

小小青蛙听风就是雨

HiedanoAkyuu、Oneman233、KR12138

2019 年 10 月 24 日

目录

1 字符串	1		
1.1 KMP	1		
1.2 EX-KMP	1		
1.3 Manacher	1		
1.4 串的最小表示	2		
1.5 后缀数组	2		
1.5.1 倍增 SA	2		
1.5.2 DC3	2		
1.6 回文自动机	2		
1.7 AC 自动机	2		
1.7.1 多模匹配	2		
1.7.2 自动机上 DP	3		
1.8 后缀自动机	4		
2 计算几何	4		
2.1 二维几何	4		
2.2 三维几何	5		
3 图论	5		
3.1 最短路	5		
3.1.1 Dijkstra	5		
3.1.2 SPFA	6		
3.1.3 Floyd	6		
3.1.4 负环	6		
3.1.5 差分约束	6		
3.2 最小生成树	6		
3.2.1 Prim	6		
3.2.2 Kruskal	6		
3.2.3 最小生成树计数	7		
3.2.4 次小生成树	7		
3.2.5 最小乘积生成树	7		
3.3 树的直径	7		
3.4 LCA	7		
3.4.1 Tarjan 离线	7		
3.4.2 倍增 LCA	7		
3.5 无向图与有向图联通性	7		
3.5.1 割点	7		
3.5.2 桥	8		
3.5.3 e-DCC	9		
3.5.4 v-DCC	9		
3.5.5 SCC	9		
3.5.6 2-SAT	9		
3.5.7 支配树	9		
3.6 二分图	9		
3.6.1 最大匹配-匈牙利	9		
3.6.2 带权匹配-KM	10		
3.7 网络流	11		
3.7.1 最大流-Dinic	11		
3.7.2 最小费用最大流-Dij+Dinic	11		
3.7.3 上下界流	12		
3.8 欧拉路	12		
3.9 Prufer 序列	12		
4 数据结构	12		
4.1 树状数组	12		
4.2 线段树	12		
4.2.1 带优先级线段树	12		
4.2.2 吉司机线段树	12		
4.2.3 线段树维护扫描线	13		
4.3 RMQ	13		
4.3.1 一维	13		
4.3.2 二维	13		
4.4 树链剖分	13		
4.4.1 点剖分	13		
4.4.2 边剖分	15		
4.5 平衡树	15		
4.5.1 Treap	15		
4.5.2 Splay	16		
4.6 动态树	16		
4.7 主席树	16		
4.8 树套树	17		
4.8.1 线段树套 Treap	17		
4.8.2 树状数组套线段树	19		
4.9 K-D Tree	20		
4.10 分治	20		
4.10.1 CDQ	20		
4.10.2 点分治	20		
4.10.3 dsu on tree	20		
4.10.4 整体二分	20		
4.11 分块	20		
4.11.1 普通分块	20		
4.11.2 莫队	21		
4.12 线性基	21		
4.13 珂朵莉树	21		
4.14 跳舞链	22		
5 动态规划	22		
5.1 SOS	22		
5.2 动态 DP	23		
5.3 插头 DP	23		
6 数学	23		
6.1 矩阵类	23		
6.2 质数筛	23		
6.2.1 埃筛	23		
6.2.2 线筛	23		
6.3 质数判定	23		
6.3.1 Miller Rabin	23		
6.4 质因数分解	23		
6.4.1 Pollard-Rho	23		
6.5 逆元	23		
6.5.1 EX-GCD 求逆元	23		
6.5.2 线性筛逆元	23		
6.5.3 阶乘逆元	23		
6.6 欧拉函数	23		
6.6.1 欧拉线筛	23		
6.6.2 求单个数的欧拉函数	23		
6.6.3 欧拉降幂	23		
6.6.4 一般积性函数求法	23		
6.7 EX-GCD	23		
6.8 CRT	23		
6.9 N 次剩余	23		
6.10 数论分块	23		
6.11 高斯消元	23		
6.11.1 普通消元	23		
6.11.2 异或方程组消元	23		
6.12 莫比乌斯反演	23		
6.12.1 莫比乌斯函数	23		
6.12.2 杜教筛	23		
6.12.3 洲阁筛	23		
6.12.4 min25 筛	23		
6.13 BSGS	23		
6.14 FFT	23		
6.15 FWT	23		
6.16 NTT	23		
6.17 数值计算	23		
6.17.1 辛普森	23		
6.17.2 自适应辛普森	23		
6.18 康拓展开	23		
6.19 卢卡斯定理	23		

7 其他23

7.1 快读快写 23

7.2 约瑟夫环 23

7.3 悬线法 23

7.4 蔡勒公式 23

7.5 三角公式 23

7.6 海伦公式 23

7.7 匹克定理 23

7.8 组合计数 23

7.8.1 计数原理 23

7.8.2 卡特兰数 23

7.8.3 Polya 23

7.8.4 二项式反演公式 23

7.8.5 斯特林反演公式 23

7.8.6 组合数恒等式 23

1 字符串

1.1 KMP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN=1000005;
5 char s1[MAXN],s2[MAXN];
6 int nxt[MAXN];
7
8 /*
9  nxt[i] s2[i-x..i-1]=s2[0..x-1]且x最大
10  即s2[0..i]的真前缀与真后缀的最大匹配
11  "ABAAB\0"=>[-1 0 0 1 1 2]
12 */
13
14 void get_fail(char *s,int l)
15 {
16     int i=0,j;
17     j=nxt[0]=-1;
18     while(i<l)
19     {
20         while(~j&&s[j]!=s[i]) j=nxt[j];
21         nxt[++i]=++j;
22     }
23 }
24
25 void kmp(char *s1,char *s2,int l1,int l2)
26 {
27     int i=0,j=0;
28     get_fail(s2,l2);
29     while(i<l1)
30     {
31         while(~j&&s1[i]!=s2[j]) j=nxt[j];
32         i++,j++;
33         if(j>=l2); //匹配上了
34     }
35 }
36
37 int main()
38 {
39     scanf("%s%s",s1,s2);
40     int l1=strlen(s1),l2=strlen(s2);
41     kmp(s1,s2,l1,l2);
42     for(int i=0;i<=l2;i++)
43         printf("%d ",nxt[i]);
44     return 0;
45 }

```

1.2 EX-KMP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  ex[i]: s1[i..l1-1]与s2的最大公共前缀长度
6  exnext[i]: s2[i..l2-1]与s2的最大公共前缀长度
7  get_exnext(s2) 求exnext[]
8  exkmp(s1,s2) 求ex[]
9 */
10
11 const int N=500005;
12 char s1[N],s2[N];

```

```

13 int ex[N],exnext[N];
14
15 void get_exnext(char s[N])
16 {
17     int n=strlen(s),p=1,j,i;
18     exnext[0]=n;
19     for(i=0;i<n-1&&s[i]==s[i+1];i++);
20     exnext[1]=i;
21     for(i=2;i<n;i++)
22         if(exnext[i-p]+i<p+exnext[p])
23             exnext[i]=exnext[i-p];
24     else
25     {
26         j=exnext[p]+p-i;
27         if(j<0) j=0;
28         while(i+j<n&&s[j]==s[i+j]) j++;
29         exnext[i]=j;
30         p=i;
31     }
32 }
33
34 void exkmp(char s1[N],char s2[N])
35 {
36     int l1=strlen(s1),l2=strlen(s2),p=0,i,j;
37     get_exnext(s2);
38     for(i=0;i<l1&&i<l2&&s1[i]==s2[i];i++);
39     ex[0]=i;
40     for(int i=1;i<l1;i++)
41     {
42         if(exnext[i-p]+i<p+ex[p])
43             ex[i]=exnext[i-p];
44         else
45         {
46             j=ex[p]+p-i;
47             if(j<0) j=0;
48             while(i+j<l1&&s1[i+j]==s2[j]) j++;
49             ex[i]=j;
50             p=i;
51         }
52     }
53 }

```

1.3 Manacher

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int N=1000005;
5 int cnt,len,ans,p[N*2];
6 char s[N],ss[N*2];
7
8 void init() //将每两个字符中插入一个字符
9 {
10     len=strlen(s),cnt=1;
11     ss[0]='!',ss[cnt]='#';
12     for(int i=0;i<len;i++)
13         ss[++cnt]=s[i],ss[++cnt]='#';
14 }
15
16 void manacher()
17 {
18     int pos=0,mx=0;
19     for(int i=1;i<=cnt;i++)
20     {

```

```

21     if(i<mx) p[i]=min(p[pos*2-i],mx-i);
22     else p[i]=1;
23     while(ss[i+p[i]]==ss[i-p[i]]) p[i]++;
24     if(mx<i+p[i]) mx=i+p[i],pos=i;
25     ans=max(ans,p[i]-1);
26 }
27 }
28
29 int main()
30 {
31     scanf("%s",s);
32     init();
33     manacher();
34     printf("%d\n",ans);
35     return 0;
36 }

```

1.4 串的最小表示

1.5 后缀数组

1.5.1 倍增 SA

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  /*
5   str[0..len-1] 原串
6   sa[1..len] 排名第i的后缀的下标[1..len]
7   Rank[1..len] 从i开始的后缀的排名[1..len]
8   height[1..len] 排名第i的后缀与排名第i-1的后缀的lcp
9   i开始的后缀与j开始的后缀的lcp (Rank[i]<Rank[j])
10  min{height[Rank[i]+1..Rank[j]]}
11  */
12
13 const int MAXN=100005;
14 const int inf=0x3f3f3f3f;
15 int wa[MAXN],wb[MAXN],wv[MAXN],wz[MAXN],sa[MAXN],Rank
    [MAXN],height[MAXN];
16 char str[MAXN];
17
18 inline bool cmp(int *r,int a,int b,int l){return r[a
    ]==r[b]&&r[a+l]==r[b+l];}
19
20 void da(const char r[],int sa[],int n,int m)
21 {
22     int i,j,p,*x=wa,*y=wb,*t;
23     for(i=0;i<m;i++) wz[i]=0;
24     for(i=0;i<n;i++) wz[x[i]=r[i]]++;
25     for(i=1;i<m;i++) wz[i]+=wz[i-1];
26     for(i=n-1;i>=0;i--) sa[--wz[x[i]]]=i;
27     for(j=1,p=1;p<n;j*=2,m=p)
28     {
29         for(p=0,i=n-j;i<n;i++) y[p++]=i;
30         for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
31         for(i=0;i<n;i++) wv[i]=x[y[i]]++;
32         for(i=0;i<m;i++) wz[i]=0;
33         for(i=0;i<n;i++) wz[wv[i]]++;
34         for(i=1;i<m;i++) wz[i]+=wz[i-1];
35         for(i=n-1;i>=0;i--) sa[--wz[wv[i]]]=y[i];
36         for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
37             x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
38     }
39 }
40

