ACM Template of Magnolia Blossom

ACM 常用模板算法

玉兰花开

Latest version at Oct.28.2022

ACM Template of Magnolia Blossom

1. 杂项

1.1. 对拍

```
#include<windows.h>
#include<iostream>
using namespace std;
typedef long long LL;
int main()
{
   int cass=0;
   while(true)
   {
      cout<<++cass<<endl;
      system("rand > std.in");
      system("AC < std.in > other.out");
      system("WA < std.in > my.out");
      if(system("fc my.out other.out"))
      {
         system("std.in");
         break;
      }
   }
   return 0;
}
```

1.2. 时间

```
#include<bits/stdc++.h>
using namespace std;
struct gettime
{
   clock_t star,ends;
   void begin()
   {
      star=clock();
   }
   void end()
   {
      ends=clock();
```

```
        cout<<"Running time :"<<(double)(ends-star)/CLOCKS_PER_SEC<<endl;
    }
}tim;
int main()
{
    tim.begin();
    tim.end();
    return 0;
}
```

1.3. 模拟退火(费马点)

```
#include<bits/stdc++.h>
#define T 100
#define delta 0.99
#define esp 1e-8
#define maxt 0x7fffffff
using namespace std;
int n;
bool flag;
double ans,t,sx,sy,tx,ty,sum;
double x[10086],y[10086];
int moved[4][2]={{0,1},{0,-1},{1,0},{-1,0}};
void search()
{
    t=T;
    ans=maxt;
    sx=sy=0;
    while(t>esp)
    {
        flag=true;
        while(flag)
        {
            flag=false;
            for(int i=0;i<=3;++i)
            {
                tx=sx+moved[i][0]*t;
                ty=sy+moved[i][1]*t;
                sum=0;
                for(int j=1;j<=n;++j)
                {
                    sum+=sqrt(pow(tx-x[j],2)+pow(ty-y[j],2));
                }
```

```
            if(ans>sum)
            {
               ans=sum;
               sx=tx;
               sy=ty;
               flag=true;
            }
         }
      }
      t*=delta;
   }
}
int main()
{
   scanf("%d",&n);
   for(int i=1;i<=n;++i)
   {
      scanf("%lf%lf",&x[i],&y[i]);
   }
   search();
   printf("%lf",ans);
}
```

1.4. 线性基(最大异或和)

```
#include<bits/stdc++.h>
#define debug(x) cout<<#x<<" = "<<x<<endl;
#define ll long long
#define maxn 55
using namespace std;
ll n,x,ans,base[maxn];
int main()
{
   n=read();
   for(int i=1;i<=n;++i)
   {
      x=read();
      for(int j=maxn;j>=0;--j)
      {
         if(x&(1LL<<j))
         {
            if(!base[j]){base[j]=x;break;}
            else x^=base[j];
```

```
        }
      }
   }
   for(int i=maxn;i>=0;--i)
       ans=max(ans,ans^base[i]);
   printf("%lld",ans);
   return 0;
}
```

1.5. 悬线法

```
#include<bits/stdc++.h>
#define debug(x) cout<<#x<<" = "<<x<<endl;
#define maxn 2005
using namespace std;
int n,m,len,ans,x,y,a[maxn][maxn];
int l[maxn][maxn],r[maxn][maxn],up[maxn][maxn];
int main()
{
   n=read(),m=read();
   for(int i=1;i<=n;++i)
   for(int j=1;j<=n;++j)
   {
      l[i][j]=r[i][j]=j;
      //将能到达的最左和最右的端点的列数赋为自己本身所在的列数（即还未扩张）
      up[i][j]=a[i][j]=1; //可以向上扩张的列数为 1（即本行）
    }
   for(int i=1;i<=m;++i)
      x=read(),y=read(),a[x][y]=0;
   for(int i=1;i<=n;++i)
   for(int j=2;j<=n;++j)
       if(a[i][j]&&a[i][j-1])
           l[i][j]=l[i][j-1];//求得 i 行 j 列向左扩展最多能扩展到的地方
   for(int i=1;i<=n;++i)
   for(int j=n-1;j>=1;--j)
       if(a[i][j]&&a[i][j+1])
           r[i][j]=r[i][j+1];//求得 i 行 j 列向右扩展最多能扩展到的地方
   for(int i=2;i<=n;++i)
   for(int j=1;j<=n;++j)
   {
      if(a[i][j]&&a[i-1][j])
      //第 i-1 行 j 列元素和第 i 行 j 列元素都符合要求，说明最大矩阵可以更新
      {
```

```
            l[i][j]=max(l[i][j],l[i-1][j]);
            r[i][j]=min(r[i][j],r[i-1][j]);
            // 取该行和上一行分别向扩展的最大距离
            up[i][j]=up[i-1][j]+1;//可以向上扩张的行数（包括本行）
        }
        len=r[i][j]-l[i][j]+1;//可扩张的区间长度
        ans=max(ans,min(len,up[i][j]));
        //由于题目要求最大正方形，在左右扩张长度和向上扩张长度取 min 就 ok 啦
    }
    printf("%d",ans);
    return 0;
}
```

2. 字符串处理

2.1. KMP

```
#include<bits/stdc++.h>
#define MAXN 1000001
using namespace std;
struct KMP
{
    int nxt[MAXN],len;
    char str[MAXN];
    void clear()
    {len=nxt[0]=nxt[1]=0;}
    /*预处理 s 串的 nxt 数组 1-bas s 结尾要加'\0'*/
    void init(char *s)
    {
        len=strlen(s+1);
        memcpy(str,s,(len+2)*sizeof(char));
        for(int i=2;i<=len;i++)
        {
            nxt[i]=nxt[i-1];
            while(nxt[i]&&s[i]!=s[nxt[i]+1]) nxt[i]=nxt[nxt[i]];
            nxt[i]+=(s[i]==s[nxt[i]+1]);
        }
    }
/*求该字符串在 s 串中所有的起始位置，仅返回第一个位置则设置为 true*/
    vector<int>match(char *s,bool isfirst=false)
    {
        int lenth=strlen(s+1);
```

```
        vector<int>pos;
        for(int i=1,j=1;i<=lenth;)
        {
            while(j!=1&&s[i]!=str[j]) j=nxt[j-1]+1;
            if(s[i]==str[j]) i++,j++;
            else i++;
            if(j==len+1)
            {
                pos.push_back(i-j+1);
                if(isfirst) return pos;
                j=nxt[len]+1;
            }
        }
        return pos;
    }
    /*求循环周期 acaca 中 ac 是一个合法周期*/
    vector<int>periodic()
    {
        vector<int>ret;
        int now=len;
        while(now)
        {
            now=nxt[now];
            ret.push_back(len-now);
        }
        return ret;
    }
    /*求循环节 acac 中 ac、acac 是循环节，aca 不是*/
    vector<int>periodic_loop()
    {
        vector<int>ret;
        for(int x:periodic())
            if(len%x==0) ret.push_back(x);
        return ret;
    }
    /*求最小循环节*/
    int min_periodic_loop(){return periodic_loop()[0];}
}kmp;
```

2.2. EX-KMP

```
//nx[i]:x[i...m-1] 与 x[0...m-1] 的最长公共前缀
//ex[i]:y[i...n-1] 与 x[0...m-1] 的最长公共前缀
void get_nx(char* pattern,int len2,long long* nx)
```

```
{
   //p 表示上一次逐个匹配的最右位置(成功匹配的最后一个字符的后一位)
   //a 表示上一次逐个匹配的开始位置
   int a=0,p=0;
   nx[0]=len2;
   for(int i=1;i<len2;i++)
   {
      if(i>=p||i+nx[i-a]>=p)
      {
         if(i>=p) p=i;
         while(p<len2&&pattern[p]==pattern[p-i]) p++;
         nx[i]=p-i;
         a=i;
      }
      else nx[i]=nx[i-a];
   }
}
void get_ex(char* s,int len1,char* t,int len2,long long* nx,long long* ex)
{
   int a=0,p=0;
   get_nx(t,len2,nx);
   for(int i=0;i<len1;i++)
   {
      if(i>=p||i+nx[i-a]>=p)
      {
         if(i>=p) p=i;
         while(p<len1&&p-i<len2&&s[p]==t[p-i]) p++;
         ex[i]=p-i;
         a=i;
      }
      else ex[i]=nx[i-a];
   }
}
```

2.3. HASH 部分知识

令 F[i]=H(Prefix[i])

则有:$H(S[l,r])=F[r]-F[l-1]*Base^{r-l+1}$

模数选择:

数据范围	选取模数	冲突概率
2^{26}~2^{27}	100663319	0.000023
2^{27}~2^{28}	201326611	0.000009
2^{28}~2^{29}	402653189	0.000001

2^{29}~2^{30}	805306457	0.000011
2^{30}~2^{31}	1610612741	0.000000

2.4. Trie 树

```
#include<bits/stdc++.h>
#define MAXN 200001
using namespace std;
struct Trie
{
    int nxt[MAXN][26],ifend[MAXN*26];
    int root=0;//虚结点
    int cnt=0;
    void clear()
    {
        root=cnt=0;
        memset(nxt,0,sizeof(nxt));
        memset(ifend,0,sizeof(ifend));
    }
    int newnode(){return ++cnt;}
    void insert(char *s)
    {
        int now=root;
        while(*s)
        {
            now=insert(now,*s-'a');
            s++;
        }
        ifend[now]=1;
    }
    int insert(int pre,int ch)
    {
       return nxt[pre][ch]? nxt[pre][ch]:nxt[pre][ch]=newnode();
    }
    bool check(char *s)
    {
       int flag=1;
       int now=root;
       while(*s)
       {
          int ch=*s-'a';
          if(nxt[now][ch]) now=nxt[now][ch];
          else
          {
             flag=0;
```

```
            break;
         }
         s++;
      }
      if(flag&&ifend[now]==1) return 1;
      else return 0;
   }
}trie;
```

2.5. 可持久化 Trie 树

```
/*
在给定的 N 个字符串中，问区间[l,r]之间是否出现了 str
*/
#include<bits/stdc++.h>
#define MAXN 200001
using namespace std;
char s[MAXN][101],str[MAXN];
struct Trie
{
    int nxt[MAXN][26],maxx[MAXN*26];
    int root[MAXN];//每时刻的跟结点
    int cnt=0;
    void clear()
    {
        cnt=0;
        memset(nxt,0,sizeof(nxt));
        memset(root,0,sizeof(root));
        memset(maxx,0,sizeof(maxx));
    }
    int newnode(){return ++cnt;}
    void insert(int t,int pos,int pre,int now)
    {//t 表示插入第几个字符串 pos 是当前字符
       if(pos>=strlen(s[t]))//s 数组存储字典
       {
          maxx[now]=t;
          return;
       }
       int ch=s[t][pos]-'a';
       if(pre)
       {
          for(int i=0;i<26;i++) nxt[now][i]=nxt[pre][i];
       }
```

```
        nxt[now][ch]=newnode();
        insert(t,pos+1,nxt[pre][ch],nxt[now][ch]);
        for(int i=0;i<26;i++)
        maxx[now]=max(maxx[nxt[now][i]],maxx[now]);
    }
    bool check(int now,int pos,int limit)
    {//now 填 root[右区间] limit 表示左区间
        if(pos>=strlen(str))//str 存储查询字串
        {
            return 1;
        }
        int ch=str[pos]-'a';
        if(maxx[nxt[now][ch]]>=limit)
        return check(nxt[now][ch],pos+1,limit);
        else return 0;
    }
}trie;
int main()
{
    trie.maxx[0]=-1;
    trie.root[0]=trie.newnode();
    trie.insert(0,0,0,trie.root[0]);
    for(int i=1;i<=5;i++)
    {
        trie.root[i]=trie.newnode();
        scanf("%s",s[i]);
        trie.insert(i,0,trie.root[i-1],trie.root[i]);
    }
    for(int i=1;i<=3;i++)
    {
        int l,r;
        cin>>l>>r;
        scanf("%s",str);
        cout<<trie.check(trie.root[r],0,l)<<endl;
    }
    return 0;
}
```

