) {
105     return os << a.val();
106 }
107 friend constexpr bool operator==(MInt lhs, MInt rhs) {
108     return lhs.val() == rhs.val();
109 }
110 friend constexpr bool operator!=(MInt lhs, MInt rhs) {
111     return lhs.val() != rhs.val();
112 }
113 };
114 template<>
115 int MInt<0>::Mod = 1;
116
117 template<int V, int P>
118 constexpr MInt<P> CInv = MInt<P>(V).inv();
119
120 constexpr int P = 998244353;
121 using Z = MInt<P>;
122
123 std::vector<int> rev;
124 template<int P>
125 std::vector<MInt<P>> roots{0, 1};
126
127 template<int P>
128 constexpr MInt<P> findPrimitiveRoot() {
129     MInt<P> i = 2;
130     int k = __builtin_ctz(P - 1);
131     while (true) {
132         if (power(i, (P - 1) / 2) != 1) {
133             break;
134         }
135         i += 1;
136     }
137     return power(i, (P - 1) >> k);
138 }
139
140 template<int P>
141 constexpr MInt<P> primitiveRoot = findPrimitiveRoot<P>();
142
143 template<>
144 constexpr MInt<998244353> primitiveRoot<998244353> {31};
145
146 template<int P>
147 constexpr void dft(std::vector<MInt<P>> &a) {
148     int n = a.size();

```

```

149 if (int(rev.size()) != n) {
150     int k = __builtin_ctz(n) - 1;
151     rev.resize(n);
152     for (int i = 0; i < n; i++) {
153         rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
154     }
155 }
156
157 for (int i = 0; i < n; i++) {
158     if (rev[i] < i) {
159         std::swap(a[i], a[rev[i]]);
160     }
161 }
162
163 if (roots<P>.size() < n) {
164     int k = __builtin_ctz(roots<P>.size());
165     roots<P>.resize(n);
166     while ((1 << k) < n) {
167         auto e = power(primitiveRoot<P>, 1 << (__builtin_ctz(P - 1) -
168             k - 1));
169         for (int i = 1 << (k - 1); i < (1 << k); i++) {
170             roots<P>[2 * i] = roots<P>[i];
171             roots<P>[2 * i + 1] = roots<P>[i] * e;
172         }
173         k++;
174     }
175     for (int k = 1; k < n; k *= 2) {
176         for (int i = 0; i < n; i += 2 * k) {
177             for (int j = 0; j < k; j++) {
178                 MInt<P> u = a[i + j];
179                 MInt<P> v = a[i + j + k] * roots<P>[k + j];
180                 a[i + j] = u + v;
181                 a[i + j + k] = u - v;
182             }
183         }
184     }
185 }
186
187 template<int P>
188 constexpr void idft(std::vector<MInt<P>> &a) {
189     int n = a.size();
190     std::reverse(a.begin() + 1, a.end());
191     dft(a);
192     MInt<P> inv = (1 - P) / n;
193     for (int i = 0; i < n; i++) {
194         a[i] *= inv;

```

```

195     }
196 }
197
198 template<int P = 998244353>
199 struct Poly : public std::vector<MInt<P>> {
200     using Value = MInt<P>;
201
202     Poly() : std::vector<Value>() {}
203     explicit constexpr Poly(int n) : std::vector<Value>(n) {}
204
205     explicit constexpr Poly(const std::vector<Value> &a) : std::vector<
206         Value>(a) {}
207     constexpr Poly(const std::initializer_list<Value> &a) : std::vector<
208         Value>(a) {}
209
210     template<class InputIt, class = std::_RequireInputIter<InputIt>>
211     explicit constexpr Poly(InputIt first, InputIt last) : std::vector<
212         Value>(first, last) {}
213
214     template<class F>
215     explicit constexpr Poly(int n, F f) : std::vector<Value>(n) {
216         for (int i = 0; i < n; i++) {
217             (*this)[i] = f(i);
218         }
219     }
220
221     constexpr Poly shift(int k) const {
222         if (k >= 0) {
223             auto b = *this;
224             b.insert(b.begin(), k, 0);
225             return b;
226         } else if (this->size() <= -k) {
227             return Poly();
228         } else {
229             return Poly(this->begin() + (-k), this->end());
230         }
231     }
232
233     constexpr Poly trunc(int k) const {
234         Poly f = *this;
235         f.resize(k);
236         return f;
237     }
238
239     constexpr friend Poly operator+(const Poly &a, const Poly &b) {
240         Poly res(std::max(a.size(), b.size()));
241         for (int i = 0; i < a.size(); i++) {
242             res[i] += a[i];
243         }

```

```

239     for (int i = 0; i < b.size(); i++) {
240         res[i] += b[i];
241     }
242     return res;
243 }
244 constexpr friend Poly operator-(const Poly &a, const Poly &b) {
245     Poly res(std::max(a.size(), b.size()));
246     for (int i = 0; i < a.size(); i++) {
247         res[i] += a[i];
248     }
249     for (int i = 0; i < b.size(); i++) {
250         res[i] -= b[i];
251     }
252     return res;
253 }
254 constexpr friend Poly operator-(const Poly &a) {
255     std::vector<Value> res(a.size());
256     for (int i = 0; i < int(res.size()); i++) {
257         res[i] = -a[i];
258     }
259     return Poly(res);
260 }
261 constexpr friend Poly operator*(Poly a, Poly b) {
262     if (a.size() == 0 || b.size() == 0) {
263         return Poly();
264     }
265     if (a.size() < b.size()) {
266         std::swap(a, b);
267     }
268     int n = 1, tot = a.size() + b.size() - 1;
269     while (n < tot) {
270         n *= 2;
271     }
272     if (((P - 1) & (n - 1)) != 0 || b.size() < 128) {
273         Poly c(a.size() + b.size() - 1);
274         for (int i = 0; i < a.size(); i++) {
275             for (int j = 0; j < b.size(); j++) {
276                 c[i + j] += a[i] * b[j];
277             }
278         }
279         return c;
280     }
281     a.resize(n);
282     b.resize(n);
283     dft(a);
284     dft(b);
285     for (int i = 0; i < n; ++i) {

```

```

286         a[i] *= b[i];
287     }
288     idft(a);
289     a.resize(tot);
290     return a;
291 }
292 constexpr friend Poly operator*(Value a, Poly b) {
293     for (int i = 0; i < int(b.size()); i++) {
294         b[i] *= a;
295     }
296     return b;
297 }
298 constexpr friend Poly operator*(Poly a, Value b) {
299     for (int i = 0; i < int(a.size()); i++) {
300         a[i] *= b;
301     }
302     return a;
303 }
304 constexpr friend Poly operator/(Poly a, Value b) {
305     for (int i = 0; i < int(a.size()); i++) {
306         a[i] /= b;
307     }
308     return a;
309 }
310 constexpr Poly &operator+=(Poly b) {
311     return (*this) = (*this) + b;
312 }
313 constexpr Poly &operator-=(Poly b) {
314     return (*this) = (*this) - b;
315 }
316 constexpr Poly &operator*=(Poly b) {
317     return (*this) = (*this) * b;
318 }
319 constexpr Poly &operator*=(Value b) {
320     return (*this) = (*this) * b;
321 }
322 constexpr Poly &operator/=(Value b) {
323     return (*this) = (*this) / b;
324 }
325 constexpr Poly deriv() const {
326     if (this->empty()) {
327         return Poly();
328     }
329     Poly res(this->size() - 1);
330     for (int i = 0; i < this->size() - 1; ++i) {
331         res[i] = (i + 1) * (*this)[i + 1];
332     }

```



```

333     return res;
334 }
335 constexpr Poly integr() const {
336     Poly res(this->size() + 1);
337     for (int i = 0; i < this->size(); ++i) {
338         res[i + 1] = (*this)[i] / (i + 1);
339     }
340     return res;
341 }
342 constexpr Poly inv(int m) const {
343     Poly x((*this)[0].inv());
344     int k = 1;
345     while (k < m) {
346         k *= 2;
347         x = (x * (Poly{2} - trunc(k) * x)).trunc(k);
348     }
349     return x.trunc(m);
350 }
351 constexpr Poly log(int m) const {
352     return (deriv() * inv(m)).integr().trunc(m);
353 }
354 constexpr Poly exp(int m) const {
355     Poly x{1};
356     int k = 1;
357     while (k < m) {
358         k *= 2;
359         x = (x * (Poly{1} - x.log(k) + trunc(k))).trunc(k);
360     }
361     return x.trunc(m);
362 }
363 constexpr Poly pow(int k, int m) const {
364     int i = 0;
365     while (i < this->size() && (*this)[i] == 0) {
366         i++;
367     }
368     if (i == this->size() || 1LL * i * k >= m) {
369         return Poly(m);
370     }
371     Value v = (*this)[i];
372     auto f = shift(-i) * v.inv();
373     return (f.log(m - i * k) * k).exp(m - i * k).shift(i * k) *
374         power(v, k);
375 }
376 constexpr Poly sqrt(int m) const {
377     Poly x{1};
378     int k = 1;
379     while (k < m) {

```

```

379         k *= 2;
380         x = (x + (trunc(k) * x.inv(k)).trunc(k)) * CInv<2, P>;
381     }
382     return x.trunc(m);
383 }
384 constexpr Poly mulT(Poly b) const {
385     if (b.size() == 0) {
386         return Poly();
387     }
388     int n = b.size();
389     std::reverse(b.begin(), b.end());
390     return ((*this) * b).shift(-(n - 1));
391 }
392 constexpr std::vector<Value> eval(std::vector<Value> x) const {
393     if (this->size() == 0) {
394         return std::vector<Value>(x.size(), 0);
395     }
396     const int n = std::max(x.size(), this->size());
397     std::vector<Poly> q(4 * n);
398     std::vector<Value> ans(x.size());
399     x.resize(n);
400     std::function<void(int, int, int)> build = [&](int p, int l, int
401         r) {
402         if (r - l == 1) {
403             q[p] = Poly{1, -x[l]};
404         } else {
405             int m = (l + r) / 2;
406             build(2 * p, l, m);
407             build(2 * p + 1, m, r);
408             q[p] = q[2 * p] * q[2 * p + 1];
409         }
410     };
411     build(1, 0, n);
412     std::function<void(int, int, int, const Poly &)> work = [&](int
413         p, int l, int r, const Poly &num) {
414         if (r - l == 1) {
415             if (l < int(ans.size())) {
416                 ans[l] = num[0];
417             }
418         } else {
419             int m = (l + r) / 2;
420             work(2 * p, l, m, num.mulT(q[2 * p + 1]).trunc(m - l));
421             work(2 * p + 1, m, r, num.mulT(q[2 * p]).trunc(r - m));
422         }
423     };
424     work(1, 0, n, mulT(q[1].inv(n)));
425     return ans;

```



