

Template of Team DFA

HPU : Codancer & Dicer

2019 年 9 月 26 日

目录

1 杂项	1	5.8 AC 自动机	18
1.1 Head & 快速读入	1	5.9 最小表示法	18
1.2 <code>__int128</code> 输入输出	1	6 C++ pbds	19
1.3 O3 优化	1	6.1 头文件	19
1.4 单调栈	1	6.2 Hash	19
1.5 打印 LIS	1	6.3 Tree	19
2 图论	1	6.4 Trie	19
2.1 Dinic 最大流	1	6.5 优先队列	19
2.2 倍增求解 LCA	2		
2.3 有向图最小环	3		
2.4 Tarjan	3		
2.4.1 缩点求 SCC	3		
2.4.2 求割点	3		
2.4.3 无向图点双连通	4		
2.4.4 无向图边双连通	4		
2.5 求无向连通图的第 K 大联通子图	5		
2.6 分层 (K) 图最短路	5		
3 数学	6		
3.1 整除分块	6		
3.2 SG 函数打表	6		
3.3 线性素数 + 莫比乌斯函数打表	6		
3.4 Lucas 定理求 $C(n, m) \% P$	6		
3.5 大数质因子分解 & 大素数检测	7		
3.6 python 通用中国剩余定理	7		
3.7 在线求组合数	8		
3.8 拉格朗日插值	8		
3.8.1 连续情况	8		
3.8.2 非连续情况	8		
3.9 辛普森自适应积分	9		
3.10 欧拉函数	9		
3.10.1 在线	9		
3.10.2 打表	9		
3.11 欧拉降幂	9		
4 数据结构	10		
4.1 线段树	10		
4.1.1 区间修改区间查询	10		
4.2 主席树	11		
4.2.1 区间第 k 小	11		
4.2.2 区间内小于等于 x 的最大值	11		
4.2.3 区间内距离 p 第 k 近的距离	12		
4.2.4 区间内小于等于 x 的最大值	12		
4.2.5 区间数的种类数	12		
4.2.6 区间内未出现过的最小自然数	13		
4.3 树链剖分	13		
4.4 树状数组	14		
4.4.1 二位偏序求矩形内点的个数	14		
4.4.2 树状数组求区间最值	15		
4.4.3 二维树状数组	16		
5 字符串	16		
5.1 序列自动机	16		
5.2 KMP 计算 next 函数	16		
5.2.1 vector 版	16		
5.2.2 KMP 匹配过程	16		
5.3 Z-function / Exkmp	16		
5.4 Manacher	16		
5.5 后缀数组	17		
5.6 双哈希	17		
5.7 字典树	17		
5.7.1 指针版	17		
5.7.2 数组版 01 字典树	18		

1 杂项

1.1 Head & 快速读入

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int N = 1e6+100;
4 const int mod = 1e9+7;
5 typedef long long ll;
6 const int INF = 0x3f3f3f3f;
7 const ll llINF = 0x3f3f3f3f3f3f3f3f;
8 #define rep(i,a,b) for(int i=(a);i<=(b);i++)
9 #define fep(i,a,b) for(int i=(a);i>=(b);i--)
10 inline bool read(ll &num) {
11     char in;bool IsN=false;
12     in=getchar();
13     if(in==EOF) return false;
14     while(in!='-'&&(in<'0'||in>'9')) in=getchar();
15     if(in=='-'){ IsN=true;num=0;}
16     else num=in-'0';
17     while(in=getchar(),in>='0'&&in<='9'){
18         num*=10,num+=in-'0';
19     }
20     if(IsN) num=-num;
21     return true;
22 }

```

1.2 ____int128 输入输出

```

1 void scan(__int128 &x)//输入{
2     x = 0;
3     int f = 1;
4     char ch;
5     if((ch = getchar()) == '-') f = -f;
6     else x = x*10 + ch-'0';
7     while((ch = getchar()) >= '0' && ch <= '9')
8         x = x*10 + ch-'0';
9     x *= f;
10 }
11 void print(__int128 x)//输出
12 {
13     if(x < 0)
14     {
15         x = -x;
16         putchar('-');
17     }
18     if(x > 9) print(x/10);
19     putchar(x%10 + '0');
20 }

```

1.3 O3 优化

```

1 #pragma GCC optimize(3,"Ofast","inline")

```

1.4 单调栈

求第 i 个数作为最大值的区间 $[l,n]$

```

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

```

```

4 const int N = 1e6+100;
5 int a[N];
6 int L[N],R[N];
7 int main(){
8     int n;
9     cin>>n;
10    stack<int> sta;
11    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
12    for(int i=1;i<=n;i++){
13        while(sta.size()&&a[sta.top()]<=a[i]) sta.pop();
14        if(sta.empty()) L[i]=1;
15        else L[i]=sta.top()+1;
16        sta.push(i);
17    }
18    while(sta.size()) sta.pop();
19    for(int i=n;i>=1;i--){
20        while(sta.size()&&a[sta.top()]<=a[i]) sta.pop();
21        if(sta.empty()) R[i]=n;
22        else R[i]=sta.top()-1;
23        sta.push(i);
24    }

```

1.5 打印 LIS

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = 60000;
5 const int INF = 0x3f3f3f3f;
6 int dp[N],fa[N],a[N],b[N],order[N],n,pos[N];
7 bool vis[N],ok[N];
8 int solve();//求b的LIS
9 int cnt=0;
10 for(int i=1;i<=n;i++){
11     if(!vis[i]) b[cnt++]=a[i];b数组
12 }
13 memset(dp,INF,sizeof(dp));
14 memset(ok,0,sizeof(ok));
15 int lpos;
16 pos[0]=-1;
17 for(int i=0;i<cnt;i++){
18     dp[lpos=(lower_bound(dp,dp+cnt,b[i])-dp)]=b[i];
19     pos[lpos]=i;
20     fa[i]=(lpos?pos[lpos-1]:-1);
21 }
22 cnt=lower_bound(dp,dp+cnt,INF)-dp;
23 int i;
24 for(i=pos[cnt-1];~fa[i];i=fa[i]){
25     ok[b[i]]=1;//说明b[i]在LIS内
26 }
27 ok[b[i]]=1;
28 return cnt;
29 }

```

2 图论

2.1 Dinic 最大流

```

1 #include<iostream>

```

```

2  #include<algorithm>
3  #include<string.h>
4  #include<cstdio>
5  #include<queue>
6  using namespace std;
7  const int INF = 0x3f3f3f3f;
8  typedef long long ll;
9
10
11
12
13 //Dinic最大流, 节点编号从0开始
14 struct MaxFlow{
15     const static ll MAX_V = 1005;
16     ll V;
17     //终点、容量、反向边
18     struct edge{
19         ll to, cap, rev;
20     };
21     vector<edge> G[MAX_V];
22     ll level[MAX_V]; //顶点到源点的距离标号
23     ll iter[MAX_V]; // 当前弧, 在其之前的边已经没有用了
24
25     void add_edge(ll from, ll to, ll cap){
26         G[from].push_back((edge){to, cap, (ll)G[to].
27             size()});
28         G[to].push_back((edge){from, 0, (ll)G[from].
29             size()-1});
30     }
31
32     // 通过BFS计算从源点出发的距离标号
33     void bfs(ll s){
34         fill(level, level + V, -1);
35         queue<ll> que;
36         level[s] = 0;
37         que.push(s);
38         while (!que.empty()){
39             ll v = que.front();
40             que.pop();
41             for (ll i=0; i< G[v].size(); i++){
42                 edge& e = G[v][i];
43                 if (e.cap > 0 && level[e.to] < 0){
44                     level[e.to] = level[v] + 1;
45                     que.push(e.to);
46                 }
47             }
48         }
49
50         //通过DFS寻找增广路
51         ll dfs(ll v, ll t, ll f){
52             if (v == t)
53                 return f;
54             for (ll &i = iter[v]; i < G[v].size(); i++){
55                 edge& e = G[v][i];
56                 if (e.cap > 0 && level[v] < level[e.to]){
57                     ll d = dfs(e.to, t, min(f, e.cap));
58                     if (d > 0){
59                         e.cap -= d;
60                         G[e.to][e.rev].cap += d;
61                         return d;
62                     }
63                 }
64             }
65             return 0;
66         }
67     }
68
69     ll max_flow(ll s, ll t){
70         ll flow = 0;
71         for (;;){
72             bfs(s);
73             if(level[t] < 0)
74                 return flow;
75             fill(iter, iter + V, 0);
76             ll f;
77             while ((f = dfs(s, t, INF)) > 0){
78                 flow += f;
79             }
80         }
81     }
82
83     void init(ll n = 0){
84         for (ll i = 0; i < V; i++){
85             G[i].clear();
86         }
87         V = n;
88     }
89 }mf;

```

```

65 }
66
67
68
69 //求解从s到t的最大流
70 ll max_flow(ll s, ll t){
71     ll flow = 0;
72     for (;;){
73         bfs(s);
74         if(level[t] < 0)
75             return flow;
76         fill(iter, iter + V, 0);
77         ll f;
78         while ((f = dfs(s, t, INF)) > 0){
79             flow += f;
80         }
81     }
82
83     void init(ll n = 0){
84         for (ll i = 0; i < V; i++){
85             G[i].clear();
86         }
87         V = n;
88     }
89 }mf;