```

```

41 void calheight(const char *r,int *sa,int n)
42 {
43     int i,j,k=0;
44     for(i=1;i<=n;i++) Rank[sa[i]]=i;
45     for(i=0;i<n;height[Rank[i+1]]=k)
46         for(k?k--:0,j=sa[Rank[i]-1];r[i+k]==r[j+k];k++);
47     for(int i=n;i>=1;--i) sa[i]++,Rank[i]=Rank[i-1];
48 }
49
50 int main()
51 {
52     scanf("%s",str);
53     int len=strlen(str);
54     da(str,sa,len+1,130); //字符的值域
55     calheight(str,sa,len);
56     for(int i=1;i<=len;i++)
57         printf("sa[%d] %d\n",i,sa[i]);
58     for(int i=1;i<=len;i++)
59         printf("Rank[%d] %d\n",i,Rank[i]);
60     for(int i=1;i<=len;i++)
61         printf("height[%d] %d\n",i,height[i]);
62     return 0;
63 }

```

1.5.2 DC3

1.6 回文自动机

1.7 AC 自动机

1.7.1 多模匹配

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  /*
5   trie静态开点+trie图优化
6   */
7
8  int sz,hd=1,nxt[1000005][26],fail[1000005],id
    [1000005],n;
9  char s[1000005];
10
11 void trie_clean()
12 {
13     sz=1;
14     memset(nxt,0,sizeof(nxt));
15     memset(fail,0,sizeof(fail));
16     memset(id,0,sizeof(id));
17 }
18
19 void trie_insert(int head,char s[],int len,int idx)
20 {
21     int p=head;
22     for(int i=0;i<len;i++)
23     {
24         int c=s[i]-'a';
25         if(!nxt[p][c]) nxt[p][c]=++sz;
26         p=nxt[p][c];
27     }
28     id[p]=idx;
29 }
30
31 void acatm_build(int head)
32 {

```

```

33 int p, tp;
34 queue<int> q;
35 q.push(head);
36 fail[head]=0;
37 while(!q.empty())
38 {
39     p=q.front();
40     q.pop();
41     for(int i=0; i<26; i++)
42         if(nxt[p][i])
43         {
44             fail[nxt[p][i]]=p==head?head:nxt[fail[p]][i];
45             q.push(nxt[p][i]);
46         }
47     else
48         nxt[p][i]=p==head?head:nxt[fail[p]][i];
49 }
50 }
51
52 int acatm_match(int head, char s[], int len)
53 {
54     int p=head, ret=0;
55     for(int i=0; i<len; i++)
56     {
57         int c=(int)s[i]-'a';
58         p=nxt[p][c];
59         for(int tp=p; tp; tp=fail[tp])
60             if(id[tp]) ret++;
61     }
62     return ret;
63 }

```

1.7.2 自动机上 DP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  每个串有个权值
6  求一个长度为n的串使得每个串的权值乘以出现次数之和最大
7 */
8
9 int fail[2005], nxt[2005][26], cnt[2005], sz, hd, n, m, dp
10 [55][2005], from[55][2005];
11 char s[105][15];
12 string dps[55][2005];
13
14 void clear()
15 {
16     sz=hd=1;
17     memset(dp, 0xc0, sizeof(dp));
18     memset(fail, 0, sizeof(fail));
19     memset(nxt, 0, sizeof(nxt));
20     memset(cnt, 0, sizeof(cnt));
21 }
22
23 void trie_insert(int head, char s[], int len, int idx)
24 {
25     int p=head;
26     for(int i=0; i<len; i++)
27     {
28         int c=s[i]-'a';
29         if(!nxt[p][c]) nxt[p][c]=++sz;

```

```

29         p=nxt[p][c];
30     }
31     cnt[p]+=idx;
32 }
33
34 void acatm_build(int head)
35 {
36     queue<int> q;
37     q.push(head);
38     while(!q.empty())
39     {
40         int p=q.front();
41         q.pop();
42         for(int i=0; i<26; i++)
43             if(nxt[p][i])
44             {
45                 fail[nxt[p][i]]=p==head?head:nxt[fail[p]][i];
46                 cnt[nxt[p][i]]+=cnt[fail[nxt[p][i]]];
47                 q.push(nxt[p][i]);
48             }
49             else
50                 nxt[p][i]=p==head?head:nxt[fail[p]][i];
51     }
52 }
53
54 bool scmp(string a, string b)
55 {
56     if(a.length()==b.length()) return a<b;
57     else return a.length()<b.length();
58 }
59
60 void solve()
61 {
62     clear();
63     scanf("%d%d", &n, &m);
64     for(int i=0; i<m; i++)
65         scanf("%s", s[i]);
66     for(int i=0; i<m; i++)
67     {
68         int x;
69         scanf("%d", &x);
70         trie_insert(hd, s[i], strlen(s[i]), x);
71     }
72     acatm_build(hd);
73
74     for(int i=0; i<=n; i++)
75         for(int j=0; j<=sz; j++)
76             dps[i][j]=string("");
77     int ans=0;
78     string anss;
79     queue<pair<int, int>> q;
80     dp[0][1]=0;
81     for(int i=0; i<n; i++)
82         for(int j=1; j<=sz; j++)
83             for(int k=0; k<26; k++)
84                 if(dp[i][j]+cnt[nxt[j][k]]>dp[i+1][nxt[j][k]]
85                     || dp[i][j]+cnt[nxt[j][k]]==dp[i+1][nxt[j][k]]
86                     && scmp(dps[i][j]+char('a'+k),
87                         dps[i+1][nxt[j][k]]))
88                 {
89                     dps[i+1][nxt[j][k]]=dps[i][j]+char('a'+k);
90                     dp[i+1][nxt[j][k]]=dp[i][j]+cnt[nxt[j][k]];

```

```

        j][k]];
    }
    for(int i=0;i<=n;i++)
        for(int j=1;j<=sz;j++)
            if(dp[i][j]>ans||dp[i][j]==ans&&scmp(dps[i][j],anss))
            {
                ans=dp[i][j];
                anss=dps[i][j];
            }
    for(int i=0;i<anss.length();i++)
        printf("%c",anss[i]);
    printf("\n");
}

int main()
{
    int _;
    scanf("%d",&_);
    while(--) solve();
    return 0;
}

```

1.8 后缀自动机

2 计算几何

2.1 二维几何

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #define db double
5 const db EPS=1e-9;
6 inline int sign(db a){return a<-EPS?-1:a>EPS;}
7 inline int cmp(db a,db b){return sign(a-b);}
8 struct P
9 {
10     db x,y;
11     P(){ }
12     P(db x,db y):x(x),y(y){ }
13     P operator+(P p){return {x+p.x,y+p.y};}
14     P operator-(P p){return {x-p.x,y-p.y};}
15     P operator*(db d){return {x*d,y*d};}
16     P operator/(db d){return {x/d,y/d};}
17     bool operator<(P p) const
18     {
19         int c=cmp(x,p.x);
20         if(c) return c==1;
21         return cmp(y,p.y)==1;
22     }
23     bool operator==(P o) const
24     {
25         return cmp(x,o.x)==0&&cmp(y,o.y)==0;
26     }
27     db distTo(P p){return (*this-p).abs();}
28     db alpha(){return atan2(y,x);}
29     void read(){scanf("%lf%lf",&x,&y);}
30     void write(){printf("%.10f,%.10f\n",x,y);}
31     db abs(){return sqrt(abs2());}
32     db abs2(){return x*x+y*y;}
33     P rot90(){return P(-y,x);}
34     P unit(){return *this/abs();}

```

```

35     int quad() const {return sign(y)==1||((sign(y)==0&&
        sign(x)>=0);}
36     db dot(P p){return x*p.x+y*p.y;}
37     db det(P p){return x*p.y-y*p.x;}
38     P rot(db an){return {x*cos(an)-y*sin(an),x*sin(an)
        +y*cos(an)};}
39 };
40
41 //For segment
42 #define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.
        x-p1.x)*(p2.y-p1.y))
43 #define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
44
45 bool chkLL(P p1,P p2,P q1,P q2) //0:parallel
46 {
47     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
48     return sign(a1+a2)!=0;
49 }
50
51 P isLL(P p1,P p2,P q1,P q2) //crossover point if
        chkLL()
52 {
53     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
54     return (p1*a2+p2*a1)/(a1+a2);
55 }
56
57 bool intersect(db l1,db r1,db l2,db r2)
58 {
59     if(l1>r1) swap(l1,r1);if(l2>r2) swap(l2,r2);
60     return !(cmp(r1,l2)==-1||cmp(r2,l1)==-1);
61 }
62
63 bool isSS(P p1,P p2,P q1,P q2)
64 {
65     return intersect(p1.x,p2.x,q1.x,q2.x)&&intersect(
        p1.y,p2.y,q1.y,q2.y)&&
66     crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<=0&&crossOp(q1
        ,q2,p1)*crossOp(q1,q2,p2)<=0;
67 }
68
69 bool isSS_strict(P p1,P p2,P q1,P q2)
70 {
71     return crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<0
        &&crossOp(q1,q2,p1)*crossOp(q1,q2,p2)<0;
72 }
73
74 bool isMiddle(db a,db m,db b)
75 {
76     return sign(a-m)==0||sign(b-m)==0||((a<m!=b<m);
77 }
78
79 bool isMiddle(P a,P m,P b)
80 {
81     return isMiddle(a.x,m.x,b.x)&&isMiddle(a.y,m.y,b.y
        );
82 }
83
84 bool onSeg(P p1,P p2,P q)
85 {
86     return crossOp(p1,p2,q)==0&&isMiddle(p1,q,p2);
87 }
88
89 bool onSeg_strict(P p1,P p2,P q)
90 {
91     return crossOp(p1,p2,q)==0&&sign((q-p1).dot(p1-p2)

```

```

    )*sign((q-p2).dot(p1-p2))<0;
}
P proj(P p1,P p2,P q)
{
    P dir=p2-p1;
    return p1+dir*(dir.dot(q-p1)/dir.abs2());
}
P reflect(P p1,P p2,P q)
{
    return proj(p1,p2,q)*2-q;
}
db nearest(P p1,P p2,P q)
{
    P h=proj(p1,p2,q);
    if(isMiddle(p1,h,p2))
        return q.distTo(h);
    return min(p1.distTo(q),p2.distTo(q));
}
db disSS(P p1,P p2,P q1,P q2) //dist of 2 segments
{
    if(isSS(p1,p2,q1,q2)) return 0;
    return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)),
        min(nearest(q1,q2,p1),nearest(q1,q2,p2)));
}
db rad(P p1,P p2)
{
    return atan2l(p1.det(p2),p1.dot(p2));
}
db area(vector<P> ps)
{
    db ret=0;
    for(int i=0;i<ps.size();i++)
        ret+=ps[i].det(ps[(i+1)%ps.size()]);
    return ret/2;
}
int contain(vector<P> ps,P p) //2:inside,1:on_seg,0:
    outside
{
    int n=ps.size(),ret=0;
    for(int i=0;i<n;i++)
    {
        P u=ps[i],v=ps[(i+1)%n];
        if(onSeg(u,v,p)) return 1;
        if(cmp(u.y,v.y)<=0) swap(u,v);
        if(cmp(p.y,u.y)>0||cmp(p.y,v.y)<=0) continue;
        ret^=crossOp(p,u,v)>0;
    }
    return ret*2;
}
vector<P> convexHull(vector<P> ps)
{
    int n=ps.size();if(n<=1) return ps;
    sort(ps.begin(),ps.end());
    vector<P> qs(n*2);int k=0;
    for(int i=0;i<n;qs[k++]=ps[i++])
        while(k>1&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
            --k;

```