2.6. 01Trie 树

```
/*
在给定的 N 个整数 A1,A2...An 中选出两个进行 xor 运算,
求得到的结果最大是多少?
```

```
*/
#include<bits/stdc++.h>
#define int long long
#define MAXN 3200001
using namespace std;
struct SegTrie
{
    int nxt[MAXN][2];
    int root=0;//虚结点
    int cnt=0;
    void clear()
   {
       root=cnt=0;
       memset(nxt,0,sizeof(nxt));
    }
    int newnode(){return ++cnt;}
    void insert(int s)
   {
       int now=root;
       for(int i=31;i>=0;i--)
       {
          now=insert(now,(s>>i)&1);
       }
    }
    int insert(int pre,int ch)
   {
      return nxt[pre][ch]? nxt[pre][ch]:nxt[pre][ch]=newnode();
    }
    int check(int s)
    {
      int res=0;
      int now=root;
       for(int i=31;i>=0;i--)
       {
          int ch=(s>>i)&1;
          if(nxt[now][ch^1]) now=nxt[now][ch^1],res+=(1<<i);
          else now=nxt[now][ch];
       }
       return res;
    }
}trie;
int n,a[MAXN];
signed main()
{
```

```
    scanf("%lld",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&a[i]);
        trie.insert(a[i]);
    }
    int maxx=-1;
    for(int i=1;i<=n;i++)
    {
        maxx=max(maxx,trie.check(a[i]));
    }
    printf("%lld\n",maxx);
    return 0;
}
```

2.7. AC 自动机

```
#include<bits/stdc++.h>
#define MAXN 5200001
using namespace std;
struct AC_Automaton
{
    int nxt[MAXN][26],val[MAXN],fail[MAXN],cnt,t;
    int id[MAXN],nod[MAXN],qrr[MAXN];
    void insert(char *s,int x)
    {//x 表示插入的第 x 个字符串
        int len=strlen(s),now=0;
        for(int i=0;i<len;i++)
        {
            int ch=s[i]-'a';
            if(!nxt[now][ch]) nxt[now][ch]=++cnt;
            now=nxt[now][ch];
        }
        val[now]++;//终止节点
        id[x]=now;
    }
    void getfail()
    {
        queue<int>q;
        for(int i=0;i<26;i++)
        if(nxt[0][i])
        {fail[nxt[0][i]]=0,q.push(nxt[0][i]);}
        while(!q.empty())
```

```
    {
        int u=q.front();q.pop();
        qrr[t++]=u;
        for(int i=0;i<26;i++)
        {
            if(nxt[u][i])
            {
                fail[nxt[u][i]]=nxt[fail[u]][i];
                q.push(nxt[u][i]);
            }
            else nxt[u][i]=nxt[fail[u]][i];
        }
    }
}
/*
给你一个文本串 S 和 n 个模式串 T1~n
请你分别求出每个模式串 Ti 在 S 中出现的次数。
*/
void query(char *s,int n)
{//n 表示有多少个字典
    int len=strlen(s);
    int now=0,ans=0;
    for(int i=0;i<len;i++)
    {
        now=nxt[now][s[i]-'a'];
        nod[now]++;
    }
    for(int i=t-1;i>=0;i--)
    {
        nod[fail[qrr[i]]]+=nod[qrr[i]];
    }
    for(int i=1;i<=n;i++)
    {
        printf("%d\n",nod[id[i]]);
    }
}
/*
给你一个文本串 S 和 n 个模式串 T1~n
请你求出模式串 Ti 在 S 中出现的总次数(不重复)。
*/
int query(char *s)
{
    int len=strlen(s);
    int now=0,ans=0;
```

```
        for(int i=0;i<len;i++)
        {
            now=nxt[now][s[i]-'a'];
            for(int t=now;t&&val[t]!=-1;t=fail[t])
           {
               ans+=val[t],val[t]=-1;
           }
        }
        return ans;
    }
}ac;
int n;
char str[MAXN];
int main()
{
   scanf("%d",&n);
   for(int i=1;i<=n;i++)
   {
      scanf("%s",str);
      ac.insert(str,i);
   }
   ac.getfail();
   scanf("%s",str);
   cout<<ac.query(str);
   return 0;
}
```

2.8. Manacher

```
#include<bits/stdc++.h>
#define MAXN 200001
using namespace std;
struct Manacher
{
    int len[MAXN];//len[i]-1 即为 i 所在位置的最长回文
    char ch[MAXN];
    int N;
    /*1-bas*/
    void init(char *s)
    {
        int n=strlen(s+1);
        ch[n*2+1]='#';
        ch[0]='@';
        ch[n*2+2]='\0';
```

```
        for(int i=n;i>=1;i--)
        {
            ch[i*2]=s[i];ch[i*2-1]='#';
        }
        N=2*n+1;
    }
    void manacher()
    {
        len[1]=1;int k=1;//k 最右回文串中心
        for(int i=2;i<=N;i++)
        {
            int p=k+len[k]-1;
            if(i<=p) len[i]=min(len[2*k-i],p-i+1);
            else len[i]=1;
            while(ch[i+len[i]]==ch[i-len[i]]) len[i]++;
            //如果是要相反 则上处==改为！=
            if(i+len[i]>k+len[k]) k=i;//本质不同回文串,可以记录 i
        }
    }
    void debug()
    {
        puts(ch);
        for (int i=1;i<=N;i++)
        {
            printf("len[%d]=%d\n",i,len[i]);
        }
    }
}mana;
char s[MAXN];
int main(){
    scanf("%s",s+1);
    mana.init(s);
    mana.manacher();
    mana.debug();
    return 0;
}
```

2.9. 回文自动机

```
/*
定义 s 的一个子串的存在值为这个子串在 s 中出现的次数乘以这个子串的长度。
对于给定的字符串 ，求所有回文子串中的最大存在值。
*/
#include<bits/stdc++.h>
```

```
#define MAXN 300001
using namespace std;
struct pam
{
   int sz,tot,last;
   int cnt[MAXN],ch[MAXN][26],len[MAXN],fail[MAXN];
   char s[MAXN];
   int node(int l)
   {
      sz++;
      memset(ch[sz],0,sizeof(ch[sz]));
      len[sz]=l;
      fail[sz]=cnt[sz]=0;
      return sz;
   }
   void clear()
   {
      sz=-1;last=0;
      s[tot=0]='$';
      node(0);
      node(-1);
      fail[0]=1;
   }
   int getfail(int x)//找后缀回文
   {
      while(s[tot-len[x]-1]!=s[tot]) x=fail[x];
      return x;
   }
   void insert(char c)
   {
      s[++tot]=c;
      int now=getfail(last);
      if(!ch[now][c-'a'])
      {
         int x=node(len[now]+2);
         fail[x]=ch[getfail(fail[now])][c-'a'];
         ch[now][c-'a']=x;
      }
      last=ch[now][c-'a'];
      cnt[last]++;
   }
   long long solve()
   {
      long long ans=0;
```

```
        for(int i=sz;i>=0;i--)
        {
            cnt[fail[i]]+=cnt[i];//cnt 统计每个回文子串被访问的次数
        }
        for(int i=1;i<=sz;i++)
        {
            ans=max(ans,1ll*len[i]*cnt[i]);
        }
        return ans;
    }
}pam;
char s[MAXN];
int main()
{
    pam.clear();
    scanf("%s",s+1);
    for (int i=1;s[i];i++)
    {
        pam.insert(s[i]);
    }
    printf("%lld\n",pam.solve());
    return 0;
}
```

2.10. 倍增 SA

```
#include<bits/stdc++.h>
#define MAXN 1000001
using namespace std;
struct Suffix
{
    //sa 是后缀排序 ht 是排序后的后缀和前一个的 LCP rk 是 i 后缀的排名
    //sa[rk[i]]=i
    int ht[MAXN],rk[MAXN],sa[MAXN],y[MAXN],c[MAXN];
    int n,m;
    char s[MAXN];
    void insert(char *str){strcpy(s+1,str+1);}
    void init()
    {
        /* 1-bas */
         n=strlen(s+1);
         m=300;//值域
        for(int i=0;i<=m;i++) c[i]=0;
        for(int i=1;i<=n;i++) c[rk[i]=s[i]]++;
        for(int i=1;i<=m;i++) c[i]+=c[i-1];
```

```
   for(int i=n;i>=1;i--) sa[c[rk[i]]--]=i;
   for(int k=1;k<=n;k<<=1)
   {
      int p=0;
      for(int i=n-k+1;i<=n;i++) y[++p]=i;
      for(int i=1;i<=n;i++)
      {
         if(sa[i]>k) y[++p]=sa[i]-k;
      }
      for(int i=0;i<=m;i++) c[i]=0;
      for(int i=1;i<=n;i++) c[rk[i]]++;
      for(int i=1;i<=m;i++) c[i]+=c[i-1];
      for(int i=n;i>=1;i--) sa[c[rk[y[i]]]--]=y[i];
      for(int i=0;i<=n;i++) swap(rk[i],y[i]);
      rk[sa[1]]=p=1;
      for(int i=2;i<=n;i++)
      {
         rk[sa[i]]=(y[sa[i]]==y[sa[i-1]]
         &&y[sa[i]+k]==y[sa[i-1]+k]? p:++p);
      }
      if(p>=n) break;
      m=p;
   }
   for(int i=1,k=0;i<=n;i++)
   {
      if(k) k--;
      int j=sa[rk[i]-1];
      while(s[i+k]==s[j+k]) k++;
      ht[rk[i]]=k;
   }
}
void LCS()//最长公共子串长度
{//输入:两个字符串通过#连接在一起
   int cre,ans=0;
   for(int i=1;i<=n;i++)
   if(s[i]=='#') cre=i;
   for(int i=2;i<=n;i++)
      if((sa[i-1]<cre&&sa[i]>cre)
      ||(sa[i-1]>cre&&sa[i]<cre))
         ans=max(ans,ht[i]);
   printf("%d\n",ans);
}
void CSN()//本质不同公共子串个数
{//输入:两个字符串通过#连接在一起
```

```
        int cre,ans=0,res=0;
        for(int i=1;i<=n;i++)
        if(s[i]=='#') cre=i;
        for(int i=2;i<=n;i++)
        {
           if((sa[i-1]<cre&&sa[i]>cre)
           ||(sa[i-1]>cre&&sa[i]<cre))
           {
              if(ht[i]>res) ans+=ht[i]-res;
              res=ht[i];//重新定义 lcp
           }
           else if(ht[i]<res) res=ht[i];
        }
        printf("%d\n",ans);
    }
    void SN()//本质不同子串的数量
    {
        long long ans=0;
         for(int i=1;i<=n;i++) ans+=(n-sa[i]+1-ht[i]);
        printf("%lld\n",ans);
    }
    void debug()
    {
         printf("%s\n",s+1);
         for(int i=1;i<=n;i++) cout<<sa[i]<<" ";cout<<endl;
         for(int i=1;i<=n;i++) cout<<ht[i]<<" ";cout<<endl;
         for(int i=1;i<=n;i++) cout<<rk[i]<<" ";cout<<endl;
    }
}sa;
char s[MAXN];
int main()
{
    scanf("%s",s+1);
    sa.insert(s);
    sa.init();
    sa.debug();
    return 0;
}
```

2.11. 后缀自动机

```
/*给出一个模式串，和多个匹配串，对于匹配串的每一个位置有一个权值
  问：既是模式串子串也是原串子串的串，所在位置的权值和最大值*/
#include<bits/stdc++.h>
#define int long long
```

```
#define MAXN 1000010
using namespace std;
struct SAM
{
   struct state
   {
      int len,fa;
      map<char,int>nxt;
   }st[MAXN];
   int sz,last;
   void init(){st[0].len=0;st[0].fa=-1;sz++;last=0;}
   /* 0-bas */
   void extend(char c)
   {
      int cur=++sz;
      st[cur].len=st[last].len+1;
      int p=last;
      while(p!=-1&&!st[p].nxt.count(c))
      {
         st[p].nxt[c]=cur;
         p=st[p].fa;
      }
      if(p==-1) st[cur].fa=0;
      else
      {
         int q=st[p].nxt[c];
         if(st[p].len+1==st[q].len) st[cur].fa=q;
         else
         {
            int clone=++sz;
            st[clone].len=st[p].len+1;
            st[clone].nxt=st[q].nxt;
            st[clone].fa=st[q].fa;
            while(p!=-1&&st[p].nxt[c]==q)
            {
               st[p].nxt[c]=clone;
               p=st[p].fa;
            }
            st[q].fa=st[cur].fa=clone;
         }
      }
      last=cur;
   }
}sam;
```

```
int n,m,k,a[MAXN];
char str[MAXN],t[MAXN];
signed main()
{
   scanf("%lld%lld%lld",&n,&m,&k);
   scanf("%s",str+1);
   sam.init();
   for(int i=1;i<=n;i++)
   {
      sam.extend(str[i]);
   }
   for(int i=1;i<=m;i++)
   {
      scanf("%lld",&a[i]);
   }
   for(int i=1;i<=k;i++)
   {
      scanf("%s",t+1);
      int ans=0,temp=0,p=0;
      for(int j=1;j<=m;j++)
      {
         if(temp+a[j]>0&&sam.st[p].nxt[t[j]])
         {
            temp+=a[j];
            p=sam.st[p].nxt[t[j]];
         }
         else
         {
            temp=0;
            p=0;
         }
         ans=max(ans,temp);
      }
      printf("%lld\n",ans);
   }
   return 0;
}
```