```

2.2 倍增求解 LCA

```

1  #include<bits/stdc++.h>
2
3  using namespace std;
4  const int N = 1e6+100;
5  vector<int> G[N];
6  long long bit[30];
7  int f[N][30];
8  int depth[N];
9  void init(){
10     bit[0]=1;
11     for(int i=1;i<=29;i++) bit[i]=(bit[i-1]<<1);
12 }
13 void dfs(int u,int par){
14     depth[u]=depth[par]+1;
15     f[u][0]=par;
16     for(int i=1;bit[i]<=depth[u];i++) f[u][i]=f[f[u][i-1]][i-1];
17     for(int i=0;i<(int)G[u].size();i++){
18         int v=G[u][i];
19         if(v!=par) dfs(v,u);
20     }
21 }
22 int lca(int x,int y){
23     if(depth[x]<depth[y]) swap(x,y);
24     for(int i=29;i>=0;i--){
25         if(depth[x]-depth[y]>=bit[i]){
26             x=f[x][i];
27         }
28     }
29     if(x==y) return x;
30     for(int i=29;i>=0;i--){
31         if(depth[x]>=(1<<i)&&f[x][i]!=f[y][i]){
32             x=f[x][i];

```

```

33     y=f[y][i];
34 }
35 }
36 return f[x][0];
37 }

```

2.3 有向图最小环

```

1 rep(k,1,n){
2     rep(i,1,k-1){
3         rep(j,1,i-1){
4             ans=min(ans,dis[i][j]+val[i][k]+val[k][j]);
5             //val代表边权
6         }
7     }
8     rep(i,1,n){
9         rep(j,1,n){
10            dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j]);
11        }
12    }
13 }

```

2.4 Tarjan

2.4.1 缩点求 SCC

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = 1e5+100;
5 typedef long long ll;
6 const int INF = 0x3f3f3f3f;
7 int n,m,scc,index;
8 vector<int> G[N];
9 ll w[N],low[N],dfn[N],minn[N],color[N],id[N];
10 bool is_instack[N];stack<int> sta;
11 //不要忘记初始化!!!!
12 void init(){
13     scc=index=0;
14     memset(low,0,sizeof(low));
15     memset(dfn,0,sizeof(dfn));
16     memset(color,0,sizeof(color));
17     memset(minn,INF,sizeof(minn));
18     memset(is_instack,0,sizeof(is_instack));
19     for(int i=1;i<=n;i++) G[i].clear();
20     while(!sta.empty()) sta.pop();
21 }
22 void Tarjan(int u){
23     low[u]=dfn[u]=++index;
24     sta.push(u);is_instack[u]=1;
25     for(auto v:G[u]){
26         if(!dfn[v]){
27             Tarjan(v);
28             low[u]=min(low[u],low[v]);
29         }
30         else if(is_instack[v]){
31             low[u]=min(low[u],dfn[v]);
32         }
33     }
34     if(low[u]==dfn[u]){
35         ++scc;
36         while(1){

```

```

37             int temp=sta.top();
38             color[temp]=scc;
39             minn[scc]=min(minn[scc],w[temp]);
40             is_instack[temp]=0;
41             sta.pop();
42             if(temp==u) break;
43         }
44     }
45 }
46
47 //main函数Tarjan用法
48 main:
49 init();
50 for(int i=1;i<=n;i++){
51     if(!dfn[i]) Tarjan(i);
52 }

```

2.4.2 求割点

割点去掉后各联通快大小

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = 1e5+100;
5 typedef long long ll;
6 int dfn[N],low[N];
7 ll n,m,siz[N];
8 int idx;
9 vector<ll> G[N],GD[N];
10 bool jud[N];//是否为割点
11 void Tarjan(int u,int fa){
12     dfn[u]=low[u]=++idx;
13     siz[u]=1;
14     int allsiz=0;
15     for(int v:G[u]){
16         if(!dfn[v]){//没有访问过
17             Tarjan(v,u);
18             siz[u]+=siz[v];
19             low[u]=min(low[u],low[v]);
20             if(low[v]>=dfn[u]){//u为割点
21                 jud[u]=1;
22                 allsiz+=siz[v];
23                 GD[u].push_back(siz[v]);
24             }
25         }
26         else if(v!=fa){
27             low[u]=min(low[u],low[v]);
28         }
29     }
30     if(jud[u]&&n-allsiz-1){
31         GD[u].push_back(n-allsiz-1);
32     }
33 }
34 int main(){
35     idx=0;
36     cin>>n>>m;
37     int u,v;
38     for(int i=1;i<=m;i++){
39         cin>>u>>v;
40         G[u].push_back(v);
41         G[v].push_back(u);
42     }
43     Tarjan(1,0);
44     for(int i=1;i<=n;i++){

```

```

45     if(!jud[i]){
46         cout<<2*(n-1)<<endl;
47     }else{
48         ll ans=n*(n-1);
49         ll now=0;
50         for(ll v:GD[i]){
51             now+=v*(v-1);
52         }
53         cout<<ans-now<<endl;
54     }
55 }
56 return 0;
57 }

```

2.4.3 无向图点双连通

```

1  #include<bits/stdc++.h>
2
3  using namespace std;
4  const int N = 2e5+100;
5  int n,m;
6  int bcc_cnt;//bcc个数
7  int dfs_clock;
8  int pre[N];
9  bool is_cut[N];//判断是否是割点
10 int bccno[N];//第i个点是属于哪一个双连通分量
11 vector<int> G[N],bcc[N];
12
13 struct edge{
14     int u,v;
15     edge(int u,int v):u(u),v(v){}
16 };
17 stack<edge> s;
18 int Tarjan(int u,int fa){
19     int lowu=pre[u]=++dfs_clock;
20     int child=0;
21     for(int v:G[u]){
22         edge e=edge(u,v);
23         if(!pre[v]){
24             s.push(e);
25             child++;
26             int lowv=Tarjan(v,u);
27             lowu=min(lowv,lowu);
28             if(lowv>=pre[u]){//找到了割点
29                 is_cut[u]=1;
30                 bcc_cnt++;
31                 bcc[bcc_cnt].clear();
32                 while(1){
33                     edge x=s.top();s.pop();
34                     if(bccno[x.u]!=bcc_cnt){
35                         bcc[bcc_cnt].push_back(x.u);
36                         bccno[x.u]=bcc_cnt;
37                     }
38                     if(bccno[x.v]!=bcc_cnt){
39                         bcc[bcc_cnt].push_back(x.v);
40                         bccno[x.v]=bcc_cnt;
41                     }
42                     if(x.u==u&&x.v==v) break;
43                 }
44             }
45         }else if(pre[v]<pre[u]&&v!=fa){
46             s.push(e);
47             lowu=min(lowu,pre[v]);
48         }
49     }
50 }

```

```

49     }
50     if(fa<0&&child==1) is_cut[u]=0;
51     return lowu;
52 }
53 void find_bcc(int n){
54     for(int i=0;i<=n;i++) pre[i]=is_cut[i]=bccno[i]=0;
55     for(int i=1;i<=n;i++){
56         if(!pre[i]) Tarjan(i,-1);
57     }
58 }
59 int main(){
60     int cnt=0;
61     cin>>n>>m;
62     int u,v;
63     for(int i=0;i<m;i++){
64         cin>>u>>v;
65         G[u].push_back(v);
66         G[v].push_back(u);
67     }
68     find_bcc(n);
69     cout<<bcc_cnt<<endl;
70     for(int i=1;i<=bcc_cnt;i++){
71         cout<<"BCC " <<i<<endl;
72         for(int v:bcc[i]) cout<<v<<" ";
73         cout<<endl;
74     }
75     return 0;
76 }

```

2.4.4 无向图边双连通

该程序是判断两点之间是否有两条不相交的路径

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N=5e4+5,M=1e5+5;
4  int n,m,vis[M<<1],ans;
5  int cnt=1,head[N],u[M],v[M];
6  int now,col,dfn[N],low[N],color[N],q,x,y;
7  stack<int> sta;
8  struct edge{int next,to;}e[M<<1];
9  inline void add(int u,int v){
10     cnt++;
11     e[cnt].next=head[u];
12     e[cnt].to=v;
13     head[u]=cnt;
14     cnt++;
15     e[cnt].next=head[v];
16     e[cnt].to=u;
17     head[v]=cnt;
18 }
19 inline void tarjan(int u)
20 {
21     dfn[u]=low[u]=++now;
22     sta.push(u);
23     for (int i=head[u];i; i=e[i].next){
24         if(!vis[i]){
25             vis[i]=vis[i^1]=1;
26             if (!dfn[e[i].to]){
27                 tarjan(e[i].to);
28                 low[u]=min(low[u],low[e[i].to]);
29             }
30             else low[u]=min(low[u],dfn[e[i].to]);
31         }
32     }
33 }

```

```

33     if (low[u]==dfn[u]){
34         color[u]=++col;
35         while (1){
36             int now=sta.top();sta.pop();
37             color[now]=col;
38             if(now==u) break;
39         }
40     }
41 }
42
43 int main(){
44     memset(head,0,sizeof(head));
45     memset(dfn,0,sizeof(head));
46     scanf("%d%d",&n,&m);
47     for (int i=1; i<=m; ++i){
48         scanf("%d%d",&u[i],&v[i]);
49         add(u[i],v[i]);
50     }
51     for (int i=1; i<=n; ++i){
52         if (!dfn[i]) tarjan(i);
53     }
54     scanf("%d",&q);
55     while(q--){
56         scanf("%d %d",&x,&y);
57         if(color[x]!=color[y]){
58             puts("No");
59         }else{
60             puts("Yes");
61         }
62     }
63     return 0;
64 }

```

2.5 求无向连通图的第 K 大联通子图

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 bitset<105> bs[105];
5 long long w[105];
6 char maze[105][105];
7 struct node{
8     bitset<105> cl;
9     long long w;
10    int last;
11 };
12 bool operator<(node a,node b){
13     return a.w>b.w;
14 }
15 priority_queue<node> q;
16 int n,k;
17 long long bfs(){
18     node now;
19     now.cl.reset();
20     now.w=0;
21     now.last=0;
22
23     q.push(now);
24     while(!q.empty()){
25         node rt=q.top();q.pop();
26         if(--k==0) return rt.w;
27
28         for(int i=rt.last+1;i<=n;i++){
29             if(((bs[i]&rt.cl).count()==rt.cl.count())){