```

    for(int i=n-2,t=k;i>=0;qs[k++]=ps[i--])
        while(k>t&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
            --k;
    qs.resize(k-1);
    return qs;
}
db convexDiameter(vector<P> ps)
{
    int n=ps.size();if(n<=1) return 0;
    int is=0,js=0;
    for(int k=1;k<n;k++) is=ps[k]<ps[is]?k:is,js=ps[js]
        <ps[k]?js:k;
    int i=is,j=js;
    db ret=ps[i].distTo(ps[j]);
    do{
        if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j])
            >=0) (++j)%=n;
        else (++i)%=n;
        ret=max(ret,ps[i].distTo(ps[j]));
    }while(i!=is||j!=js);
    return ret;
}

```

2.2 三维几何

3 图论

3.1 最短路

3.1.1 Dijkstra

```

1 #include <bits/stdc++.h>
2 #define mkp(a,b) make_pair(a,b)
3 #define fst first
4 #define snd second
5 //luogu P4779
6 using namespace std;
7 typedef pair<int,int> pii;
8 const int inf=0x3f3f3f3f;
9 const int N=1000005;
10
11 struct edge
12 {
13     int y,v;
14     edge(int Y,int V):y(Y),v(V){}
15 };
16 vector<edge> e[N];
17 void add(int x,int y,int v)
18 {
19     e[x].push_back(edge(y,v));
20 }
21
22 int n,m,s;
23 int dis[N];
24 bool vis[N];
25
26 void dij(int s)
27 {
28     memset(dis,0x3f,sizeof(dis));
29     dis[s]=0;
30     priority_queue<pii,vector<pii>,greater<pii>> q;
31     q.push(mkp(0,s));
32     while(!q.empty())

```



```

33 {
34     int x=q.top().snd;
35     q.pop();
36     if(vis[x]) continue;
37     vis[x]=1;
38     for(auto y:e[x])
39     {
40         if(dis[x]+y.v<dis[y.y])
41         {
42             dis[y.y]=dis[x]+y.v;
43             q.push(mkp(dis[y.y],y.y));
44         }
45     }
46 }
47 }
48
49 int main()
50 {
51     scanf("%d%d%d",&n,&m,&s);
52     for(int i=1,x,y,z;i<=m;++i)
53     {
54         scanf("%d%d%d",&x,&y,&z);
55         add(x,y,z);
56     }
57     dij(s);
58     for(int i=1;i<=n;++i)
59         printf("%d ",dis[i]==inf?2147483647:dis[i]);
60     return 0;
61 }

```

3.1.2 SPFA

3.1.3 Floyd

3.1.4 负环

3.1.5 差分约束

3.2 最小生成树

3.2.1 Prim

```

1 #include <bits/stdc++.h>
2 //luogu P3366
3 using namespace std;
4
5 /*
6  prim/kruskal一定要注意解决重边
7 */
8
9 const int N=5005;
10 const int inf=0x3f3f3f3f;
11
12 int n,m;
13 int mp[N][N];
14 int dis[N];
15
16 int prim(int s)
17 {
18     int sum=0;
19     int cnt=0;
20     for(int i=1;i<=n;++i)
21         dis[i]=mp[s][i];
22     cnt++;
23     while(1)
24     {

```

```

25         int mn=inf;
26         int now=-1;
27         for(int i=1;i<=n;++i)
28         {
29             if(dis[i]!=0&&dis[i]<mn)
30             {
31                 mn=dis[i];
32                 now=i;
33             }
34         }
35         if(now==-1) break;
36         sum+=dis[now];
37         dis[now]=0;
38         cnt++;
39         for(int i=1;i<=n;++i)
40         {
41             if(dis[i]!=0&&mp[now][i]<dis[i])
42                 dis[i]=mp[now][i];
43         }
44     }
45     if(cnt<n) return -1;
46     else return sum;
47 }
48
49 int main()
50 {
51     scanf("%d%d",&n,&m);
52     memset(mp,0x3f,sizeof(mp));
53     for(int i=1;i<=n;++i)
54         mp[i][i]=0;
55     for(int i=1,x,y,z;i<=m;++i)
56     {
57         scanf("%d%d%d",&x,&y,&z);
58         mp[x][y]=min(mp[x][y],z);
59         mp[y][x]=min(mp[y][x],z);
60     }
61     int ans=prim(1);
62     if(ans==-1) puts("orz");
63     else printf("%d",ans);
64     return 0;
65 }

```

3.2.2 Kruskal

```

1 #include <bits/stdc++.h>
2 //luogu P3366
3 using namespace std;
4
5 /*
6  prim/kruskal一定要注意解决重边
7 */
8
9 const int N=200005;
10
11 int n,m;
12 struct node
13 {
14     int x,y,z;
15 }o[N];
16
17 bool cmp(node a,node b)
18 {
19     return a.z<b.z;
20 }

```

```

21
22 int f[5005];
23 int _find(int x)
24 {
25     if(x!=f[x]) f[x]=_find(f[x]);
26     return f[x];
27 }
28 void _merge(int x,int y)
29 {
30     x=_find(x),y=_find(y);
31     if(x!=y) f[x]=y;
32 }
33
34 int kk()
35 {
36     for(int i=1;i<=n;++i)
37         f[i]=i;
38     sort(o+1,o+1+m,cmp);
39     int sum=0;
40     for(int i=1;i<=m;++i)
41     {
42         if(_find(o[i].x)!=_find(o[i].y))
43         {
44             sum+=o[i].z;
45             _merge(o[i].x,o[i].y);
46         }
47     }
48     int tmp=_find(1);
49     for(int i=2;i<=n;++i)
50         if(_find(i)!=tmp)
51             return -1;
52     return sum;
53 }
54
55 int main()
56 {
57     scanf("%d",&n,&m);
58     for(int i=1;i<=m;++i)
59         scanf("%d%d",&o[i].x,&o[i].y,&o[i].z);
60     int ans=kk();
61     if(ans==-1) puts("orz");
62     else printf("%d",ans);
63     return 0;
64 }

```

3.2.3 最小生成树计数

3.2.4 次小生成树

3.2.5 最小乘积生成树

3.3 树的直径

3.4 LCA

3.4.1 Tarjan 离线

3.4.2 倍增 LCA

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5     预处理 O(nlogn)
6     单次查询 O(logn)
7 */

```

```

8
9 const int MAXN=500005;
10 int n,q,dep[MAXN],s,lg[MAXN],fa[MAXN][32];
11 vector<int> e[MAXN];
12
13 void dfs(int now,int pa)
14 {
15     dep[now]=dep[pa]+1;
16     fa[now][0]=pa;
17     for(int i=1;(1<<i)<=dep[now];i++)
18         fa[now][i]=fa[fa[now][i-1]][i-1];
19     for(auto to:e[now])
20         if(to!=pa) dfs(to,now);
21 }
22
23 int lca(int x,int y)
24 {
25     if(dep[x]<dep[y]) swap(x,y);
26     while(dep[x]>dep[y]) x=fa[x][lg[dep[x]-dep[y]]-1];
27     if(x==y) return x;
28     for(int i=lg[dep[x]]-1;i>=0;i--)
29         if(fa[x][i]!=fa[y][i])
30             x=fa[x][i],y=fa[y][i];
31     return fa[x][0];
32 }
33
34 int main()
35 {
36     for(int i=1;i<MAXN;i++)
37         lg[i]=lg[i-1]+(1<<lg[i-1]==i);
38     scanf("%d%d",&n,&q,&s);
39     for(int i=0,x,y;i<n-1;i++)
40     {
41         scanf("%d%d",&x,&y);
42         e[x].push_back(y),e[y].push_back(x);
43     }
44     dep[0]=0;
45     dfs(s,0);
46     for(int i=0,x,y;i<q;i++)
47     {
48         scanf("%d%d",&x,&y);
49         printf("%d\n",lca(x,y));
50     }
51     return 0;
52 }

```

3.5 无向图与有向图联通性

3.5.1 割点

```

1 #include <bits/stdc++.h>
2 #define int long long
3 //luogu P3469
4
5 /*
6     tarjan求割点的算法中，如果不保证连通性，应该使用被注释
7     掉的遍历方法
8     part数组储存了被这个割点分成的不同的几块各自的大小
9 */
10
11 using namespace std;
12 const int N=100005;
13
14 int n,m,x,y;
15 vector<int> e[N],part[N];

```

```

15 bool is[N];
16 int dfn[N], low[N], timer=0;
17 int sz[N];
18
19 void tarjan(int u, int f)
20 {
21     dfn[u]=low[u]=++timer;
22     sz[u]++; //
23     int son=0, tmp=0;
24     for(auto v:e[u])
25     {
26         if(dfn[v]==0)
27         {
28             tarjan(v, u);
29             sz[u]+=sz[v]; //
30             low[u]=min(low[u], low[v]);
31             if(low[v]>=dfn[u]&&u!=f)
32             {
33                 is[u]=1;
34                 tmp+=sz[v]; //
35                 part[u].push_back(sz[v]); //
36             }
37             if(u==f) son++;
38         }
39         low[u]=min(low[u], dfn[v]);
40     }
41     if(son>=2&&u==f) is[u]=1; //point on the top
42     if(is[u]&&n-tmp-1!=0)
43         part[u].push_back(n-tmp-1); //
44 }
45
46 signed main()
47 {
48     scanf("%lld%lld", &n, &m);
49     for(int i=1; i<=m; ++i)
50     {
51         scanf("%lld%lld", &x, &y);
52         e[x].push_back(y), e[y].push_back(x);
53     }
54     /*
55     for(int i=1; i<=n; ++i)
56         if(!dfn[i]) tarjan(i, i);
57     */
58     tarjan(1, 0);
59     for(int i=1; i<=n; ++i)
60     {
61         if(!is[i]) printf("%lld\n", 2*(n-1));
62         else{
63             int tmp=0;
64             for(auto j:part[i])
65                 tmp+=j*(j-1);
66             printf("%lld\n", n*(n-1)-tmp);
67         }
68     }
69     return 0;
70 }

```

3.5.2 桥

```

1 #include <bits/stdc++.h>
2 #define mkp make_pair
3 //uva796
4 using namespace std;
5 const int N=1000005;