3. 数学知识

3.1. Miller-Rabin

```
//在 TIMES*(logn)^3 复杂度内判素数
```

```
#include<bits/stdc++.h>
#define LL long long
using namespace std;
LL Quick_Pow(LL a,LL b,LL p)
{
   LL sum=1;
   for(;b;b>>=1,a=a*a%p) if(b&1) sum=sum*a%p;
   return sum;
}
const int TIMES=10;
mt19937 g(rand());
bool Witness(LL a,LL n)
{
   LL u=n-1,t=0;
   while(u%2==0) t++,u>>=1;
   LL x=Quick_Pow(a,u,n);
   if(x==1) return false;
   for(int i=1;i<=t;i++,x=x*x%n)
      if(x!=n-1&&x!=1&&x*x%n==1) return true;
   return x!=1;
}
bool Miller_Rabin(LL n)
{
   if(n==2) return true;
   if(n<2||!(n&1)) return false;
   srand(time(NULL));
   for(int i=1;i<=TIMES;i++)
      if(Witness(g()%(n-1)+1,n)) return false;
   return true;
}
int main()
{
   for(int i=1;i<=100;i++)
   {
      cout<<i<<' '<<Miller_Rabin(i)<<endl;
   }
}
```

3.2. 博弈 SG 函数

```
#include<bits/stdc++.h>
#define MAXN 1001
using namespace std;
int sg[MAXN],f[MAXN],cnt,m,n,temp,T;
```

```
int SG(int n)
{
   if(sg[n]!=-1) return sg[n];
   if(n==0) return sg[0]=0;
   multiset<int>s;
   for(int i=1;i<=n;i++) {s.insert(SG(n-i));}
   for(int i=0;;i++) {if(s.find(i)==s.end()) break;}
   return sg[n]=i;
}
int main()
{
   memset(sg,-1,sizeof(sg));
   for(int i=1;i<=2;i++) sg[i]=SG(i);
   return 0;
}
```

3.3. XOR 前缀和

```
#include<bits/stdc++.h>//计算从 1 开始到 n 的异或和
using namespace std;
int calcXOR(int n)
{
   if(n%4==0) return n;
   if(n%4==1) return 1;
   if(n%4==2) return n+1;
   else return 0;
}
int main()
{int n;cin>>n;cout<<calcXOR(n);return 0;}
```

3.4. 分解质因数

```
#include<bits/stdc++.h>
#define LL long long
using namespace std;
const int TIMES=10;
LL Quick_Mul(LL a,LL b,LL p)
{
   LL tmp=(a*b-(LL)((long double)a/p*b+1e-8)*p);
   if(tmp<0)return tmp+p;
   else return tmp;
}
LL Quick_Pow(LL a,LL b,LL p)
{
```

```
    LL sum=1;
    for(a%=p; b; b>>=1,a=Quick_Mul(a,a,p))
        if(b&1) sum=Quick_Mul(sum,a,p);
    return sum;
}
mt19937 g(rand());
bool Witness(LL a,LL n)
{
    LL u=n-1,t=0;
    while(u%2==0) t++,u>>=1;
    LL x=Quick_Pow(a,u,n);
    if(x==1) return false;
    for(int i=1;i<=t;i++,x=Quick_Mul(x,x,n))
        if(x!=n-1&&x!=1&&Quick_Mul(x,x,n)==1) return true;
    return x!=1;
}
bool Miller_Rabin(LL n)
{
    if(n==2) return true;
    if(n<2||!(n&1)) return false;
    for(int i=1;i<=TIMES;i++)
        if(Witness(g()%(n-1)+1,n)) return false;
    return true;
}
LL Pollar_Rho(LL n)
{
    if(!(n&1)) return 2;
    while(true)
    {
        LL a=g()%(n-1)+1,b=a,c=g()%(n-1)+1;
        if(c==2) c=3;
        while(true)
        {
            a=(Quick_Mul(a,a,n)+c)%n;
            b=(Quick_Mul(b,b,n)+c)%n;
            b=(Quick_Mul(b,b,n)+c)%n;
            if(a==b) break; //环
            LL d=__gcd(abs(b-a),n);
            if(d>1) return d;
        }
    }
}
LL Find_Fac(LL n)
{//找出最小因子
```

```
    if(Miller_Rabin(n)) return n;
    LL p=Pollar_Rho(n);
    return min(Find_Fac(p),Find_Fac(n/p));
}
int main()
{
    LL x;cin>>x;
    while(x>1)
    {
        LL tmp=Find_Fac(x);
        while(x%tmp==0) x/=tmp;
        cout<<tmp<<endl;
    }
    return 0;
}
```

3.5. 线性逆元

```
inv[1]=1;
for(int i=2;i<=n;++i)
  inv[i]=MOD-(long long)MOD/i*inv[MOD%i]%MOD;
```

3.6. 矩阵 Fibonacci

```
#include<bits/stdc++.h>
#define MAXN 10000001
#define int unsigned long long
using namespace std;
int p;
struct juzhen
{int g[101][101];}f,r;
inline juzhen matrixMultiple(juzhen &a,juzhen &b)
{
    juzhen ans;
    memset(ans.g,0,sizeof(ans.g));
    for(int i=1;i<=2;i++)
        for(int j=1;j<=2;j++)
            if(a.g[i][j])
                for(int k=1;k<=2;k++)
                    ans.g[i][k]+=((a.g[i][j]%p)*(b.g[j][k]%p))%p;
    return ans;
}
void matrix(juzhen &x)
{x.g[1][1]=1;x.g[2][1]=0;x.g[1][2]=0;x.g[2][2]=1;}
```

```
void q_power(int b)
{
   matrix(r);juzhen tmp=f;
   while(b)
   {
      if(b&1){r=matrixMultiple(r,tmp);}
      tmp=matrixMultiple(tmp,tmp);
      b>>=1;
   }
}
int main()
{
   int n;scanf("%d%d",&n,&p);n=n+2;
   f.g[1][1]=1;f.g[1][2]=1;f.g[2][1]=1;f.g[2][2]=0;
   if(n<=1){printf("1");return 0;}
   q_power(n-2);
   printf("%d",(r.g[1][1]+r.g[2][1])%p-1);
   return 0;
}
```

3.7. 快速幂

```
#include<bits/stdc++.h>
using namespace std;
long long f(long long x,long long y,long long z)
{
   long long ans=1%z;
   for(;y;y>>=1) {if(y&1) ans=ans*x%z;x=x*x%z;} return ans;
}
int main()
{int x,y,mod;cin>>x>>y>>mod;cout<<f(x,y,mod);return 0;}
```

3.8. 逆序对

```
#include<bits/stdc++.h>
#define MAXN 500001
using namespace std;
long long a[MAXN],n,m,temp[MAXN],temx[MAXN],ans;
char done[11];
int lowbit(int x){return x&(-x);}
int query(int x)
{
```

```
    long long sum=0;
    for(int i=x;i>=1;i-=lowbit(i))
        sum+=a[i];
    return sum;
}
void update(int x,int y)
{
    for(int i=x;i<=n;i+=lowbit(i)) a[i]+=y;
}
bool cmp(int x,int y)
{
    if(temp[x]==temp[y]) return x>y;
    return temp[x]>temp[y];
}
int main()
{
    scanf("%lld",&n);
    for(int i=1;i<=n;i++) {scanf("%lld",&temp[i]);temx[i]=i;}
    sort(temx+1,temx+n+1,cmp);
    for(int i=1;i<=n;i++)
        {update(temx[i],1);ans+=query(temx[i]-1);}
    printf("%lld\n",ans);
    return 0;
}
```

3.9. 欧拉函数

1~N 之间与 N 互质的数的个数
性质 1、1~n 中与 n 互质的数的和为 n*phi(n)/2
性质 2、若 a,b 互质，则 phi(ab)=phi(a)*phi(b)

```
#include<bits/stdc++.h>
#define MAXN 100001
using namespace std;
int v[MAXN],prime[MAXN],phi[MAXN],m;
void primes(int n)
{
    memset(v,0,sizeof(v));m=0;
    phi[1]=1,phi[2]=2;
    for(int i=2;i<=n;i++)
    {
        if(v[i]==0) {v[i]=i;prime[++m]=i;phi[i]=i-1;}
        for(int j=1;j<=m;j++)
        {
```

```
            if(prime[j]>v[i]||prime[j]>n/i) break;
            v[i*prime[j]]=prime[j];
            if(i%prime[j]==0)
               {phi[i*prime[j]]=phi[i]*prime[j];break;}
            phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
    for(int i=1;i<=n;i++) cout<<i<<":"<<phi[i]<<endl;
}
int main()
{int n;scanf("%d",&n);primes(n);return 0;}
```

3.10. 线性素数筛

```
#include<bits/stdc++.h>
#define MAXN 100001
using namespace std;
int prime[MAXN],n;
void getPrime(int n)
{
    memset(prime,0,sizeof(prime));
    for(int i=2;i<=n;i++)
    {
        if(!prime[i]) prime[++prime[0]]=i;
        for(int j=1;j<=prime[0]&&prime[j]<=n/i;j++)
        {
            prime[prime[j]*i]=1;
            if(i%prime[j]==0) break;
        }
    }
}
int main()
{
    scanf("%d",&n);
    getPrime(n);
    for(int i=1;i<=prime[0];i++)
    {
        printf("%d ",prime[i]);
    }
    return 0;
}
```

3.11. 拓展欧几里德和逆元

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
int exgcd(int a,long long b,int &x,int &y)//解 ax+by=gcd(a,b)
{  if(a==0&&b==0) return -1;//无最大公约数
   if(b==0){x=1;y=0;return a;}
   int d=exgcd(b,a%b,y,x);
   y-=a/b*x;
   return d;
}
int mod_reverse(int a,int n)//解 ax≡1(mod n)
{  int x,y;
   int d=exgcd(a,n,x,y);
   if(d==1) return (x%n+n)%n;
   else return -1;
}
int main()
{
   return 0;
}
```

3.12. 高次同余方程 BSGS

```
//a^x=b(mod n) 求解上式 0<=x<n 的解
#include<bits/stdc++.h>
#define MOD 76543
using namespace std;
int hs[MOD],head[MOD],nxt[MOD],id[MOD],top;
void insert(int x,int y)
{
   int k=x%MOD;
   hs[top]=x,id[top]=y,nxt[top]=head[k],head[k]=top++;
}
int find(int x)
{
   int k=x%MOD;
   for(int i=head[k];i!=-1;i=nxt[i])
      if(hs[i]==x) return id[i];
   return -1;
}
int BSGS(int a,int b,int n)
```