```

```

30         node pt=rt;
31         pt.cl[i]=1;
32         pt.last=i;
33         pt.w+=w[i];
34         q.push(pt);
35     }
36 }
37 }
38 return -1;
39 }
40 int main(){
41     cin>>n>>k;
42     for(int i=1;i<=n;i++) cin>>w[i];
43     for(int i=1;i<=n;i++){
44         for(int j=1;j<=n;j++){
45             cin>>maze[i][j];
46             if(maze[i][j]=='1'){
47                 bs[i][j]=1;
48             }
49         }
50     }
51     cout<<bfs()<<endl;
52     return 0;
53 }

```

2.6 分层 (K) 图最短路

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef pair<int ,int> pii;
5 const int INF = 0x3f3f3f3f;
6 const int N = 5e6+10;
7 const int M = 5e6+10;
8
9 struct EDGE{
10     int next;
11     int to;
12     int w;
13 }edge[M];
14
15 int n,m,k,s,e,cnt = 1;
16 int head[N],dis[N];
17 bool inq[N];
18
19 void add(int u, int v, int w){
20     edge[cnt].next = head[u];
21     edge[cnt].to = v;
22     edge[cnt].w = w;
23     head[u] = cnt++;
24 }
25
26 struct NODE{
27     int id,dist;
28 }q,p;
29 bool operator < (NODE a, NODE b){
30     return a.dist > b.dist;
31 }
32 void Dijkstra(){
33     memset(dis, INF, sizeof dis);
34     memset(inq, 0, sizeof inq);
35     priority_queue<NODE> que;
36     p.id = s; p.dist = 0;
37     dis[s] = 0; que.push(p);

```

```

38 while(!que.empty()){
39     q = que.top(); que.pop();
40     if(inq[q.id]) continue;
41     inq[q.id] = true;
42     for(int i=head[q.id]; ~i; i=edge[i].next){
43         int u = edge[i].to;
44         if(dis[u] > q.dist + edge[i].w){
45             dis[u] = q.dist + edge[i].w;
46             p.id = u;
47             p.dist = dis[u];
48             que.push(p);
49         }
50     }
51 }
52 int ans = INF;
53 for(int i=0; i<=k; ++i) ans = min(ans, dis[e + i*n]);
54 printf("%d\n", ans);
55 }
56 int main(int argc, char const *argv[])
57 {
58     int u, v, w;
59     memset(head, -1, sizeof head);
60     scanf("%d %d %d %d %d", &n, &m, &s, &e, &k);
61     for(int i=1; i<=m; ++i){
62         scanf("%d %d %d", &u, &v, &w);
63         for(int j=0; j<=k; ++j){
64             add(u + j*n, v + j*n, w);
65             add(v + j*n, u + j*n, w);
66             if(j != k){
67                 add(u + j*n, v + (j+1)*n, 0);
68                 add(v + j*n, u + (j+1)*n, 0);
69             }
70         }
71     }
72     Dijkstra();
73     return 0;
74 }

```

3 数学

3.1 整除分块

计算 $\sum_{i=1}^n \lfloor \frac{n}{i} \rfloor$

```

1 for(int l=1, r; l<=n; l=r+1)
2 {
3     r=n/(n/l);
4     ans+=(r-l+1)*(n/l);
5 }

```

3.2 SG 函数打表

```

1 int op[110], sg[11000];
2 int k, N;
3 vector<int> s;
4 void getSG(){
5     sg[0] = 0;
6     for(int i=1; i<=N; ++i){
7         s.clear();
8         for(int j=1; i>=op[j] && j<=k; ++j)
9             s.push_back(sg[i - op[j]]);
10        for(int j=0; ; ++j){

```

```

11            if(count(s.begin(), s.end(), j) == 0){
12                sg[i] = j;
13                break;
14            }
15        }
16    }
17 }

```

3.3 线性素数 + 莫比乌斯函数打表

```

1 int miu[MAXN+10], check[MAXN+10], prime[MAXN+10];
2 void Mobius()
3 {
4     memset(check, false, sizeof(check));
5     miu[1] = 1;
6     int tot = 0;
7     for(int i = 2; i <= MAXN; i++)
8     {
9         if( !check[i] )
10         {
11             prime[tot++] = i;
12             miu[i] = -1;
13         }
14         for(int j = 0; j < tot; j++)
15         {
16             if(i * prime[j] > MAXN) break;
17             check[i * prime[j]] = true;
18             if( i % prime[j] == 0 )
19             {
20                 miu[i * prime[j]] = 0;
21                 break;
22             }
23             else
24             {
25                 miu[i * prime[j]] = -miu[i];
26             }
27         }
28     }
29 }

```

3.4 Lucas 定理求 $C(n, m) \% P$

```

1 typedef long long LL;
2
3 LL mod;
4
5 inline LL pow(LL a, LL b) //快速幂是为了求逆元
6 {
7     LL ans = 1;
8     for(; b >= 1; a = a * a % mod)
9         if(b & 1)
10             ans = ans * a % mod;
11     return ans;
12 }
13
14 LL farc[1000005];
15
16 inline void prepare(LL a)
17 {
18     farc[0]=1;
19     for(LL i = 1; i <= a; ++i)
20         farc[i]=farc[i-1]*i%mod;

```



```

21 }
22
23 inline LL Csmall(LL m, LL n) //  $C(m,n) = (n!)/(m!(n-m)!)$ 
24 {
25     if(n < m)
26         return 0;
27     return fac[n] * pow(fac[m], mod-2) % mod * pow(
28         fac[n-m], mod-2) % mod; // 费马小定理求逆元
29 }
30 inline LL C(LL m, LL n)
31 {
32     if(n < m)
33         return 0;
34     if(!n)
35         return 1; // Lucas 的边界条件
36     return C(m/mod, n/mod) % mod * Csmall(m%mod, n%mod
37         ) % mod; // 上面证明的 Lucas 定理

```

3.5 大数质因子分解 & 大素数检测

```

1  ll Abs( ll a ){ return a<0?-a:a; }
2  ll Min( ll a , ll b ){ return a<b?a:b; }
3  ll Max( ll a , ll b ){ return a>b?a:b; }
4  ll Gcd( ll a , ll b ){ return b==0?a:Gcd( b , a%b );
5  }
6  ll arr[5] = { 2,3,5,233,331 };
7  ll Qmul( ll a , ll b , ll mod )
8  {
9      ll res = 0;
10     while ( b )
11     {
12         if ( b&1 )
13             res = ( res+a )%mod;
14         a = ( a+a )%mod;
15         b = b>>1;
16     }
17     return res;
18 }
19 ll Qpow( ll a , ll b , ll mod )
20 {
21     ll res = 1;
22     while ( b )
23     {
24         if ( b&1 )
25             res = Qmul( res , a , mod );
26         a = Qmul( a , a , mod );
27         b = b>>1;
28     }
29     return res;
30 }
31 bool Miller_Rabin( ll n )
32 {
33     if ( n==2 ) return true;
34     if ( n < 2 || n%2==0 ) return false;
35     ll m = n-1, k = 0;
36     while ( m%2==0 ) k++, m>>=1;
37     for ( int I=0 ; I<5 ; I++ )
38     {
39         ll a = arr[I]%(n-1)+1;
40         ll x = Qpow( a , m , n );
41         for ( int j=1 ; j<=k ; j++ )

```

```

41     {
42         ll y = Qmul( x , x , n );
43         if ( y==1&&x!=1&&x!=n-1 )
44             return false;
45         x = y;
46     }
47     if ( x!=1 ) return false;
48 }
49 return true;
50 }
51 ll fac[110], tol = 0;
52 ll Pollard_rho( ll x , ll c )
53 {
54     ll I=1,k=2;
55     ll x0 = rand()%x;
56     ll y0 = x0;
57     while ( 1 )
58     {
59         I++;
60         x0 = ( Qmul( x0 , x0 , x )+c )%x;
61         ll d0 = Gcd( Abs( y0-x0 ) , x );
62         if ( d0!=1&&d0!=x ) return d0;
63         if ( y0==x0 ) return x;
64         if ( I == k ) { y0=x0; k+=k; }
65     }
66 }
67 void Findfac( ll n )
68 {
69     if ( Miller_Rabin( n ) )
70     {
71         fac[tol++] = n;
72         return;
73     }
74     ll p = n;
75     while ( p>=n )
76         p = Pollard_rho( p , rand()%(n-1)+1 );
77     Findfac( p );
78     Findfac( n/p );
79 }
80 ll exgcd( ll a, ll b, ll &x, ll &y )
81 {
82     if(b==0)
83     {
84         x=1,y=0;
85         return a;
86     }
87     ll g=exgcd(b,a%b,x,y);
88     ll tmp=x;x=y;y=tmp-a/b*y;
89     return g;
90 }

```

3.6 python 通用中国剩余定理

```

1  """
2  n 方程个数
3  a1 r1: x = a1 (mod r1)
4  flag 是否有解
5  """
6  def egcd(a, b):
7      if 0 == b:
8          return 1, 0, a
9      x, y, q = egcd(b, a % b)
10     x, y = y, (x - a // b * y)
11     return x, y, q

```