```

424 }
425 };
426
427 template<int P = 998244353>
428 Poly<P> berlekampMassey(const Poly<P> &s) {
429     Poly<P> c;
430     Poly<P> oldC;
431     int f = -1;
432     for (int i = 0; i < s.size(); i++) {
433         auto delta = s[i];
434         for (int j = 1; j <= c.size(); j++) {
435             delta -= c[j - 1] * s[i - j];
436         }
437         if (delta == 0) {
438             continue;
439         }
440         if (f == -1) {
441             c.resize(i + 1);
442             f = i;
443         } else {
444             auto d = oldC;
445             d *= -1;
446             d.insert(d.begin(), 1);
447             MInt<P> df1 = 0;
448             for (int j = 1; j <= d.size(); j++) {
449                 df1 += d[j - 1] * s[f + 1 - j];
450             }
451             assert(df1 != 0);
452             auto coef = delta / df1;
453             d *= coef;
454             Poly<P> zeros(i - f - 1);
455             zeros.insert(zeros.end(), d.begin(), d.end());
456             d = zeros;
457             auto temp = c;
458             c += d;
459             if (i - temp.size() > f - oldC.size()) {
460                 oldC = temp;
461                 f = i;
462             }
463         }
464     }
465     c *= -1;
466     c.insert(c.begin(), 1);
467     return c;
468 }
469
470

```

```

471 template<int P = 998244353>
472 MInt<P> linearRecurrence(Poly<P> p, Poly<P> q, i64 n) {
473     int m = q.size() - 1;
474     while (n > 0) {
475         auto newq = q;
476         for (int i = 1; i <= m; i += 2) {
477             newq[i] *= -1;
478         }
479         auto newp = p * newq;
480         newq = q * newq;
481         for (int i = 0; i < m; i++) {
482             p[i] = newp[i * 2 + n % 2];
483         }
484         for (int i = 0; i <= m; i++) {
485             q[i] = newq[i * 2];
486         }
487         n /= 2;
488     }
489     return p[0] / q[0];
490 }
491
492 void sieve(int n) {
493     minp.assign(n + 1, 0);
494     primes.clear();
495
496     for (int i = 2; i <= n; i++) {
497         if (minp[i] == 0) {
498             minp[i] = i;
499             primes.push_back(i);
500         }
501
502         for (auto p : primes) {
503             if (i * p > n) {
504                 break;
505             }
506             minp[i * p] = p;
507             if (p == minp[i]) {
508                 break;
509             }
510         }
511     }
512 }
513
514 int f[N + 1];
515
516 void solve() {
517     int n;

```

```

518 std::cin >> n;
519
520 std::vector<int> a(n);
521 for (int i = 0; i < n; i++) {
522     std::cin >> a[i];
523 }
524
525 bool same = true;
526 for (int i = 1; i < n; i++) {
527     if (f[a[i]] != f[a[0]]) {
528         same = false;
529     }
530 }
531 if (same && (f[a[0]] == 0 || n % 2 == 1)) {
532     std::cout << "Bob\n";
533 } else {
534     std::cout << "Alice\n";
535 }
536 }
537
538 int main() {
539     std::ios::sync_with_stdio(false);
540     std::cin.tie(nullptr);
541
542     sieve(N);
543
544     for (int i = 5; i <= N; i += 2) {
545         f[i] = minp[i - 2] == i - 2 ? 1 ^ f[i - 2] : 0;
546     }
547
548     Poly<P> p[2];
549     p[0].resize(N + 1);
550     p[1].resize(N + 1);
551     f[4] = 1;
552     for (int i = 3; i <= N; i += 2) {
553         if (minp[i] == i) {
554             p[f[i]][i] = 1;
555         }
556     }
557     auto a = p[0] * p[0];
558     auto b = p[1] * p[1];
559     for (int i = 6; i <= N; i += 2) {
560         if (a[i] != 0 && b[i] != 0) {
561             f[i] = 3;
562         } else if (a[i] != 0) {
563             f[i] = 1;
564         }
565     }

```

```

565 }
566
567 int t;
568 std::cin >> t;
569
570 while (t--) {
571     solve();
572 }
573
574 return 0;
575 }

```

## 4.24 FWT

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  //is=+1 : fwt
5  //is=-1 : ifwt
6
7  //basically, we need +,-,* for class T
8  // for xor_fwt and xor_convolution, we need /2 (or change it to
9      *2^(-1) manually)
10
11 template <class T>
12 void and_fwt(T f[],int ldn,int is=1)
13 {
14     int n=(1<<ldn);
15     for(int ldm=1;ldm<=ldn;ldm++)
16     {
17         int m=(1<<ldm),mh=m>>1;
18         for(int r=0;r<n;r+=m)
19         {
20             int t1=r,t2=r+mh;
21             for(int j=0;j<mh;j++,t1++,t2++)
22             {
23                 T u=f[t1],v=f[t2];
24                 if(is>0)f[t1]=u+v,f[t2]=v;
25                 else f[t1]=u-v,f[t2]=v;
26             }
27         }
28     }
29
30     template <class T>
31     void or_fwt(T f[],int ldn,int is=1)

```

```

32 {
33     int n=(1<<ldn);
34     for(int ldm=1;ldm<=ldn;ldm++)
35     {
36         int m=(1<<ldm),mh=m>>1;
37         for(int r=0;r<n;r+=m)
38         {
39             int t1=r,t2=r+mh;
40             for(int j=0;j<mh;j++,t1++,t2++)
41             {
42                 T u=f[t1],v=f[t2];
43                 if(is>0)f[t1]=u,f[t2]=u+v;
44                 else f[t1]=u,f[t2]=v-u;
45             }
46         }
47     }
48 }
49
50 template <class T>
51 void xor_fwt(T f[],int ldn,int is=1)
52 {
53     int n=(1<<ldn);
54     for(int ldm=1;ldm<=ldn;ldm++)
55     {
56         int m=(1<<ldm),mh=m>>1;
57         for(int r=0;r<n;r+=m)
58         {
59             int t1=r,t2=r+mh;
60             for(int j=0;j<mh;j++,t1++,t2++)
61             {
62                 T u=f[t1],v=f[t2];
63                 if(is>0)f[t1]=u+v,f[t2]=u-v;
64                 else f[t1]=(u+v)/2,f[t2]=(u-v)/2;
65             }
66         }
67     }
68 }
69
70 // will **change** f[] and g[], result is stored in f[]
71 template <class T>
72 void and_convolution(T f[],T g[],int n)
73 {
74     int ldn=__lg(n);assert((1<<ldn)==n);
75     and_fwt(f,ldn);and_fwt(g,ldn);
76     for(int i=0;i<n;i++)f[i]=f[i]*g[i];
77     and_fwt(f,ldn,-1);
78 }

```

```

79
80 template <class T>
81 void or_convolution(T f[],T g[],int n)
82 {
83     int ldn=__lg(n);assert((1<<ldn)==n);
84     or_fwt(f,ldn);or_fwt(g,ldn);
85     for(int i=0;i<n;i++)f[i]=f[i]*g[i];
86     or_fwt(f,ldn,-1);
87 }
88
89 template <class T>
90 void xor_convolution(T f[],T g[],int n)
91 {
92     int ldn=__lg(n);assert((1<<ldn)==n);
93     xor_fwt(f,ldn);xor_fwt(g,ldn);
94     for(int i=0;i<n;i++)f[i]=f[i]*g[i];
95     xor_fwt(f,ldn,-1);
96 }

```

## 4.25 BM

```

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>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
16 // head
17
18 ll n;
19 namespace linear_seq {
20     const int N=10010;
21     ll res[N],base[N],_c[N],_md[N];
22
23     vector<int> Md;
24     void mul(ll *a,ll *b,int k) {

```

```

25 rep(i,0,k+k) _c[i]=0;
26 rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
27 for (int i=k+k-1;i>=k;i--) if (_c[i])
28 rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%
    mod;
29 rep(i,0,k) a[i]=_c[i];
30 }
31 int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+ ...
32 ll ans=0,pnt=0;
33 int k=SZ(a);
34 assert(SZ(a)==SZ(b));
35 rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
36 Md.clear();
37 rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
38 rep(i,0,k) res[i]=base[i]=0;
39 res[0]=1;
40 while ((1ll<pnt)<=n) pnt++;
41 for (int p=pnt;p>=0;p--) {
42 mul(res,res,k);
43 if ((n>p)&1) {
44 for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
45 rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%
        mod;
46 }
47 }
48 rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
49 if (ans<0) ans+=mod;
50 return ans;
51 }
52 VI BM(VI s) {
53 VI C(1,1),B(1,1);
54 int L=0,m=1,b=1;
55 rep(n,0,SZ(s)) {
56 ll d=0;
57 rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
58 if (d==0) ++m;
59 else if (2*L<=n) {
60 VI T=C;
61 ll c=mod-d*powmod(b,mod-2)%mod;
62 while (SZ(C)<SZ(B)+m) C.pb(0);
63 rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
64 L=n+1-L; B=T; b=d; m=1;
65 } else {
66 ll c=mod-d*powmod(b,mod-2)%mod;
67 while (SZ(C)<SZ(B)+m) C.pb(0);
68 rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
69 ++m;

```

```

70 }
71 }
72 return C;
73 }
74 int gao(VI a,ll n) {
75 VI c=BM(a);
76 c.erase(c.begin());
77 rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
78 return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
79 }
80 };
81
82 int main() {
83 /*push_back 进去前 8~10 项左右、最后调用 gao 得第 n 项*/
84 vector<int>v;
85 v.push_back(3);
86 v.push_back(9);
87 v.push_back(20);
88 v.push_back(46);
89 v.push_back(106);
90 v.push_back(244);
91 v.push_back(560);
92 v.push_back(1286);
93 v.push_back(2956);
94 v.push_back(6794);
95 int nCase;
96 scanf("%d",&nCase);
97 while(nCase--){
98 scanf("%lld",&n);
99 printf("%lld\n",1LL * linear_seq::gao(v,n-1) % mod); //求第n项
100 }
101 }