```
{
    memset(head,-1,sizeof(head));
    top=1;
    if(b==1) return 0;
    int m=sqrt(n*1.0),j;
    long long x=1,p=1;
    for(int i=0;i<m;++i,p=p*a%n) insert(p*b%n,i);
    for(long long i=m;;i+=m)
    {
        if((j=find(x=x*p%n))!=-1) return i-j;
        if(i>n)break;
    }
    return -1;
}
```

3.13. 线性递推魔法

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for(int i=a;i<n;i++)
#define per(i,a,n) for(int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
ll n;
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];
    vector<int> Md;
    void mul(ll *a,ll *b,int k)
    {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if(a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for(int i=k+k-1;i>=k;i--) if(_c[i])
```

```
        rep(j,0,SZ(Md))
          _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
    rep(i,0,k) a[i]=_c[i];
}
int solve(ll n,VI a,VI b)
{ //a 系数 b 初值 b[n+1]=a[0]*b[n]+...
    ll ans=0,pnt=0;
    int k=SZ(a);
    assert(SZ(a)==SZ(b));
    rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
    Md.clear();
    rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
    rep(i,0,k) res[i]=base[i]=0;
    res[0]=1;
    while((1ll<<pnt)<=n) pnt++;
    for(int p=pnt;p>=0;p--)
    {
        mul(res,res,k);
        if ((n>>p)&1)
        {
            for(int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
            rep(j,0,SZ(Md))
                res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
        }
    }
    rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
    if (ans<0) ans+=mod;
    return ans;
}
VI BM(VI s)
{
    VI C(1,1),B(1,1);
    int L=0,m=1,b=1;
    rep(n,0,SZ(s))
    {
        ll d=0;
        rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
        if(d==0) ++m;
        else if(2*L<=n)
        {
            VI T=C;
            ll c=mod-d*powmod(b,mod-2)%mod;
            while (SZ(C)<SZ(B)+m) C.pb(0);
            rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
```

```
                L=n+1-L; B=T; b=d; m=1;
            }
            else
            {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n)
    {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};
int main() {
    vector<int>v;
    v.push_back([a1]);
    ...//尽量 push 尽可能多的前几项
    v.push_back([ax]);
    scanf("%lld", &n);
    printf("%lld\n",1LL*linear_seq::gao(v,n-1)%mod);
}
```

4. 数据结构

4.1. 区间改值区间极值&线段树

```
#include<bits/stdc++.h>
#define MAXN 100001
#define INF 0x3f3f3f3f
#define lefts root<<1
#define righs root<<1|1
using namespace std;
long long num[MAXN];
int minx,maxx,sum,n,m,x,y,k,mele;
struct node
```

```
{
   int left,right,lazy;
   long long value,maxx;
}tree[MAXN*4];
inline void pushnow(int root,int changenum)
{
      tree[root].value=(tree[root].right-tree[root].left+1)*changenum;
   tree[root].maxx=changenum;
   tree[root].lazy=changenum;
}
inline void pushup(int root)
{
   tree[root].value=tree[lefts].value+tree[righs].value;
   tree[root].maxx=max(tree[lefts].maxx,tree[righs].maxx);
   return;
}
inline void pushdown(int root)
{
   if(tree[root].lazy)
   {
      pushnow(lefts,tree[root].lazy);
      pushnow(righs,tree[root].lazy);
      tree[root].lazy=0;
   }
}
inline void build(int root,int left,int right)
{
   tree[root].left=left;
   tree[root].right=right;
   if(left==right)
   {
      tree[root].value=num[left];
      tree[root].maxx=num[left];
      tree[root].lazy=0;
      return;
   }
   int mid=(left+right)>>1;
   build(lefts,left,mid);
   build(righs,mid+1,right);
   pushup(root);
   return;
}
inline void update(int root,int left,int right,int changenum)
{
```

```
    if(tree[root].left>=left&&tree[root].right<=right)
    {
        pushnow(root,changenum);
        return;
    }
    int mid=(tree[root].left+tree[root].right)>>1;
    pushdown(root);
    if(left<=mid) update(lefts,left,right,changenum);
    if(right>mid) update(righs,left,right,changenum);
    pushup(root);
}
inline long long query(int root,int left,int right)
{
    if(left<=tree[root].left&&tree[root].right<=right)
    {
        return tree[root].maxx;
    }
    int mid=(tree[root].left+tree[root].right)>>1;
    pushdown(root);
    long long ans=-INF;
    if(left<=mid) ans=max(ans,query(lefts,left,right));
    if(right>mid) ans=max(ans,query(righs,left,right));
    pushup(root);
    return ans;
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&num[i]);
    }
    build(1,1,n);
    for(int i=1;i<=m;i++)
    {
        scanf("%d",&mele);
        if(mele==1)
        {
            scanf("%d%d%d",&x,&y,&k);
            update(1,x,y,k);
        }
        else
        {
            scanf("%d%d",&x,&y);
```

```
            printf("%lld\n",query(1,x,y));
        }
    }
    return 0;
}
```

4.2. 区间加值区间极值&线段树

```
#include<bits/stdc++.h>
#define MAXN 100001
#define lefts root<<1
#define righs root<<1|1
using namespace std;
long long num[MAXN];
int minx,maxx,sum,n,m,x,y,k,mele;
struct node
{
    int left,right,lazy;
    long long value;
}tree[MAXN*4];
inline void pushnow(int root,int addnum)
{
    tree[root].value+=(tree[root].right-tree[root].left+1)*addnum;
    tree[root].lazy+=addnum;
}
inline void pushup(int root)
{
    tree[root].value=tree[lefts].value+tree[righs].value;
    return;
}
inline void pushdown(int root)
{
    if(tree[root].lazy)
    {
        pushnow(lefts,tree[root].lazy);
        pushnow(righs,tree[root].lazy);
        tree[root].lazy=0;
    }
}
inline void build(int root,int left,int right)
{
    tree[root].left=left;
```

```
    tree[root].right=right;
    if(left==right)
    {
        tree[root].value=num[left];
        tree[root].lazy=0;
        return;
    }
    int mid=(left+right)>>1;
    build(lefts,left,mid);
    build(righs,mid+1,right);
    pushup(root);
    return;
}
inline void update(int root,int left,int right,int addnum)
{
    if(tree[root].left>=left&&tree[root].right<=right)
    {
        pushnow(root,addnum);
        return;
    }
    int mid=(tree[root].left+tree[root].right)>>1;
    pushdown(root);
    if(left<=mid) update(lefts,left,right,addnum);
    if(right>mid) update(righs,left,right,addnum);
    pushup(root);
}
inline long long query(int root,int left,int right)
{
    if(left<=tree[root].left&&tree[root].right<=right)
    {
        return tree[root].value;
    }
    int mid=(tree[root].left+tree[root].right)>>1;
    pushdown(root);
    long long ans=0;
    if(left<=mid) ans+=query(lefts,left,right);
    if(right>mid) ans+=query(righs,left,right);
    pushup(root);
    return ans;
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
```

```
    {
        scanf("%lld",&num[i]);
    }
    build(1,1,n);
    for(int i=1;i<=m;i++)
    {
        scanf("%d",&mele);
        if(mele==1)
        {
            scanf("%d%d%d",&x,&y,&k);
            update(1,x,y,k);
        }
        else
        {
            scanf("%d%d",&x,&y);
            printf("%lld\n",query(1,x,y));
        }
    }
    return 0;
}
```

4.3. 动态开点线段树

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=1.5e7;
int n,q,l,r,k,root,cnt;
int sum[maxn],tag[maxn],son[maxn][2];
void build(int &pos)
{
    if(!pos) pos=++cnt,tag[pos]=-1;
}
void pushdown(int l,int r,int pos)
{
    int mid=l+r>>1;
    build(son[pos][0]),build(son[pos][1]);
    int lson=son[pos][0],rson=son[pos][1];
    tag[lson]=tag[rson]=tag[pos];
    sum[lson]=tag[pos]*(mid-l+1);
    sum[rson]=tag[pos]*(r-mid);
    tag[pos]=-1;
}
```

```
void modify(int L,int R,int l,int r,int k,int &pos)
{
   if(l>R||r<L) return;
   build(pos);
   if(L<=l&&r<=R)
   {
      sum[pos]=k*(r-l+1);
      tag[pos]=k;
      return;
   }
   if(tag[pos]!=-1) pushdown(l,r,pos);
   int mid=l+r>>1;
   modify(L,R,l,mid,k,son[pos][0]);
   modify(L,R,mid+1,r,k,son[pos][1]);
   sum[pos]=sum[son[pos][0]]+sum[son[pos][1]];
}
int main()
{
   n=read(),q=read();
   modify(1,n,1,n,1,root);
   while(q--)
   {
      l=read(),r=read(),k=read()-1;
      modify(l,r,1,n,k,root);
      printf("%d\n",sum[root]);
   }
   return 0;
}
```

4.4. 数链剖分

```
#include<bits/stdc++.h>
#define int long long
#define MAXN 100001
using namespace std;
int n,m,r,rt,MOD,cnt,num[MAXN];
int head[MAXN],fa[MAXN],dep[MAXN],son[MAXN],size[MAXN],
top[MAXN],id[MAXN],rk[MAXN];
int x,y,op,k;
struct edge
{int next,to;}e[MAXN<<1];
struct node
{
```

```
       int left,right,value,lazy,righs,lefts;
}tree[MAXN<<2];
inline void insert(int now,int aim)
{
       e[++cnt].to=aim;
       e[cnt].next=head[now];
    head[now]=cnt;
    return;
}
void dfs1(int x)
{
    size[x]=1,dep[x]=dep[fa[x]]+1;
    for(int i=head[x];i;i=e[i].next)
    {
       int temp=e[i].to;
       if(temp!=fa[x])
        {
            fa[temp]=x,dfs1(temp);size[x]+=size[temp];
            if(size[son[x]]<size[temp]) son[x]=temp;
        }
   }
    return;
}
void dfs2(int x,int tp)
{
    top[x]=tp,id[x]=++cnt,rk[cnt]=x;
    if(son[x]) dfs2(son[x],tp);
    for(int i=head[x];i;i=e[i].next)
    {
       int temp=e[i].to;
       if(temp!=fa[x]&&temp!=son[x]) dfs2(temp,temp);
   }
    return;
}
inline void pushnow(int root,int addnum)
{
   (tree[root].value+=(tree[root].right-tree[root].left+1)*addnum)%=MOD;
   (tree[root].lazy+=addnum)%=MOD;
   return;
}
inline void pushup(int root)
{
       tree[root].value=(tree[tree[root].lefts].value+tree[tree[root].rig
hs].value)%MOD;
```

```
    return;
}
inline void pushdown(int root)
{
    if(tree[root].lazy)
    {
        pushnow(tree[root].lefts,tree[root].lazy);
        pushnow(tree[root].righs,tree[root].lazy);
        tree[root].lazy=0;
    }
    return;
}
inline void build(int root,int left,int right)
{
   tree[root].left=left;
   tree[root].right=right;
   if(left==right)
   {
      tree[root].value=num[rk[left]];
      tree[root].lazy=0;
      return;
   }
   int mid=(left+right)>>1;
   tree[root].lefts=cnt++;
   tree[root].righs=cnt++;
   build(tree[root].lefts,left,mid);
   build(tree[root].righs,mid+1,right);
   pushup(root);
   return;
}
inline void update(int root,int left,int right,int addnum)
{
   if(tree[root].left>=left&&tree[root].right<=right)
   {
      pushnow(root,addnum);
      return;
   }
   int mid=(tree[root].left+tree[root].right)>>1;
   pushdown(root);
   if(left<=mid) update(tree[root].lefts,left,right,addnum);
   if(right>mid) update(tree[root].righs,left,right,addnum);
   pushup(root);
}
inline int query(int root,int left,int right)
```

```
{
   if(left<=tree[root].left&&tree[root].right<=right)
   {
      return tree[root].value;
   }
   int mid=(tree[root].left+tree[root].right)>>1;
   pushdown(root);
   int ans=0;
   if(left<=mid) ans+=query(tree[root].lefts,left,right);
   if(right>mid) ans+=query(tree[root].righs,left,right);
   pushup(root);
   return ans%MOD;
}
inline int sum(int x,int y)
{
    int res=0;
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]]) swap(x,y);
        (res+=query(rt,id[top[x]],id[x]))%=MOD;
        x=fa[top[x]];
    }
    if(id[x]>id[y]) swap(x,y);
    return (res+query(rt,id[x],id[y]))%MOD;
}
inline void updates(int x,int y,int addnum)
{
    while(top[x]!=top[y])
    {
        if(dep[top[x]]<dep[top[y]]) swap(x,y);
        update(rt,id[top[x]],id[x],addnum);
        x=fa[top[x]];
    }
    if(id[x]>id[y]) swap(x,y);
    update(rt,id[x],id[y],addnum);
}
signed main()
{
    scanf("%lld%lld%lld%lld",&n,&m,&r,&MOD);//r 是根节点序号
    for(int i=1;i<=n;i++) scanf("%lld",&num[i]);//初值
    for(int i=1;i<n;i++)
    {
        scanf("%lld%lld",&x,&y);//建边
        insert(x,y),insert(y,x);
```

