

Template of Team DFA

HPU : Codancer & Dicer

2019 年 10 月 30 日

目录

1 杂项	1		
1.1 Head & 快速读入	1		
1.2 <code>__int128</code> 输入输出	1		
1.3 O3 优化	1		
1.4 单调栈	1		
1.5 打印 LIS	1		
2 图论	1		
2.1 Dinic 最大流	1		
2.2 倍增求解 LCA	2		
2.3 有向图最小环	2		
2.4 带权二分图匹配-KM	2		
2.5 Tarjan	3		
2.5.1 缩点求 SCC	3		
2.5.2 2-SAT	4		
2.5.3 无向图点双连通	5		
2.5.4 无向图边双连通	5		
2.6 求无向连通图的第 K 大联通子图	6		
2.7 分层 (K) 图最短路	6		
2.8 点分治	7		
3 数学	7		
3.1 整除分块	7		
3.2 SG 函数打表	8		
3.3 线性素数 + 莫比乌斯函数打表	8		
3.4 高斯消元	8		
3.5 Lucas 定理求 $C(n, m) \% P$	8		
3.6 大数质因子分解 & 大素数检测	9		
3.7 python 通用中国剩余定理	9		
3.8 在线求组合数	10		
3.9 拉格朗日插值	10		
3.9.1 连续情况	10		
3.9.2 非连续情况	10		
3.10 辛普森自适应积分	11		
3.11 欧拉函数	11		
3.11.1 在线	11		
3.11.2 打表	11		
3.12 欧拉降幂	11		
4 数据结构	12		
4.1 线段树	12		
4.1.1 区间修改区间查询	12		
4.2 主席树	13		
4.2.1 区间第 k 小	13		
4.2.2 区间内小于等于 x 的最大值	13		
4.2.3 区间内距离 p 第 k 近的距离	13		
4.2.4 区间内小于等于 x 的最大值	14		
4.2.5 区间数的种类数	14		
4.2.6 区间内未出现过的最小自然数	14		
4.3 树链剖分	15		
4.4 树状数组	16		
4.4.1 二位偏序求矩形内点的个数	16		
4.4.2 树状数组求区间最值	17		
4.4.3 二维树状数组	18		
5 字符串	18		
5.1 序列自动机	18		
5.2 KMP 计算 next 函数	18		
5.2.1 vector 版	18		
5.2.2 KMP 匹配过程	18		
5.3 Z-function / Exkmp	18		
5.4 Manacher	18		
5.5 后缀数组	19		
5.6 双哈希	19		
5.7 字典树	19		
5.7.1 指针版	19		
5.7.2 数组版 01 字典树	19		
5.8 AC 自动机	20		
5.9 最小表示法	20		
6 计算几何	20		
6.1 基本的定义	20		
6.2 点和线	21		
6.2.1 计算投影的坐标	21		
6.2.2 计算 p 关于线段 s 的对称点	21		
6.2.3 判断线段是否相交	21		
6.2.4 线段平行和正交判断	21		
6.2.5 计算线段的交点	21		
6.2.6 点到直线的距离	21		
6.2.7 点到线段的最近距离	21		
6.3 多边形	22		
6.3.1 多边形面积	22		
6.3.2 判断多边形是否是凸包	22		
6.3.3 点和多边形的关系	22		
6.3.4 计算点集中的凸包	22		
6.3.5 直线和圆的交点	22		
6.4 旋转卡壳	22		
6.4.1 计算凸包直径	22		
7 C++ pbds	23		
7.1 头文件	23		
7.2 Hash	23		
7.3 Tree	23		
7.4 Trie	23		
7.5 优先队列	23		