```

## 4.26 矩阵求逆

```

1 #include<bits/stdc++.h>
2 #define int64 long long
3 using namespace std;
4 const int64 mod=1e9+7;
5 int64 a[410][410];
6 int n,is[410],js[410];
7 void exgcd(int a,int b,int &x,int &y){
8     if(!b)return x=1,y=0,void();
9     exgcd(b,a%b,y,x);y-=x*(a/b);
10 }
11 int inv(int p){

```

```

12  int x,y;exgcd(p,mod,x,y);
13  return (x+mod)%mod;
14  }
15  void inv(){
16      for(int k=1;k<=n;k++){
17          for(int i=k;i<=n;i++) // 1
18              for(int j=k;j<=n;j++)if(a[i][j]){
19                  is[k]=i,js[k]=j;break;
20              }
21          for(int i=1;i<=n;i++) // 2
22              swap(a[k][i],a[is[k]][i]);
23          for(int i=1;i<=n;i++)
24              swap(a[i][k],a[i][js[k]]);
25          if(!a[k][k]){
26              puts("No Solution");
27              exit(0);
28          }
29          a[k][k]=inv(a[k][k]); // 3
30          for(int j=1;j<=n;j++)if(j!=k) // 4
31              (a[k][j]*=a[k][k])%=mod;
32          for(int i=1;i<=n;i++)if(i!=k) // 5
33              for(int j=1;j<=n;j++)if(j!=k)
34                  (a[i][j]+=mod-a[i][k]*a[k][j]%mod)%=mod;
35          for(int i=1;i<=n;i++)if(i!=k) // 就是这里不同
36              a[i][k]=(mod-a[i][k]*a[k][k]%mod)%mod;
37      }
38      for(int k=n;k--){ // 6
39          for(int i=1;i<=n;i++)
40              swap(a[js[k]][i],a[k][i]);
41          for(int i=1;i<=n;i++)
42              swap(a[i][is[k]],a[i][k]);
43      }
44  }
45  int main(){
46      scanf("%d",&n);
47      for(int i=1;i<=n;i++)
48          for(int j=1;j<=n;j++)
49              scanf("%lld",a[i]+j);
50      inv();
51      for(int i=1;i<=n;i++)
52          for(int j=1;j<=n;j++)
53              printf("%lld%c",a[i][j],j==n?'\\n':' ');
54      return 0;
55  }

```

## 4.27 矩阵乘法 &amp; 快速幂

```

1  template<typename T>
2  struct matrix
3  {
4      int siz;
5      vector<vector<T>>>a;
6      matrix(int _siz)
7      {
8          siz=_siz;
9          a=vector(siz,vector<T>(siz));
10     }
11     void resize(int _siz)
12     {
13         siz=_siz;
14         a=vector(siz,vector<T>(siz));
15     }
16     vector<T>& operator[](int x)
17     {
18         assert(x>=0&&x<siz);
19         return a[x];
20     }
21     matrix<T> operator*(matrix<T>&b)
22     {
23         assert(siz==b.siz);
24         matrix<T> res(siz);
25         for(int i=0;i<siz;i++)
26         {
27             for(int j=0;j<siz;j++)
28             {
29                 for(int k=0;k<siz;k++)
30                 {
31                     res[i][j]+=a[i][k]*b[k][j];
32                 }
33             }
34         }
35         return res;
36     }
37     friend matrix<T> operator^(matrix<T> a,int n)
38     {
39         matrix<T> res(a.siz);
40         for(int i=0;i<a.siz;i++)res[i][i]=1;
41         while(n)
42         {
43             if(n&1)res=res*a;
44             a=a*a;n>>=1;
45         }

```

```

46     return res;
47 }
48 };

```

#### 4.28 整除分块

```

1 //向上取整
2 for(int l=1;l<=n;l++)
3 {
4     int k=(n+l-1)/l;
5     int r=(n-1)/(k-1);
6     solve();
7     l=r;
8 }

```

#### 4.29 自适应辛普森积分

$$\int_a^b f(x)dx \approx \frac{b-a}{6} \left[ f(a) + 4 \times f\left(\frac{a+b}{2}\right) + f(b) \right]$$

```

1 long double simpson(long double a, long double b) {
2     return (f(a) + 4 * f((a + b) / 2) + f(b)) * (b - a) / 6;
3 }
4
5 long double integral(long double L, long double R, long double eps) {
6     long double mid = (L + R) / 2;
7     long double S_total = simpson(L, R);
8     long double S_left = simpson(L, mid);
9     long double S_right = simpson(mid, R);
10    long double delta = S_left + S_right - S_total;
11    if (fabs(delta) < 15 * eps)
12        return S_left + S_right + delta / 15;
13    return integral(L, mid, eps / 2) + integral(mid, R, eps / 2);
14 }

```

#### 4.30 最小二乘法

给定  $n$  个点, 求一条直线  $y = kx + b$  最匹配 (写出方差式子求偏导)

$$\begin{cases} \Sigma xk + nb - \Sigma y = 0 \\ \Sigma x^2k + \Sigma xb - \Sigma xy = 0 \end{cases} \rightarrow \begin{cases} k = \frac{-\Sigma x\Sigma y + n\Sigma xy}{\Sigma x\Sigma x - \Sigma x^2n} \\ b = \frac{-\Sigma x^2\Sigma y + \Sigma x\Sigma xy}{\Sigma x^2n - \Sigma x\Sigma x} \end{cases}$$

```

1 void ols(double n, double*x, double*y, double&k, double&b) { //ordinary
2     least squares
3     double sumx = 0, sumy = 0, sumxx = 0, sumxy = 0;
4     for(int i=1;i<=n;++i){
5         sumx += x[i];
6         sumy += y[i];
7         sumxx += x[i] * x[i];
8         sumxy += x[i] * y[i];
9     }
10    k = (sumx * sumy - n * sumxy) / (sumx * sumx - sumxx * n);
11    b = -(sumxx * sumy - sumx * sumxy) / (sumx * sumx - sumxx * n);
12 }
13 double variance(double* x, double* y, double& k, double& b) {
14     double res = 0;
15     for (int i = 1; i <= n; ++i) {
16         res += (k * x[i] + b - y[i]) * (k * x[i] + b - y[i]);
17     }
18     return res;
19 }

```

#### 4.31 高精度

```

1 struct Int {
2     int sign;
3     std::vector<int> v;
4
5     Int() : sign(1) {}
6     Int(const std::string &s) { *this = s; }
7     Int(int v) {
8         char buf[21];
9         sprintf(buf, "%d", v);
10        *this = buf;
11    }
12    void zip(int unzip) {
13        if (unzip == 0) {
14            for (int i = 0; i < (int)v.size(); i++)
15                v[i] = get_pos(i * 4) + get_pos(i * 4 + 1) * 10 + get_pos(i * 4 + 2) * 100 + get_pos(i * 4 + 3) * 1000;
16        } else
17            for (int i = (v.resize(v.size() * 4), (int)v.size() - 1), a; i >= 0; i--)
18                a = (i % 4 >= 2) ? v[i / 4] / 100 : v[i / 4] % 100, v[i] = (i & 1) ? a / 10 : a % 10;
19        setsign(1, 1);
20    }
21 }

```

```

21 int get_pos(unsigned pos) const { return pos >= v.size() ? 0 : v[
    pos]; }
22 Int &setsign(int newsign, int rev) {
23     for (int i = (int)v.size() - 1; i > 0 && v[i] == 0; i--)
24         v.erase(v.begin() + i);
25     sign = (v.size() == 0 || (v.size() == 1 && v[0] == 0)) ? 1 : (
        rev ? newsign * sign : newsign);
26     return *this;
27 }
28 std::string to_str() const {
29     Int b = *this;
30     std::string s;
31     for (int i = (b.zip(1), 0); i < (int)b.v.size(); ++i)
32         s += char(*(b.v.rbegin() + i) + '0');
33     return (sign < 0 ? "-" : "") + (s.empty() ? std::string("0") : s
        );
34 }
35 bool absless(const Int &b) const {
36     if (v.size() != b.v.size()) return v.size() < b.v.size();
37     for (int i = (int)v.size() - 1; i >= 0; i--)
38         if (v[i] != b.v[i]) return v[i] < b.v[i];
39     return false;
40 }
41 Int operator-() const {
42     Int c = *this;
43     c.sign = (v.size() > 1 || v[0]) ? -c.sign : 1;
44     return c;
45 }
46 Int &operator=(const std::string &s) {
47     if (s[0] == '-')
48         *this = s.substr(1);
49     else {
50         for (int i = (v.clear(), 0); i < (int)s.size(); ++i)
51             v.push_back(*(s.rbegin() + i) - '0');
52         zip(0);
53     }
54     return setsign(s[0] == '-' ? -1 : 1, sign = 1);
55 }
56 bool operator<(const Int &b) const {
57     return sign != b.sign ? sign < b.sign : (sign == 1 ? absless(b)
        : b.absless(*this));
58 }
59 bool operator==(const Int &b) const { return v == b.v && sign == b.
    sign; }
60 Int &operator+=(const Int &b) {
61     if (sign != b.sign) return *this = (*this) - -b;
62     v.resize(std::max(v.size(), b.v.size()) + 1);