```

```

6 typedef pair<int, int> pii;
7 inline int read(){
8     char ch=getchar(); int s=0, w=1;
9     while(ch<48||ch>57){if(ch=='-')w=-1; ch=getchar();}
10    while(ch>=48&&ch<=57){s=(s<<1)+(s<<3)+ch-48; ch=
11        getchar();}
12    return s*w;
13 }
14 inline void write(int x){
15     if(x<0) putchar('-'), x=-x;
16     if(x>9) write(x/10);
17     putchar(x%10+48);
18 }
19 int n;
20 int dfn[N], low[N], timer=0;
21 int fa[N];
22 vector<int> e[N];
23 vector<pii> ans;
24
25 void tarjan(int u, int f)
26 {
27     fa[u]=f;
28     dfn[u]=low[u]=++timer;
29     for(auto v:e[u])
30     {
31         if(!dfn[v])
32         {
33             tarjan(v, u);
34             low[u]=min(low[u], low[v]);
35             //if(dfn[u]<low[v]) is[u][v]=1;
36             //u is v's father
37         }
38         else if(v!=f) low[u]=min(low[u], dfn[v]);
39     }
40 }
41
42 void init()
43 {
44     timer=0;
45     for(int i=0; i<n; ++i) dfn[i]=low[i]=fa[i]=0;
46     for(int i=0; i<n; ++i) e[i].clear();
47     ans.clear();
48 }
49
50 void gao()
51 {
52     for(int i=0; i<n; ++i)
53         if(!dfn[i]) tarjan(i, -1);
54     for(int i=0; i<n; ++i)
55     {
56         int F=fa[i];
57         if(F!=-1&&dfn[F]<low[i])
58             ans.emplace_back(min(F, i), max(F, i));
59     }
60     sort(ans.begin(), ans.end());
61     printf("%d critical links\n", (int)ans.size());
62     for(auto i:ans)
63         printf("%d - %d\n", i.first, i.second);
64     puts("");
65 }
66
67 int main()
68 {
69     while(~scanf("%d", &n))

```

```

70 {
71     if(n==0)
72     {
73         puts("0 critical links");
74         puts("");
75         continue;
76     }
77     init();
78     for(int i=0,x,y,z;i<n;++i)
79     {
80         scanf("%d %d",&x,&y);
81         for(int i=0;i<y;++i)
82             z=read(),
83             e[x].push_back(z),
84             e[z].push_back(x);
85     }
86     gao();
87 }
88 return 0;
89 }

```

3.5.3 e-DCC

3.5.4 v-DCC

3.5.5 SCC

3.5.6 2-SAT

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 //luogu P4782
4
5 /*
6  2-SAT用于求解有n个布尔变量x1-xn和m个需要满足的条件
7  每个条件形式为xi=0(1)||xj=0(1), 是否有可行解
8  注意要开两倍空间建反向边
9  */
10
11 const int N=2e6+5;
12
13 int n,m,a,va,b,vb;
14 int dfn[N],low[N],timer=0;
15 stack<int> s;
16 bool vis[N];
17 vector<int> e[N];
18 int co[N],color=0;
19
20 void add(int x,int y)
21 {
22     e[x].push_back(y);
23 }
24
25 void tarjan(int u)
26 {
27     dfn[u]=low[u]=++timer;
28     s.push(u);
29     vis[u]=1;
30     for(auto v:e[u])
31     {
32         if(!dfn[v])
33             tarjan(v),
34             low[u]=min(low[u],low[v]);
35         else if(vis[v])
36             low[u]=min(low[u],dfn[v]);
37     }

```

```

38     if(low[u]==dfn[u])
39     {
40         int v;
41         color++;
42         do
43         {
44             v=s.top();
45             s.pop();
46             vis[v]=0;
47             co[v]=color;
48         }
49         while(u!=v);
50     }
51 }
52
53 bool solve()
54 {
55     for(int i=1;i<=2*n;++i)
56         if(!dfn[i]) tarjan(i);
57     for(int i=1;i<=n;++i)
58         if(co[i]==co[i+n])
59             return 0;
60     return 1;
61 }
62
63 int main()
64 {
65     scanf("%d%d",&n,&m);
66     for(int i=1;i<=m;++i)
67     {
68         scanf("%d%d%d%d",&a,&va,&b,&vb);
69         int nota=va^1,notb=vb^1;
70         add(a+nota*n,b+vb*n); //not a and b
71         add(b+notb*n,a+va*n); //not b and a
72     }
73     if(solve())
74     {
75         puts("POSSIBLE");
76         for(int i=1;i<=n;++i)
77             printf("%d ",co[i]>co[i+n]);
78     }
79     else puts("IMPOSSIBLE");
80     return 0;
81 }

```

3.5.7 支配树

3.6 二分图

3.6.1 最大匹配-匈牙利

```

1 #include <bits/stdc++.h>
2 //luogu P3386
3 using namespace std;
4
5 /*
6  hungary每一次遍历必须要清空vis数组
7  */
8
9 const int N=1005;
10
11 vector<int> e[N];
12 bool vis[N];
13 int match[N],rematch[N];
14

```

```

15 bool dfs(int u)
16 {
17     for(auto v:e[u])
18     {
19         if(!vis[v]){
20             vis[v]=1;
21             if(match[v]==0||dfs(match[v]))
22             {
23                 match[v]=u;
24                 rematch[u]=v;
25                 return 1;
26             }
27         }
28     }
29     return 0;
30 }
31
32 int n,m,k;
33
34 int main()
35 {
36     scanf("%d%d%d",&n,&m,&k);
37     for(int i=1,x,y;i<=k;++i)
38     {
39         scanf("%d%d",&x,&y);
40         if(x>n||y>m) continue;
41         e[x].push_back(y);
42     }
43     int ans=0;
44     for(int i=1;i<=n;++i)
45     {
46         memset(vis,0,sizeof(vis));
47         if(dfs(i)) ans++;
48     }
49     printf("%d",ans);
50     return 0;
51 }

```

3.6.2 带权匹配-KM

```

1 #include <bits/stdc++.h>
2 //hdu 2255
3 using namespace std;
4
5 /*
6  KM仅用于最大带权匹配一定是最大匹配的情况中
7 */
8
9 const int N=305;
10 const int inf=0x3f3f3f3f;
11
12 int n,mp[N][N];
13 int la[N],lb[N],delta;
14 bool va[N],vb[N];
15 int match[N];
16
17 bool dfs(int x)
18 {
19     va[x]=1;
20     for(int y=1;y<=n;++y)
21     {
22         if(!vb[y]){
23             if(la[x]+lb[y]==mp[x][y])
24                 {

```

```

25                 vb[y]=1;
26                 if(!match[y]||dfs(match[y]))
27                 {
28                     match[y]=x;
29                     return 1;
30                 }
31             }
32         else
33             delta=min(delta,la[x]+lb[y]-mp[x][y]);
34     }
35 }
36 return 0;
37 }
38
39 int km()
40 {
41     for(int i=1;i<=n;++i)
42     {
43         match[i]=0;
44         la[i]=-inf;
45         lb[i]=0;
46         for(int j=1;j<=n;++j)
47         {
48             la[i]=max(la[i],mp[i][j]);
49         }
50     }
51     for(int i=1;i<=n;++i)
52     {
53         while(1)
54         {
55             memset(va,0,sizeof(va));
56             memset(vb,0,sizeof(vb));
57             delta=inf;
58             if(dfs(i)) break;
59             for(int j=1;j<=n;++j)
60             {
61                 if(va[j]) la[j]-=delta;
62                 if(vb[j]) lb[j]+=delta;
63             }
64         }
65     }
66     int ans=0;
67     for(int i=1;i<=n;++i)
68         ans+=mp[match[i]][i];
69     return ans;
70 }
71
72 int main()
73 {
74     while(~scanf("%d",&n))
75     {
76         memset(mp,-0x3f,sizeof(mp));
77         for(int i=1;i<=n;++i)
78         {
79             for(int j=1;j<=n;++j)
80             {
81                 scanf("%d",&mp[i][j]);
82             }
83         }
84         printf("%d\n",km());
85     }
86     return 0;
87 }

```

3.7 网络流

3.7.1 最大流-Dinic

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 /*
6  s,t 超级源、超级汇
7  cur[] 当前弧优化
8  时间复杂度 O(n^2*m)
9 */
10
11 const int MAXN=10005;
12 const ll inf=0x3f3f3f3f3f3f3f11;
13 int n,m,s,t,tot,dis[MAXN],cur[MAXN];
14 struct edge
15 {
16     int to,cap,rev;
17     edge(){}
18     edge(int to,int cap,int rev):to(to),cap(cap),rev(
19         rev){}
20 }
21 vector<edge> E[MAXN];
22
23 inline void add_edge(int x,int y,int f)
24 {
25     E[x].emplace_back(y,f,E[y].size());
26     E[y].emplace_back(x,0,E[x].size()-1);
27 }
28
29 int bfs()
30 {
31     for(int i=1;i<=n;i++) dis[i]=0x3f3f3f3f;
32     dis[s]=0;
33     queue<int> q;
34     q.push(s);
35     while(!q.empty())
36     {
37         int now=q.front();q.pop();
38         for(int i=0;i<E[now].size();i++)
39         {
40             edge &e=E[now][i];
41             if(dis[e.to]>dis[now]+1&&e.cap)
42             {
43                 dis[e.to]=dis[now]+1;
44                 if(e.to==t) return 1;
45                 q.push(e.to);
46             }
47         }
48     }
49     return 0;
50 }
51
52 ll dfs(int now,ll flow)
53 {
54     if(now==t) return flow;
55     ll rest=flow,k;
56     for(int i=cur[now];i<E[now].size();i++)
57     {
58         edge &e=E[now][i];
59         if(e.cap&&dis[e.to]==dis[now]+1)
60         {
61             cur[now]=i;
62             k=dfs(e.to,min(rest,(long long)e.cap));
63             E[e.to][e.rev].cap+=k;
64             rest-=k;
65         }
66     }
67     return flow-rest;
68 }
69
70 ll dinic()
71 {
72     ll ret=0,delta;
73     while(bfs())
74     {
75         for(int i=1;i<=n;i++) cur[i]=0;
76         while(delta=dfs(s,inf)) ret+=delta;
77     }
78     return ret;
79 }

```

```

62         e.cap-=k;
63         E[e.to][e.rev].cap+=k;
64         rest-=k;
65     }
66 }
67 return flow-rest;
68 }
69
70 ll dinic()
71 {
72     ll ret=0,delta;
73     while(bfs())
74     {
75         for(int i=1;i<=n;i++) cur[i]=0;
76         while(delta=dfs(s,inf)) ret+=delta;
77     }
78     return ret;
79 }