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 二分图最大匹配

```

2 #include<bits/stdc++.h>
3
4 using namespace std;
5 const int N = 3000;
6 const int INF = 0x3f3f3f3f;
7 int S,T,n,m,w[N],dep[N],head[N],to[N],num=1,sum=0,x,
    nxt[N];
8 bool vis[N];
9 void add(int u,int v,int ww){
10     num++;
11     to[num]=v;nxt[num]=head[u];w[num]=ww;head[u]=num;
12     num++;
13     to[num]=u;nxt[num]=head[v];w[num]=0;head[v]=num;
14 }
15 queue<int> q;
16 bool bfs(){
17     while(!q.empty()) q.pop();
18     memset(vis,0,sizeof(vis));
19     while(!q.empty()) q.pop();
20     vis[S]=1;q.push(S);dep[S]=1;
21     while(!q.empty()){
22         int u=q.front();q.pop();
23         for(int i=head[u];i;i=nxt[i]){
24             int v=to[i];
25             if(vis[v]||w[i]<=0) continue;
26             dep[v]=dep[u]+1;
27             vis[v]=1;q.push(v);
28         }
29     }
30     return vis[T];
31 }
32 int dfs(int u,int d){
33     if(u==T||d==0){
34         return d;
35     }
36     int ret=0;
37     for(int i=head[u];i;i=nxt[i]){
38         int v=to[i];
39         if(dep[v]!=dep[u]+1||w[i]<=0) continue;
40         int flow=dfs(v,min(d,w[i]));
41         d-=flow;ret+=flow;
42         w[i]-=flow;w[i^1]+=flow;
43         if(d==0) break;
44     }
45     if(ret==0) dep[u]=-1;
46     return ret;
47 }
48 int main(){
49     scanf("%d %d",&n,&m);
50     S=0;T=n+1;
51     int u,v;
52     for(int i=1;i<=m;i++) add(S,i,1);
53     for(int i=m+1;i<=n;i++) add(i,T,1);
54     while(~scanf("%d %d",&u,&v)){
55         add(u,v,1);
56     }
57     while(bfs()) sum+=dfs(S,INF);
58     printf("%d\n",sum);
59     return 0;
60 }

```

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 带权二分图匹配-KM

```

1 //O(n^3) /by jls
2 #include<cstdio>
3 #include<iostream>
4 #include<string.h>
5 #include<queue>

```

```

6 #include<algorithm>
7
8 using namespace std;
9 const int N = 500;
10 int n,nx,ny,m,num,w[N][N],boy[N],gir[N],slack[N],pre[
    N],lx[N],ly[N];
11 bool vx[N],vy[N];
12 queue<int> q;
13 void Changematch(int k){
14     while(pre[k]){
15         boy[k]=pre[k];
16         int ne=gir[pre[k]];
17         gir[boy[k]]=k;
18         k=ne;
19     }
20 }
21 void Getmatch(int s){
22     while(!q.empty()) q.pop();q.push(s);
23     for(int i=1;i<=n;i++) slack[i]=1e9,vx[i]=vy[i]=pre
        [i]=0; //初始化
24     while(1){
25         while(!q.empty()){
26             int k=q.front();q.pop();
27             vx[k]=1;
28             int tmp;
29             for(int i=1;i<=n;i++){ //增广
30                 if(vy[i]==0&&(tmp=lx[k]+ly[i]-w[k][i])<
                    slack[i]){
31                     pre[i]=k;
32                     if(tmp==0){
33                         if(boy[i]==0){
34                             Changematch(i);return ;
35                         }
36                         vy[i]=1;q.push(boy[i]);
37                     }else{
38                         slack[i]=tmp;
39                     }
40                 }
41             }
42         }
43         int delta=1e9,where=0;
44         for(int i=1;i<=n;i++) if(vy[i]==0&&slack[i]<
            delta) delta=slack[i],where=i;
45         for(int i=1;i<=n;i++){
46             if(vx[i]) lx[i]-=delta;
47             if(vy[i]) ly[i]+=delta;
48             else{
49                 slack[i]-=delta;
50             }
51         }
52         if(boy[where]==0){
53             Changematch(where); return ;
54         }
55         vy[where]=1;q.push(boy[where]);
56     }
57 }
58 int main(){
59     scanf("%d %d %d",&nx,&ny,&m);
60     n=max(nx,ny);
61     int u,v,ww;
62     long long ans=0;
63     for(int i=1;i<=m;i++){
64         scanf("%d %d %d",&u,&v,&ww);
65         w[u][v]=max(w[u][v],ww);
66     }

```

```

67     for(int i=1;i<=n;i++){
68         for(int j=1;j<=n;j++) lx[i]=max(lx[i],w[i][j])
            ;
69     }
70     for(int i=1;i<=n;i++) Getmatch(i);
71     for(int i=1;i<=n;i++) ans+=(lx[i]+ly[i]);
72     printf("%lld\n",ans);
73     memset(gir,0,sizeof(gir));
74     for(int i=1;i<=ny;i++){
75         if(w[boy[i]][i]) gir[boy[i]]=i;
76         else gir[boy[i]]=0;
77     }
78     for(int i=1;i<=nx;i++){
79         printf("%d ",gir[i]);
80     }
81     puts("");
82     return 0;
83 }