```

```

63     for (int i = 0, carry = 0; i < (int)b.v.size() || carry; i++) {
64         carry += v[i] + b.get_pos(i);
65         v[i] = carry % 10000, carry /= 10000;
66     }
67     return setsign(sign, 0);
68 }
69 Int operator+(const Int &b) const {
70     Int c = *this;
71     return c += b;
72 }
73 void add_mul(const Int &b, int mul) {
74     v.resize(std::max(v.size(), b.v.size()) + 2);
75     for (int i = 0, carry = 0; i < (int)b.v.size() || carry; i++) {
76         carry += v[i] + b.get_pos(i) * mul;
77         v[i] = carry % 10000, carry /= 10000;
78     }
79 }
80 Int operator-(const Int &b) const {
81     if (b.v.empty() || b.v.size() == 1 && b.v[0] == 0) return *this;
82     if (sign != b.sign) return (*this) + -b;
83     if (absless(b)) return -(b - *this);
84     Int c;
85     for (int i = 0, borrow = 0; i < (int)v.size(); i++) {
86         borrow += v[i] - b.get_pos(i);
87         c.v.push_back(borrow);
88         c.v.back() -= 10000 * (borrow >= 31);
89     }
90     return c.setsign(sign, 0);
91 }
92 Int operator*(const Int &b) const {
93     if (b < *this) return b * *this;
94     Int c, d = b;
95     for (int i = 0; i < (int)v.size(); i++, d.v.insert(d.v.begin(),
        0))
96         c.add_mul(d, v[i]);
97     return c.setsign(sign * b.sign, 0);
98 }
99 Int operator/(const Int &b) const {
100     Int c, d;
101     d.v.resize(v.size());
102     double db = 1.0 / (b.v.back() + (b.get_pos((unsigned)b.v.size()
        - 2) / 1e4) +
103         (b.get_pos((unsigned)b.v.size() - 3) + 1) / 1e8);
104     for (int i = (int)v.size() - 1; i >= 0; i--) {
105         c.v.insert(c.v.begin(), v[i]);
106         int m = (int)((c.get_pos((int)b.v.size()) * 10000 + c.get_pos
            ((int)b.v.size() - 1)) * db);

```

```

107     c = c - b * m, c.setsign(c.sign, 0), d.v[i] += m;
108     while (!(c < b))
109         c = c - b, d.v[i] += 1;
110 }
111 return d.setsign(sign * b.sign, 0);
112 }
113 Int operator%(const Int &b) const { return *this - *this / b * b; }
114 bool operator>(const Int &b) const { return b < *this; }
115 bool operator<=(const Int &b) const { return !(b < *this); }
116 bool operator>=(const Int &b) const { return !(*this < b); }
117 bool operator!=(const Int &b) const { return !(*this == b); }
118 };

```

## 5 Dynamic Programming

### 5.1 最长公共子序列

```

1 if(a[i]==b[j])dp[i][j]=dp[i-1][j-1]+1;
2 else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);

```

### 5.2 背包

#### 5.2.1 01 背包

```

1 //一个旅行者有一个最多能装M公斤的背包，现在有n件物品，它们的重量分别是W1, W2
   .....Wn; 它们的价格为C1, C2.....Cn, 求旅行者能获得的最大总价值。
2
3 #include<bits/stdc++.h>
4 using namespace std;
5 int w[1003],c[1003],f[1003];
6 int n,m;
7 int main()
8 {
9     scanf("%d%d",&n,&m);
10    for(int i = 1;i <= n;i++)
11    {
12        scanf("%d%d",&w[i],&c[i]);
13        for(int j = m;j >= w[i];j--)
14        {
15            f[j] = max(f[j],f[j - w[i]] + c[i]);
16        }
17    }
18    printf("%d",f[m]);
19    return 0;
20 }

```

#### 5.2.2 完全背包

```

1 //有N种物品和一个容量为V的背包，每种物品都有无限件可以用。第i种物品的费用是w[i]
   , 价值是c[i]。求解将哪些物品装入背包的费用总和不超过背包容量，且价值总和最大。
2
3 #include<cstdio>
4 #include<cstring>
5 #include<cmath>
6 #include<algorithm>
7 using namespace std;
8 int m,n;
9 int w[1003],c[1003],f[1003];

```



```

10 int main()
11 {
12     scanf("%d%d",&n,&m);
13     for(int i = 1;i <= n;i++)
14     {
15         scanf("%d%d",&w[i],&c[i]);
16         for(int j = w[i];j <= m;j++)//01背包是倒序，完全背包是顺序
17         {
18             f[j] = max(f[j],f[j - w[i]] + c[i]);
19         }
20     }
21     printf("%d",f[m]);
22     return 0;
23 }

```

### 5.2.3 多重背包

```

1 //有N种物品和一个容量为V的背包。第i种物品最多有n[i]，每件费用是w[i]，价值是c[i]
2 //。求解将哪些物品放入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。
3 #include<stdio>
4 #include<string>
5 #include<cmath>
6 #include<algorithm>
7 using namespace std;
8 int m,n;
9 int w[1003],c[1003],s[1003],f[1003];
10 int main()
11 {
12     scanf("%d%d",&n,&m);
13     for(int i = 1;i <= n;i++)
14     {
15         scanf("%d%d%d",&w[i],&c[i],&s[i]);
16     }
17     for(int i = 1;i <= n;i++)
18     {
19         for(int j = m;j >= 0;j--)
20         {
21             for(int k = 0;k <= s[i];k++)
22             {
23                 if(j - k * w[i] < 0)break;
24                 f[j] = max(f[j],f[j - k * w[i]] + k * c[i]);
25             }
26         }
27     }
28     printf("%d",f[m]);

```

```

29     return 0;
30 }

```

### 5.2.4 多种背包混合问题

```

1 //一个旅行者有一个最多能装V公斤的背包，现在有n件物品，已知它们的价值和所占体积。有
2 //的物品可以一次取一件，有的物品可以取无限次，有的物品可以取得个数有上限。求最大
3 //价值。
4 //无法理解出题人为什么非要把这三个背包放在一起，有毒吧。。。
5 #include<stdio>
6 #include<string>
7 #include<cmath>
8 #include<algorithm>
9 using namespace std;
10 int m,n;
11 int w[1003],c[1003],s[1003],f[1003];
12 int main()
13 {
14     scanf("%d%d",&n,&m);
15     for(int i = 1;i <= n;i++)
16     {
17         scanf("%d%d%d",&w[i],&c[i],&s[i]);
18     }
19     for(int i = 1;i <= n;i++)
20     {
21         if(s[i] == 0)
22         {
23             for(int j = w[i];j <= m;j++)
24             {
25                 f[j] = max(f[j],f[j - w[i]] + c[i]);
26             }
27         }
28         else{
29             for(int j = 1;j <= s[i];j++)
30             {
31                 for(int k = m;k >= w[i];k--)
32                 {
33                     f[k] = max(f[k],f[k - w[i]] + c[i]);
34                 }
35             }
36         }
37     }
38     printf("%d",f[m]);
39     return 0;

```

## 5.2.5 二维费用的背包问题

//小橙书上关于这个讲了一大堆，大意就是限制条件有之前的一个变成了两个，就是由之前得背包重量，变成了现在的氢气和氧气的含量。

//例题：潜水员

//潜水员有一个带2种气体的气缸：一个为氧气，一个为氮气。他有一定数量的气缸。每个气缸都有重量和气体容量。潜水员为了完成他的工作需要特定数量的氧和氢。他完成工作所需气缸的总重的最低限度是多少

```
#include<stdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
//啊啊啊啊de了一万年bug终于过了
int u,v,k;
int f[101][101];
int a[1001],b[1001],c[1001];
int main()
{
    memset(f,10001,sizeof(f));//要把数组初始化为一个很大的数
    f[0][0] = 0;
    scanf("%d%d%d",&v,&u,&k);
    for(int i = 1;i <= k;i++)
    {
        scanf("%d%d%d",&a[i],&b[i],&c[i]);
    }
    for(int i = 1;i <= k;i++)
    {
        for(int j = v;j >= 0;j--)
        {
            for(int x = u;x >= 0;x--)
            {
                int t1 = j + a[i],t2 = x + b[i];//设一个变量方便计算
                if(t1 > v)t1 = v;//如果超出需求量就用需求量代替
                if(t2 > u)t2 = u;//这样就可以用f[v][u]来表示最优解了
                f[t1][t2] = min(f[t1][t2],f[j][x] + c[i]);
            }
        }
    }
    printf("%d",f[v][u]);
    return 0;
}
/*
5 60
```

```
5
3 36 120
10 25 129
5 50 250
1 45 130
4 20 119
*/
//正解是最后输出249
```

## 5.2.6 分组的背包问题

//有完没完，难道我真的要把自己的大好人生都用来学习背包吗!!!

//问题：

//有N件物品和一个容量为V的背包。第i件物品的费用是w[i]，价值是c[i]。这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。求解将哪些物品放入背包中可使这组物品的费用总和不超过背包容量，且价值总和最大

//对每一组物品，可以选择其中的一件或者一件都不选

```
#include<stdio>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
//这段代码简直有毒，我照着小橙书一个字一个字地敲得，结果竟然过不了样例
int v,n,t;
int w[33],c[33];
int a[13][33],dp[203];
int main()
{
    scanf("%d%d%d",&v,&n,&t);
    for(int i = 1;i <= n;i++)
    {
        int p;
        scanf("%d%d%d",&w[i],&c[i],&p);
        a[p][++a[p][0]] = i;//记录下来它属于哪个组
    }
    for(int x = 1;x <= t;x++)
    {
        for(int j = v;j >= 0;j--)
        {
            for(int i = 1;i <= a[x][0];i++)
            {
                if(j >= w[a[x][i]])
```

```

34     {
35         int q = a[x][i];
36         dp[j] = max(dp[j] - w[q]] + c[q],dp[j]);
37     }
38 }
39 }
40 }
41 printf("%d",dp[v]);
42 return 0;
43 }
44 /*
45 10 6 3
46 2 1 1
47 3 3 1
48 4 8 2
49 6 9 2
50 2 8 3
51 3 9 3
52 */
53
54 //输出 20

```

### 5.2.7 有依赖的背包问题

```

1  /*
2  这种背包问题的物品间存在某种依赖关系。若i依附于j，表示若选物品i，则必须选物品j。对
   每个主件i的附件集合先进行一次01背包，将主件转化为一个物品组，然后就可以按照分组
   背包来进行计算了。
3  详情见金明的预算方案https://blog.csdn.net/qq\_42914224/article/details/82594657
4  题目描述
5  金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明自己专用的很宽敞的房间。
   更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了
   算，只要不超过N元钱就行”。今天一早，金明就开始做预算了，他把想买的物品分为两
   类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：
6  主件 附件
7  电脑 打印机，扫描仪
8  书柜 图书
9  书桌 台灯，文具
10 工作椅 无
11 如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有0个、
12 1个或2个附件。附件不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的N
   元。于是，他把每件物品规定了一个重要度，分为5等：用整数
13 185表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是10元的整数倍）。他希望
   在不超过N元（可以等于N元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