```

3.7.2 最小费用最大流-Dij+Dinic

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef pair<int,int> pii;
4
5 /*
6  第一遍跑的spfa,然后是加上势函数的dij,玄学
7  h[] 势函数
8  cur[] 当前弧优化
9  msf 最大流时的最小费用
10 s,t 超级源、超级汇
11 时间复杂度 O(n^2*m)
12 */
13
14 const int MAXN=2005;
15 const int inf=0x3f3f3f3f;
16 int msf,s,t,cur[MAXN],dis[MAXN],vis[MAXN],h[MAXN];
17 struct edge
18 {
19     int to,val,cap,rev;
20     edge(){}
21     edge(int to,int cap,int val,int rev):to(to),cap(
22         cap),val(val),rev(rev){}
23 }
24 vector<edge> E[MAXN];
25
26 inline void add_edge(int x,int y,int f,int cost)
27 {
28     E[x].emplace_back(y,f,cost,E[y].size());
29     E[y].emplace_back(x,0,-cost,E[x].size()-1);
30 }
31
32 int dij()
33 {
34     fill(dis,dis+t+1,inf);
35     priority_queue<pii,vector<pii>,greater<pii>> q;
36     q.emplace(0,s);dis[s]=0;
37     while(!q.empty())
38     {
39         pii p=q.top();q.pop();
40         int now=p.second;
41         if(dis[now]<p.first) continue;
42         for(int i=0;i<E[now].size();i++)
43         {

```

```

43     edge &e=E[now][i];
44     if(e.cap>0&&dis[e.to]>p.first+e.val+h[now]-
45         h[e.to])
46     {
47         dis[e.to]=p.first+e.val+h[now]-h[e.to];
48         q.emplace(dis[e.to],e.to);
49     }
50 }
51 return dis[t]!=inf;
52 }
53
54 int dfs(int now,int flow)
55 {
56     if(now==t) return flow;
57     int rest=flow,k;
58     vis[now]=1;
59     for(int i=cur[now];i<E[now].size();i++)
60     {
61         edge &e=E[now][i];
62         if(e.cap&&dis[now]+e.val+h[now]-h[e.to]==dis[e
63             .to]&&!vis[e.to])
64         {
65             cur[now]=i;
66             k=dfs(e.to,min(e.cap,rest));
67             e.cap-=k;
68             E[e.to][e.rev].cap+=k;
69             rest-=k;
70             msf+=k*e.val;
71         }
72     }
73     vis[now]=0;
74     return flow-rest;
75 }
76
77 int dinic()
78 {
79     int ret=0,delta;
80     while(dij())
81     {
82         for(int i=s;i<=t;i++) cur[i]=0;
83         while(delta=dfs(s,inf)) ret+=delta;
84         for(int i=s;i<=t;i++) h[i]+=(dis[i]==inf)?0:
85             dis[i];
86     }
87     return ret;
88 }

```

3.7.3 上下界流

3.8 欧拉路

```

1 #include <bits/stdc++.h>
2 //luogu P2731
3 using namespace std;
4 const int N=505;
5
6 /*
7  euler_path一定要找到正确的起点
8 */
9
10 int n;
11 int mp[N][N];
12 stack<int> st;
13 int deg[N];

```

```

14
15 void dfs(int x)
16 {
17     for(int i=1;i<=500;++i)
18     {
19         if(mp[x][i])
20         {
21             mp[x][i]--;
22             mp[i][x]--;
23             dfs(i);
24         }
25     }
26     st.push(x);
27 }
28
29 int main()
30 {
31     scanf("%d",&n);
32     for(int i=1,x,y;i<=n;++i)
33     {
34         scanf("%d%d",&x,&y);
35         mp[x][y]++;
36         mp[y][x]++;
37         deg[x]++;
38         deg[y]++;
39     }
40     int s=1;
41     for(int i=1;i<=500;++i)
42     {
43         if(deg[i]%2==1)
44         {
45             s=i;
46             break;
47         }
48     }
49     dfs(s);
50     while(!st.empty())
51     {
52         printf("%d\n",st.top());
53         st.pop();
54     }
55     return 0;
56 }

```

3.9 Prufer 序列

4 数据结构

4.1 树状数组

4.2 线段树

4.2.1 带优先级线段树

4.2.2 吉司机线段树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 /*
6  modify 将区间大于x的数变成x
7  query 询问区间和
8  单次复杂度 O(log^2(n))
9 */

```

```

10
11 const ll INF=0xc0c0c0c0c0c0c0c0ll;
12 const int MAXN=200005;
13 ll seg[MAXN<<2],m1[MAXN<<2],m2[MAXN<<2],cnt[MAXN<<2],
    tag[MAXN<<2],a[MAXN];
14 int n,q;
15
16 void pushdown(int rt)
17 {
18     if(!tag[rt]) return;
19     ll y=m1[rt];
20     if(y<m1[rt<<1])
21     {
22         tag[rt<<1]=1;
23         seg[rt<<1]-=(m1[rt<<1]-y)*cnt[rt<<1];
24         m1[rt<<1]=y;
25     }
26     if(y<m1[rt<<1|1])
27     {
28         tag[rt<<1|1]=1;
29         seg[rt<<1|1]-=(m1[rt<<1|1]-y)*cnt[rt<<1|1];
30         m1[rt<<1|1]=y;
31     }
32     tag[rt]=0;
33 }
34
35 void pushup(int rt)
36 {
37     seg[rt]=seg[rt<<1]+seg[rt<<1|1];
38     if(m1[rt<<1]==m1[rt<<1|1])
39     {
40         m1[rt]=m1[rt<<1];
41         cnt[rt]=cnt[rt<<1]+cnt[rt<<1|1];
42         m2[rt]=max(m2[rt<<1],m2[rt<<1|1]);
43     }
44     else if(m1[rt<<1]>m1[rt<<1|1])
45     {
46         m1[rt]=m1[rt<<1];
47         cnt[rt]=cnt[rt<<1];
48         m2[rt]=max(m2[rt<<1],m1[rt<<1|1]);
49     }
50     else
51     {
52         m1[rt]=m1[rt<<1|1];
53         cnt[rt]=cnt[rt<<1|1];
54         m2[rt]=max(m2[rt<<1|1],m1[rt<<1]);
55     }
56 }
57
58 void build(int rt,int l,int r)
59 {
60     tag[rt]=0;
61     if(l==r)
62     {
63         seg[rt]=m1[rt]=a[l];
64         cnt[rt]=1;
65         m2[rt]=INF;
66         return;
67     }
68     int m=l+r>>1;
69     if(l<=m) build(rt<<1,l,m);
70     if(m<r) build(rt<<1|1,m+1,r);
71     pushup(rt);
72 }
73

```

```

74 void modify(int rt,int l,int r,int L,int R,ll y)
75 {
76     if(y>=m1[rt]) return;
77     if(L<=l&&r<=R&&y>m2[rt])
78     {
79         tag[rt]=1;
80         seg[rt]-=(m1[rt]-y)*cnt[rt];
81         m1[rt]=y;
82         return;
83     }
84     pushdown(rt);
85     int m=l+r>>1;
86     if(L<=m) modify(rt<<1,l,m,L,R,y);
87     if(m<R) modify(rt<<1|1,m+1,r,L,R,y);
88     pushup(rt);
89 }
90
91 ll query(int rt,int l,int r,int L,int R)
92 {
93     if(L<=l&&r<=R) return seg[rt];
94     int m=l+r>>1;
95     pushdown(rt);
96     ll ret=0;
97     if(L<=m) ret+=query(rt<<1,l,m,L,R);
98     if(m<R) ret+=query(rt<<1|1,m+1,r,L,R);
99     pushup(rt);
100     return ret;
101 }

```

4.2.3 线段树维护扫描线

4.3 RMQ

4.3.1 一维

4.3.2 两维

4.4 树链剖分

4.4.1 点剖分

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 /*
6     轻重链剖分 单次复杂度  $O(\log^2(n))$ 
7     a[i] 表示dfs标号为i的点的值,而非点i的值
8     1 x y z 表示将树从x到y结点最短路径上所有节点值都加上z
9     2 x y 表示求树从x到y结点最短路径上所有节点值之和
10    3 x z 表示将以x为根节点的子树内所有节点值都加上z
11    4 x 表示求以x为根节点的子树内所有节点值之和
12 */
13
14 const int MAXN=100005;
15 ll mod,lazy[MAXN<<2],seg[MAXN<<2],a[MAXN],tmp[MAXN];
16 int n,q,r,cnt,tot,dep[MAXN],top[MAXN],id[MAXN],son[
    MAXN],num[MAXN],fa[MAXN];
17 vector<int> e[MAXN];
18
19 void dfs1(int now,int f)
20 {
21     dep[now]=dep[f]+1;
22     fa[now]=f;
23     num[now]=1;
24     son[now]=0;

```



```

25     for(auto to:e[now])
26     {
27         if(to==f) continue;
28         dfs1(to,now);
29         num[now]+=num[to];
30         if(num[to]>num[son[now]]) son[now]=to;
31     }
32 }
33
34 void dfs2(int now,int f)
35 {
36     id[now]=++cnt;
37     top[now]=f;
38     if(son[now]) dfs2(son[now],f);
39     for(auto to:e[now])
40         if(to!=fa[now]&&to!=son[now])
41             dfs2(to,to);
42 }
43
44 inline void pushdown(int rt,ll lnum,ll rnum)
45 {
46     if(!lazy[rt]) return;
47     seg[rt<<1]=(seg[rt<<1]+lazy[rt]*lnum%mod)%mod;
48     seg[rt<<1|1]=(seg[rt<<1|1]+lazy[rt]*rnum%mod)%mod;
49     lazy[rt<<1]=(lazy[rt<<1]+lazy[rt])%mod;
50     lazy[rt<<1|1]=(lazy[rt<<1|1]+lazy[rt])%mod;
51     lazy[rt]=0;
52 }
53
54 inline void pushup(int rt)
55 {
56     seg[rt]=(seg[rt<<1]+seg[rt<<1|1])%mod;
57 }
58
59 void build(int rt,int l,int r)
60 {
61     lazy[rt]=0;
62     if(l==r)
63     {
64         seg[rt]=a[l]%mod;
65         return;
66     }
67     int m=l+r>>1;
68     if(l<=m) build(rt<<1,l,m);
69     if(m<r) build(rt<<1|1,m+1,r);
70     pushup(rt);
71 }
72
73 void modify(int rt,int l,int r,int L,int R,ll x)
74 {
75     if(L<=l&&r<=R)
76     {
77         lazy[rt]=(lazy[rt]+x)%mod;
78         seg[rt]=(seg[rt]+x*(r-l+1)%mod)%mod;
79         return;
80     }
81     int m=l+r>>1;
82     pushdown(rt,m-l+1,r-m);
83     if(L<=m) modify(rt<<1,l,m,L,R,x);
84     if(m<R) modify(rt<<1|1,m+1,r,L,R,x);
85     pushup(rt);
86 }
87
88 ll query(int rt,int l,int r,int L,int R)
89 {

```