```

12 n = int(input().split)
13 flag = False
14 a1, r1 = map(int, input().split())
15 for _ in range(n-1):
16     a2, r2 = map(int, input().split())
17     R = r2-r1
18     x, y, d = egcd(a1, a2)
19     tmp = a2//d
20     if R%d != 0:
21         flag = True
22     r1=((x*R//d)%tmp+tmp)%tmp*a1+r1
23     a1=a1*(a2//d)
24 lcm = a1
25 ans = (r1%lcm+lcm)%lcm

```

3.7 在线求组合数

```

1 void init(){
2     fact[0]=inv[1]=factinv[0]=inv[0]=fact[1]=factinv
3     [1]=1;
4     for(int i=2;i<=MAXN;i++){
5         fact[i]=(fact[i-1]%mod*i)%mod;
6         inv[i]=(mod-mod/i)*inv[mod%i]%mod;
7         factinv[i]=factinv[i-1]*inv[i]%mod;
8     }
9     ll c(ll n,ll m){
10         return fact[n]*factinv[m]%mod*factinv[n-m]%mod;
11     }

```

3.8 拉格朗日插值

3.8.1 连续情况

以计算 $\sum_{i=1}^n i^k$ 为例

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = 1e6+100;
5 typedef long long ll;
6 const ll mod = 1e9+7;
7 ll p[N],x[N],s1[N],s2[N],ifac[N];
8 ll qpow(ll a,ll b){
9     ll ans=1;
10    while(b){
11        if(b&1) ans=(ans%mod*a%mod)%mod;
12        a=(a%mod*a%mod)%mod;
13        b>>=1;
14    }
15    return (ans%mod+mod)%mod;
16 }
17
18 //拉格朗日插值, n项, 每个点的坐标为(x_i,y_i), 求第xi项的
19 //值, 保证x是连续的一段
20 ll lagrange(ll n, ll *x, ll *y, ll xi) {
21     ll ans = 0;
22     s1[0] = (xi-x[0])%mod, s2[n+1] = 1;
23     for (ll i = 1; i <= n; i++) s1[i] = 1ll*s1[i-1]*(
24         xi-x[i])%mod;
25     for (ll i = n; i >= 0; i--) s2[i] = 1ll*s2[i+1]*(
26         xi-x[i])%mod;
27     ifac[0] = ifac[1] = 1;

```

```

25     for (ll i = 2; i <= n; i++) ifac[i] = -1ll*mod/i*
26         ifac[mod%i]%mod;
27     for (ll i = 2; i <= n; i++) ifac[i] = 1ll*ifac[i]*
28         ifac[i-1]%mod;
29     for (ll i = 0; i <= n; i++)
30         (ans += 1ll*y[i]*(i == 0 ? 1 : s1[i-1])%mod*s2
31             [i+1]%mod
32             *ifac[i]%mod*((n-i)&1) ? -1 : 1)*ifac[n-i
33             ]%mod) %= mod;
34     return (ans+mod)%mod;
35 }
36 int main(){
37     ll n,k;
38     cin>>n>>k;
39     if(k==0){
40         cout<<n<<endl;
41         return 0;
42     }
43     p[0]=0;
44     for(ll i=1;i<=k+2;i++) p[i]=(p[i-1]%mod+qpow(i,k))
45         %mod;
46     for(ll i=1;i<=k+2;i++) x[i]=i;
47     if(n<=k+2){
48         cout<<p[n]<<endl;
49     }
50     else{
51         cout<<lagrange(k+2,x,p,n)<<endl;
52     }
53     return 0;
54 }

```

3.8.2 非连续情况

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = 1e6+100;
5 typedef long long ll;
6 const ll mod = 998244353;
7 struct point{
8     ll x,y;
9 }p[N];
10 int n,k;
11 ll qpow(ll a,ll b,ll mod){
12     ll ans=1;
13     while(b){
14         if(b&1){
15             ans=(ans%mod*a%mod)%mod;
16         }
17         a=(a%mod*a%mod)%mod;
18         b>>=1;
19     }
20     return ans%mod;
21 }
22 ll Lagrange(int k){
23     ll ans=0;
24     for(int j=1;j<=n;j++){//
25         ll base1=1;
26         ll base2=1;
27         for(int i=1;i<=n;i++){//lj(k)基函数
28             if(j==i) continue;
29             base1=(base1%mod*((k-p[i].x)%mod+mod)%mod)%
30                 mod;

```

```

30     base2=(base2%mod*((p[j].x-p[i].x)%mod+mod)%
31         mod)%mod;
32     }
33     ans=(ans%mod+(p[j].y%mod*base1%mod*qpow(base2,
34         mod-2,mod)%mod)%mod)%mod;
35     }
36     return ans;
37 }
38 int main(){
39     cin>>n>>k;
40     for(int i=1;i<=n;i++) cin>>p[i].x>>p[i].y;
41     cout<<Lagrange(k)<<endl;
42     return 0;
43 }

```

3.9 辛普森自适应积分

```

1  #include<cstdio>
2  #include<cmath>
3  double a, b, c, d, L, R;
4  double F(double x) {
5      return (c * x + d) / (a * x + b);
6  }
7  double sim(double l, double r) {
8      return (F(l) + F(r) + 4 * F((l + r) / 2)) * (r - l)
9          / 6;
10 }
11 double asr(double L, double R, double eps, double ans
12 ) {
13     double mid = (L + R) / 2;
14     double LL = sim(L, mid), RR = sim(mid, R);
15     if(fabs(LL + RR - ans) < eps) return LL + RR;
16     else return asr(L, mid, eps / 2, sim(L, mid)) +
17         asr(mid, R, eps / 2, sim(mid, R));
18 }
19 main() {
20     #ifdef WIN32
21     freopen("a.in", "r", stdin);
22     #endif
23     scanf("%lf %lf %lf %lf %lf %lf", &a, &b, &c, &d, &L, &R);
24     printf("%lf", asr(L, R, 1e-6, sim(L, R)));
25 }

```

3.10 欧拉函数

比 n 小的与 n 互质的数的个数

3.10.1 在线

```

1  int euler(int n)//返回euler(n)
2  {
3      int i;
4      int res = n, a = n;
5      for(i = 2; i*i <= a; ++i)
6      {
7          if(a%i == 0)
8          {
9              res -= res/i; //p(n) = (p - p/p1)(1 - 1/p2).....
10             while(a%i == 0) a/=i;
11         }
12     }
13     return res;
14 }

```

```

14     if(a > 1) res -= res/a;//存在大于sqrt(a)的质因子
15     return res;
16 }

```

3.10.2 打表

```

1  void SE()//select euler//类似于素数筛选法
2  {
3      int i, j;
4      euler[1] = 1;
5      for(i = 2; i < Max; ++i) euler[i]=i;
6      for(i = 2; i < Max; ++i)
7      {
8          if(euler[i] == i)//这里出现的肯定是素数
9          {
10             for(j = i; j < Max; j += i)//然后更新含有它的数
11             {
12                 euler[j] = euler[j]/i*(i - 1); // n*(1 - 1/p1)...*(1 - 1/pk).先除后乘
13             }
14         }
15     }
16     //for (int i = 1; i <= 20; ++i) printf("%d ", euler[i]);
17 }

```

3.11 欧拉降幂

降幂公式:

$$a^{b\%p} = \begin{cases} a^{b\% \phi(p)\%p} & \gcd(a, p) = 1 \\ a^{b\%p} & \gcd(a, p) \neq 1, b < \phi(p) \\ a^{b\% \phi(p) + \phi(p)\%p} & \gcd(a, p) \neq 1, \phi(p) \leq b \end{cases}$$

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N = 2e6+100;
4  const int mod = 1e9+7;
5  typedef long long ll;
6  const int INF = 0x3f3f3f3f;
7  const ll llINF = 0x3f3f3f3f3f3f3f3f;
8  #define rep(i,a,b) for(int i=(a);i<=(b);i++)
9  #define fep(i,a,b) for(int i=(a);i>=(b);i--)
10 inline bool read(ll &num) {
11     char in;bool IsN=false;
12     in=getchar();
13     if(in==EOF) return false;
14     while(in!='-'&&(in<'0' || in>'9')) in=getchar();
15     if(in=='-'){ IsN=true;num=0;}
16     else num=in-'0';
17     while(in=getchar(), in>='0'&&in<='9'){
18         num*=10,num+=in-'0';
19     }
20     if(IsN) num=-num;
21     return true;
22 }
23 ll ph[N];
24 void init(){
25     rep(i,1,N-10){
26         ph[i]=i;
27     }
28     rep(i,2,N-10){

```

```

29     if(ph[i]==i){
30         for(int j=i;j<=N-10;j+=i){
31             ph[j]=ph[j]/i*(i-1);
32         }
33     }
34 }
35 }
36 ll qpow(ll a,ll b,ll mod){
37     ll ans=1;
38     while(b){
39         if(b&1) ans=(ans%mod*a%mod)%mod;
40         a=(a%mod*a%mod)%mod;
41         b>>=1;
42     }
43     return ans%mod;
44 }
45 bool check(ll a,ll b,ll m){
46     if(b==0) return 1>=ph[m];
47     if(b==1) return a>=ph[m];
48     ll ans=1;
49     if(ans>=ph[m]) return 1;
50     rep(i,1,b-1){
51         rep(j,1,a){
52             ans*=a;
53             if(ans>=ph[m]) return 1;
54         }
55     }
56     return 0;
57 }
58 ll solve(ll a,ll b,ll m){
59     if(m==1) return 0;
60     if(b==0) return 1%m;
61     if(b==1) return a%m;
62     if(__gcd(a,m)==1){
63         return qpow(a,solve(a,b-1,ph[m]),m);
64     }
65     else{
66         if(check(a,b-1,m)){
67             return qpow(a,solve(a,b-1,ph[m])+ph[m],m);
68         }
69         else return qpow(a,solve(a,b-1,m),m);
70     }
71 }
72 ll T,a,b,m;
73 int main(){
74     //freopen("1.in", "r", stdin);
75     read(T);
76     init();
77     //cout<<ph[1000000]<<endl;
78     while(T--){
79         read(a);read(b);read(m);
80         printf("%lld\n",solve(a,b,m)%m);
81     }
82     return 0;
83 }