```

设第*j*件物品的价格为 $v[j]$ ，重要度为 $w[j]$ ，共选中了*k*件物品，编号依次为  
 $j_1, j_2, \dots, j_k$ ，则所求的总和为：  
 $v[j_1] \times w[j_1] + v[j_2] \times w[j_2] + \dots + v[j_k] \times w[j_k]$ 。  
 请你帮助金明设计一个满足要求的购物单。  
 输入输出格式  
 输入格式：  
 第1行，为两个正整数，用一个空格隔开：  
*Nm*  
 （其中*N* ( $< 32000$ ) 表示总钱数，  
*m* ( $< 60$ ) 为希望购买物品的个数。） 从第2行到第  
*m*+1行，第*j*行给出了编号为*j*01的物品的的基本数据，每行有3个非负整数*vpq*（其中*v*表示该物  
 品的价格 ( $v < 10000$ )，*p*表示该物品的重要度 ( $185$ )，*q*表示该物品是主件还是附件。  
 如果*q*=0，表示该物品为主件，如果*q*>0，表示该物品为附件，*q*是所属主件的编号）

输出格式：

一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值 ( $< 200000$ )。

输入输出样例

输入样例#1:

```

32 1000 5
33 800 2 0
34 400 5 1
35 300 5 1
36 400 3 0
37 500 2 0
38 输出样例#1:
39 2200

```

对于每个主件和两个附件

可能存在五种情况

只选主件，主件和一个附件，主件和另一个附件，主件和两个附件，全都不选。

所以就枚举每个主件，然后对于这个主件比较五种情况的大小

\*/

```

47 #include<cstdio>
48 #include<cstring>
49 #include<cmath>
50 #include<algorithm>
51 using namespace std;
52 int n,m;
53 int c[63][63];
54 //昨晚没过就是因为数组的第二维开小了
55 int v[63],p[63],dp[32003];
56 int main()
57 {
58     scanf("%d%d",&n,&m);

```

```

59 n = n / 10;
60 for(int i = 1; i <= m; i++)
61 {
62     int q;
63     scanf("%d%d%d", &v[i], &p[i], &q);
64     v[i] = v[i] / 10;
65     p[i] *= v[i];
66     c[q][++c[q][0]] = i;
67     //主件为q的附件的个数 (q = 0时, 还能智能地记录主件个数)
68 }
69 for(int i = 1; i <= c[0][0]; i++) //枚举主件
70 {
71     for(int j = n; j >= v[i]; j--)
72     {
73         int v1 = (j - v[c[0][i]]) >= 0 ? dp[j - v[c[0][i]]] + p[c[0][i]] : 0;
74         int v2 = (j - v[c[0][i]] - v[c[c[0][i]][1]]) >= 0 ? dp[j - v[c[0][i]] - v[c[c[0][i]][1]]] + p[c[0][i]] + p[c[c[0][i]][1]] : 0;
75         int v3 = (j - v[c[0][i]] - v[c[c[0][i]][2]]) >= 0 ? dp[j - v[c[0][i]] - v[c[c[0][i]][2]]] + p[c[0][i]] + p[c[c[0][i]][2]] : 0;
76         int v4 = (j - v[c[0][i]] - v[c[c[0][i]][1]] - v[c[c[0][i]][2]]) >= 0 ? dp[j - v[c[0][i]] - v[c[c[0][i]][1]] - v[c[c[0][i]][2]]] + p[c[0][i]] + p[c[c[0][i]][1]] + p[c[c[0][i]][2]] : 0;
77         dp[j] = max(dp[j], max(v1, max(v2, max(v3, v4))));
78     }
79 }
80 printf("%d", dp[n] * 10);
81 return 0;
82 }

```

### 5.3 状压 dp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int INF=1e9;
4 int mp[50][50];
5 struct node
6 {
7     int now, set;
8 };
9 long long dp[21][1<<21];
10 bool vis[21][1<<21];
11 //设dp状态i,j表示当前在i城市, 已经访问的城市集合为j

```

```

12 queue<node>q;
13 int main()
14 {
15     int n;
16     scanf("%d", &n);
17     for(int i=1; i<=n; i++)
18         for(int j=1; j<=n; j++) mp[i][j]=INF;
19
20     for(int i=1; i<=n; i++)
21         for(int j=0; j<=1<<n; j++) dp[i][j]=INF;
22
23     int m;
24     scanf("%d", &m);
25     for(int i=1; i<=m; i++)
26     {
27         int ta, tb, tc;
28         scanf("%d%d%d", &ta, &tb, &tc);
29         mp[ta][tb]=min(mp[ta][tb], tc);
30         mp[tb][ta]=min(mp[tb][ta], tc);
31     }
32     dp[1][0]=0;
33     vis[1][0]=1;
34     q.push({1, 0});
35     while(!q.empty())
36     {
37         node p=q.front();
38         q.pop();
39         int now=p.now, set=p.set;
40         vis[now][set]=0;
41         for(int i=1; i<=n; i++)
42         {
43             if((set&(1<<(i-1)))||i==now);
44             else
45             {
46                 if(dp[i][set|(1<<(i-1))]>dp[now][set]+mp[now][i])
47                 {
48                     dp[i][set|(1<<(i-1))]=dp[now][set]+mp[now][i];
49
50                     if (vis[i][set|(1<<(i-1))]==0)
51                     {
52                         q.push({i, set|(1<<(i-1))});
53                         vis[i][set|(1<<(i-1))]=1;
54                     }
55                 }
56             }
57         }
58     }

```

```

59 if (n==1)printf("0");
60 else printf("%lld\n",dp[1][(1<n)-1]);
61 //dis[i]表示城市1到城市i的最短路距离
62 //dis[t]现在还不是最短路,那么肯定存在j,dis[j]+w(j,t)<dis[t]
63 //queue<int>q;q.push(i),q.front();
64 return 0;
65 }

```

## 5.4 数位 dp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int cnt=0;
4 int a[30];
5 long long dp[70][2];
6 long long dfs(int pos,int limit,int lead,int is6){
7     if(pos==0) return 1;
8     if(!limit&&dp[pos][is6]!=-1) return dp[pos][is6];
9     int up=(limit?a[pos]:9);
10    long long ans=0;
11    for(int i=0;i<=up;i++){
12        if(i==4) continue;
13        if(i==2&&is6) continue;
14        ans+=dfs(pos-1,limit&&(i==up),lead&&i==0,i==6);
15    }
16    if(!limit) dp[pos][is6]=ans;
17    return ans;
18 }
19
20 long long solve(long long x){
21    cnt=0;
22    while(x){
23        a[++cnt]=x%10;
24        x/=10;
25    }
26    return dfs(cnt,1,0,0);
27 }
28
29 int main(){
30    long long n,m;
31    memset(dp,-1,sizeof(dp));
32    while(scanf("%lld %lld",&n,&m)!=EOF){
33        if(n==0&&m==0) break;
34        printf("%lld\n",solve(m)-solve(n-1));
35    }
36 }

```

## 5.5 带障碍的路径计数

```

1 //复杂度为 $O(m^2)$ ,m为障碍的个数,此代码包含m个障碍和n个终点,每个终点互不影响
2 void solve()
3 {
4     binom_init();
5     int n,m;
6     cin>>n>>m;
7     vector<int>x(m+n+1),y(m+n+1),vis(m+n+1);
8     vector<Z>dp(m+n+1);
9     for(int i=1;i<=m;i++)cin>>x[i]>>y[i];
10    for(int i=1;i<=n;i++)x[m+i]=i,y[m+i]=n-i+1;
11    function<Z(int)>dfs=[&](int i)->Z
12    {
13        if(vis[i])return dp[i];
14        vis[i]=1;
15        Z res=0;
16        for(int j=1;j<=m;j++)
17        {
18            if(j!=i&&x[j]<=x[i]&&y[j]<=y[i])
19            {
20                res+=dfs(j)*binom(x[i]-x[j]+y[i]-y[j],x[i]-x[j]);
21            }
22        }
23        return dp[i]=binom(x[i]-1+y[i]-1,x[i]-1)-res;
24    };
25    Z ans=0;
26    for(int i=m+1;i<=n+m;i++)
27    {
28        ans+=dfs(i);
29    }
30    cout<<ans;
31 }

```

## 6 Geometry

### 6.1 二维

```

1 namespace Geometry{
2     const double eps=1e-9;
3     const double PI=acos(-1.0);
4     inline int dcmp(double x){
5         if(fabs(x)<eps)return 0;
6         return x<0?-1:1;
7     }
8     inline double Hypot(double a,double b){
9         return sqrt(a*a+b*b);
10    }
11    struct Point{
12        double x,y;
13        Point(){ }
14        Point(double x,double y):x(x),y(y){ }
15    };
16    typedef Point Vector;
17    Vector operator + (const Vector &a,const Vector &b){
18        return Vector(a.x+b.x,a.y+b.y);
19    }
20    Vector operator - (const Vector &a,const Vector &b){
21        return Vector(a.x-b.x,a.y-b.y);
22    }
23    Vector operator * (const Vector &a,const double k){
24        return Vector(a.x*k,a.y*k);
25    }
26    Vector operator / (const Vector &a,const double k){
27        return Vector(a.x/k,a.y/k);
28    }
29    bool operator == (const Vector &a,const Vector &b){
30        return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
31    }
32    inline double Dot(Vector a,Vector b){
33        return a.x*b.x+a.y*b.y;
34    }
35    inline double Cross(Vector a,Vector b){
36        return a.x*b.y-a.y*b.x;
37    }
38    inline double Length(Vector a){
39        return sqrt(Dot(a,a));
40    }
41    inline double Angle(Vector a,Vector b){
42        return acos(Dot(a,b)/(Length(a)*Length(b)));
43    }