```

2.5 Tarjan

2.5.1 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 }

```

2.5.2 2-SAT

```

1  /*
2  2-SAT
3  SAT是np完全问题，我们讨论的是2-SAT
4  对于n个bool变量，m组关系，每组关系有两个条件，只要满足其一
   即可
5  判断能否给这n个变量赋值使其满足条件，并输出变量的值
6  */
7  #include<bits/stdc++.h>
8
9  using namespace std;
10 const int N = 3000+100;
11 int n,m,scc,idx,color[N],low[N],dfn[N];
12 vector<int> G[N];
13 bool is_instack[N];stack<int> sta;
14 void init(int n){
15     scc=idx=0;
16     for(int i=1;i<=2*n;i++){
17         low[i]=dfn[i]=is_instack[i]=color[i]=0;
18         G[i].clear();
19     }
20     while(!sta.empty()) sta.pop();
21 }
22 void Tarjan(int u){
23     low[u]=dfn[u]=++idx;
24     sta.push(u);is_instack[u]=1;
25     for(int v:G[u]){
26         if(!dfn[v]){
27             Tarjan(v);
28             low[u]=min(low[u],low[v]);
29         }else if(is_instack[v]){
30             low[u]=min(low[u],dfn[v]);
31         }
32     }
33     if(low[u]==dfn[u]){
34         ++scc;
35         while(1){
36             int temp=sta.top();
37             color[temp]=scc;
38             is_instack[temp]=0;
39             sta.pop();
40             if(temp==u) break;
41         }
42     }
43 }
44 //i是妻子，i+n是丈夫
45 int main(){
46     while(~scanf("%d %d",&n,&m)){
47         int a,b,va,vb;
48         init(n);
49         for(int i=1;i<=m;i++){

```

```

50         scanf("%d %d %d %d",&a,&b,&va,&vb);
51         a++;b++;
52         G[a+n*(va&1)].push_back(b+n*(vb^1));
53         G[b+n*(vb&1)].push_back(a+n*(va^1));
54     }
55     for(int i=1;i<=(n<<1);i++){
56         if(!dfn[i]) Tarjan(i);
57     }
58     bool flag=0;
59     for(int i=1;i<=n;i++){
60         if(color[i]==color[i+n]){
61             puts("NO");
62             flag=1;
63             break;
64         }
65     }
66     if(!flag) puts("YES");
67 }
68 return 0;
69 }
70 \end{stlisting}
71 \subsubsection{求割点}
72 割点去掉后各联通快大小
73 \begin{lstlisting}
74 #include<bits/stdc++.h>
75
76 using namespace std;
77 const int N = 1e5+100;
78 typedef long long ll;
79 int dfn[N],low[N];
80 ll n,m,siz[N];
81 int idx;
82 vector<ll> G[N],GD[N];
83 bool jud[N];//是否为割点
84 void Tarjan(int u,int fa){
85     dfn[u]=low[u]=++idx;
86     siz[u]=1;
87     int allsiz=0;
88     for(int v:G[u]){
89         if(!dfn[v]){//没有访问过
90             Tarjan(v,u);
91             siz[u]+=siz[v];
92             low[u]=min(low[u],low[v]);
93             if(low[v]>=dfn[u]){//u为割点
94                 jud[u]=1;
95                 allsiz+=siz[v];
96                 GD[u].push_back(siz[v]);
97             }
98         }
99         else if(v!=fa){
100             low[u]=min(low[u],low[v]);
101         }
102     }
103     if(jud[u]&&n-allsiz-1){
104         GD[u].push_back(n-allsiz-1);
105     }
106 }
107 int main(){
108     idx=0;
109     cin>>n>>m;
110     int u,v;
111     for(int i=1;i<=m;i++){
112         cin>>u>>v;
113         G[u].push_back(v);
114         G[v].push_back(u);

```

```

115 }
116 Tarjan(1,0);
117 for(int i=1;i<=n;i++){
118     if(!jud[i]){
119         cout<<2*(n-1)<<endl;
120     }else{
121         ll ans=n*(n-1);
122         ll now=0;
123         for(ll v:GD[i]){
124             now+=v*(v-1);
125         }
126         cout<<ans-now<<endl;
127     }
128 }
129 return 0;
130 }

```

2.5.3 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 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 }

```

2.5.4 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 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 }

```

2.6 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 }

```

2.7 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", &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 }

```

2.8 点分治

```

1 typedef long long ll;
2 const int MOD = 1e9+7;
3 const int MAXN = 1e4 + 7;
4
5 int n, root, size, tot = 0;
6 int son[MAXN], f[MAXN], head[MAXN];
7 int dep[MAXN]; bool vis[MAXN];
8 struct node{
9     int u, w, nxt;
10 }; vector<node> E;
11 void add(int u, int v, int w) {
12     E.push_back(node{v, w, head[u]});
13     head[u] = tot++;
14 }
15 void init() {
16     memset(head, -1, sizeof head);
17     memset(vis, 0, sizeof vis);
18     E.clear(); tot = 0;
19 }
20 void get_rt(int x, int fa = 0) {

```