```

4 数据结构

4.1 线段树

4.1.1 区间修改区间查询

```

1 #include<bits/stdc++.h>
2 using namespace std;

```

```

3 typedef long long ll;
4 const ll maxn = 1000050;
5 ll ans[maxn<<1],a[maxn],mod;
6 ll add[maxn],mult[maxn];
7 inline void pushup(ll rt){
8     ans[rt]=(ans[rt<<1]+ans[rt<<1|1])%mod;
9 }
10 void pushdown(ll rt,ll l,ll r){
11     ll mid=(l+r)>>1;
12     ans[rt<<1]=(ans[rt<<1]*mult[rt]+add[rt]*(mid-l+1))%mod;
13     ans[rt<<1|1]=(ans[rt<<1|1]*mult[rt]+add[rt]*(r-mid))%mod;
14
15     mult[rt<<1]=(mult[rt]*mult[rt<<1])%mod;
16     mult[rt<<1|1]=(mult[rt]*mult[rt<<1|1])%mod;
17
18     add[rt<<1]=(add[rt<<1]*mult[rt]+add[rt])%mod;
19     add[rt<<1|1]=(add[rt<<1|1]*mult[rt]+add[rt])%mod;
20
21     add[rt]=0;
22     mult[rt]=1;
23     return ;
24 }
25 inline void buildtree(ll rt,ll l,ll r){
26     mult[rt]=1;
27     add[rt]=0;
28     if(l==r){
29         ans[rt]=a[l];
30         return ;
31     }
32     ll mid=(l+r)>>1;
33     buildtree(rt<<1,l,mid);
34     buildtree(rt<<1|1,mid+1,r);
35     pushup(rt);
36 }
37 inline void update1(ll n1,ll nr,ll l,ll r,ll rt,ll k)
38 {
39     if(n1<=l&&r<=nr){
40         ans[rt]=(ans[rt]*k)%mod;
41         add[rt]=(add[rt]*k)%mod;
42         mult[rt]=(mult[rt]*k)%mod;
43         return ;
44     }
45     pushdown(rt,l,r);
46     ll mid=(l+r)>>1;
47     if(n1<=mid){
48         update1(n1,nr,l,mid,rt<<1,k);
49     }
50     if(nr>mid) update1(n1,nr,mid+1,r,rt<<1|1,k);
51     pushup(rt);
52 }
53 inline void update2(ll n1,ll nr,ll l,ll r,ll rt,ll k)
54 {
55     if(n1<=l&&nr>=r){
56         add[rt]=(add[rt]+k)%mod;
57         ans[rt]=(ans[rt]+k*(r-l+1))%mod;
58         return ;
59     }
60     pushdown(rt,l,r);
61     ll mid=(l+r)>>1;
62     if(n1<=mid){
63         update2(n1,nr,l,mid,rt<<1,k);
64     }
65     if(nr>mid) update2(n1,nr,mid+1,r,rt<<1|1,k);
66 }

```

```

64     pushup(rt);
65 }
66 ll query(ll nl,ll nr,ll l,ll r,ll rt){
67     ll res=0;
68     if(nl<=l&&r<=nr){
69         return ans[rt]%mod;
70     }
71     ll mid=(l+r)>>1;
72     pushdown(rt,l,r);
73     if(nl<=mid) res=(res%mod+query(nl,nr,l,mid,rt<<1))
74         %mod;
75     if(nr>mid) res=(res%mod+query(nl,nr,mid+1,r,rt
76         <<1|1))%mod;
77     return res;
78 }
79 int main(){
80     ios::sync_with_stdio(0);
81     cin.tie(0);
82     cout.tie(0);
83     ll n,m,op,x,y,k;
84     cin>>n>>m>>mod;
85     for(ll i=1;i<=n;i++) cin>>a[i];
86     buildtree(1,1,n);
87     while(m--){
88         cin>>op;
89         if(op==1){
90             cin>>x>>y>>k;
91             update1(x,y,1,n,1,k);
92         }
93         else if(op==2){
94             cin>>x>>y>>k;
95             update2(x,y,1,n,1,k);
96         }
97         else{
98             cin>>x>>y;
99             cout<<query(x,y,1,n,1)<<endl;
100         }
101     }
102     return 0;
103 }

```

4.2 主席树

4.2.1 区间第 k 小

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = 1e5+100;
5 struct node{
6     int l,r,num;
7 }T[N*30];
8 vector<int> v;
9 int n,m,a[N],t,cnt,roots[N];
10 int getid(int x){
11     return lower_bound(v.begin(),v.end(),x)-v.begin()
12         +1;
13 }
14 void update(int l,int r,int &x,int y,int pos){
15     T[++cnt]=T[y];T[cnt].num++;x=cnt;
16     if(l==r) return ;
17     int mid=(l+r)>>1;
18     if(pos<=mid) update(l,mid,T[x].l,T[y].l,pos);
19     else update(mid+1,r,T[x].r,T[y].r,pos);

```

```

19 }
20 int query(int l,int r,int x,int y,int k){
21     if(l==r) return l;
22     int sum=T[T[x].l].num-T[T[y].l].num;
23     int mid=(l+r)>>1;
24     if(sum>=k) return query(l,mid,T[x].l,T[y].l,k);
25     else return query(mid+1,r,T[x].r,T[y].r,k-sum);
26 }
27 int main(){
28     scanf("%d",&t);
29     while(t--){
30         v.clear();
31         cnt=0;
32         scanf("%d %d",&n,&m);
33         for(int i=1;i<=n;i++){
34             scanf("%d",&a[i]);
35             v.push_back(a[i]);
36         }
37         sort(v.begin(),v.end());
38         v.erase(unique(v.begin(),v.end()),v.end());
39         for(int i=1;i<=n;i++) update(1,n,roots[i],
40             roots[i-1],getid(a[i]));
41         while(m--){
42             int l,r,k;
43             scanf("%d %d %d",&l,&r,&k);
44             printf("%d\n",v[query(1,n,roots[r],roots[l
45                 -1],k)-1]);
46         }
47     }
48     return 0;
49 }

```

4.2.2 区间内小于等于 x 的最大值

```

1 struct node
2 {
3     ll sum,l,r;
4 }t[maxn*32];
5 int cnt;
6 void update(ll l,ll r,ll &x,ll y,ll pos){
7     t[++cnt]=t[y];t[cnt].sum++;x=cnt;//复制节点并且更新
8     if(l==r) return ;
9     int mid=(l+r)>>1;
10    if(mid>=pos) update(l,mid,t[x].l,t[y].l,pos);
11    else update(mid+1,r,t[x].r,t[y].r,pos);
12 }
13 int query(int a,int b,int x,int l,int r)
14 {
15     if(l==r)
16     {
17         if(l==x)
18             return 0;
19         else
20             return 1;
21     }
22     int mid=(l+r)>>1;
23     int xx=t[t[b].l].sum-t[t[a].l].sum;
24     int yy=t[t[b].r].sum-t[t[a].r].sum;
25     int res=0;
26     if(yy&&x>mid)
27         res=query(t[a].r,t[b].r,x,mid+1,r);
28     if(xx&&!res)
29         res=query(t[a].l,t[b].l,x,l,mid);
30     return res;

```

```

31 }
32
33 for(int i=1;i<=n;i++) update(1,n,roots[i],roots[i-1],
    a[i]);
34 query(roots[L-1],roots[R],x,1,n);

```

4.2.3 区间内距离 p 第 k 近的距离

```

1 struct node
2 {
3     ll sum,l,r;
4 }t[maxn*32];
5 int cnt;
6 void update(ll l,ll r,ll &x,ll y,ll pos){
7     t[++cnt]=t[y];t[cnt].sum++;x=cnt;//复制节点并且更新
8     if(l==r) return ;
9     int mid=(l+r)>>1;
10    if(mid>=pos) update(1,mid,t[x].l,t[y].l,pos);
11    else update(mid+1,r,t[x].r,t[y].r,pos);
12 }
13 int query(int a,int b,int x,int l,int r)
14 {
15     if(l==r)
16     {
17         if(l==x)
18             return 0;
19         else
20             return 1;
21     }
22     int mid=(l+r)>>1;
23     int xx=t[t[b].l].sum-t[t[a].l].sum;
24     int yy=t[t[b].r].sum-t[t[a].r].sum;
25     int res=0;
26     if(yy&&x>mid)
27         res=query(t[a].r,t[b].r,x,mid+1,r);
28     if(xx&&!res)
29         res=query(t[a].l,t[b].l,x,l,mid);
30     return res;
31 }
32
33 for(int i=1;i<=n;i++) update(1,n,roots[i],roots[i-1],
    a[i]);
34 query(roots[L-1],roots[R],x,1,n);

```

4.2.4 区间内小于等于 x 的最大值

```

1 struct node
2 {
3     ll sum,l,r;
4 }t[maxn*32];
5 int cnt;
6 void update(ll l,ll r,ll &x,ll y,ll pos){
7     t[++cnt]=t[y];t[cnt].sum++;x=cnt;//复制节点并且更新
8     if(l==r) return ;
9     int mid=(l+r)>>1;
10    if(mid>=pos) update(1,mid,t[x].l,t[y].l,pos);
11    else update(mid+1,r,t[x].r,t[y].r,pos);
12 }
13 int query(int a,int b,int x,int l,int r)
14 {
15     if(l==r)
16     {
17         if(l==x)

```