```

```

44    inline double Distance(Point a,Point b){
45        return Hypot(a.x-b.x,a.y-b.y);
46    }
47    inline double TriangleArea(Point &a,Point &b,Point &c){
48        return fabs(Cross(b-a,c-a)/2);
49    }
50    inline Vector Rotate(Vector a,double ang){
51        return Vector(a.x*cos(ang)-a.y*sin(ang),a.x*sin(ang)+a.y*cos(ang));
52    }
53    struct StdLine{
54        double a,b,c,angle;
55        Point s,e;
56        StdLine(){ }
57        StdLine(Point _s,Point _e){
58            s=_s;
59            e=_e;
60            a=s.y-e.y;
61            b=e.x-s.x;
62            c=s.x*e.y-e.x*s.y;
63            angle=atan2(e.y-s.y,e.x-s.x);
64        }
65    };
66    struct Line{
67        Point s,e;
68        Line(){ }
69        Line(Point _s,Point _e){
70            s=_s;
71            e=_e;
72        }
73        bool isPointOnSegment(Point &p){
74            return dcmp(Cross(p-s,e-s))==0&&dcmp(Dot(p-s,p-e))<=0;
75        }
76        bool isPointOnLine(Point &p){
77            return dcmp(Cross(p-s,e-s))==0;
78        }
79        bool isParallel(Line &l){
80            return dcmp(Cross(e-s,l.e-l.s))==0;
81        }
82        double getDistance(Point &p){
83            return fabs(Cross(p-s,p-e)/Distance(s,e));
84        }
85        double getDistanceToSegment(Point &p){
86            if(Dot(e-s,p-s)<0)return Distance(s,p);
87            if(Dot(s-e,p-e)<0)return Distance(e,p);
88            return getDistance(p);
89        }

```

```

90 Point intersectPoint(Line &l){
91     return s+(e-s)*(Cross(l.e-l.s,l.s-s)/Cross(l.e-l.s,e-s));
92 }
93 Point symmetryPoint(Point &p){
94     double a=e.x-s.x,b=e.y-s.y;
95     double t=((p.x-s.x)*a+(p.y-s.y)*b)/(a*a+b*b);
96     return Point(2*s.x+2*a*t-p.x,2*s.y+2*b*t-p.y);
97 }
98 Line getPlumbLine(double inc=20.0){
99     Point p=Point((s.x+e.x)/2.0,(s.y+e.y)/2.0);
100     if(dcmp(s.x-e.x)==0)return Line(p,Point(p.x+inc,p.y));
101     if(dcmp(s.y-e.y)==0)return Line(p,Point(p.x,p.y+inc));
102     return Line(p,Point(p.x+inc,p.y-(inc/((s.y-e.y)/(s.x-e.x)))));
103 }
104 };
105 typedef Point Segment;
106 inline int getIntersectSegment(Segment a,Segment b,Segment &res){
107     if(a.y<b.x||a.x>b.y){
108         res=Segment(0,0);
109         return 0;
110     }
111     if(a.x<=b.x&&a.y>=b.y)res=b;
112     else if(b.x<=a.x&&b.y>=a.y)res=a;
113     else res=Segment(max(a.x,b.x),min(a.y,b.y));
114     return 1;
115 }
116 struct Rectangle{
117     Point a,b;
118     Rectangle(){}
119     Rectangle(Point a,Point b):a(a),b(b){}
120     double area(){
121         return fabs((a.x-b.x)*(a.y-b.y));
122     }
123     Rectangle getIntersect(Rectangle &r){
124         Segment x,y;
125         getIntersectSegment(Segment(a.x,b.x),Segment(r.a.x,r.b.x),x);
126         getIntersectSegment(Segment(a.y,b.y),Segment(r.a.y,r.b.y),y);
127         return Rectangle(Point(x.x,y.x),Point(x.y,y.y));
128     }
129 };
130 struct Polygon{
131     int n; vector<Point> p;
132     Polygon(){n=0;}
133     void init(){
134         p.clear();
135         n=0;

```

```

136     }
137     void push(Point &a){
138         p.push_back(a);
139         n++;
140     }
141     void pop(){
142         p.pop_back();
143         n--;
144     }
145     Point back(){
146         return p[n-1];
147     }
148     bool insidePolygon(Point &a){
149         double ang=0;
150         for(int i=0;i<n;i++)
151             ang+=Angle(p[i]-a,p[(i+1)%n]-a);
152         return dcmp(fabs(ang)-PI*2)==0;
153     }
154     double area(Point &o){
155         double sum=0;
156         for(int i=0;i<n;i++)
157             sum+=Cross(p[i]-o,p[(i+1)%n]-o);
158         return fabs(sum/2);
159     }
160     Polygon graham(){//凸包
161         for(int i=1;i<n;i++)
162             if(p[i].y<p[0].y||dcmp(p[i].y-p[0].y)==0&&p[i].x<p[0].x)
163                 swap(p[i],p[0]);
164         sort(p.begin()+1,p.end(),[&](Point a,Point b){
165             double r=Cross(a-p[0],b-p[0]);
166             if(dcmp(r)==0)return dcmp(Distance(p[0],a)-Distance(p[0],b))<0;
167             return r>0;
168         });
169         Polygon poly;
170         poly.push(p[0]);
171         poly.push(p[1]);
172         for(int i=2;i<n;i++){
173             while(poly.n>=2&&Cross(poly.p[poly.n-1]-poly.p[poly.n-2],p[i]-poly.p[poly.n-2])<0)
174                 poly.pop();
175             poly.push(p[i]);
176         }
177         return poly;
178     }
179 };
180 struct Circle{

```



```

181 Point o; double r;
182 Circle(){}
183 Circle(Point o,double r):o(o),r(r){}
184 double getArcLength(double ang){
185     return ang*r;
186 }
187 double getSectorArea(double ang){
188     return getArcLength(ang)*r/2;
189 }
190 Point getPoint(double ang){
191     return Point(o.x+r*cos(ang),o.y+r*sin(ang));
192 }
193 int getTangentPoints(Point &p,vector<Point> &vec){
194     double d=Distance(p,o),b=atan2(p.y-o.y,p.x-o.x);
195     double delta=acos(r/d);
196     vec.push_back(getPoint(b+delta));
197     if(dcmp(delta)){
198         vec.push_back(getPoint(b-delta));
199         return 2;
200     }
201     return 1;
202 }
203 int getIntersectPoints(Line &l,vector<Point> &vec){
204     if(dcmp(l.getDistance(o)-r)>0)return 0;
205     Vector v=l.e-l.s;
206     Point p=l.s+v*(Dot(o-l.s,v))/Dot(v,v);
207     double b=sqrt(r*r-Dot(p-o,p-o));
208     if(dcmp(b)==0){
209         vec.push_back(p);
210         return 1;
211     }
212     v=v/Length(v)*b;
213     vec.push_back(p+v);
214     vec.push_back(p-v);
215     return 2;
216 }
217 int getIntersectPoints(Circle &c,vector<Point> &vec){
218     double d=Distance(o,c.o);
219     if(dcmp(d)==0){
220         if(dcmp(r-c.r)==0)return -1;
221         return 0;
222     }
223     if(dcmp(r+c.r-d)<0)return 0;
224     if(dcmp(fabs(r-c.r)-d)>0)return 0;
225     double ang=Angle(c.o-o,Vector(1,0));
226     double delta=acos((r*r+d*d-c.r*c.r)/(2*r*d));
227     vec.push_back(getPoint(ang+delta));

```

```

228         if(dcmp(delta)){
229             vec.push_back(getPoint(ang-delta));
230             return 2;
231         }
232         return 1;
233     }
234 };
235 }
236 using namespace Geometry;

```

## 6.2 三维

```

1 namespace Geometry3 {
2     const double eps=1e-9;
3     inline int dcmp(double x){
4         if(fabs(x)<eps)return 0;
5         return x<0?-1:1;
6     }
7     inline double Hypot(double a,double b,double c){
8         return sqrt(a*a+b*b+c*c);
9     }
10    struct Point3{
11        double x,y,z;
12        Point3(){}
13        Point3(double x,double y,double z):x(x),y(y),z(z){}
14    };
15    typedef Point3 Vector3;
16    Vector3 operator + (const Vector3 &a,const Vector3 &b){
17        return Vector3(a.x+b.x,a.y+b.y,a.z+b.z);
18    }
19    Vector3 operator - (const Vector3 &a,const Vector3 &b){
20        return Vector3(a.x-b.x,a.y-b.y,a.z-b.z);
21    }
22    Vector3 operator * (const Vector3 &a,const double k){
23        return Vector3(a.x*k,a.y*k,a.z*k);
24    }
25    Vector3 operator / (const Vector3 &a,const double k){
26        return Vector3(a.x/k,a.y/k,a.z/k);
27    }
28    bool operator == (const Vector3 &a,const Vector3 &b){
29        return dcmp(a.x-b.x)==0&& dcmp(a.y-b.y)==0&& dcmp(a.z-b.z)==0;
30    }
31    inline double Dot(Vector3 a,Vector3 b){
32        return a.x*b.x+a.y*b.y+a.z*b.z;
33    }
34    inline double Length(Vector3 a){

```



```

35     return sqrt(Dot(a,a));
36 }
37 inline Vector3 CrossVector(Vector3 v1,Vector3 v2){
38     return Vector3(v1.y*v2.z-v1.z*v2.y,v1.z*v2.x-v1.x*v2.z,v1.x*v2.y
39         -v1.y*v2.x);
40 }
41 inline double Cross(Vector3 a,Vector3 b){
42     return Length(CrossVector(a,b));
43 }
44 inline double Distance(Point3 a,Point3 b){
45     return Hypot(a.x-b.x,a.y-b.y,a.z-b.z);
46 }
47 struct Line3{
48     Point3 s,e;
49     Line3(){}
50     Line3(Point3 _s,Point3 _e){
51         s=_s;
52         e=_e;
53     }
54     bool isPointOnSegment(Point3 &p){
55         return dcmp(Cross(p-s,e-s))==0&&dcmp(Dot(p-s,p-e))<=0;
56     }
57     bool isPointOnLine(Point3 &p){
58         return dcmp(Cross(p-s,e-s))==0;
59     }
60     bool isParallel(Line3 &l){
61         return dcmp(Cross(e-s,l.e-l.s))==0;
62     }
63     double getDistance(Point3 &p){
64         return fabs(Cross(p-s,p-e)/Distance(s,e));
65     }
66     double getDistanceToSegment(Point3 &p){
67         if(Dot(e-s,p-s)<0)return Distance(s,p);
68         if(Dot(s-e,p-e)<0)return Distance(e,p);
69         return getDistance(p);
70     }
71     double getDistanceToLine(Line3 &l){
72         Vector3 x=CrossVector(s-e,l.s-l.e);
73         return fabs(Dot(x,s-l.s))/Length(x);
74     }
75     double getAngleCosineToLine(Line3 &l){
76         return Dot(s-e,l.s-l.e)/Length(s-e)/Length(l.s-l.e);
77     }
78 };
79 struct Plane{
80     Point3 a,b,c,v;
81     Plane(){}