```

21     son[x] = 1; f[x] = 0;
22     for(int j = head[x]; ~j; j = E[j].nxt) {
23         int u = E[j].u, w = E[j].w;
24         if(vis[u] || u == fa) continue;
25         get_rt(u, x);
26         son[x] += son[u];
27         f[x] = max(f[x], son[u]);
28     }
29     f[x] = max(f[x], size - son[x]);
30     if(f[x] < f[root]) root = x;
31 }
32
33 void get_dep(int x, int fa) {
34     //得到每个节点到根节点的距离
35     for(int j = head[x]; ~j; j = E[j].nxt) {
36         int u = E[j].u, w = E[j].w;
37         if(vis[u] || u == fa) continue;
38         dep[u] = dep[x] + w;
39         get_dep(u, x);
40     }
41 }
42 void calc(int x, int op) {
43     get_dep(x, 0);
44     //得到深度, 处理答案
45     //update ans
46 }
47 void solve(int x) {
48     dep[x] = 0; calc(x, 1); vis[x] = 1;
49     for(int j = head[x]; ~j; j = E[j].nxt) {
50         int u = E[j].u, w = E[j].w;
51         if(vis[u]) continue;
52         dep[u] = w; calc(u, -1);
53         root = 0; size = son[u];
54         get_rt(u);
55         solve(root);
56     }
57 }
58 int main(int argc, char const *argv[])
59 {
60     while(~scanf("%d", &n)) {
61         init();
62         int u, v, w;
63         rep(i, 1, n-1) {
64             scanf("%d %d %d", &u, &v, &w);
65             add(u, v, w); add(v, u, w);
66         }
67         root = 0; f[0] = size = n;
68         get_rt(1, 0);
69         //init ans
70         solve(root);
71         //print ans
72     }
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 高斯消元

```
1 //计算n*(n+1)矩阵的解, ans[i]即为所求
2 #include<bits/stdc++.h>
3
4 using namespace std;
5 const double eps = 1e-7;
```

```
6 int n;
7 double a[110][110],ans[110];
8 void Gauss(int n){
9     for(int i=1;i<=n;i++){
10        int r=i;
11        for(int j=i+1;j<=n;j++){
12            if(fabs(a[r][i])<fabs(a[j][i])){
13                r=j;
14            }
15        }
16        if(fabs(a[r][i])<eps) return ;
17        if(i!=r)swap(a[i],a[r]);
18        double div=a[i][i];
19        for(int j=i;j<=n;j++){
20            a[i][j]/=div;
21        }
22        for(int j=i+1;j<=n;j++){
23            div=a[j][i];
24            for(int k=i;k<=n;k++){
25                a[j][k]-=a[i][k]*div;
26            }
27        }
28    }
29    ans[n]=a[n][n+1];
30    for(int i=n-1;i>=1;i--){
31        ans[i]=a[i][n+1];
32        for(int j=i+1;j<=n;j++){
33            ans[i]-=(a[i][j]*ans[j]);
34        }
35    } //回带操作
36 }
```

3.5 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-
24     m)!)
25 {
26     if(n < m)
27         return 0;
28     return farc[n] * pow(farc[m], mod-2) % mod * pow(
29         farc[n-m], mod-2) % mod; // 费马小定理求逆元
30 }
```

```

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

```

3.6 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++ )
42         {
43             ll y = Qmul( x , x , n );
44             if ( y==1&&x!=1&&x!=n-1 )
45                 return false;
46             x = y;
47         }
48         if ( x!=1 ) return false;
49     }
50     return true;

```

```

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 }

```

3.7 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

```

3.8 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]*i%mod)%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     }

```

3.9 拉格朗日插值

3.9.1 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;
28     for (ll i = 2; i <= n; i++) ifac[i] = -1ll*mod/i*
29         ifac[mod%i]%mod;
30     for (ll i = 2; i <= n; i++) ifac[i] = 1ll*ifac[i]*
31         ifac[i-1]%mod;
32     for (ll i = 0; i <= n; i++)
33         (ans += 1ll*y[i]*(i == 0 ? 1 : s1[i-1])%mod*s2
34             [i+1]%mod
35             *ifac[i]%mod*(((n-i)&1) ? -1 : 1)*ifac[n-i
36                 ]%mod) %= mod;
37     return (ans+mod)%mod;
38 }

```