```
    }
    cnt=0,dfs1(r),dfs2(r,r);
    cnt=0,build(rt=cnt++,1,n);
    for(int i=1;i<=m;i++)
    {
        scanf("%lld",&op);
        if(op==1)//将树从 x 到 y 结点最短路径上所有节点的值都加上 z
        {
            scanf("%lld%lld%lld",&x,&y,&k);
            updates(x,y,k);
        }
        else if(op==2)//将树从 x 到 y 结点路径上所有节点的值都加上 z
        {
            scanf("%lld%lld",&x,&y);
            printf("%lld\n",sum(x,y));
        }
        else if(op==3)//将以 x 为根节点的子树内所有节点值都加上 z
        {
            scanf("%lld%lld",&x,&y);
            update(rt,id[x],id[x]+size[x]-1,y);
        }
        else//求以 x 为根节点的子树内所有节点值之和
        {
               scanf("%lld",&x);
               printf("%lld\n",query(rt,id[x],id[x]+size[x]-1))
        }
    }
    return 0;
}
```

4.5. Splay 平衡树

```
#include<bits/stdc++.h>
#define MAXN 100001
using namespace std;
int root,cnt;
struct tree
{
   int value,fa,left,right,num;//相同值的个数
   int leftnum,rightnum;//左右子树结点个数
}tree[MAXN];
void zag(int x)
{
```

```
    int y=tree[x].fa;
    tree[y].right=tree[x].left;
    if(tree[x].left) tree[tree[x].left].fa=y;
    tree[x].fa=tree[y].fa;
    if(tree[y].fa)
    {
        if(y==tree[tree[y].fa].left)
            tree[tree[y].fa].left=x;
        else tree[tree[y].fa].right=x;
    }
    tree[x].left=y;tree[y].fa=x;
    tree[y].rightnum=tree[x].leftnum;
    tree[x].leftnum=tree[y].leftnum+tree[y].rightnum+1;
}
void zig(int x)
{
    int y=tree[x].fa;
    tree[y].left=tree[x].right;
    if(tree[x].right) tree[tree[x].right].fa=y;
    tree[x].fa=tree[y].fa;
    if(tree[y].fa)
    {
        if(y==tree[tree[y].fa].left)
            tree[tree[y].fa].left=x;
        else tree[tree[y].fa].right=x;
    }
    tree[x].right=y;tree[y].fa=x;
    tree[y].leftnum=tree[x].rightnum;
    tree[x].rightnum=tree[y].leftnum+tree[y].rightnum+1;
}
void splay(int x)
{
    int p;
    while(tree[x].fa)
    {
        p=tree[x].fa;
        if(tree[p].fa==0)
        {
            if(x==tree[p].left) zig(x);
            else zag(x);
            break;
        }
        if(x==tree[p].left)
        {
```

```
            if(p==tree[tree[p].fa].left) {zig(p);zig(x);}
            else {zig(x);zag(x);}
        }
        else
        {
            if(p==tree[tree[p].fa].right) {zag(p);zag(x);}
            else {zag(x);zig(x);}
        }
    }
    root=x;
}
int find_exist(int x)
{
    int p=root;
    while(p)
    {
        if(x==tree[p].value) {splay(p);return 1;}
        if(x<tree[p].value) p=tree[p].left;
        else p=tree[p].right;
    }
    return 0;
}
int find_kth(int k)
{
    int p=root;
    if(tree[p].leftnum+tree[p].rightnum+1<k) {return -1;}
    while(1)
    {
        if(tree[p].rightnum+1==k) return tree[p].value;
        if(tree[p].rightnum>=k) p=tree[p].right;
        else {k=k-tree[p].rightnum-1;p=tree[p].left;}
    }
}
void insert(int x)
{
    int p=root,f;
    while(p)
    {
        f=p;
        if(x<=tree[p].value)
            {tree[p].leftnum++;p=tree[p].left;}
        else
            {tree[p].rightnum++;p=tree[p].right;}
    }
```

```
    tree[++cnt].value=x;tree[cnt].fa=0;
    tree[cnt].left=tree[cnt].right=0;
    tree[cnt].leftnum=tree[cnt].rightnum=0;
    if(root==0) {root=cnt;return;}
    tree[cnt].fa=f;
    if(x<=tree[f].value) tree[f].left=cnt;
    else tree[f].right=cnt;
    splay(cnt);
}
void delet(int x)
{
    find_exist(x);
    int p=root,L=tree[p].left,R=tree[p].right;
    if(!L&&!R){root=0;return;}
    if(!L){root=R;tree[R].fa=0;return;}
    if(!R){root=L;tree[L].fa=0;return;}
    p=L;tree[L].fa=0;
    while(tree[p].right) p=tree[p].right;
    splay(p);
    tree[p].right=R;tree[R].fa=p;
    tree[p].rightnum=tree[R].leftnum+tree[R].rightnum+1;
    return;
}
int maximum(int root)
{
    int p=root;
    while(tree[p].right) p=tree[p].right;
    return tree[p].value;
}
int minimum(int root)
{
    int p=root;
    while(tree[p].left) p=tree[p].left;
    return tree[p].value;
}
int pred(int x)//前驱
{
    find_exist(x);
    int p=tree[root].left;
    while(p)
    {
        if(tree[p].right==0) break;
        p=tree[p].right;
    }
```

```
    if(p) return tree[p].value;
    return -1;//没有前驱
}
int succ(int x)//后继
{
    find_exist(x);
    int p=tree[root].right;
    while(p)
    {
        if(tree[p].left==0) break;
        p=tree[p].left;
    }
    if(p) return tree[p].value;
    return -1;//没有后继
}
int main()
{
    int n,temp; scanf("%d",&n);
    for(int i=1;i<=n;i++){scanf("%d",&temp);insert(temp);}
    return 0;
}
```

4.6. 二维树状数组

```
#include<bits/stdc++.h>
#define MAXN 2001
using namespace std;
int n,m,T,a,b,x,y,z,mkp[MAXN][MAXN];
char opt[2];
inline int lowbit(int x) {return x&(-x);}
inline void update(int x,int y,int k)
{
    for(int i=x;i<=n;i+=lowbit(i))
        for(int j=y;j<=m;j+=lowbit(j))
            mkp[i][j]+=k;
}
inline int query(int x,int y)
{
    int ans=0;
    for(int i=x;i>=1;i-=lowbit(i))
        for(int j=y;j>=1;j-=lowbit(j))
            ans+=mkp[i][j];
    return ans;
```

```
}
int main()
{
   scanf("%d%d%d",&n,&m,&T);
   while(T--)
   {
      scanf("%s",opt);
      if(opt[0]=='C')
         {scanf("%d%d%d",&x,&y,&z);update(x,y,z);}
      else
      {
            scanf("%d%d%d%d",&a,&b,&x,&y);
            printf("%d\n",query(x,y)+query(a-1,b-1)-query(x,b-1)-query(a-
            1,y));
      }
   }
   return 0;
}
```

4.7. 并查集路径压缩

```
#include<bits/stdc++.h>
#define MAXN 5001
using namespace std;
int n,m,p,fa[MAXN];
int find(int x)
{
   if(fa[x]==x) return x;
   else return find(fa[x]);
}
int main()
{
   scanf("%d%d%d",&n,&m,&p);
   for(int i=1;i<=n;i++) fa[i]=i;
   for(int i=1;i<=m;i++)
   {
      int a,b;
      scanf("%d%d",&a,&b);
      int r1=find(a),r2=find(b);
      if(r1!=r2) fa[r1]=r2;
   }
   for(int i=1;i<=p;i++)
   {
      int a,b;
```

```
        scanf("%d%d",&a,&b);
        int r1=find(a),r2=find(b);
        if(r1==r2) printf("YES\n");
        else printf("NO\n");
}
    return 0;
}
```

4.8. 主席树(区间第 k 大)

```
#include<bits/stdc++.h>
#define MAXN 100001
using namespace std;
int n,m,cnt;
int root[MAXN],rank[MAXN];
struct node
{
    int L,R,sum;
    node(){sum=0;}
}tree[MAXN*20];
struct Value
{
    int x,id;
}value[MAXN];
bool cmp(Value v1,Value v2) {return v1.x<v2.x;}
void init()
{
    cnt=1;root[0]=0;
    tree[0].L=tree[0].R=tree[0].sum=0;
}
void update(int num,int &rt,int l,int r)
{
    tree[cnt++]=tree[rt];rt=cnt-1;
    tree[rt].sum++;
    if(l==r) return;
    int mid=(l+r)>>1;
    if(num<=mid) update(num,tree[rt].L,l,mid);
    else update(num,tree[rt].R,mid+1,r);
}
int query(int i,int j,int k,int l,int r)
{
    int d=tree[tree[j].L].sum-tree[tree[i].L].sum;
    if(l==r) return l;
```

```
    int mid=(l+r)>>1;
    if(k<=d) return query(tree[i].L,tree[j].L,k,l,mid);
    else return query(tree[i].R,tree[j].R,k-d,mid+1,r);
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&value[i].x);
        value[i].id=i;
    }
    sort(value+1,value+n+1,cmp);
    for(int i=1;i<=n;i++)
    {
        rank[value[i].id]=i;
    }
    init();
    for(int i=1;i<=n;i++)
    {
        root[i]=root[i-1];
        update(rank[i],root[i],1,n);
    }
    int left,right,k;
    for(int i=1;i<=m;i++)
    {
          scanf("%d%d%d",&left,&right,&k);
          printf("%d\n",value[query(root[left-1],root[right],k,1,n)].x);
    }
    return 0;
}
```

4.9. 主席树(t 时刻求值)

```
#include<bits/stdc++.h>
#define maxn 1000005<<5
#define debug(x) cout<<#x<<" = "<<x<<endl;
using namespace std;
int n,m,val,cnt,tim,opt,pos;
int a[maxn],root[maxn],w[maxn],son[maxn][2];
void build(int &now,int l,int r)
{
   now=++cnt;
   if(l==r){w[now]=a[l];return;}
```

```
    int mid=(l+r)>>1;
    build(son[now][0],l,mid);
    build(son[now][1],mid+1,r);
}
void add(int &now,int pre,int l,int r,int pos,int val)
{
    now=++cnt;w[now]=w[pre];
    son[now][0]=son[pre][0],son[now][1]=son[pre][1];
    if(l==r){w[now]=val;return;}
    int mid=(l+r)>>1;
    if(mid>=pos) add(son[now][0],son[pre][0],l,mid,pos,val);
    else add(son[now][1],son[pre][1],mid+1,r,pos,val);
}
int query(int now,int l,int r,int pos)
{
    if(l==r) return w[now];
    int mid=(l+r)>>1;
    if(mid>=pos) return query(son[now][0],l,mid,pos);
    else return query(son[now][1],mid+1,r,pos);
}
int main()
{
    n=read(),m=read();
    for(int i=1;i<=n;++i) a[i]=read();
    build(root[0],1,n);
    for(int i=1;i<=m;++i)
    {
        tim=read(),opt=read(),pos=read();
        switch(opt)
        {
            case 1:{val=read();add(root[i],root[tim],1,n,pos,val);break;}
            case 2:{printf("%d\n",query(root[tim],1,n,pos));
                    root[i]=root[tim];break;}
        }
    }
    return 0;
}
```

4.10. ST 表

```
#include<bits/stdc++.h>
#define maxn 100005
using namespace std;
```

```
int n,m,l,r;
long long maxi[maxn*2][21];
long long query(int l,int r)
{
   int k=log(r-l+1)/log(2);
   return max(maxi[l][k],maxi[r-(1<<k)+1][k]);
}
int main()
{
   scanf("%d%d",&n,&m);
   for(int i=1;i<=n;++i) maxi[i][0]=read();
   for(int i=1;i<=log(n)/log(2)+1;++i)
   {
      for(int j=1;j+(1<<i)-1<=n;++j)
      {
         maxi[j][i]=max(maxi[j][i-1],maxi[j+(1<<(i-1))][i-1]);
      }
   }
   for(int i=1;i<=m;++i)
   {
      l=read();r=read();
      printf("%lld\n",query(l,r));
   }
   return 0;
}
```

4.11. 带修莫队

```
#include<bits/stdc++.h>
#define maxn 60000+5
#define maxw 1000000+5
using namespace std;
char op;
int n,m,x,y,blo,ans,qnum,cnum;
int a[maxn],pos[maxn],cnt[maxw],print[maxn];
struct qwq
{
   int ql,qr,id,pre;
}q[maxn];
struct qvq
{
   int seat,now;
}c[maxn];
```