```

```

81     Plane(Point3 _a,Point3 _b,Point3 _c){
82         a=_a;
83         b=_b;
84         c=_c;
85         v=CrossVector(a-b,a-c);
86     }
87     Plane(Point3 _a,Vector3 _v){
88         a=_a;
89         v=_v;
90     }
91     bool isPointOnTriangle(Point3 &p){
92         return dcmp(Cross(a-b,a-c)-Cross(p-a,p-b)-Cross(p-b,p-c)-
93             Cross(p-c,p-a))==0;
94     }
95     bool isParallel(Plane &p){
96         return dcmp(Cross(v,p.v))==0;
97     }
98     int getIntersectLine(Plane &p,Line3 &l){
99         Vector3 r=CrossVector(v,p.v),v=CrossVector(v,r);
100         double d=Dot(p.v,v);
101         if(dcmp(d)==0) return 0;
102         l.s=a+v*(Dot(p.v,p.a-a)/d);
103         l.e=l.s+r;
104         return 1;
105     }
106     double getDistance(Point3 &p){
107         return fabs(Dot(v,p-a))/Length(v);
108     }
109     double getAngleCosineToPlane(Plane &p){
110         return Dot(v,p.v)/Length(v)/Length(p.v);
111     }
112     double getAngleSineToLine(Line3 &l){
113         return Dot(l.s-l.e,v)/Length(l.s-l.e)/Length(v);
114     }
115 };
116 struct Circle3{
117     Point3 o; double r;
118     Circle3(){}
119     Circle3(Point3 o,double r):o(o),r(r){}
120     int getIntersectPoints(Line3 &l,vector<Point3> &vec){
121         if(dcmp(l.getDistance(o)-r)>0)return 0;
122         Vector3 v=l.e-l.s;
123         Point3 p=l.s+v*(Dot(o-l.s,v))/Dot(v,v);
124         double b=sqrt(r*r-Dot(p-o,p-o));
125         if(dcmp(b)==0){
126             vec.push_back(p);
127             return 1;

```

```

127     }
128     v=v/Length(v)*b;
129     vec.push_back(p+v);
130     vec.push_back(p-v);
131     return 2;
132 }
133 };
134 }
135 using namespace Geometry3;

```

## 7 Others

### 7.1 头文件

```

1 #include<algorithm>
2 #include<cctype>
3 #include<chrono> //poj不能用
4 #include<cmath>
5 #include<cstdio>
6 #include<cstring>
7 #include<functional>
8 #include<iomanip>
9 #include<iostream>
10 #include<map>
11 #include<queue>
12 #include<random> //poj不能用
13 #include<set>
14 #include<sstream>
15 #include<stack>
16 #include<string>
17 #include<utility>
18 #include<vector>

```

### 7.2 吸氧大法

```

1 #pragma GCC optimize(2)
2 #pragma GCC optimize(3)
3 #pragma GCC optimize("Ofast")
4 #pragma GCC optimize("inline")
5 #pragma GCC optimize("-fgcse")
6 #pragma GCC optimize("-fgcse-lm")
7 #pragma GCC optimize("-fipa-sra")
8 #pragma GCC optimize("-ftree-pre")
9 #pragma GCC optimize("-ftree-vrp")
10 #pragma GCC optimize("-fpeephole2")
11 #pragma GCC optimize("-ffast-math")
12 #pragma GCC optimize("-fsched-spec")
13 #pragma GCC optimize("unroll-loops")
14 #pragma GCC optimize("-falign-jumps")
15 #pragma GCC optimize("-falign-loops")
16 #pragma GCC optimize("-falign-labels")
17 #pragma GCC optimize("-fdevirtualize")
18 #pragma GCC optimize("-fcaller-saves")
19 #pragma GCC optimize("-fcrossjumping")
20 #pragma GCC optimize("-fthread-jumps")
21 #pragma GCC optimize("-funroll-loops")

```

```

22 #pragma GCC optimize("-fwhole-program")
23 #pragma GCC optimize("-freorder-blocks")
24 #pragma GCC optimize("-fschedule-insns")
25 #pragma GCC optimize("inline-functions")
26 #pragma GCC optimize("-ftree-tail-merge")
27 #pragma GCC optimize("-fschedule-insns2")
28 #pragma GCC optimize("-fstrict-aliasing")
29 #pragma GCC optimize("-fstrict-overflow")
30 #pragma GCC optimize("-falign-functions")
31 #pragma GCC optimize("-fcse-skip-blocks")
32 #pragma GCC optimize("-fcse-follow-jumps")
33 #pragma GCC optimize("-fsched-interblock")
34 #pragma GCC optimize("-fpartial-inlining")
35 #pragma GCC optimize("no-stack-protector")
36 #pragma GCC optimize("-freorder-functions")
37 #pragma GCC optimize("-findirect-inlining")
38 #pragma GCC optimize("-fhoist-adjacent-loads")
39 #pragma GCC optimize("-frerun-cse-after-loop")
40 #pragma GCC optimize("inline-small-functions")
41 #pragma GCC optimize("-finline-small-functions")
42 #pragma GCC optimize("-ftree-switch-conversion")
43 #pragma GCC optimize("-foptimize-sibling-calls")
44 #pragma GCC optimize("-fexpensive-optimizations")
45 #pragma GCC optimize("-funsafe-loop-optimizations")
46 #pragma GCC optimize("inline-functions-called-once")
47 #pragma GCC optimize("-fdelete-null-pointer-checks")
48 // 简易
49 #pragma GCC optimize(2)
50 #pragma GCC optimize(3)
51 #pragma GCC optimize("Ofast,unroll-loops")

```

### 7.3 迭代器

```

1 vector<int>v(10,1);
2 for(vector<int>::iterator it=v.begin();it!=v.end();it++)
3 {
4     // ...
5 }

```

### 7.4 lambda 函数

```

1 function<void(int,int)>dfs=[&](int now,int flag)
2 {
3     // ...

```

```

4 };
5 function<int(int,int)>func=[&](int x,int y)->int
6 {
7     // ...
8 };

```

### 7.5 结构体

```

1 struct node
2 {
3     string name;
4     int cnt;
5     node(){}
6     node(string s,int c){name=s,cnt=c;}
7     bool operator<(const node& a)const // 结构体排序
8     {
9         if(cnt!=a.cnt)
10            return cnt>a.cnt; // 先按数量从大到小
11        if(name.size()!=a.name.size())
12            return name.size()>a.name.size(); // 再按标签长度从小到大
13        return name<a.name; // 最后按字典序, string可以直接进行对比
14    }
15 };

```

### 7.6 快读快写

```

1 namespace fastIO
2 {
3     #define gc h==d&&(d=(h=buf)+fread(buf,1,100000,stdin),h==d)?EOF:*h
4     ++ // 不能用fread则换成getchar
5     static char buf[100000],*h=buf,*d=buf; // 缓存开大可减少读入时间, 看题目给
6     的空间
7     template<typename T>
8     inline void read(T&x)
9     {
10         int f=1;x=0;char c(gc);
11         while(c>'9' || c<'0'){if(c=='-') f=-1;c=gc;}
12         while(c<='9' && c>='0')x=(x<<1)+(x<<3)+(c^48),c=gc;
13         x*=f;
14     }
15     template<typename T>
16     void print(T x)
17     {
18         if(x<0){putchar('-');x=~(x-1);}

```

```

17 static int s[20],top=0;
18 while(x){s[++top]=x%10;x/=10;}
19 if(!top)s[++top]=0;
20 while(top)putchar(s[top--]+'0');
21 }
22 }
23 using namespace fastIO;

```

```

1 namespace fastIO
2 {
3     #ifdef LOCAL
4     #define DEBUG
5     #endif
6     #define BUF_SIZE 100000
7     #define DECIMAL 6
8     #define LL long long
9     static bool IOerror=0;
10    #ifdef DEBUG
11    inline char nc(){char ch=getchar();if(ch== -1)IOerror=1;return ch;}
12    #else
13    inline char nc(){static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=
        buf+BUF_SIZE;if(p1==pend){p1=buf;pend=buf+fread(buf,1,BUF_SIZE,
        stdin);if(pend==p1){IOerror=1;return -1;}}return *p1++;}
14    #endif
15    inline bool blank(char ch){return ch==' '||ch=='\n'||ch=='\r'||ch==
        '\t';}
16    template<class T> inline bool read(T&x){bool sign=0;char ch=nc();x
        =0;for(;blank(ch);ch=nc());if(IOerror)return false;if(ch=='-')
        sign=1,ch=nc();for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';if(
        sign)x=-x;return true;}
17    inline bool read(unsigned long long&x){char ch=nc();x=0;for(;blank(
        ch);ch=nc());if(IOerror)return false;for(;ch>='0'&&ch<='9';ch=
        nc())x=x*10+ch-'0';return true;}
18    inline bool read(unsigned int&x){char ch=nc();x=0;for(;blank(ch);ch
        =nc());if(IOerror)return false;for(;ch>='0'&&ch<='9';ch=nc())x=
        x*10+ch-'0';return true;}
19    inline bool read(double&x){bool sign=0;char ch=nc();x=0;for(;blank(
        ch);ch=nc());if(IOerror)return false;if(ch=='-')sign=1,ch=nc();
        for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';if(ch=='.'){double
        tmp=1;ch=nc();for(;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(
        ch-'0');if(sign)x=-x;return true;}
20    inline bool read(char*s){char ch=nc();for(;blank(ch);ch=nc());if(
        IOerror)return false;for(;!blank(ch)&&!IOerror;ch=nc())*s++=ch
        ;*s=0;return true;}
21    inline bool read(char&c){for(c=nc();blank(c);c=nc());if(IOerror){c
        =-1;return false;}return true;}
22    inline bool read(std::string&s){s.clear();char ch=nc();for(;blank(