```

32 int main(){
33     ll n,k;
34     cin>>n>>k;
35     if(k==0){
36         cout<<n<<endl;
37         return 0;
38     }
39     p[0]=0;
40     for(ll i=1;i<=k+2;i++) p[i]=(p[i-1]%mod+qpow(i,k))
41         %mod;
42     for(ll i=1;i<=k+2;i++) x[i]=i;
43     if(n<=k+2){
44         cout<<p[n]<<endl;
45     }
46     else{
47         cout<<lagrange(k+2,x,p,n)<<endl;
48     }
49     return 0;
50 }

```

3.9.2 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;
31             base2=(base2%mod*((p[j].x-p[i].x)%mod+mod)%
32                 mod)%mod;
33         }
34         ans=(ans%mod+(p[j].y%mod*base1%mod*qpow(base2,
35             mod-2,mod)%mod)%mod)%mod;
36     }
37     return ans;
38 }
39 int main(){
40     cin>>n>>k;
41     for(int i=1;i<=n;i++) cin>>p[i].x>>p[i].y;
42     cout<<Lagrange(k)<<endl;
43 }

```

```

40     return 0;
41 }

```

3.10 辛普森自适应积分

```

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

```

3.11 欧拉函数

比 n 小的与 n 互质的数的个数

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

```

3.11.2 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 }

```

3.12 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%mod;
61     if(b==1) return a%mod;
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))
13         %mod;

```

```

13     ans[rt<<1|1]=(ans[rt<<1|1]*mult[rt]+add[rt]*(r-mid
14         ))%mod;
15     mult[rt<<1]=(mult[rt]*mult[rt<<1])%mod;
16     mult[rt<<1|1]=(mult[rt]*mult[rt<<1|1])%mod;
17     add[rt<<1]=(add[rt<<1]*mult[rt]+add[rt])%mod;
18     add[rt<<1|1]=(add[rt<<1|1]*mult[rt]+add[rt])%mod;
19     add[rt]=0;
20     mult[rt]=1;
21     return ;
22 }
23 inline void buildtree(ll rt,ll l,ll r){
24     mult[rt]=1;
25     add[rt]=0;
26     if(l==r){
27         ans[rt]=a[l];
28         return ;
29     }
30     ll mid=(l+r)>>1;
31     buildtree(rt<<1,l,mid);
32     buildtree(rt<<1|1,mid+1,r);
33     pushup(rt);
34 }
35 inline void update1(ll n1,ll nr,ll l,ll r,ll rt,ll k)
36 {
37     if(n1<=l&&r<=nr){
38         ans[rt]=(ans[rt]*k)%mod;
39         add[rt]=(add[rt]*k)%mod;
40         mult[rt]=(mult[rt]*k)%mod;
41         return ;
42     }
43     pushdown(rt,l,r);
44     ll mid=(l+r)>>1;
45     if(n1<=mid){
46         update1(n1,nr,l,mid,rt<<1,k);
47     }
48     if(nr>mid) update1(n1,nr,mid+1,r,rt<<1|1,k);
49     pushup(rt);
50 }
51 inline void update2(ll n1,ll nr,ll l,ll r,ll rt,ll k)
52 {
53     if(n1<=l&&nr>=r){
54         add[rt]=(add[rt]+k)%mod;
55         ans[rt]=(ans[rt]+k*(r-l+1))%mod;
56         return ;
57     }
58     pushdown(rt,l,r);
59     ll mid=(l+r)>>1;
60     if(n1<=mid){
61         update2(n1,nr,l,mid,rt<<1,k);
62     }
63     if(nr>mid) update2(n1,nr,mid+1,r,rt<<1|1,k);
64     pushup(rt);
65 }
66 ll query(ll n1,ll nr,ll l,ll r,ll rt){
67     ll res=0;
68     if(n1<=l&&r<=nr){
69         return ans[rt]%mod;
70     }
71     ll mid=(l+r)>>1;
72     pushdown(rt,l,r);
73     if(n1<=mid) res=(res%mod+query(n1,nr,l,mid,rt<<1))
74         %mod;

```

```

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

```

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()
        +1;
12 }
13 void update(int l,int r,int &x,int y,int pos){
14     T[++cnt]=T[y];T[cnt].num++;x=cnt;
15     if(l==r) return ;
16     int mid=(l+r)>>1;
17     if(pos<=mid) update(l,mid,T[x].l,T[y].l,pos);
18     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],
            roots[i-1],getid(a[i]));
40         while(m--){
41             int l,r,k;
42             scanf("%d %d %d",&l,&r,&k);
43             printf("%d\n",v[query(1,n,roots[r],roots[l
                -1],k)-1]);
44         }
45     }
46     return 0;
47 }

```

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(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],
34      a[i]);
35  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(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],
34     a[i]);
35 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 struct SegTree{
18     struct Node{
19         int l, r;
20         ll lz, sum;
21         int mid(){return (l+r)>>1;}

```