```

90     if(L<=l&&r<=R) return seg[rt];
91     int m=l+r>>1;
92     ll ret=0;
93     pushdown(rt,m-l+1,r-m);
94     if(L<=m) ret=(ret+query(rt<<1,l,m,L,R))%mod;
95     if(m<R) ret=(ret+query(rt<<1|1,m+1,r,L,R))%mod;
96     pushup(rt);
97     return ret;
98 }
99
100 int main()
101 {
102     scanf("%d%d%d%lld",&n,&q,&r,&mod);
103     for(int i=1;i<=n;i++) scanf("%lld",&tmp[i]);
104     for(int i=1,x,y;i<=n;i++)
105     {
106         scanf("%d%d",&x,&y);
107         e[x].push_back(y),e[y].push_back(x);
108     }
109     num[0]=0,dep[r]=0;
110     dfs1(r,r);
111     dfs2(r,r);
112     for(int i=1;i<=n;i++) a[id[i]]=tmp[i];
113     build(1,1,n);
114
115     while(q--)
116     {
117         int op,x,y,ll z;
118         scanf("%d%d",&op,&x);
119         if(op==4)
120         {
121             printf("%lld\n",query(1,1,n,id[x],id[x]+num
122                 [x]-1));
123             continue;
124         }
125         if(op==1)
126         {
127             scanf("%d%lld",&y,&z);z%=mod;
128             while(top[x]!=top[y])
129             {
130                 if(dep[top[x]]<dep[top[y]]) swap(x,y);
131                 modify(1,1,n,id[top[x]],id[x],z);
132                 x=fa[top[x]];
133             }
134             if(dep[x]>dep[y]) swap(x,y);
135             modify(1,1,n,id[x],id[y],z);
136         }
137         else if(op==2)
138         {
139             scanf("%d",&y);
140             ll ans=0;
141             while(top[x]!=top[y])
142             {
143                 if(dep[top[x]]<dep[top[y]]) swap(x,y);
144                 ans=(ans+query(1,1,n,id[top[x]],id[x]))%
145                     mod;
146                 x=fa[top[x]];
147             }
148             if(dep[x]>dep[y]) swap(x,y);
149             ans=(ans+query(1,1,n,id[x],id[y]))%mod;
150             printf("%lld\n",ans);
151         }
152         else
153         {
154             scanf("%lld",&z);z%=mod;

```

```

153         modify(1,1,n,id[x],id[x]+num[x]-1,z);
154     }
155 }
156 return 0;
157 }

```

4.4.2 边剖分

4.5 平衡树

4.5.1 Treap

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN=1e5+5;
5 const int inf=0x7fffffff;
6 int n,op,x;
7
8 /*
9  树内初始化时有无穷大和无穷小两个结点
10  _delete(root,x) 删除一个x
11  _insert(root,x) 插入一个x
12  getRank(root,x) 返回x的排名+1(包含了无穷小)
13  getVal(root,x+1) 返回排名为x的数
14  getPrev(x) x的前驱
15  getNext(x) x的后继
16 */
17
18 namespace Treap
19 {
20     int tot,root;
21     struct node
22     {
23         int cnt,val,dat,siz,lc,rc;
24     }bst[MAXN];
25
26     inline void pushup(int rt)
27     {
28         bst[rt].siz=bst[rt].cnt;
29         if(bst[rt].lc) bst[rt].siz+=bst[bst[rt].lc].siz;
30         if(bst[rt].rc) bst[rt].siz+=bst[bst[rt].rc].siz;
31     }
32
33     inline void zig(int &rt)
34     {
35         int p=bst[rt].lc;
36         bst[rt].lc=bst[p].rc;
37         bst[p].rc=rt;
38         rt=p;
39         pushup(bst[rt].rc);pushup(rt);
40     }
41
42     inline void zag(int &rt)
43     {
44         int p=bst[rt].rc;
45         bst[rt].rc=bst[p].lc;
46         bst[p].lc=rt;
47         rt=p;
48         pushup(bst[rt].lc);pushup(rt);
49     }
50
51     int new_node(int val)

```

```

52     {
53         bst[++tot].val=val;
54         bst[tot].dat=rand();
55         bst[tot].siz=bst[tot].cnt=1;
56         bst[tot].lc=bst[tot].rc=0;
57         return tot;
58     }
59
60     void build()
61     {
62         new_node(-inf);new_node(inf);
63         root=1,bst[1].rc=2;
64         pushup(1);
65     }
66
67     void _insert(int &rt,int val)
68     {
69         if(rt==0)
70         {
71             rt=new_node(val);
72             return;
73         }
74         if(bst[rt].val==val)
75         {
76             bst[rt].cnt++;
77             pushup(rt);
78             return;
79         }
80         if(val<bst[rt].val)
81         {
82             _insert(bst[rt].lc,val);
83             if(bst[rt].dat<bst[bst[rt].lc].dat) zig(rt);
84         }
85         else
86         {
87             _insert(bst[rt].rc,val);
88             if(bst[rt].dat<bst[bst[rt].rc].dat) zag(rt);
89         }
90         pushup(rt);
91     }
92
93     void _delete(int &rt,int val)
94     {
95         if(rt==0) return;
96         if(bst[rt].val==val)
97         {
98             if(bst[rt].cnt>1)
99             {
100                 bst[rt].cnt--;
101                 pushup(rt);
102                 return;
103             }
104             if(bst[rt].rc||bst[rt].lc)
105             {
106                 if(bst[rt].rc==0||bst[bst[rt].rc].dat<bst[bst[rt].lc].dat)
107                     zig(rt),_delete(bst[rt].rc,val);
108                 else
109                     zag(rt),_delete(bst[rt].lc,val);
110                 pushup(rt);
111             }
112             else rt=0;
113         }

```

```

114     return;
115 }
116 if(val<bst[rt].val) _delete(bst[rt].lc,val);
117 else _delete(bst[rt].rc,val);
118 pushup(rt);
119 }
120
121 int getPrev(int val)
122 {
123     int ret=1,rt=root;
124     while(rt)
125     {
126         if(bst[rt].val==val)
127         {
128             if(bst[rt].lc)
129             {
130                 rt=bst[rt].lc;
131                 while(bst[rt].rc) rt=bst[rt].rc;
132                 ret=rt;
133             }
134             break;
135         }
136         if(bst[rt].val<val&&bst[rt].val>bst[ret].val) ret=rt;
137         if(val<bst[rt].val) rt=bst[rt].lc;
138         else rt=bst[rt].rc;
139     }
140     return bst[ret].val;
141 }
142
143 int getNext(int val)
144 {
145     int ret=2,rt=root;
146     while(rt)
147     {
148         if(bst[rt].val==val)
149         {
150             if(bst[rt].rc)
151             {
152                 rt=bst[rt].rc;
153                 while(bst[rt].lc) rt=bst[rt].lc;
154                 ret=rt;
155             }
156             break;
157         }
158         if(bst[rt].val>val&&bst[rt].val<bst[ret].val) ret=rt;
159         if(val<bst[rt].val) rt=bst[rt].lc;
160         else rt=bst[rt].rc;
161     }
162     return bst[ret].val;
163 }
164
165 int getRank(int rt,int val)
166 {
167     if(rt==0) return 0;
168     if(val==bst[rt].val) return bst[bst[rt].lc].siz+1;
169     if(val<bst[rt].val) return getRank(bst[rt].lc,val);
170     else return bst[bst[rt].lc].siz+bst[rt].cnt+getRank(bst[rt].rc,val);
171 }
172
173 int getVal(int rt,int k)

```

```

174 {
175     if(rt==0) return inf;
176     if(bst[bst[rt].lc].siz>=k) return getVal(bst[rt].lc,k);
177     if(bst[bst[rt].lc].siz+bst[rt].cnt>=k) return bst[rt].val;
178     return getVal(bst[rt].rc,k-bst[bst[rt].lc].siz-bst[rt].cnt);
179 }
180 }
181
182 int main()
183 {
184     using namespace Treap;
185     srand(time(0));
186     build();
187     scanf("%d",&n);
188     while(n-->0)
189     {
190         scanf("%d%d",&op,&x);
191         if(op==1) _insert(root,x);
192         else if(op==2) _delete(root,x);
193         else if(op==3) printf("%d\n",getRank(root,x)-1);
194         else if(op==4) printf("%d\n",getVal(root,x+1));
195         else if(op==5) printf("%d\n",getPrev(x));
196         else if(op==6) printf("%d\n",getNext(x));
197     }
198     return 0;
199 }

```

4.5.2 Splay

4.6 动态树

4.7 主席树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  /*
5   离散化+区间k小
6  */
7
8  const int MAXN=200005;
9  int n,m,a[MAXN],tmp[MAXN],org[MAXN],root[MAXN],tot=0;
10 struct tree
11 {
12     int cnt,lc,rc;
13 }seg[30*MAXN];
14
15 int build(int l,int r)
16 {
17     int p=tot++;
18     if(l==r)
19     {
20         seg[p].cnt=0;
21         return p;
22     }
23     int m=l+r>>1;
24     seg[p].lc=build(l,m);
25     seg[p].rc=build(m+1,r);
26     seg[p].cnt=seg[seg[p].lc].cnt+seg[seg[p].rc].cnt;
27     return p;

```

```

28 }
29
30 int modify(int rt,int l,int r,int x)
31 {
32     int p=tot++;
33     seg[p]=seg[rt];
34     if(l==r)
35     {
36         seg[p].cnt++;
37         return p;
38     }
39     int m=l+r>>1;
40     if(x<=m) seg[p].lc=modify(seg[rt].lc,l,m,x);
41     else seg[p].rc=modify(seg[rt].rc,m+1,r,x);
42     seg[p].cnt=seg[seg[p].lc].cnt+seg[seg[p].rc].cnt;
43     return p;
44 }
45
46 int query(int p,int q,int l,int r,int k)
47 {
48     if(l==r) return l;
49     int m=l+r>>1;
50     int lcnt=seg[seg[q].lc].cnt-seg[seg[p].lc].cnt;
51     if(lcnt>=k) return query(seg[p].lc,seg[q].lc,l,m,k);
52     else return query(seg[p].rc,seg[q].rc,m+1,r,k-lcnt);
53 }
54
55 int main()
56 {
57     scanf("%d%d",&n,&m);
58     for(int i=1;i<=n;i++)
59         scanf("%d",a+i),tmp[i]=a[i];
60     sort(tmp+1,tmp+n+1);
61     root[0]=build(1,n);
62     for(int i=1;i<=n;i++)
63     {
64         int k=lower_bound(tmp+1,tmp+n+1,a[i])-tmp;
65         org[k]=a[i];
66         a[i]=k;
67         root[i]=modify(root[i-1],1,n,a[i]);
68     }
69     while(m--)
70     {
71         int x,y,k;
72         scanf("%d%d%d",&x,&y,&k);
73         printf("%d\n",org[query(root[x-1],root[y],1,n,k)]);
74     }
75     return 0;
76 }

```

4.8 树套树

4.8.1 线段树套 Treap

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5     空间 O(nlogn)
6     单点修改,区间rank,前驱后继(不存在则为±2147483647) 单
7     次 O(log^2(n))
8     区间排名为k的值 单次 O(log^3(n))