```

18         return 0;
19     else
20         return 1;
21 }
22 int mid=(l+r)>>1;
23 int xx=t[t[b].l].sum-t[t[a].l].sum;
24 int yy=t[t[b].r].sum-t[t[a].r].sum;
25 int res=0;
26 if(yy&&x>mid)
27     res=query(t[a].r,t[b].r,x,mid+1,r);
28 if(xx&&!res)
29     res=query(t[a].l,t[b].l,x,l,mid);
30 return res;
31 }
32
33 for(int i=1;i<=n;i++) update(1,n,roots[i],roots[i-1],
    a[i]);
34 query(roots[L-1],roots[R],x,1,n);

```

4.2.5 区间数的种类数

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int N = 1e6+5;
4 int n,q,a[N],p[N];
5
6 int rt[N*40],ls[N*40],rs[N*40],sum[N*40],cnt=0;
7 void up(int pre,int& o,int l,int r,int pos,int val) {
8     o++;cnt;
9     ls[o]=ls[pre];
10    rs[o]=rs[pre];
11    sum[o]=sum[pre]+val;
12    if(l==r) return ;
13    int m=(l+r)/2;
14    if(pos<=m) up(ls[pre],ls[o],l,m,pos,val);
15    else up(rs[pre],rs[o],m+1,r,pos,val);
16 }
17
18 int qu(int o,int l,int r,int ql,int qr) {
19     if(ql<=l && qr>=r) return sum[o];
20     int ans = 0,m = (l+r)/2;
21     if(ql<=m) ans += qu(ls[o],l,m,ql,qr);
22     if(qr>m) ans += qu(rs[o],m+1,r,ql,qr);
23     return ans;
24 }
25
26 int main(){
27     scanf("%d",&n);
28     for(int i=1;i<=n;i++) {
29         scanf("%d",&a[i]);
30         if(!p[a[i]]) {
31             up(rt[i-1],rt[i],1,n,i,1);
32         }else {
33             int tp;
34             up(rt[i-1],tp,1,n,p[a[i]],-1);
35             up(tp,rt[i],1,n,i,1);
36         }
37         p[a[i]] = i;
38     }
39     scanf("%d",&q);
40     while(q--) {
41         int l,r;
42         scanf("%d%d",&l,&r);
43         int ans = qu(rt[r],1,n,l,r);

```

```

44     printf("%d\n",ans);
45 }
46     return 0;
47 }

```

4.2.6 区间内未出现过的最小自然数

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  const int N = 2e5+5;
5  const int M = 1e9;
6
7  int rt[N*30],ls[N*30],rs[N*30],mn[N*30],cnt=0;
8  void up(int pre,int& o,int l,int r,int val,int pos) {
9      o=++cnt;
10     ls[o]=ls[pre];
11     rs[o]=rs[pre];
12     mn[o]=pos;
13     if(l==r) return ;
14     int m=(l+r)/2;
15     if(val<=m) up(ls[pre],ls[o],l,m,val,pos);
16     else up(rs[pre],rs[o],m+1,r,val,pos);
17     mn[o]=min(mn[ls[o]],mn[rs[o]]);
18 }
19
20 int qu(int o,int l,int r,int pos) {
21     if(l==r) return l;
22     int m=(l+r)/2;
23     if( mn[ls[o]]<pos ) return qu(ls[o],l,m,pos);
24     return qu(rs[o],m+1,r,pos);
25 }
26
27 int a[N],n,q,l,r;
28 int main(){
29     scanf("%d%d",&n,&q);
30     for(int i=1;i<=n;i++) {
31         scanf("%d",&a[i]);
32         up(rt[i-1],rt[i],0,M,a[i],i);
33     }
34     while(q--) {
35         scanf("%d%d",&l,&r);
36         int ans = qu(rt[r],0,M,l);
37         printf("%d\n",ans);
38     }
39     return 0;
40 }

```

4.3 树链剖分

```

1  #pragma GCC optimize(2)
2  #include<bits/stdc++.h>
3  #define rep(i, a, b) for(int i = (a); i <= (int)(b); ++i)
4  #define per(i, a, b) for(int i = (a); i >= (int)(b); --i)
5  #define debug(x) cerr << #x << ' ' << x << endl;
6  #define ls x<<1
7  #define rs x<<1|1
8  using namespace std;
9
10 typedef long long ll;

```

```

11 const int MAXN = 1e6 + 7;
12
13 int son[MAXN], fa[MAXN], dep[MAXN], siz[MAXN], top[
14     MAXN], tid[MAXN], rnk[MAXN], w[MAXN];
15 vector<int> G[MAXN];
16 int n, m, s, cur = 0;
17
18 struct SegTree{
19     struct Node{
20         int l, r;
21         ll lz, sum;
22         int mid(){return (l+r)>>1;}
23         int size(){return (r-l+1);}
24     }s[MAXN<<2];
25     inline void pushdown(int x){
26         s[ls].lz += s[x].lz;
27         s[ls].sum += s[x].lz * s[ls].size();
28         s[rs].lz += s[x].lz;
29         s[rs].sum += s[x].lz * s[rs].size();
30         s[x].lz = 0;
31     }
32     inline void pushup(int x){
33         s[x].sum = s[ls].sum + s[rs].sum;
34     }
35     inline void build(int x, int l, int r){
36         s[x].l = l; s[x].r = r;
37         if(l == r){
38             s[x].lz = 0;
39             s[x].sum = w[rnk[l]];
40             return;
41         }
42         int mid = s[x].mid();
43         build(ls, l, mid);
44         build(rs, mid + 1, r);
45         pushup(x);
46     }
47     inline ll query(int x, int l, int r){
48         if(s[x].l == l && s[x].r == r) return s[x].sum;
49         ;
50         pushdown(x);
51         int mid = s[x].mid();
52         if(r <= mid) return query(ls, l, r);
53         else if(l > mid) return query(rs, l, r);
54         else return query(ls, l, mid) + query(rs, mid
55             + 1, r);
56     }
57     inline void updata(int x, int l, int r, int v){
58         if(s[x].l == l && s[x].r == r){
59             s[x].lz += v;
60             s[x].sum += 1LL * v * s[x].size();
61             return;
62         }
63         pushdown(x);
64         int mid = s[x].mid();
65         if(r <= mid) updata(ls, l, r, v);
66         else if(l > mid) updata(rs, l, r, v);
67         else {
68             updata(ls, l, mid, v);
69             updata(rs, mid + 1, r, v);
70         }
71         pushup(x);
72     }
73 }st;
74 void dfs1(int x, int f = 0){
75     son[x] = -1;

```



```

73     siz[x] = 1;
74     dep[x] = dep[f] + 1;
75     fa[x] = f;
76     for(int u: G[x]){
77         if(u == f) continue;
78         dfs1(u, x);
79         siz[x] += siz[u];
80         if(son[x] == -1 || siz[son[x]] < siz[u]) son[x]
            = u;
81     }
82 }
83
84 void dfs2(int x, int t){
85     top[x] = t;
86     cur++;
87     tid[x] = cur;
88     rnk[cur] = x;
89     if(son[x] == -1) return;
90     dfs2(son[x], t);
91     for(int u: G[x]){
92         if(u != son[x] && u != fa[x]) dfs2(u, u);
93     }
94 }
95
96 //链上更新
97 inline void linkadd(int u, int v, int w){
98     int fu = top[u], fv = top[v];
99     while(fu != fv){
100         if(dep[fu] >= dep[fv]){
101             st.updata(1, tid[fu], tid[u], w);
102             u = fa[fu];
103         } else {
104             st.updata(1, tid[fv], tid[v], w);
105             v = fa[fv];
106         }
107         fu = top[u];
108         fv = top[v];
109     }
110     if(tid[u] > tid[v]) swap(u, v);
111     st.updata(1, tid[u], tid[v], w);
112 }
113
114 //链上求和查询
115 inline ll linkquery(int u, int v){
116     int fu = top[u], fv = top[v];
117     ll res = 0;
118     while(fu != fv){
119         if(dep[fu] >= dep[fv]){
120             res += st.query(1, tid[fu], tid[u]);
121             u = fa[fu];
122         } else {
123             res += st.query(1, tid[fv], tid[v]);
124             v = fa[fv];
125         }
126         fu = top[u];
127         fv = top[v];
128     }
129     if(tid[u] > tid[v]) swap(u, v);
130     res += st.query(1, tid[u], tid[v]);
131     return res;
132 }
133
134 //子树查询
135 inline ll subtreequery(int x){

```

```

137     return st.query(1, tid[x], tid[x] + siz[x] - 1);
138 }
139
140 //子树更新
141 inline void subtreeadd(int x, int w){
142     return st.updata(1, tid[x], tid[x] + siz[x] - 1, w
        );
143 }
144
145 //查询LAC
146 inline int lca(int u, int v){
147     int fu = top[u], fv = top[v];
148     while(fu != fv){
149         if(dep[fu] >= dep[fv]) u = fa[fu];
150         else v = fa[fv];
151         fu = top[u];
152         fv = top[v];
153     }
154     if(dep[u] > dep[v]) swap(u, v);
155     return u;
156 }
157 int main() {
158     scanf("%d %d %d", &n, &m, &s);
159     rep(i, 1, n) scanf("%d", &w[i]);
160     int u, v, w, op;
161     rep(i, 1, n-1){
162         scanf("%d %d", &u, &v);
163         G[u].push_back(v);
164         G[v].push_back(u);
165     }
166     dfs1(s);
167     dfs2(s, s);
168     st.build(1, 1, n);
169     while(m--){
170         scanf("%d %d", &op, &u);
171         if(op == 1) {
172             scanf("%d %d", &v, &w);
173             linkadd(u, v, w);
174         } else if(op == 2) {
175             printf("%lld\n", linkquery(u, u));
176         } else {
177             printf("%lld\n", subtreequery(u));
178         }
179     }
180     return 0;
181 }

```

4.4 树状数组

4.4.1 二位偏序求矩形内点的个数

离线算法，以南京网赛 A 为例

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int N = 1e6+100;
4 const int mod = 1e9+7;
5 typedef long long ll;
6 const int INF = 0x3f3f3f3f;
7 const ll llINF = 0x3f3f3f3f3f3f3f3f;
8 #define rep(i,a,b) for(int i=(a);i<=(b);i++)
9 #define fep(i,a,b) for(int i=(a);i>=(b);i--)
10 inline bool read(ll &num) {
11     char in;bool IsN=false;
12     in=getchar();