```

22         int size(){return (r-l+1);}
23     }s[MAXN<<2];
24     inline void pushdown(int x){
25         s[ls].lz += s[x].lz;
26         s[ls].sum += s[x].lz * s[ls].size();
27         s[rs].lz += s[x].lz;
28         s[rs].sum += s[x].lz * s[rs].size();
29         s[x].lz = 0;
30     }
31     inline void pushup(int x){
32         s[x].sum = s[ls].sum + s[rs].sum;
33     }
34     inline void build(int x, int l, int r){
35         s[x].l = l; s[x].r = r;
36         if(l == r){
37             s[x].lz = 0;
38             s[x].sum = w[rnk[l]];
39             return;
40         }
41         int mid = s[x].mid();
42         build(ls, l, mid);
43         build(rs, mid + 1, r);
44         pushup(x);
45     }
46     inline ll query(int x, int l, int r){
47         if(s[x].l == l && s[x].r == r) return s[x].sum
48         ;
49         pushdown(x);
50         int mid = s[x].mid();
51         if(r <= mid) return query(ls, l, r);
52         else if(l > mid) return query(rs, l, r);
53         else return query(ls, l, mid) + query(rs, mid
54             + 1, r);
55     }
56     inline void updata(int x, int l, int r, int v){
57         if(s[x].l == l && s[x].r == r){
58             s[x].lz += v;
59             s[x].sum += 1LL * v * s[x].size();
60             return;
61         }
62         pushdown(x);
63         int mid = s[x].mid();
64         if(r <= mid) updata(ls, l, r, v);
65         else if(l > mid) updata(rs, l, r, v);
66         else {
67             updata(ls, l, mid, v);
68             updata(rs, mid + 1, r, v);
69         }
70         pushup(x);
71     }
72 }st;
73 void dfs1(int x, int f = 0){
74     son[x] = -1;
75     siz[x] = 1;
76     dep[x] = dep[f] + 1;
77     fa[x] = f;
78     for(int u: G[x]){
79         if(u == f) continue;
80         dfs1(u, x);
81         siz[x] += siz[u];
82         if(son[x] == -1 || siz[son[x]] < siz[u]) son[x
83             ] = u;
84     }
85 }

```

```

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){
136     return st.query(1, tid[x], tid[x] + siz[x] - 1);
137 }
138
139 //子树更新
140 inline void subtreeadd(int x, int w){
141     return st.updata(1, tid[x], tid[x] + siz[x] - 1, w);
142 }
143
144 //查询LCA
145 inline int lca(int u, int v){
146     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-=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
55 ll dig(ll x){
56     ll ans=0;
57     while(x){
58         ans+=x%10;
59         x/=10;
60     }
61     return ans;
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
116             [i]-1}]+mmp[{x_1[i]-1,y_1[i]-1}]);
117         }
118     }
119     return 0;
120 }

```

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        }
15    }
16    ll query(int l, int r){
17        ll ans = 0;
18        while(r >= 1){
19            ans = max(a[r], ans);
20            r--;
21            while(r >= 1 + lowbit(r)){
22                ans = max(e[r], ans);
23                r -= lowbit(r);
24            }
25        }
26        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)
18         :1;//关键
19     while(i-p[i]>=0&&i+p[i]<(int)t.length()&&t[i-p
20         [i]]==t[i+p[i]]) p[i]++;
21     if(i+p[i]-1>maxxright){
22         maxxright=i+p[i]-1;
23         pos=i;
24     }
25     ans=max(ans,p[i]);
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])
21                 ?1:0);
22         }
23         for(int i=0;i<=n;i++) ran[i]=tmp[i];
24     }
25 }

```

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]%
17         mod1)%mod1+mod1)%mod1;
18 }
19 ull gethash2(int l,int r){
20     if(l==0) return hashes2[r]%mod2;
21     return (hashes2[r]-(hashes2[l-1]%mod2*p2[r-l+1]%
22         mod2)+mod2)%mod2;
23 }
24 int main(){
25     int n;cin>>n>>s;
26     hashes1[0]=s[0];p1[0]=1;
27     hashes2[0]=s[0];p2[0]=1;
28     for(int i=1;i<=n;i++){
29         hashes1[i]=(hashes1[i-1]*base1%mod1+(ull)s[i]%
30             mod1)%mod1;
31         hashes2[i]=(hashes2[i-1]*base2%mod2+(ull)s[i]%
32             mod2)%mod2;
33         p1[i]=(base1*p1[i-1])%mod1;
34         p2[i]=(base2*p2[i-1])%mod2;
35     }
36 }

```

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 +
15             1
16     }
17
18     int get(string s) {
19         int cur = 0, sz = s.size();
20         for(int i = 0; i < sz; ++i) {
21             int to = s[i] - 'a';
22             if(!tr[cur][to]) return 0;
23             cur = tr[cur][to];
24         }
25         return cnt[cur];
26     }
27 }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 计算几何

6.1 基本的定义

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4 #define eps (1e-10);
5 //点
6 struct Point{