```
bool cmp(qwq a,qwq b)
{
   if(pos[a.ql]!=pos[b.ql]) return a.ql<b.ql;
   if(pos[a.qr]!=pos[b.qr]) return a.qr<b.qr;
   return a.pre<b.pre;
}
void add(int x){if(++cnt[x]==1)ans++;}
void del(int x){if(--cnt[x]==0)ans--;}
void work(int t,int i)
{
   if(c[t].seat>=q[i].ql&&c[t].seat<=q[i].qr)
   {
      if(--cnt[a[c[t].seat]]==0) ans--;
      if(++cnt[c[t].now]==1) ans++;
   }
   swap(c[t].now,a[c[t].seat]);
}
void solve()
{
   int l=1,r=0,t=0;
   for(int i=1;i<=qnum;++i)
   {
      while(q[i].ql>l)del(a[l++]);
      while(q[i].ql<l)add(a[--l]);
      while(q[i].qr<r)del(a[r--]);
      while(q[i].qr>r)add(a[++r]);
      while(t<q[i].pre) work(++t,i);
      while(t>q[i].pre) work(t--,i);
      print[q[i].id]=ans;
   }
}
int main()
{
   n=read();m=read();
   for(int i=1;i<=n;++i) a[i]=read();
   for(int i=1;i<=m;++i)
   {
      op=getchar();x=read();y=read();
      if(op=='Q')
      {
         q[++qnum].id=qnum;
         q[qnum].pre=cnum;
         q[qnum].ql=x,q[qnum].qr=y;
      }
```

```
        else
        {
            c[++cnum].seat=x;
            c[cnum].now=y;
        }
    }
    blo=pow(n,0.666666);
    for(int i=1;i<=n;++i) pos[i]=i/blo+1;
    sort(q+1,q+qnum+1,cmp);
    solve();
    for(int i=1;i<=qnum;++i)
        printf("%d\n",print[i]);
    return 0;
}
```

4.12. 对顶堆维护动态中位数

```
#include<bits/stdc++.h>
#define LL long long
using namespace std;
int x,ans,tmp,T;
priority_queue<int> q1;
priority_queue<int,vector<int>,greater<int> > q2;
int main()
{
    T=read();
    while(T)
    {
        x=read();
        if(!x)
        {
            T--;
            while(!q1.empty()) q1.pop();
            while(!q2.empty()) q2.pop();
            continue;
        }
        if(x==-1)
        {
            if(q1.size()>=q2.size())
            {
                ans=q1.top();
                q1.pop();
            }
```

```
            else
            {
                ans=q2.top();
                q2.pop();
            }
            printf("%d\n",ans);
            continue;
        }
        if(q1.empty()) q1.push(x);
        else if(!q1.empty()&&x>=q1.top())
        {
            q2.push(x);
            if(q1.size()+1<q2.size())
            {
                tmp=q2.top();
                q2.pop();
                q1.push(tmp);
            }
        }
        else if(!q1.empty()&&x<q1.top())
        {
            q1.push(x);
            if(q1.size()>q2.size()+1)
            {
                tmp=q1.top();
                q1.pop();
                q2.push(tmp);
            }
        }
    }
    return 0;
}
```

4.13. 扫描线

```
#include<bits/stdc++.h>
#define LL unsigned long long
using namespace std;
const int maxn=1e6+10;
LL ans,val[maxn<<2];
int n,w,x1,x2,y1,y2,num,pre,ed;
int pos[maxn<<2],mini[maxn<<2],tag[maxn<<2],ll[maxn<<2],rr[maxn<<2];
struct qwq
```

```
{
   int x1,x2,w,h;
}edge[maxn<<2];
void addedge(int x1,int x2,int w,int h)
{
   edge[++num].x1=x1,edge[num].x2=x2;
   edge[num].w=w,edge[num].h=h;
}

bool cmp(qwq a,qwq b)
{
   return a.h<b.h;
}
int len(int num)
{
   return rr[num]-ll[num]+1;
}
void pushdown(int num)
{
   mini[num<<1]+=tag[num];
   mini[num<<1|1]+=tag[num];
   tag[num<<1]+=tag[num];
   tag[num<<1|1]+=tag[num];
   tag[num]=0;
}
void build(int l,int r,int num)
{
   ll[num]=l,rr[num]=r;
   val[num]=pos[r+1]-pos[l];
   if(l==r) return;
   int mid=l+r>>1;
   build(l,mid,num<<1);
   build(mid+1,r,num<<1|1);
}
void add(int l,int r,int k,int num)
{
   if(l>rr[num]||r<ll[num]) return;
   if(l<=ll[num]&&r>=rr[num])
   {
      mini[num]+=k;
      tag[num]+=k;
      return;
   }
   if(tag[num]) pushdown(num);
```

```
    add(l,r,k,num<<1);
    add(l,r,k,num<<1|1);
    mini[num]=min(mini[num<<1],mini[num<<1|1]);
}
LL query(int l,int r,int num)
{
    if(l>rr[num]||r<ll[num]) return 0;
    if(l<=ll[num]&&r>=rr[num]&&mini[num]) return val[num];
    if(l==r) return 0;
    if(tag[num]) pushdown(num);
    return query(l,r,num<<1)+query(l,r,num<<1|1);
}
int main()
{
    n=read();
    for(int i=1;i<=n;++i)
    {
        x1=read(),y1=read(),x2=read(),y2=read();
        pos[2*i-1]=x1,pos[2*i]=x2;
        addedge(x1,x2,1,y1);
        addedge(x1,x2,-1,y2);
    }
    sort(edge+1,edge+1+2*n,cmp);
    sort(pos+1,pos+1+2*n);
    ed=unique(pos+1,pos+1+2*n)-pos-1;
    pre=edge[1].h;
    build(1,ed-1,1);
    for(int i=1;i<=2*n;++i)
    {
        x1=edge[i].x1,x2=edge[i].x2,w=edge[i].w;
        int pos1=lower_bound(pos+1,pos+1+ed,x1)-pos;
        int pos2=lower_bound(pos+1,pos+1+ed,x2)-pos;
        ans+=1ull*(edge[i].h-pre)*query(1,ed-1,1);
        add(pos1,pos2-1,w,1);
        pre=edge[i].h;
    }
    printf("%lld",ans);
    return 0;
}
```

4.14. 树上莫队

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

```
#define debug(x) cout<<#x<<" = "<<x<<endl;
#define maxn 40010
#define maxm 100010
using namespace std;
int n,m,blo,u,v,num,cnt,ans;
int fa[maxn],sz[maxn],node[maxn],wson[maxn],head[maxn];
int st[maxn],ed[maxn],dep[maxn],quantity[maxn],vis[maxn];
int val[maxn],color[maxn],pos[maxn],print[maxm],fir[maxn];

struct qwq
{
   int to,nxt;
}e[maxn<<1];
struct qvq
{
   int l,r,id,lca;
}q[maxm];
bool cmp(qvq a,qvq b)
{
   if(pos[a.l]==pos[b.l])
       return a.r<b.r;
   return a.l<b.l;
}
void addedge(int u,int v)
{
   e[++num].to=v;
   e[num].nxt=fir[u];
   fir[u]=num;
}
void discretization()//离散化
{
   sort(val+1,val+1+n);
   int end=unique(val+1,val+1+n)-val-1;
   for(int i=1;i<=n;++i)
       color[i]=lower_bound(val+1,val+end+1,color[i])-val;
}
void dfs1(int u,int f)
{
   fa[u]=f,sz[u]=1,st[u]=++cnt;
   dep[u]=dep[f]+1,node[cnt]=u;//node ->欧拉序
   for(int i=fir[u];i;i=e[i].nxt)
   {
      int v=e[i].to;
      if(v==f) continue;
```

```
        dfs1(v,u);
        sz[u]+=sz[v];
        if(sz[v]>sz[wson[u]]) wson[u]=v;;
    }
    ed[u]=++cnt;node[cnt]=u;
}
void dfs2(int u,int top)
{
    head[u]=top;
    if(wson[u]) dfs2(wson[u],top);
    for(int i=fir[u];i;i=e[i].nxt)
    {
        int v=e[i].to;
        if(v!=wson[u]&&v!=fa[u])
            dfs2(v,v);
    }
}
int Lca(int x,int y)
{
    while(head[x]!=head[y])
    {
        if(dep[head[x]]<dep[head[y]]) swap(x,y);
        x=fa[head[x]];
    }
    return dep[x]<dep[y]?x:y;
}
void add(int x){if(++quantity[x]==1) ans++;}
void del(int x){if(--quantity[x]==0) ans--;}
void modify(int x){vis[x]?del(color[x]):add(color[x]);vis[x]^=1;}
void resetq()
{
    for(int i=1;i<=m;++i)
    {
        if(st[q[i].l]>st[q[i].r])
            swap(q[i].l,q[i].r);
        int lca=Lca(q[i].l,q[i].r);
        if(lca==q[i].l) q[i].l=st[q[i].l],q[i].r=st[q[i].r];
        else q[i].l=ed[q[i].l],q[i].r=st[q[i].r],q[i].lca=lca;
    }
    sort(q+1,q+1+m,cmp);
}
void moqueue()
{
    int l=1,r=0;
```

```
    for(int i=1;i<=m;++i)
    {
        while(q[i].l>l) modify(node[l++]);
        while(q[i].l<l) modify(node[--l]);
        while(q[i].r>r) modify(node[++r]);
        while(q[i].r<r) modify(node[r--]);
        if(q[i].lca) modify(q[i].lca);
        print[q[i].id]=ans;
        if(q[i].lca) modify(q[i].lca);
    }
}
int main()
{
    n=read(),m=read();blo=sqrt(n);
    for(int i=1;i<=n;++i) val[i]=color[i]=read();
    discretization();
    for(int i=1;i<=2*n;++i) pos[i]=i/blo+1;
    for(int i=1;i<n;++i)
    {
        u=read(),v=read();
        addedge(u,v),addedge(v,u);
    }
     for(int i=1;i<=m;++i)
        q[i].l=read(),q[i].r=read(),q[i].id=i;
    dfs1(1,0);dfs2(1,1);
    resetq();moqueue();
    for(int i=1;i<=m;++i)
         printf("%d\n",print[i]);
    return 0;
}
```

4.15. 左偏树(可并堆)

```
#include<bits/stdc++.h>
#define maxn 200100
using namespace std;
int n,m,opt,x,y;
int ch[maxn][2],fa[maxn],dis[maxn],val[maxn];
int find(int x)
{
    return fa[x]?find(fa[x]):x;
}
int merge(int x,int y)
```

```
{
   if(!x||!y) return x+y;
   if(val[x]>val[y]||(val[x]==val[y]&&x>y)) swap(x,y);
   ch[x][1]=merge(ch[x][1],y);
   fa[ch[x][1]]=x;
   if(dis[ch[x][0]]<dis[ch[x][1]])
      swap(ch[x][0],ch[x][1]);
   dis[x]=dis[ch[x][1]]+1;
   return x;
}
void del(int x)
{
   val[x]=0;
   fa[ch[x][0]]=fa[ch[x][1]]=0;
   merge(ch[x][0],ch[x][1]);
}
int main()
{
   n=read(),m=read();
   for(int i=1;i<=n;++i) val[i]=read();
   while(m--)
   {
      opt=read();
      switch(opt)
      {
         case 1:
         {
            x=read(),y=read();
            if(val[x]&&val[y]&&find(x)!=find(y))
               merge(find(x),find(y));
            break;
         }
         case 2:
         {
            x=read();
            if(!val[x]) puts("-1");
            else printf("%d\n",val[find(x)]),del(find(x));
            break;
         }
      }
   }
   return 0;
}
```

4.16. dlx