```

```

8 */
9
10 const int inf=2147483647;
11 const int MAXN=50005;
12 int root[MAXN<<2],n,m,a[MAXN];
13 struct Treap
14 {
15     int tot;
16     struct node
17     {
18         int lc,rc,dat,val,cnt,siz;
19     }bst[MAXN*4*20];
20
21     int newnode(int v)
22     {
23         bst[++tot].val=v;
24         bst[tot].dat=rand();
25         bst[tot].siz=bst[tot].cnt=1;
26         bst[tot].lc=bst[tot].rc=0;
27         return tot;
28     }
29
30     void zig(int &rt)
31     {
32         int p=bst[rt].lc;
33         bst[rt].lc=bst[p].rc;
34         bst[p].rc=rt;
35         rt=p;
36         pushup(bst[rt].rc);
37         pushup(rt);
38     }
39
40     void zag(int &rt)
41     {
42         int p=bst[rt].rc;
43         bst[rt].rc=bst[p].lc;
44         bst[p].lc=rt;
45         rt=p;
46         pushup(bst[rt].lc);
47         pushup(rt);
48     }
49
50     void pushup(int rt)
51     {
52         bst[rt].siz=bst[rt].cnt;
53         if(bst[rt].lc) bst[rt].siz+=bst[bst[rt].lc].siz;
54         if(bst[rt].rc) bst[rt].siz+=bst[bst[rt].rc].siz;
55     }
56
57     int build()
58     {
59         int rt=newnode(-inf);
60         bst[rt].rc=newnode(inf);
61         pushup(rt);
62         return rt;
63     }
64
65     void _delete(int &rt,int x)
66     {
67         if(bst[rt].val==x)
68         {
69             if(bst[rt].cnt>1)
70             {

```

```

71         bst[rt].cnt--;
72         pushup(rt);
73         return;
74     }
75     if(bst[rt].lc||bst[rt].rc)
76     {
77         if(bst[rt].rc==0||bst[bst[rt].rc].dat<
78             bst[bst[rt].lc].dat)
79             zig(rt),_delete(bst[rt].rc,x);
80         else
81             zag(rt),_delete(bst[rt].lc,x);
82         pushup(rt);
83     }
84     else rt=0;
85     return;
86 }
87 if(x<bst[rt].val) _delete(bst[rt].lc,x);
88 else _delete(bst[rt].rc,x);
89 pushup(rt);
90 }
91 void _insert(int &rt,int x)
92 {
93     if(rt==0)
94     {
95         rt=newnode(x);
96         return;
97     }
98     if(bst[rt].val==x) bst[rt].cnt++;
99     else if(x<bst[rt].val)
100     {
101         _insert(bst[rt].lc,x);
102         if(bst[bst[rt].lc].dat>bst[rt].dat) zig(rt)
103             ;
104     }
105     else
106     {
107         _insert(bst[rt].rc,x);
108         if(bst[bst[rt].rc].dat>bst[rt].dat) zag(rt)
109             ;
110     }
111     pushup(rt);
112 }
113 int get_rank(int rt,int x)
114 {
115     if(!rt) return 1;
116     if(bst[rt].val==x) return bst[bst[rt].lc].siz
117         +1;
118     if(x<bst[rt].val) return get_rank(bst[rt].lc,x
119         );
120     else return get_rank(bst[rt].rc,x)+bst[bst[rt]
121         ].lc].siz+bst[rt].cnt;
122 }
123 int get_num(int rt,int x)
124 {
125     if(!rt) return 0;
126     if(bst[rt].val==x) return bst[bst[rt].lc].siz+
127         bst[rt].cnt;
128     if(x<bst[rt].val) return get_num(bst[rt].lc,x)
129         ;
130     else return get_num(bst[rt].rc,x)+bst[bst[rt].
131         lc].siz+bst[rt].cnt;
132 }

```

```

127
128 int get_prev(int rt,int x)
129 {
130     int ret=-inf;
131     while(rt)
132     {
133         if(bst[rt].val==x)
134         {
135             if(bst[rt].lc)
136             {
137                 rt=bst[rt].lc;
138                 while(bst[rt].rc) rt=bst[rt].rc;
139                 ret=bst[rt].val;
140             }
141             break;
142         }
143         if(bst[rt].val<x&&bst[rt].val>ret) ret=bst[
144             rt].val;
145         if(x<bst[rt].val) rt=bst[rt].lc;
146         else rt=bst[rt].rc;
147     }
148     return ret;
149 }
150 int get_nxt(int rt,int x)
151 {
152     int ret=inf;
153     while(rt)
154     {
155         if(bst[rt].val==x)
156         {
157             if(bst[rt].rc)
158             {
159                 rt=bst[rt].rc;
160                 while(bst[rt].lc) rt=bst[rt].lc;
161                 ret=bst[rt].val;
162             }
163             break;
164         }
165         if(bst[rt].val>x&&bst[rt].val<ret) ret=bst[
166             rt].val;
167         if(x<bst[rt].val) rt=bst[rt].lc;
168         else rt=bst[rt].rc;
169     }
170     return ret;
171 }
172 }treap;
173 void build(int rt,int l,int r)
174 {
175     root[rt]=treap.build();
176     if(l==r) return;
177     int m=l+r>>1;
178     build(rt<<1,l,m);
179     build(rt<<1|1,m+1,r);
180 }
181 void modify(int rt,int l,int r,int x,int v,int y)
182 {
183     if(y==-1) treap._delete(root[rt],v);
184     else treap._insert(root[rt],v);
185     if(l==r) return;
186     int m=l+r>>1;
187     if(x<=m) modify(rt<<1,l,m,x,v,y);
188     else modify(rt<<1|1,m+1,r,x,v,y);
189 }

```

```

190 }
191
192 int query(int rt,int l,int r,int op,int L,int R,int x
    )
193 {
194     if(L<=l&&r<=R)
195     {
196         if(op==1) return treap.get_rank(root[rt],x)-2;
197         if(op==2) return treap.get_num(root[rt],x)-1;
198         if(op==4) return treap.get_prev(root[rt],x);
199         if(op==5) return treap.get_nxt(root[rt],x);
200     }
201     int m=l+r>>1,ret;
202     if(op==1||op==2)
203     {
204         ret=0;
205         if(L<=m) ret+=query(rt<<1,l,m,op,L,R,x);
206         if(m<R) ret+=query(rt<<1|m+1,r,op,L,R,x);
207     }
208     if(op==4)
209     {
210         ret=-inf;
211         if(L<=m) ret=max(ret,query(rt<<1,l,m,op,L,R,x)
            );
212         if(m<R) ret=max(ret,query(rt<<1|m+1,r,op,L,R
            ,x));
213     }
214     if(op==5)
215     {
216         ret=inf;
217         if(L<=m) ret=min(ret,query(rt<<1,l,m,op,L,R,x)
            );
218         if(m<R) ret=min(ret,query(rt<<1|m+1,r,op,L,R
            ,x));
219     }
220     return ret;
221 }
222
223 int main()
224 {
225     srand(time(0));
226     scanf("%d%d",&n,&m);
227     build(1,1,n);
228     for(int i=1;i<=n;i++)
229     {
230         scanf("%d",a+i);
231         modify(1,1,n,i,a[i],1);
232     }
233     while(m-->0)
234     {
235         int op,l,r,k,pos;
236         scanf("%d",&op);
237         if(op==1)
238         {
239             scanf("%d%d%d",&l,&r,&k);
240             printf("%d\n",query(1,1,n,op,l,r,k)+1);
241         }
242         else if(op==2)
243         {
244             scanf("%d%d%d",&l,&r,&k);
245             int L=-inf,R=inf,mid;
246             while(L<R)
247             {
248                 mid=(L+R+1)>>1;
249                 if(query(1,1,n,1,l,r,mid)+1>k) R=mid-1;

```

```

        else L=mid;
    }
    printf("%d\n",L);
}
else if(op==3)
{
    scanf("%d%d",&pos,&k);
    modify(1,1,n,pos,a[pos],-1);
    a[pos]=k;
    modify(1,1,n,pos,k,1);
}
else
{
    scanf("%d%d%d",&l,&r,&k);
    printf("%d\n",query(1,1,n,op,l,r,k));
}
}
return 0;
}

```

4.8.2 树状数组套线段树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5 带单点修区间k小
6 用的时候注意下空间 时空 O(nlog^2(n))
7 外层 add(pos,x,y) 空间上为pos的点且值域上为x的点加上y
8 query(l,r,k) 询问区间[l,r]里k小
9 内层 modify 值域线段树动态开点
10 query 值域线段树区间k小
11 VAL 值域大小
12 */
13
14 const int MAXN=200005;
15 int n,a[MAXN],X[MAXN],Y[MAXN],c1,c2,VAL;
16 struct SEG
17 {
18     int root[MAXN],lc[MAXN*500],rc[MAXN*500],cnt[MAXN
        *500],tot;
19     void modify(int &rt,int l,int r,int x,int y)
20     {
21         if(rt==0) rt=++tot;
22         cnt[rt]+=y;
23         if(l==r) return;
24         int m=l+r>>1;
25         if(x<=m) modify(lc[rt],l,m,x,y);
26         else modify(rc[rt],m+1,r,x,y);
27     }
28     int query(int l,int r,int k)
29     {
30         if(l==r) return l;
31         int sum=0,m=l+r>>1;
32         for(int i=0;i<c1;i++) sum+=cnt[lc[X[i]]];
33         for(int i=0;i<c2;i++) sum+=cnt[lc[Y[i]]];
34         if(sum>=k)
35         {
36             for(int i=0;i<c1;i++) X[i]=lc[X[i]];
37             for(int i=0;i<c2;i++) Y[i]=lc[Y[i]];
38             return query(l,m,k);
39         }
40         else
41         {

```