```

```

7   double x,y;
8   Point(double _x=0,double _y=0):x(_x),y(_y){}
9   Point operator + (Point p){return Point(x+p.x,y+p.
10      y);}
11   Point operator - (Point p){return Point(x-p.x,y-p.
12      y);}
13   Point operator * (double a){return Point(a*x,a*y)
14      ;}
15   Point operator / (double a){return Point(x/a,y/a)
16      ;}
17   double norm(){return x*x+y*y;}//模的平方
18   double ABS() {return sqrt(norm());};//模
19 };
20 //线段
21 struct Segment{
22     Point p1,p2;
23 };
24 //点积
25 double dot(Point a,Point b){
26     return a.x*b.x+a.y*b.y;
27 }
28 //叉积
29 double cross(Point a,Point b){
30     return a.x*b.y-a.y*b.x;
31 }

```

```

7   if(cross(B.p1-A.p1,B.p2-A.p1)*cross(B.p1-A.p2,B.p2
8      -A.p2)>0) return 0;
9   return 1;
10 }

```

6.2.4 线段平行和正交判断

```

1   bool Parallel(Segment a,Segment b){
2       Point alp=a.p2-a.p1;
3       Point beta=b.p2-b.p1;
4       if(cross(alp,beta)==0){
5           return 1;
6       }
7       return 0;
8   }
9   //判断线段正交
10  bool Orthogonal(Segment a,Segment b){
11      Point alp=a.p2-a.p1;
12      Point beta=b.p2-b.p1;
13      if(dot(alp,beta)==0){
14          return 1;
15      }
16      return 0;
17  }

```

6.2 点和线

6.2.1 计算投影的坐标

```

1   //p在S上的投影坐标
2   Point Projection(Point p,Segment s){
3       Point alp=p-s.p1;Point beta=s.p2-s.p1;
4       double res=dot(alp,beta)/beta.norm();
5       return s.p1+(beta*res);
6   }

```

6.2.2 计算 p 关于线段 s 的对称点

```

1   Point Reflection(Point p,Segment s){
2       Point p1=s.p1;Point p2=s.p2;
3       double A=(p1.y-p2.y);
4       double B=-(p1.x-p2.x);
5       double C=(p1.x-p2.x)*p1.y-p1.x*A;
6       return {((B*B-A*A)*p.x-2*A*B*p.y-2*A*C)/(A*A+B*B)
7          ,((A*A-B*B)*p.y-2*A*B*p.x-2*B*C)/(A*A+B*B)};
8   }

```

6.2.3 判断线段是否相交

```

1   bool Intersect(Segment A ,Segment B){
2       if(max(B.p1.x,B.p2.x)<min(A.p1.x,A.p2.x)) return
3          0;
4       if(max(B.p1.y,B.p2.y)<min(A.p1.y,A.p2.y)) return
5          0;
6       if(max(A.p1.y,A.p2.y)<min(B.p1.y,B.p2.y)) return
7          0;
8       if(max(A.p1.x,A.p2.x)<min(B.p1.x,B.p2.x)) return
9          0;
10      if(cross(A.p1-B.p1,A.p2-B.p1)*cross(A.p1-B.p2,A.p2
11         -B.p2)>0) return 0;

```

6.2.5 计算线段的交点

```

1   Point IntersectPoint(Segment A ,Segment B){
2       double a=A.p1.y-A.p2.y;
3       double b=A.p2.x-A.p1.x;
4       double e=A.p1.x*(A.p1.y-A.p2.y)-A.p1.y*(A.p1.x-A.
5          p2.x);
6       double c=B.p1.y-B.p2.y;
7       double d=B.p2.x-B.p1.x;
8       double f=B.p1.x*(B.p1.y-B.p2.y)-B.p1.y*(B.p1.x-B.
9          p2.x);
10      double ansx=(e*d-f*b)/(a*d-c*b);
11      double ansy=(a*f-c*e)/(a*d-c*b);
12      if(ansx>=0) ansx=fabs(ansx);
13      if(ansy>=0) ansy=fabs(ansy);
14      return {ansx,ansy};
15  }

```

6.2.6 点到直线的距离

```

1   //点到直线的距离
2   double PLDis(Point a,Segment s){
3       double A=s.p1.y-s.p2.y;
4       double B=s.p2.x-s.p1.x;
5       double C=(s.p1.x-s.p2.x)*s.p1.y-(s.p1.y-s.p2.y)*s.
6          p1.x;
7       return fabs(A*a.x+B*a.y+C)/sqrt((A*A+B*B));
8   }