```
#include<bits/stdc++.h>
#define maxn 1000+5
using namespace std;
int n,m,d,num,head;
int U[maxn],D[maxn],L[maxn],R[maxn];
int sz[maxn],h[maxn],col[maxn],row[maxn];
//sz:一列有多少个 h：行头元素 col&&row：当前点所在的行数和列数
void prepare()
{
   num=m;head=0;
   for(int i=0;i<=m;++i)
   {
      U[i]=D[i]=i;sz[i]=0;
      L[i]=i-1;R[i]=i+1;
   }
   L[head]=m;R[m]=head;
   memset(h,-1,sizeof(h));
}
void link(int r,int c)//将 r 行 c 列元素插入双向十字循环链表
{
   num++;sz[c]++;
   col[num]=c;row[num]=r;
   U[num]=U[c],D[num]=c;
   U[D[num]]=D[U[num]]=num;
   if(h[r]<0) h[r]=L[num]=R[num]=num;
    else L[num]=h[r],R[num]=R[h[r]],L[R[h[r]]]=num,R[h[r]]=num;
}
void remove(int c)//删除某一列中出现 1 的所有行
{
   L[R[c]]=L[c],R[L[c]]=R[c];
   for(int i=D[c];i!=c;i=D[i])
      for(int j=R[i];j!=i;j=R[j])
         U[D[j]]=U[j],D[U[j]]=D[j],sz[col[j]]--;
}
void resume(int c)//重置某一列中出现 1 的所有行
{
   for(int i=D[c];i!=c;i=D[i])
       for(int j=R[i];j!=i;j=R[j])
          U[D[j]]=D[U[j]]=j,sz[col[j]]++;
    L[R[c]]=R[L[c]]=c;
}
int dance(int dep)
```

```
{
   if(R[head]==head)return true;
   int c=R[head];
   for(int i=R[head];i!=head;i=R[i])if(sz[i]<sz[c]) c=i;
//选取出现 1 元素次数最少的一列进行删除
   remove(c);
   for(int i=D[c];i!=c;i=D[i])
   {
      for(int j=R[i];j!=i;j=R[j]) remove(col[j]);
      if(dance(dep+1)) return true;
      for(int j=L[i];j!=i;j=L[j]) resume(col[j]);
   }
   resume(c);
   return false;
}
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
       prepare();
       for(int i=1;i<=n;++i)
       {
          for(int j=1;j<=m;++j)
          {
             scanf("%d",&d);
             if(d) link(i,j);
          }
       }
       if(dance(1)) printf("Yes, I found it\n");
       else printf("It is impossible\n");
   }
   return 0;
}
```

4.17. 势能线段树(区间取 min)

```
#include<bits/stdc++.h>
#define LL long long
#define inf 0x7fffffff
#define maxn 1000000+10
using namespace std;
LL sum[maxn<<2];
int n,m,x,y,opt,T,t;
```

```
int a[maxn],ll[maxn<<2],rr[maxn<<2],maxi[maxn<<2];
int seg[maxn<<2],tag[maxn<<2],cnt[maxn<<2];
void pushup(int num)
{
   sum[num]=sum[num<<1]+sum[num<<1|1];
   maxi[num]=max(maxi[num<<1],maxi[num<<1|1]);
   if(maxi[num]==maxi[num<<1]&&maxi[num]==maxi[num<<1|1])
      seg[num]=max(seg[num<<1],seg[num<<1|1])
,cnt[num]=cnt[num<<1]+cnt[num<<1|1];
   else if(maxi[num]==maxi[num<<1])
      seg[num]=max(seg[num<<1],maxi[num<<1|1]),cnt[num]=cnt[num<<1];
   else if(maxi[num]==maxi[num<<1|1])
      seg[num]=max(seg[num<<1|1],maxi[num<<1]),cnt[num]=cnt[num<<1|1];
}
int len(int num)
{
   return rr[num]-ll[num]+1;
}
void pushdown(int num)
{
   tag[num<<1]=min(tag[num<<1],tag[num]);
   tag[num<<1|1]=min(tag[num<<1|1],tag[num]);
   if(tag[num]<maxi[num<<1]&&tag[num]>=seg[num<<1])
      sum[num<<1]+=1ll*(tag[num]-maxi[num<<1])*cnt[num<<1];
   if(tag[num]<maxi[num<<1|1]&&tag[num]>=seg[num<<1|1])
      sum[num<<1|1]+=1ll*(tag[num]-maxi[num<<1|1])*cnt[num<<1|1];
   maxi[num<<1]=min(maxi[num<<1],tag[num]);
   maxi[num<<1|1]=min(maxi[num<<1|1],tag[num]);
   tag[num]=inf;
}
void build(int l,int r,int num)
{
   ll[num]=l,rr[num]=r,tag[num]=inf;
   if(l==r)
   {
      sum[num]=maxi[num]=a[l];
      cnt[num]=1,seg[num]=0;
      return;
   }
   int mid=(l+r)>>1;
   build(l,mid,num<<1);
   build(mid+1,r,num<<1|1);
   pushup(num);
}
```

```
void update(int l,int r,int k,int num)
{
   if(ll[num]>r||rr[num]<l||maxi[num]<=k) return;
   if(ll[num]>=l&&rr[num]<=r&&k>seg[num])
   {
      tag[num]=min(tag[num],k);
      sum[num]+=1ll*(k-maxi[num])*cnt[num];
      maxi[num]=k;
      return;
   }
   if(tag[num]!=inf) pushdown(num);
   update(l,r,k,num<<1);
   update(l,r,k,num<<1|1);
   pushup(num);
}
LL qsum(int l,int r,int num)
{
   if(ll[num]>r||rr[num]<l) return 0;
   if(ll[num]>=l&&rr[num]<=r) return sum[num];
   if(tag[num]!=inf) pushdown(num);
   return qsum(l,r,num<<1)+qsum(l,r,num<<1|1);
}
int qmax(int l,int r,int num)
{
   if(ll[num]>r||rr[num]<l) return 0;
   if(ll[num]>=l&&rr[num]<=r) return maxi[num];
   if(tag[num]!=inf) pushdown(num);
   return max(qmax(l,r,num<<1),qmax(l,r,num<<1|1));
}
int main()
{
   T=read();
   while(T--)
   {
      n=read(),m=read();
      for(int i=1;i<=n;++i) a[i]=read();
      build(1,n,1);
      while(m--)
      {
         opt=read();
         switch(opt)
         {
            case 0:x=read(),y=read(),t=read();update(x,y,t,1);break;
            case 1:x=read(),y=read();printf("%d\n",qmax(x,y,1));break;
```

```
            case 2:x=read(),y=read(),printf("%lld\n",qsum(x,y,1));break;
         }
      }
   }
   return 0;
}
```

4.18. 线段树二分

```
#include<bits/stdc++.h>
using namespace std;
const int N=4e5+10;
int n,b[N],L[N],R[N],c[N];
struct SegmentTree
{
   int Min[N*4],Max[N*4],b1[N*4],b2[N*4];
   int queryMin(int p,int l,int r,int s,int t,int v,int opt)
{
      if(Min[p]>=v) return -1;
      if(l==r) return l;
      if(b1[p]<1e9)
{
         Min[p<<1]=min(Min[p<<1],b1[p]);
Min[p<<1|1]=min(Min[p<<1|1],b1[p]);
         b1[p<<1]=min(b1[p<<1],b1[p]);b1[p<<1|1]=min(b1[p<<1|1],b1[p]);
         b1[p]=1e9;
      }
      int mid=l+r>>1;
      if(t<=mid) return queryMin(p<<1,l,mid,s,t,v,opt);
      else if(s>mid) return queryMin(p<<1|1,mid+1,r,s,t,v,opt);
      else
{
         if(opt==0)
{//求的是 s 到 t 上第一个比 v 小的位置
            int kk=queryMin(p<<1,l,mid,s,mid,v,opt);
            if(kk!=-1) return kk;
            else return queryMin(p<<1|1,mid+1,r,mid+1,t,v,opt);
         }else{//求的是 s 到 t 上最后一个比 v 小的位置
            int kk=queryMin(p<<1|1,mid+1,r,mid+1,t,v,opt);
            if(kk!=-1) return kk;
            else return queryMin(p<<1,l,mid,s,mid,v,opt);
         }
      }
```

```
    }
    int queryMax(int p,int l,int r,int s,int t,int v,int opt)
{
        if(Max[p]<=v) return -1;
        if(l==r) return l;
        if(b2[p])
{
            Max[p<<1]=max(Max[p<<1],b2[p]);
Max[p<<1|1]=max(Max[p<<1|1],b2[p]);
            b2[p<<1]=max(b2[p<<1],b2[p]);b2[p<<1|1]=max(b2[p<<1|1],b2[p]);
            b2[p]=0;
        }
        int mid=l+r>>1;
        if(t<=mid) return queryMax(p<<1,l,mid,s,t,v,opt);
        else if(s>mid) return queryMax(p<<1|1,mid+1,r,s,t,v,opt);
        else
{
            if(opt==0)
{//求的是 s 到 t 上第一个比 v 大的位置
                int kk=queryMax(p<<1,l,mid,s,mid,v,opt);
                if(kk!=-1) return kk;
                else return queryMax(p<<1|1,mid+1,r,mid+1,t,v,opt);
            }else{//求的是 s 到 t 上最后一个比 v 大的位置
                int kk=queryMax(p<<1|1,mid+1,r,mid+1,t,v,opt);
                if(kk!=-1) return kk;
                else return queryMax(p<<1,l,mid,s,mid,v,opt);
            }
        }
    }
    int query(int p,int l,int r,int s,int t,int opt)
{
        if(l>=s&&r<=t)
{
            if(opt==0) return Min[p];
            else return Max[p];
        }
        if(opt==0&&b1[p]<1e9)
{
            Min[p<<1]=min(Min[p<<1],b1[p]);
Min[p<<1|1]=min(Min[p<<1|1],b1[p]);
            b1[p<<1]=min(b1[p<<1],b1[p]);b1[p<<1|1]=min(b1[p<<1|1],b1[p]);
            b1[p]=1e9;
        }else if(opt==1&&b2[p]){
            Max[p<<1]=max(Max[p<<1],b2[p]);
```

```
Max[p<<1|1]=max(Max[p<<1|1],b2[p]);
         b2[p<<1]=max(b2[p<<1],b2[p]);b2[p<<1|1]=max(b2[p<<1|1],b2[p]);
         b2[p]=0;
      }
      int mid=l+r>>1;
      if(opt==0)
{
         int minn=1e9;
         if(s<=mid) minn=min(minn,query(p<<1,l,mid,s,t,opt));
         if(t>mid) minn=min(minn,query(p<<1|1,mid+1,r,s,t,opt));
         return minn;
      }else{
         int maxx=0;
         if(s<=mid) maxx=max(maxx,query(p<<1,l,mid,s,t,opt));
         if(t>mid) maxx=max(maxx,query(p<<1|1,mid+1,r,s,t,opt));
         return maxx;
      }
   }
   void update(int p,int l,int r,int s,int t,int v)
{
      if(l>=s&&r<=t)
{
         Min[p]=min(Min[p],v);
         Max[p]=max(Max[p],v);
         b1[p]=min(b1[p],v);
         b2[p]=max(b2[p],v);
         return ;
      }
      if(b1[p]<1e9&&l!=r)
{
         Min[p<<1]=min(Min[p<<1],b1[p]);
Min[p<<1|1]=min(Min[p<<1|1],b1[p]);
         b1[p<<1]=min(b1[p<<1],b1[p]);b1[p<<1|1]=min(b1[p<<1|1],b1[p]);
         b1[p]=1e9;
      }
      if(b2[p]&&l!=r)
{
         Max[p<<1]=max(Max[p<<1],b2[p]);
Max[p<<1|1]=max(Max[p<<1|1],b2[p]);
         b2[p<<1]=max(b2[p<<1],b2[p]);b2[p<<1|1]=max(b2[p<<1|1],b2[p]);
         b2[p]=0;
      }
      int mid=l+r>>1;
      if(s<=mid) update(p<<1,l,mid,s,t,v);
```

```
        if(t>mid) update(p<<1|1,mid+1,r,s,t,v);
        Min[p]=min(Min[p<<1],Min[p<<1|1]);
        Max[p]=max(Max[p<<1],Max[p<<1|1]);
    }
}tr;
int main()
{
    int t;cin>>t;
    while(t--)
{
        scanf("%d",&n);
        for(int i=1;i<=n;i++)
{
            scanf("%d%d%d",&L[i],&R[i],&c[i]);
            b[i]=L[i];b[i+n]=R[i];
        }
        sort(b+1,b+2*n+1);
        int m=unique(b+1,b+1+2*n)-b-1;
        for(int i=1;i<=4*m;i++) tr.Min[i]=tr.b1[i]=1e9,tr.Max[i]=tr.b2[i]=0;
        for(int i=1;i<=n;i++)
{
            L[i]=lower_bound(b+1,b+1+m,L[i])-b;
            R[i]=lower_bound(b+1,b+1+m,R[i])-b;
            tr.update(1,1,m,L[i],R[i],c[i]);
        }
        for(int i=1;i<=n;i++)
{
            int ans=1e9;
            int minn=tr.query(1,1,m,L[i],R[i],0);
            int maxx=tr.query(1,1,m,L[i],R[i],1);
            if(minn<=c[i]&&c[i]<=maxx&&minn!=maxx)
{
                printf("0 ");
                continue;
            }
            if(L[i]>1)
{
                int lef=0;
                if(tr.queryMin(1,1,m,1,L[i]-1,c[i],1)!=-1)
lef=max(lef,tr.queryMin(1,1,m,1,L[i]-1,c[i],1));
                if(tr.queryMax(1,1,m,1,L[i]-1,c[i],1)!=-1)
lef=max(lef,tr.queryMax(1,1,m,1,L[i]-1,c[i],1));
                if(lef!=0)
                ans=min(ans,b[L[i]]-b[lef]);
```