```

42     for(int i=0;i<c1;i++) X[i]=rc[X[i]];
43     for(int i=0;i<c2;i++) Y[i]=rc[Y[i]];
44     return query(m+1,r,k-sum);
45 }
46 }
47 }seg;
48
49 void add(int pos,int x,int y)
50 {
51     for(;pos<=n;pos+=pos&-pos) seg.modify(seg.root[pos]
52     ],1,VAL,x,y);
53 }
54 int query(int l,int r,int k)
55 {
56     c1=c2=0;
57     for(int i=l-1;i;i-=i&-i) X[c1++]=seg.root[i];
58     for(int i=r;i;i-=i&-i) Y[c2++]=seg.root[i];
59     return seg.query(1,VAL,k);
60 }

```

4.9 K-D Tree

4.10 分治

4.10.1 CDQ

4.10.2 点分治

4.10.3 dsu on tree

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 /*
6 统计每颗子树内的出现次数最多的数(们)的和
7 复杂度 O(nlogn)
8 */
9
10 int n,c[100005],cnt[100005],mx,son[100005],siz
11 [100005],hson;
12 ll ans[100005],sum;
13 vector<int> e[100005];
14
15 void dfs1(int now,int fa)
16 {
17     son[now]=0,siz[now]=1;
18     for(auto to:e[now])
19     {
20         if(to==fa) continue;
21         dfs1(to,now);
22         siz[now]+=siz[to];
23         if(siz[to]>siz[son[now]]) son[now]=to;
24     }
25 }
26
27 void cal(int now,int fa,int y)
28 {
29     cnt[c[now]]+=y;
30     if(cnt[c[now]]==mx) sum+=c[now];
31     else if(cnt[c[now]]>mx) mx=cnt[c[now]],sum=c[now];
32     for(auto to:e[now])
33         if(to!=fa&&to!=hson) cal(to,now,y);
34 }

```

```

35 void dfs2(int now,int fa,int keep)
36 {
37     for(auto to:e[now])
38     {
39         if(to==fa||to==son[now]) continue;
40         dfs2(to,now,0);
41     }
42     if(son[now]) dfs2(son[now],now,1);
43     hson=son[now];
44     cal(now,fa,1);
45     hson=0;
46     ans[now]=sum;
47     if(!keep) cal(now,fa,-1),sum=0,mx=0;
48 }
49
50 int main()
51 {
52     scanf("%d",&n);
53     for(int i=1;i<=n;i++) scanf("%d",c+i);
54     for(int i=1,x,y;i<n;i++)
55     {
56         scanf("%d%d",&x,&y);
57         e[x].push_back(y),e[y].push_back(x);
58     }
59     dfs1(1,1);
60     dfs2(1,1,1);
61     for(int i=1;i<=n;i++) printf("%lld ",ans[i]);
62     return 0;
63 }

```

4.10.4 整体二分

4.11 分块

4.11.1 普通分块

```

1 #include <bits/stdc++.h>
2 //luogu P3203
3 using namespace std;
4 const int N=500005;
5
6 int n,m,tot;
7 int a[N],cnt[N],pos[N];
8 int id[N],from[N],to[N];
9 int o,x,y;
10
11 void modify(int i)
12 {
13     if(i+a[i]>n)
14     {
15         pos[i]=i;
16         cnt[i]=0;
17         return;
18     }
19     if(id[i]==id[i+a[i]])
20     {
21         pos[i]=pos[i+a[i]];
22         cnt[i]=cnt[i+a[i]]+1;
23     }
24     else
25     {
26         pos[i]=i+a[i];
27         cnt[i]=1;
28     }
29 }

```

```

30
31 void ask(int x)
32 {
33     int p=x,res=0;
34     while(p!=pos[p])
35         res+=cnt[p],
36         p=pos[p];
37     printf("%d\n",res+1);
38 }
39
40 int main()
41 {
42     scanf("%d",&n);
43     tot=(int)sqrt(n);
44     for(int i=1;i<=tot;++i)
45     {
46         from[i]=(i-1)*tot+1;
47         to[i]=i*tot;
48     }
49     if(to[tot]<n)
50     {
51         tot++;
52         from[tot]=to[tot-1];
53         to[tot]=n;
54     }
55     for(int i=1;i<=tot;++i)
56     {
57         for(int j=from[i];j<=to[i];++j)
58             id[j]=i;
59     }
60     for(int i=1;i<=n;++i)
61         scanf("%d",&a[i]);
62     for(int i=n;i>=1;--i)
63         modify(i);
64     scanf("%d",&m);
65     while(m--)
66     {
67         scanf("%d",&o);
68         if(o==2)
69         {
70             scanf("%d%d",&x,&y);
71             x++;
72             a[x]=y;
73             for(int i=x;i>=from[id[x]];--i)
74                 modify(i);
75         }
76         else if(o==1)
77         {
78             scanf("%d",&x);
79             x++;
80             ask(x);
81         }
82     }
83     return 0;
84 }

```

4.11.2 莫队

4.12 线性基

4.13 珂朵莉树

```

1 #include <bits/stdc++.h>
2 #define int long long //be careful
3 //CF896C

```

```

4 using namespace std;
5
6 /*
7     珂朵莉树的左右split顺序很重要，并且set集合一开始不要为
8     空，否则会RE
9 */
10 const int N=1000005;
11
12 int qpow(int a,int b,int mod)
13 {
14     int res=1,tmp=a%mod;
15     while(b)
16     {
17         if(b&1) res=res*tmp%mod;
18         tmp=tmp*tmp%mod;
19         b>>=1;
20     }
21     return res;
22 }
23
24 struct node
25 {
26     int l,r;
27     mutable int v;
28     node(int L,int R=-1,int V=0):l(L),r(R),v(V){}
29     bool operator < (const node& o)const{return l<o.l;};
30 };
31 set<node> s;
32 typedef set<node>::iterator it;
33
34 it split(int pos)
35 {
36     it i=s.lower_bound(node(pos));
37     if(i!=s.end()&&i->l==pos) return i;
38     --i;
39     int L=i->l,R=i->r,V=i->v;
40     s.erase(i);
41     s.insert(node(L,pos-1,V));
42     return s.insert(node(pos,R,V)).first;
43 }
44
45 void assign(int l,int r,int val)
46 {
47     it ir=split(r+1),il=split(l);
48     s.erase(il,ir);
49     s.insert(node(l,r,val));
50 }
51
52 void add(int l,int r,int val)
53 {
54     it ir=split(r+1),il=split(l);
55     for(;il!=ir;il++)
56         il->v+=val;
57 }
58
59 int rk(int l,int r,int k)
60 {
61     vector<pair<int,int>> v;
62     it ir=split(r+1),il=split(l);
63     for(;il!=ir;il++)
64         v.emplace_back(il->v,il->r-il->l+1);
65     sort(v.begin(),v.end());
66     for(int i=0;i<v.size();++i)

```



```

67     {
68         k-=v[i].second;
69         if(k<=0) return v[i].first;
70     }
71     return -1; //can't find
72 }
73
74 int sum(int l,int r,int ex,int mod)
75 {
76     it ir=split(r+1),il=split(l);
77     int res=0;
78     for(;il!=ir;il++)
79         res=(res+qpow(il->v,ex,mod)*(il->r-il->l+1)%
80             mod)%mod;
81     return res;
82 }
83
84 inline int read(){
85     char ch=getchar();int s=0,w=1;
86     while(ch<48||ch>57){if(ch=='-')w=-1;ch=getchar();}
87     while(ch>=48&&ch<=57){s=(s<<1)+(s<<3)+ch-48;ch=
88         getchar();}
89     return s*w;
90 }
91
92 inline void write(int x){
93     if(x<0)putchar('-'),x=-x;
94     if(x>9)write(x/10);
95     putchar(x%10+48);
96 }
97 //Fast I/O
98
99 int n,m,seed,vmax,a[N];
100 int rnd()
101 {
102     int ret=seed;
103     seed=(seed*7+13)%1000000007;
104     return ret;
105 }
106
107 signed main()
108 {
109     n=read(),m=read(),seed=read(),vmax=read();
110     for(int i=1;i<=n;++i)
111     {
112         a[i]=(rnd()%vmax)+1;
113         s.insert(node(i,i,a[i]));
114     }
115     for(int i=1;i<=m;++i)
116     {
117         int op=(rnd()%4)+1;
118         int l=(rnd()%n)+1;
119         int r=(rnd()%n)+1;
120         if(l>r) swap(l,r);
121         int x,y;
122         if(op==3) x=(rnd()%(r-l+1))+1;
123         else x=(rnd()%vmax)+1;
124         if(op==4) y=(rnd()%vmax)+1;
125         switch(op)
126         {
127             case 1:
128                 add(l,r,x);break;
129             case 2:
130                 assign(l,r,x);break;
131             case 3:
132                 write(rk(l,r,x)),puts("");break;

```

```

130         case 4:
131             write(sum(l,r,x,y)),puts("");break;
132     }
133 }
134 return 0;
135 }
136 }

```

4.14 跳舞链

5 动态规划

5.1 SOS

```

1 for(int i=0;i<(1<<N);i++) dp[i]=a[i];
2 for(int i=0;i<N;i++)
3     for(int mask=0;mask<(1<<N);mask++)
4         if(mask&(1<<i))
5             dp[mask]+=dp[mask^(1<<i)];

```

5.2 动态 DP

5.3 插头 DP

6 数学

6.1 矩阵类

6.2 质数筛

6.2.1 埃筛

6.2.2 线筛

6.3 质数判定

6.3.1 Miller Rabin

6.4 质因数分解

6.4.1 Pollard-Rho

6.5 逆元

6.5.1 EX-GCD 求逆元

6.5.2 线性筛逆元

6.5.3 阶乘逆元

6.6 欧拉函数

6.6.1 欧拉线筛

6.6.2 求单个数的欧拉函数

6.6.3 欧拉降幂

6.6.4 一般积性函数求法

6.7 EX-GCD

6.8 CRT

6.9 N 次剩余

6.10 数论分块

6.11 高斯消元

6.11.1 普通消元

6.11.2 异或方程组消元

6.12 莫比乌斯反演

6.12.1 莫比乌斯函数

6.12.2 杜教筛

6.12.3 洲阁筛

6.12.4 min25 筛

6.13 BSGS

6.14 FFT

6.15 FWT

6.16 NTT

6.17 数值计算

6.17.1 辛普森

6.17.2 自适应辛普森

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const double eps=1e-12;
4
5  /*
6   调用 asr(l,r,simpson(l,r))
7  */
8
9  inline double f(double x)
10 {
11     return x; //被积函数
12 }
13
14 double simpson(double l,double r)
15 {
16     double mid=(l+r)/2;
17     return (f(l)+4*f(mid)+f(r))*(r-l)/6;
18 }
19
20 double asr(double l,double r,double ans)
21 {
22     double mid=(l+r)/2;
23     double l1=simpson(l,mid),r1=simpson(mid,r);
24     if(fabs(l1+r1-ans)<eps) return l1+r1;
25     return asr(l,mid,l1)+asr(mid,r,r1);
26 }
27
28 int main()
29 {
30
31     return 0;
32 }

```

6.18 康拓展开

6.19 卢卡斯定理

7 其他

7.1 快读快写

7.2 约瑟夫环

7.3 悬线法

7.4 蔡勒公式

7.5 三角公式

7.6 海伦公式

7.7 匹克定理

7.8 组合计数

7.8.1 计数原理

7.8.2 卡特兰数

7.8.3 Polya

7.8.4 二项式反演公式

7.8.5 斯特林反演公式

7.8.6 组合数恒等式