```

6.2.7 点到线段的最近距离

```

1   double Segdis(Point A ,Segment B){
2       Point alp1=A-B.p1;
3       Point beta1=B.p2-B.p1;
4       Point alp2=A-B.p2;

```



```

5 Point beta2=B.p1-B.p2;
6 if(dot(alp1,beta1)<0||dot(alp2,beta2)<0){
7     return min(Pointdis(A,B.p1),Pointdis(A,B.p2));
8 }
9 double a=B.p1.y-B.p2.y;
10 double b=B.p2.x-B.p1.x;
11 double c=B.p1.y*(B.p1.x-B.p2.x)-B.p1.x*(B.p1.y-B.
12     p2.y);
13 return fabs(a*A.x+b*A.y+c)/sqrt(a*a+b*b);

```

6.3 多边形

6.3.1 多边形面积

```

1 vector<Point> polygon;
2 double Area(vector<Point> polygon){
3     double ans=0;
4     int n=(int)polygon.size();
5     for(int i=0;i<n;i++){
6         ans+=cross(polygon[i],polygon[(i+1)%n]);
7     }
8     return fabs(ans/2);
9 }

```

6.3.2 判断多边形是否是凸包

```

1 bool Isconvex(vector<Point> polygon){
2     int n=(int)polygon.size();
3     polygon.push_back(polygon[0]);
4     polygon.push_back(polygon[2]);
5     for(int i=0;i<n;i++){
6         Point a=polygon[i+1]-polygon[i];
7         Point b=polygon[i+2]-polygon[i+1];
8         if(cross(a,b)<0) return 0;
9     }
10    return 1;
11 }

```

6.3.3 点和多边形的关系

```

1 //1代表在多边形上, 2代表在内部, 0代表在外部
2 int Contain(vector<Point> G,Point p){
3     int n = G.size();
4     bool x=0;
5     for(int i=0;i<n;i++){
6         Point a=G[i]-p,b=G[(i+1)%n]-p;
7         if(abs(cross(a,b))<eps && dot(a,b)<eps) return
8             1;
9         if(a.y>b.y) swap(a,b);
10        if(a.y<eps&&eps<b.y&&cross(a,b)>eps) x=!x;
11    }
12    return (x?2:0);

```

6.3.4 计算点集中的凸包

```

1 vector<Point> Andrew(vector<Point> G){
2     sort(G.begin(),G.end(),cmp);
3     vector<Point> up,down;

```

```

4     int n=(int)G.size();
5     up.push_back(G[0]);
6     up.push_back(G[1]);
7     down.push_back(G[n-1]);
8     down.push_back(G[n-2]);
9     for(int i=2;i<n;i++){
10        while(up.size()>1&&cross(up[up.size()-2]-up[up
11            .size()-1],G[i]-up[up.size()-1])<0){
12            up.pop_back();
13        }
14        up.push_back(G[i]);
15    }
16    for(int i=n-3;i>=0;i--){
17        while(down.size()>1&&cross(down[down.size()
18            -2]-down[down.size()-1],G[i]-down[down.
19            size()-1])<0){
20            down.pop_back();
21        }
22        down.push_back(G[i]);
23    }
24    vector<Point> ans;
25    for(int i=down.size()-1;i>=1;i--) ans.push_back(
26        down[i]);
27    for(int i=up.size()-1;i>=1;i--) ans.push_back(up[i
28        ]);
29    return ans;

```

6.3.5 直线和圆的交点

```

1 vector<Point> CCL(Segment s,Point o,double r){
2     vector<Point> res;
3     Point x=Projection(o,s);
4     double dis=PLDis(o,s);
5     if(dis>r){//距离>r没有交点
6         return res;
7     }
8     if(dis==r){//只有一个交点
9         res.push_back(x);
10        res.push_back(x);
11    }
12    double beta=sqrt(r*r-dis*dis);//勾股定理
13    Point pp=s.p2-s.p1;
14    pp=pp/pp.ABS();//单位向量
15    Point ans1=x-pp*beta;
16    Point ans2=x+pp*beta;
17    res.push_back(ans1);
18    res.push_back(ans2);
19    return res;
20 }

```

6.4 旋转卡壳

6.4.1 计算凸包直径

```

1 //旋转卡壳计算凸包直径
2 double Diameter(vector<Point> G){
3     double ans=0;
4     int n=G.size();
5     for(int i=0,k=0;i<n;i++){
6         while((G[i]-G[k]).norm()<(G[i]-G[(k+1)%n]).
7             norm()) k=(k+1)%n;
8         ans=max(ans,(G[i]-G[k]).ABS());

```



```

8     }
9     return ans;
10 }

```

7 C++ pbds

7.1 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;

```

7.2 Hash

```

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

```

7.3 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)

```

7.4 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。

```

7.5 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

```