```



```

13     if(in==EOF) return false;
14     while(in!='-'&&(in<'0' || in>'9')) in=getchar();
15     if(in=='-'){ IsN=true; num=0; }
16     else num=in-'0';
17     while(in=getchar(), in>='0'&&in<='9'){
18         num*=10, num+=in-'0';
19     }
20     if(IsN) num=-num;
21     return true;
22 }
23 ll T,n,p,m,c[N];
24 int lowbit(ll x){
25     return (x&(-x));
26 }
27 void add(ll x,ll v){
28     for(;x<N;x+=lowbit(x)){
29         c[x]+=v;
30         //cout<<x<< ' '<<v<<endl;
31     }
32 }
33 ll query(ll x){
34     ll ans=0;
35     for(;x>=1;x-=lowbit(x)){
36         ans+=c[x];
37     }
38     return ans;
39 }
40 //BIT
41 struct point{
42     ll x,y;
43     int flag;
44 }pp[600000];
45 bool cmp(point a,point b){
46     if(a.x==b.x){
47         if(a.y==b.y){
48             return a.flag<b.flag;
49         }
50         return a.y<b.y;
51     }
52     return a.x<b.x;
53 }
54 ll dig(ll x){
55     ll ans=0;
56     while(x){
57         ans+=x%10;
58         x/=10;
59     }
60     return ans;
61 }
62 }
63 ll cal(ll x,ll y){//计算(x,y)处的值
64     x=x-n/2-1;
65     y=y-n/2-1;
66     ll t=max(abs(x),abs(y));
67     if(x>=y) return n*n-4*t*t-2*t-x-y;
68     else return n*n-4*t*t+2*t+x+y;
69 }
70 map<pair<ll,ll>,ll> mmp;
71 ll yy[N],id[N];
72 ll x_1[100001],y_1[100001],x_2[100001],y_2[100001];
73 int main(){
74     read(T);
75     while(T--){
76         mmp.clear();

```

```

77     read(n);read(m);read(p);
78     memset(c,0,sizeof(c));
79     ll x,y,x_1,y_1,x_2,y_2;
80     rep(i,1,m){
81         read(x);read(y);
82         pp[i]={x,y,0};
83     }
84     rep(i,1,p){
85         read(x_1);read(y_1);read(x_2);read(y_2);
86         x_1[i]=x_1;y_1[i]=y_1;x_2[i]=x_2;y_2[i]=y_2;
87         pp[++m]={x_1-1,y_1-1,1};
88         pp[++m]={x_2,y_2,1};
89         pp[++m]={x_1-1,y_2,1};
90         pp[++m]={x_2,y_1-1,1};
91     }
92     rep(i,1,m) yy[i]=pp[i].y;
93     sort(yy+1,yy+m+1);
94     int siz=unique(yy+1,yy+m+1)-yy-1;
95     sort(pp+1,pp+m+1,cmp);
96     rep(i,1,m){
97         id[i]=lower_bound(yy+1,yy+siz+1,pp[i].y)-yy;
98     }
99     //离散化
100     rep(i,1,m){
101         if(pp[i].flag==0){
102             add(id[i],dig(cal(pp[i].x,pp[i].y)));
103             //cout<<pp[i].x<< ' '<<pp[i].y<< ' '<<id[i]
104             //<<dig(cal(pp[i].x,pp[i].y))<< endl;
105         }
106         else{
107             mmp[{pp[i].x,pp[i].y}]=query(id[i]);
108             //cout<<pp[i].x<< ' '<<pp[i].y<< ' '<<
109             //query(id[i])<<endl;
110         }
111     }
112     rep(i,1,p){
113         //cout<<x_2[i]<< ' '<<y_2[i]<<endl;
114         printf("%lld\n",mmp[{x_2[i],y_2[i]}]-mmp[
115             [{x_1[i]-1,y_2[i]}]-mmp[{x_2[i],y_1[i]-1}]+mmp[
116             [{x_1[i]-1,y_1[i]-1}]]);
117     }
118     return 0;
119 }

```

4.4.2 树状数组求区间最值

```

1 struct BIT{
2     ll e[MAXN];
3     int lowbit(int x){
4         return x & -x;
5     }
6     void upd(int x){
7         int lx;
8         while(x <= n){
9             e[x] = a[x];
10            lx = lowbit(x);
11            for(int I = 1; I < lx; I <= 1) e[x] = max(
12                e[x], e[x-I]);
13            x += lowbit(x);
14        }

```

```

14 }
15 ll query(int l, int r){
16     ll ans = 0;
17     while(r >= 1){
18         ans = max(a[r], ans);
19         r--;
20         while(r >= 1 + lowbit(r)){
21             ans = max(e[r], ans);
22             r -= lowbit(r);
23         }
24     }
25     return ans;
26 }
27 }bit;

```

4.4.3 二维树状数组

查询二维前缀和

```

1 #include<iostream>
2 #include<string.h>
3 #include<algorithm>
4 #include<stdio.h>
5
6 using namespace std;
7 const int N = 1e3+100;
8 int c[N][N];
9 int n;
10 int lowbit(int x){
11     return x&(-x);
12 }
13 void update(int x,int y,int k){
14     for(int i=x;i<=n;i+=lowbit(i)){
15         for(int j=y;j<=n;j+=lowbit(j)){
16             c[i][j]+=k;
17         }
18     }
19 }
20 int query(int x,int y){
21     int ans=0;
22     for(int i=x;i>=1;i-=lowbit(i)){
23         for(int j=y;j>=1;j-=lowbit(j)){
24             ans+=c[i][j];
25         }
26     }
27     return ans;
28 }

```

5 字符串

5.1 序列自动机

```

1 int nxt[N][27]; //距离i往后最近的字母j的位置
2 void init(char *s){
3     int l=strlen(s);
4     for(int i=0;i<26;i++) nxt[l][i]=INF;
5     for(int i=l-1;i>=0;i--){
6         for(int j=0;j<26;j++){
7             nxt[i][j]=nxt[i+1][j];
8         }
9         nxt[i][s[i]-'a']=i;
10    }
11 }

```

5.2 KMP 计算 next 函数

5.2.1 vector 版

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = 2e6+100;
5 int n,m;int a[N],b[N];
6 vector<int> res;
7 vector<int> cal(vector<int> a){
8     int n=(int)a.size();
9     vector<int> nxt(n);
10    for(int i=1;i<n;i++){
11        int j=nxt[i-1];
12        while(j>0&&a[i]!=a[j]) j=nxt[j-1];
13        if(a[i]==a[j]) j++;
14        nxt[i]=j;
15    }
16    return nxt;
17 }

```

5.2.2 KMP 匹配过程

```

1 int kmp(){
2     int i,j;
3     i=j=0;
4     while(i<n&&j<m){
5         if(s[i]==t[j]){
6             i++;j++;
7         }
8         else if(!j){
9             i++;
10        }
11        else{
12            j=nxt[j-1];
13        }
14    }
15    if(j==m) return i-m+1;
16    else return -1;
17 }

```

5.3 Z-function / Exkmp

```

1 vector<int> z_function(string s) {
2     int n = (int)s.length();
3     vector<int> z(n);
4     for (int i = 1, l = 0, r = 0; i < n; ++i) {
5         if (i <= r) z[i] = min(r - i + 1, z[i - l]);
6         while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
7         if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
8     }
9     return z;
10 }

```

5.4 Manacher

```

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

```

```

4  const int N = 1e6+100;
5  int p[N];
6  string s;
7  int manacher(string s){
8      string t="";
9      t+='*';
10     for(int i=0;i<(int)s.size();i++){
11         t+=s[i];
12         t+='*';
13     }
14     int ans=0;
15     int pos=0;int maxxright=0;
16     for(int i=0;i<(int)t.length();i++){
17         p[i]=maxxright>i?min(p[2*pos-i],maxxright-i):1; //关键
18         while(i-p[i]>=0&&i+p[i]<(int)t.length()&&t[i-p[i]]==t[i+p[i]]) p[i]++;
19         if(i+p[i]-1>maxxright){
20             maxxright=i+p[i]-1;
21             pos=i;
22         }
23         ans=max(ans,p[i]);
24     }
25     return ans-1;
26 }
27 int main(){
28     cin>>s;
29     cout<<manacher(s)<<endl;
30     return 0;
31 }

```

5.5 后缀数组

从 1 到 n 输出 $sa[i]$ $sa[i]$ 代表排名为 i 的下标 $ran[i]$ 代表下标为 i 的排名