```

```

        ch);ch=nc());if(IOerror)return false;for(;!blank(ch)&&!IOerror;
        ch=nc())s+=ch;return true;}
23    template<class T,class ... U> bool read(T&h,U& ... t){return read(h)
        &&read(t ... );}
24    struct Ostream_fwrite
25    {
26        #ifdef DEBUG
27        inline void out(char ch) { putchar(ch); }
28        #else
29        char*buf,*p1,*pend;
30        Ostream_fwrite(){buf=new char[BUF_SIZE];p1=buf;pend=buf+BUF_SIZE
            ;}
31        inline void out(char ch){if(p1==pend){fwrite(buf,1,BUF_SIZE,
            stdout);p1=buf;}*p1++=ch;}
32        #endif
33        template<class T> inline void print(T x){static char s[41],*s1;
            s1=s;if(!x)*s1++='0';if(x<0)out('-'),x=-x;while(x)*s1++=x
            %10+'0',x/=10;while(s1--!=s)out(*s1);}
34        inline void print(char ch){out(ch);}
35        inline void print(unsigned long long x){static char s[41],*s1;s1
            =s;if(!x)*s1++='0';while(x)*s1++=x%10+'0',x/=10;while(s1--!=
            s)out(*s1);}
36        inline void print(unsigned int x){static char s[41],*s1;s1=s;if
            (!x)*s1++='0';while(x)*s1++=x%10+'0',x/=10;while(s1--!=s)out
            (*s1);}
37        inline void print(double x,int y=DECIMAL){static LL mul
            []={1,10,100,1000,10000,100000,
            1000000,10000000,100000000,1000000000,
            10000000000LL,100000000000LL,
            1000000000000LL,10000000000000LL,
            100000000000000LL,1000000000000000LL,
            10000000000000000LL,10000000000000000LL};
            if(x<-1e-12)out('-'),x=-x;x*=mul[y];LL x1=(LL)floor(x);if(x-
            floor(x)>=0.5)++x1;LL x2=x1/mul[y],x3=x1-x2*mul[y];print(
            x2);if(y>0){out('.');for(size_t i=1;i<y&&x3*mul[i]<mul[y]
            ;out('0'),++i);print(x3);}
44        inline void print(char*s){while(*s)out(*s++);}
45        inline void print(const char*s){while(*s)out(*s++);}
46        inline void print(std::string s){print(s.c_str());}
47        #ifdef DEBUG
48        inline void flush(){if(p1!=buf){fwrite(buf,1,p1-buf,stdout);p1=
            buf;}}
49        ~Ostream_fwrite(){flush();}
50        #endif // !DEBUG
51    }Ostream;
52    template<class T> inline void print(T x){Ostream.print(x);}
53    template<class T> inline void println(T x){Ostream.print(x);Ostream

```

```

        .out('\n');}
54 inline void print(double x,int y=DECIMAL){Ostream.print(x,y);}
55 inline void println(double x,int y=DECIMAL){Ostream.print(x,y);
    Ostream.out('\n');}
56 template<class T,class ... U> void print(const T&h,const U&... t){
    print(h);Ostream.out(' ');print(t...);}
57 template<class T,class ... U> void println(const T&h,const U&... t){
    print(h);Ostream.out(' ');println(t...);}
58 #ifndef DEBUG
59 inline void flush(){Ostream.flush();}
60 #endif
61 #undef LL
62 #undef DECIMAL
63 #undef BUF_SIZE
64 //本地 getchar/putchar , 提交 fread/fwrite
65 //本地支持终端交互输入输出
66 }
67 using namespace fastIO;

```

```

1 void read(int& x)
2 {
3     char c = getchar(); x = 0;
4     while (c < '0' || c > '9') c = getchar();
5     while (c <= '9' && c >= '0') x = x * 10 + c - 48, c = getchar();
6 }
7 inline int read()
8 {
9     int s=0,w=1;char ch=getchar();
10    for(;!isdigit(ch);ch=getchar())
11        if(ch=='-')w=-1;
12    for(;ch>='0'&&ch<='9';ch=getchar())
13        s=(s<<1)+(s<<3)+(ch^48);
14    return (w==1?-s:s);
15 }
16 inline void print(int x) {
17     if (x == 0){printf("0\n");return;}
18     int num = 0; char c[15];
19     while (x) c[++num] = (x % 10) + 48, x /= 10;
20     while (num) putchar(c[num--]);
21     putchar('\n');
22 }

```

## 7.7 取模

```

1 template<typename T>
2 int normalize(T value, int mod) {

```

```

3     if (value < -mod || value >= 2 * mod) value %= mod;
4     if (value < 0) value += mod;
5     if (value >= mod) value -= mod;
6     return value;
7 }
8
9 template<int mod>
10 struct static_modular_int {
11     using mint = static_modular_int<mod>;
12
13     int value;
14
15     static_modular_int() : value(0) {}
16     static_modular_int(const mint &x) : value(x.value) {}
17
18     template<typename T, typename U = std::enable_if_t<std::is_integral<T>::value>>
19     static_modular_int(T value) : value(normalize(value, mod)) {}
20
21     template<typename T>
22     mint power(T degree) const {
23         degree = normalize(degree, mod - 1);
24         mint prod = 1, a = *this;
25         for (; degree > 0; degree >>= 1, a *= a)
26             if (degree & 1)
27                 prod *= a;
28
29         return prod;
30     }
31
32     mint inv() const {
33         return power(-1);
34     }
35
36     mint& operator=(const mint &x) {
37         value = x.value;
38         return *this;
39     }
40
41     mint& operator+=(const mint &x) {
42         value += x.value;
43         if (value >= mod) value -= mod;
44         return *this;
45     }
46
47     mint& operator-=(const mint &x) {
48         value -= x.value;

```

```

49     if (value < 0) value += mod;
50     return *this;
51 }
52
53 mint& operator*=(const mint &x) {
54     value = int64_t(value) * x.value % mod;
55     return *this;
56 }
57
58 mint& operator/=(const mint &x) {
59     return *this *= x.inv();
60 }
61
62 friend mint operator+(const mint &x, const mint &y) {
63     return mint(x) += y;
64 }
65
66 friend mint operator-(const mint &x, const mint &y) {
67     return mint(x) -= y;
68 }
69
70 friend mint operator*(const mint &x, const mint &y) {
71     return mint(x) *= y;
72 }
73
74 friend mint operator/(const mint &x, const mint &y) {
75     return mint(x) /= y;
76 }
77
78 mint& operator++() {
79     ++value;
80     if (value == mod) value = 0;
81     return *this;
82 }
83
84 mint& operator--() {
85     --value;
86     if (value == -1) value = mod - 1;
87     return *this;
88 }
89
90 mint operator++(int) {
91     mint prev = *this;
92     value++;
93     if (value == mod) value = 0;
94     return prev;
95 }

```

```

96
97 mint operator--(int) {
98     mint prev = *this;
99     value--;
100     if (value == -1) value = mod - 1;
101     return prev;
102 }
103
104 mint operator-() const {
105     return mint(0) - *this;
106 }
107
108 bool operator==(const mint &x) const {
109     return value == x.value;
110 }
111
112 bool operator!=(const mint &x) const {
113     return value != x.value;
114 }
115
116 bool operator<(const mint &x) const {
117     return value < x.value;
118 }
119
120 template<typename T>
121 explicit operator T() {
122     return value;
123 }
124
125 friend std::istream& operator>>(std::istream &in, mint &x) {
126     std::string s;
127     in >> s;
128     x = 0;
129     for (const auto c : s)
130         x = x * 10 + (c - '0');
131
132     return in;
133 }
134
135 friend std::ostream& operator<<(std::ostream &out, const mint &x) {
136     return out << x.value;
137 }
138
139 static int primitive_root() {
140     if constexpr (mod == 1000000007) return 5;
141     if constexpr (mod == 998244353) return 3;
142     if constexpr (mod == 786433) return 10;

```



```

143     static int root = -1;
144     if (root != -1)
145         return root;
146
147     std::vector<int> primes;
148     int value = mod - 1;
149     for (int i = 2; i * i <= value; i++)
150         if (value % i == 0) {
151             primes.push_back(i);
152             while (value % i == 0)
153                 value /= i;
154         }
155
156     if (value != 1) primes.push_back(value);
157     for (int r = 2;; r++) {
158         bool ok = true;
159         for (auto p : primes) {
160             if ((mint(r).power((mod - 1) / p)).value == 1) {
161                 ok = false;
162                 break;
163             }
164         }
165         if (ok) return root = r;
166     }
167 }
168
169 };
170
171 // constexpr int MOD = 1000000007;
172 constexpr int MOD = 998244353;
173 using mint = static_modular_int<MOD>;

```

## 7.8 Int128

```

1 std::ostream &operator<<(std::ostream &os, i128 n) {
2     std::string s;
3     while (n) {
4         s += '0' + n % 10;
5         n /= 10;
6     }
7     std::reverse(s.begin(), s.end());
8     return os << s;
9 }
10
11 i128 sqrt(i128 x) //开方
12 {

```

```

13     i128 y=(i128)ceil(sqrt((long double)x));
14     for(;y*y<=x;++y);
15     for(--y;y*y>x;--y);
16     return y;
17 }

```

## 7.9 gp\_hash\_table

```

1 #include<bits/extc++.h>
2 using namespace __gnu_pbds;
3
4 struct custom_hash {
5     static uint64_t splitmix64(uint64_t x) {
6         // http://xorshift.di.unimi.it/splitmix64.c
7         x += 0x9e3779b97f4a7c15;
8         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
9         x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
10        return x ^ (x >> 31);
11    }
12
13    size_t operator()(uint64_t x) const {
14        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now()
15            .time_since_epoch().count();
16        return splitmix64(x + FIXED_RANDOM);
17    }
18 };
19 gp_hash_table<long long, int, custom_hash> safe_hash_table;

```

## 7.10 关同步流

```

1 ios::sync_with_stdio(false);cin.tie(0);

```

## 7.11 计时器

```

1 clock_t start,finish;
2 double totaltime;
3 start=clock();
4 // ...
5 finish=clock();
6 totaltime=(double)(finish-start)/CLOCKS_PER_SEC;
7 cout<<"\n此程序的运行时间为"<<totaltime<<"秒！"<<endl;

```