```

1  string s;
2  int ran[N],tmp[N],sa[N];
3  int n,k;
4  bool cmp(int i,int j){
5      if(ran[i]!=ran[j]) return ran[i]<ran[j];
6      int ri=i+k<=n?ran[i+k]:-1;
7      int rj=j+k<=n?ran[j+k]:-1;
8      return ri<rj;
9  }
10 void construct_sa(string s,int *sa){
11     n=(int)s.length();
12     for(int i=0;i<=n;i++){
13         sa[i]=i;
14         ran[i]=i<n?s[i]:-1;
15     }
16     for(k=1;k<=n;k*=2){
17         sort(sa,sa+n+1,cmp);
18         tmp[sa[0]]=0;
19         for(int i=1;i<=n;i++){
20             tmp[sa[i]]=tmp[sa[i-1]]+(cmp(sa[i-1],sa[i])?1:0);
21         }
22         for(int i=0;i<=n;i++) ran[i]=tmp[i];
23     }
24 }

```

5.6 双哈希

```

1  #include<bits/stdc++.h>
2
3  using namespace std;
4  typedef long long ull;
5  typedef long long ll;
6  const int N = 1e6+100;
7  const int base1 = 233;
8  const int base2 = 2333;
9  const int mod1 = 1e9+9;
10 const int mod2 = 1e9+7;
11 ull hashes1[N],p1[N];
12 ull hashes2[N],p2[N];
13 string s;
14 ull gethash1(int l,int r){
15     if(l==0) return hashes1[r]%mod1;
16     return (hashes1[r]-(hashes1[l-1]%mod1*p1[r-l+1]%mod1)%mod1+mod1)%mod1;
17 }
18 ull gethash2(int l,int r){
19     if(l==0) return hashes2[r]%mod2;
20     return (hashes2[r]-(hashes2[l-1]%mod2*p2[r-l+1]%mod2)+mod2)%mod2;
21 }
22 int main(){
23     int n;cin>>n>>s;
24     hashes1[0]=s[0];p1[0]=1;
25     hashes2[0]=s[0];p2[0]=1;
26     for(int i=1;i<=n;i++){
27         hashes1[i]=(hashes1[i-1]*base1%mod1+(ull)s[i]%mod1)%mod1;
28         hashes2[i]=(hashes2[i-1]*base2%mod2+(ull)s[i]%mod2)%mod2;
29         p1[i]=(base1*p1[i-1])%mod1;
30         p2[i]=(base2*p2[i-1])%mod2;
31     }
32 }

```

5.7 字典树

5.7.1 指针版

```

1  struct Tire {
2      const int MAXN = 4e5 + 7;
3      int tr[MAXN][26], tot = 0;
4      int cnt[MAXN];
5
6      void insert(string s ) {
7          int cur = 0, sz = s.size();
8          cnt[cur] ++; //插入的字符串个数
9          for(int i = 0; i < sz; ++i) {
10             int to = s[i] - 'a';
11             if(!tr[cur][to]) tr[cur][to] = ++tot;
12             cur = tr[cur][to];
13         }
14         cnt[cur]++; //当前节点所表示的字符串的出现次数 + 1
15     }
16
17     int get(string s) {
18         int cur = 0, sz = s.size();
19         for(int i = 0; i < sz; ++i) {
20             int to = s[i] - 'a';
21             if(!tr[cur][to]) return 0;
22             cur = tr[cur][to];

```

```

23     }
24     return cnt[cur];
25 }
26 }trie;

```

5.7.2 数组版 01 字典树

从数组 a 中找出一个数使得 k 和它 xor 最大

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 const int N = (1e5+100)*33;
5 int tree[N][2];int a[N];
6 bool vis[N];
7 int tot;
8 void insert(int x){
9     int now=0;
10    for(int i=31;i>=0;i--){
11        int id=(x>>i)&1;
12        if(!tree[now][id]) tree[now][id]=++tot;
13        now=tree[now][id];
14    }
15 }
16 int find(int x){
17     int ans=0;
18     int now=0;
19     for(int i=31;i>=0;i--){
20         int id=!(x>>i)&1;
21         ans*=2;
22         if(tree[now][id]){
23             ans++;now=tree[now][id];
24         }
25         else now=tree[now][!id];
26     }
27     return ans;
28 }
29 int main(){
30     int T,n,m,s,all;
31     all=1;
32     scanf("%d",&T);
33     while(T--){
34         tot=0;
35         memset(tree,0,sizeof(tree));
36         // memset(vis,0,sizeof(vis));
37         scanf("%d %d",&n,&m);
38         for(int i=1;i<=n;i++) scanf("%d",&a[i]);
39         for(int i=1;i<=n;i++){
40             insert(a[i]);
41         }
42         printf("Case #d:\n",all++);
43         while(m--){
44             scanf("%d",&s);
45             printf("%d\n",find(s)^s);
46         }
47     }
48     return 0;
49 }

```

5.8 AC 自动机

```

1 namespace AC{
2     const int MAXN = 1e6 + 7;

```

```

3     //注意字符集大小
4     int tr[MAXN][30], tot = 0;
5     int cnt[MAXN], fail[MAXN];
6     void insert(string s){
7         int cur = 0;
8         for(int i = 0; i < s.size(); ++i){
9             int to = s[i] - 'a';
10            if(tr[cur][to] == 0) tr[cur][to] = ++tot;
11            cur = tr[cur][to];
12        }
13        cnt[cur]++;
14    }
15    queue<int> que;
16    void build(){
17        for(int i = 0; i < 26; ++i){
18            if(tr[0][i]) que.push(tr[0][i]);
19        }
20        while(!que.empty()){
21            int cur = que.front();
22            que.pop();
23            for(int i = 0; i < 26; ++i){
24                if(tr[cur][i]){
25                    fail[tr[cur][i]] = tr[fail[cur]][i];
26                    que.push(tr[cur][i]);
27                } else tr[cur][i] = tr[fail[cur]][i];
28            }
29        }
30    }
31    //注意每次匹配结束 cnt 数组发生改变
32    //如果需要再次匹配 需要重新建树
33    int query(string s){
34        int cur = 0, res = 0;
35        for(int i = 0; i < s.size(); ++i){
36            int to = s[i] - 'a';
37            cur = tr[cur][to];
38            for(int j = cur; j && cnt[j] != -1; j =
39                fail[j]){
40                res += cnt[j];
41                cnt[j] = -1;
42            }
43        }
44        return res;
45    }

```

5.9 最小表示法

```

1 int k = 0, i = 0, j = 1;
2 while (k < n && i < n && j < n) {
3     if (sec[(i + k) % n] == sec[(j + k) % n]) {
4         k++;
5     } else {
6         //最大表示法只需要将此处的 > 更改为 < 即可
7         sec[(i + k) % n] > sec[(j + k) % n] ? i = i + k +
8             1 : j = j + k + 1;
9         if (i == j) i++;
10        k = 0;
11    }
12 }
i = min(i, j);

```

6 C++ pbds

6.1 头文件

```
1 #include<ext/pb_ds/assoc_container.hpp>
2 #include<ext/pb_ds/tree_policy.hpp> //用tree
3 #include<ext/pb_ds/hash_policy.hpp> //用hash
4 #include<ext/pb_ds/trie_policy.hpp> //用trie
5 #include<ext/pb_ds/priority_queue.hpp> //用
   priority_queue
6 using namespace __gnu_pbds;
```

6.2 Hash

```
1 cc_hash_table<int,bool> h; //拉链法
2 gp_hash_table<int,bool> h; //探测法 稍快
```

6.3 Tree

```
1 #define pii pair<int,int>
2 #define mp(x,y) make_pair(x,y)
3 tree<pii,null_type,less<pii>,rb_tree_tag,
   tree_order_statistics_node_update> tr;
4 pii //存储的类型
5 null_type //无映射(低版本g++为null_mapped_type)
6 less<pii> //从小到大排序
7 rb_tree_tag //红黑树
8 tree_order_statistics_node_update //更新方式
9 tr.insert(mp(x,y)); //插入;
10 tr.erase(mp(x,y)); //删除;
11 tr.order_of_key(pii(x,y)); //求排名 从0开始
12 tr.find_by_order(x); //找k小值, 返回迭代器 从0开始
13 tr.join(b); //将b并入tr, 前提是两棵树类型一样且没有重复
   元素
14 tr.split(v,b); //分裂, key小于等于v的元素属于tr, 其余的
   属于b
15 tr.lower_bound(x); //返回第一个大于等于x的元素的迭代器
16 tr.upper_bound(x); //返回第一个大于x的元素的迭代器
17 //元素不能重复
18 //以上所有操作的时间复杂度均为O(logn)
```

6.4 Trie

```
1 typedef trie<string,null_type,
   trie_string_access_traits<>,pat_trie_tag,
   trie_prefix_search_node_update> tr;
2 //第一个参数必须为字符串类型, tag也有别的tag, 但pat最快,
   与tree相同, node_update支持自定义
3 tr.insert(s); //插入s
4 tr.erase(s); //删除s
5 tr.join(b); //将b并入tr
6 pair//pair的使用如下:
7 pair<tr::iterator,tr::iterator> range=base.
   prefix_range(x);
8 for(tr::iterator it=range.first;it!=range.second;it
   ++)
9     cout<<*it<<' '<<endl;
10 //pair中第一个是起始迭代器, 第二个是终止迭代器, 遍历过去就
   可以找到所有字符串了。
```

6.5 优先队列

```
1 priority_queue<int,greater<int>,TAG> Q; //小根堆, 大根
   堆写less<int>
2 /*其中的TAG为类型, 有以下几种:
3 pairing_heap_tag
4 thin_heap_tag
5 binomial_heap_tag
6 rc_binomial_heap_tag
7 binary_heap_tag
8 其中pairing_help_tag最快*/
9 Q.push(x);
10 Q.pop();
11 Q.top();
12 Q.join(b);
13 Q.empty();
14 Q.size();
15 Q.modify(it,6);
16 Q.erase(it);
17 //以上操作我都不讲了, pbds里的优先队列还可以用迭代器遍历
```