```
          }
          if(R[i]<m)
{
             int rig=1e9;
             if(tr.queryMin(1,1,m,R[i]+1,m,c[i],0)!=-1)
rig=min(rig,tr.queryMin(1,1,m,R[i]+1,m,c[i],0));
if(tr.queryMax(1,1,m,R[i]+1,m,c[i],0)!=-1)
rig=min(rig,tr.queryMax(1,1,m,R[i]+1,m,c[i],0));
             if(rig!=1e9)
             ans=min(ans,b[rig]-b[R[i]]);
          }
          printf("%d ",ans);
       }
       cout<<endl;
    }
    return 0;
}
```

4.19. dsu

```
#include<bits/stdc++.h>
#define LL long long
using namespace std;
const int MAXN = 1e5 + 10;
int N, col[MAXN], son[MAXN], siz[MAXN], cnt[MAXN], Mx, Son;
LL sum = 0, ans[MAXN];
vector<int> v[MAXN];
void dfs(int x, int fa)
{
    siz[x] = 1;
for(int i = 0; i < v[x].size(); i++)
{
        int to = v[x][i];
        if(to == fa) continue;
        dfs(to, x);
        siz[x] += siz[to];
        if(siz[to] > siz[son[x]]) son[x] = to;//轻重链剖分
    }
}
void add(int x, int fa, int val)
{
    cnt[col[x]] += val;//这里可能会因题目而异
    if(cnt[col[x]] > Mx) Mx = cnt[col[x]], sum = col[x];
```

```
    else if(cnt[col[x]] == Mx) sum += (LL)col[x];
for(int i = 0; i < v[x].size(); i++)
{
        int to = v[x][i];
        if(to == fa || to == Son) continue;
        add(to, x, val);
    }
}
void dfs2(int x, int fa, int opt)
{
    for(int i = 0; i < v[x].size(); i++) {
        int to = v[x][i];
        if(to == fa) continue;
        if(to != son[x]) dfs2(to, x, 0);
//暴力统计轻边的贡献，opt = 0 表示递归完成后消除对该点的影响
    }
if(son[x]) dfs2(son[x], x, 1), Son = son[x];
//统计重儿子的贡献，不消除影响
    add(x, fa, 1); Son = 0;//暴力统计所有轻儿子的贡献
    ans[x] = sum;//更新答案
    if(!opt) add(x, fa, -1), sum = 0, Mx = 0;//如果需要删除贡献的话就删掉
}
int main()
{
    N = read();
    for(int i = 1; i <= N; i++) col[i] = read();
    for(int i = 1; i <= N - 1; i++)
    {
        int x = read(), y = read();
        v[x].push_back(y); v[y].push_back(x);
    }
    dfs(1, 0);
    dfs2(1, 0, 0);
    for(int i = 1; i <= N; i++) printf("%I64d ", ans[i]);
    return 0;
}
```

5. 图论

5.1. 图论 Dijkstra

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

```
#define MAXN 100001
#define INF 2139062143
using namespace std;
int now,aim,wealth,n,m;
int head[MAXN],vis[MAXN],dis[MAXN],cnt,top;
priority_queue<pair<int,int> >q;
struct node
{
   int to,next,cost;
}e[MAXN];
void insert(int now,int aim,int wealth)
{
   e[++cnt].to=aim;e[cnt].next=head[now];
   e[cnt].cost=wealth;head[now]=cnt;
}
void dijkstra()
{
   memset(dis,127,sizeof(dis));
   memset(vis,0,sizeof(vis));
   dis[1]=0;
   q.push(make_pair(0,1));
   while(!q.empty())
   {
      int u=q.top().second;q.pop();
      if(vis[u]) continue; vis[u]=1;
      for(int i=head[u];i;i=e[i].next)
      {
         int temp=e[i].to;
         if(dis[temp]>dis[u]+e[i].cost)
         {
            dis[temp]=dis[u]+e[i].cost;
            q.push(make_pair(-dis[temp],temp));
         }
      }
   }
}
int main()
{
   scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++)
   {
      scanf("%d%d%d",&now,&aim,&wealth);
      insert(now,aim,wealth);
   }
```

```
    dijkstra();
    for(int i=1;i<=n;i++) printf("%d\n",dis[i]);
    return 0;
}
```

5.2. 图论 SPFA

```
#include<bits\stdc++.h>
#define MAXN 100001
using namespace std;
int head[MAXN],vis[MAXN],dis[MAXN];
int cnt,top;
queue<int>q;
struct node
{
    int to,next,cost;
}e[MAXN];
void insert(int now,int aim,int wealth)
{
    e[++cnt].to=aim;e[cnt].next=head[now];
    e[cnt].cost=wealth;head[now]=cnt;
}
void SPFA(int start,int end)
{
    memset(dis,127,sizeof(dis));
    memset(vis,0,sizeof(vis));
    q.push(start);
    vis[start]=1;dis[start]=0;
    while(!q.empty())
    {
        int u=q.front();q.pop();
        vis[u]=0;
        for(int i=head[u];i!=0;i=e[i].next)
        {
            int temp=e[i].to;
            if(dis[temp]>dis[u]+e[i].cost)
            {
                dis[temp]=dis[u]+e[i].cost;
                if(!vis[temp]){q.push(temp);vis[temp]=1;}
            }
        }
    }
}
int main()
```

```
{
   int now,aim,wealth,n,m; scanf("%d%d",&n,&m);
   for(int i=1;i<=m;i++)
   {
      scanf("%d%d%d",&now,&aim,&wealth);
      insert(now,aim,wealth);
   }
   SPFA(1,n);
   for(int i=1;i<=n;i++) printf("%d\n",dis[i]);
   return 0;
}
```

5.3. KM 求最优匹配

```
/*
本题建的负边权求得最优匹配
KM 算法只能求完全匹配
如果边权为正的非完全匹配图 可以加 0 边
*/
#include<bits/stdc++.h>
using namespace std;
const int maxn=100+10;
const int inf=1e9+10;
int n,m,toth,totm,ans;
char s[maxn][maxn];
int w[maxn][maxn],posh[maxn][2],posm[maxn][2];
int valx[maxn],valy[maxn],match[maxn];
bool visx[maxn],visy[maxn];
bool dfs(int u)
{
   visx[u]=true;
   for(int v=1;v<=totm;++v)
   {
      if(visy[v]||valx[u]+valy[v]!=w[u][v]) continue;
      visy[v]=true;
      if(!match[v]||dfs(match[v]))
      {
         match[v]=u;
         return true;
      }
   }
   return false;
}
int main()
```

```
{
   while(~scanf("%d%d",&n,&m))
   {
      if(!n&&!m) break;
      toth=totm=ans=0;
      for(int i=1;i<=n;++i)
      {
         scanf("%s",s[i]+1);
         for(int j=1;j<=m;++j)
         {
            if(s[i][j]=='H')
               posh[++toth][0]=i,posh[toth][1]=j;
            if(s[i][j]=='m')
               posm[++totm][0]=i,posm[totm][1]=j;
         }
      }
      for(int i=1;i<=toth;++i) valx[i]=-inf;
      for(int i=1;i<=totm;++i) valy[i]=match[i]=0;
      for(int i=1;i<=toth;++i)
      for(int j=1;j<=totm;++j)
      {
         w[i][j]=-(abs(posh[i][0]-posm[j][0])
                   +abs(posh[i][1]-posm[j][1]));
         valx[i]=max(w[i][j],valx[i]);
      }
      for(int i=1;i<=toth;++i)
      {
         while(1)
         {
            for(int j=1;j<=toth;++j) visx[j]=false;
            for(int j=1;j<=totm;++j) visy[j]=false;
            if(dfs(i)) break;
            int delta=inf;
            for(int j=1;j<=toth;++j)
            {
               if(!visx[j]) continue;
               for(int k=1;k<=totm;++k)
                  if(!visy[k]) delta=min(delta,valx[j]+valy[k]-w[j][k]);
            }
            if(delta==inf) return -1;
            for(int j=1;j<=toth;++j)
               if(visx[j]) valx[j]-=delta;
            for(int j=1;j<=totm;++j)
               if(visy[j]) valy[j]+=delta;
```

```
            }
        }
        for(int i=1;i<=toth;++i) ans-=valx[i];
        for(int i=1;i<=totm;++i) ans-=valy[i];
        printf("%d\n",ans);
    }
    return 0;
}
```

5.4. kruskal 重构树

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=2e5+10;
const int inf=2e9;
int n,m,q,x,y,num,node,pos;
int fir[maxn],fa[maxn][31],a[maxn],head[maxn],w[maxn]={inf},val[maxn];
struct qvq
{
    int u,v,w;
}edge[maxn];
struct qwq
{
    int to,nxt;
}e[maxn<<1];
void addedge(int u,int v)
{
    e[++num].to=v;
    e[num].nxt=fir[u];
    fir[u]=num;
}
bool cmp(qvq a,qvq b)
{
    return a.w<b.w;
}
int find(int x)
{
    return head[x]=head[x]==x?x:find(head[x]);
}
void dfs(int u,int f)
{
    fa[u][0]=f,val[u]+=a[u];
    for(int i=1;i<=30;++i)
```

```
        fa[u][i]=fa[fa[u][i-1]][i-1];
    for(int i=fir[u];i;i=e[i].nxt)
    {
        int v=e[i].to;
        if(v==f) continue;
        dfs(v,u);
        val[u]+=val[v];
    }
}
int findtop(int u)
{
    int sum=val[u]+y;
    for(int i=30;i>=0;--i)
        if(w[fa[u][i]]<=sum) u=fa[u][i];
    return u;
}
int main()
{
    n=read(),m=read(),q=read();node=n;
    for(int i=1;i<=n;++i) a[i]=read();
    for(int i=1;i<=m;++i)
        edge[i].u=read(),edge[i].v=read(),edge[i].w=read();
    sort(edge+1,edge+1+m,cmp);
    for(int i=1;i<=2*n;++i) head[i]=i;
    for(int i=1;i<=m;++i)
    {
        int u=find(edge[i].u);
        int v=find(edge[i].v);
        if(u==v) continue;
        w[++node]=edge[i].w;
        head[u]=node,head[v]=node;
        addedge(node,u);
        addedge(node,v);
    }
    dfs(node,0);
    while(q--)
    {
        x=read(),y=read();
        while((pos=findtop(x))!=x) x=pos;
        printf("%lld\n",val[x]+y);
    }
    return 0;
}
```

5.5. 边双缩点

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=5e3+10;
const int maxm=1e4+10;
int num,indx,cnt,n,m,u,v;
int fir[maxn],dfn[maxn],low[maxn],color[maxn],deg[maxn];
struct qvq
{
   int u,v;
}edge[maxm<<1];
struct qwq
{
   int to,nxt;
   bool tag;
}e[maxm<<1];
void addedge(int u,int v)
{
   e[num].to=v;
   e[num].nxt=fir[u];
   fir[u]=num++;
}
void tarjan(int u,int f)
{
   low[u]=dfn[u]=++indx;
   for(int i=fir[u];~i;i=e[i].nxt)
   {
      int v=e[i].to;
      if(v==f) continue;
      if(!dfn[v])
      {
         tarjan(v,u);
         low[u]=min(low[u],low[v]);
         if(low[v]>dfn[u])
            e[i].tag=e[i^1].tag=true;
      }
      else low[u]=min(low[u],dfn[v]);
   }
}
void dfs(int u,int p)
{
   color[u]=p;
   for(int i=fir[u];~i;i=e[i].nxt)
```

```
    {
        int v=e[i].to;
        if(color[v]||e[i].tag) continue;
        dfs(v,p);
    }
}
int main()
{
    n=read(),m=read();
    for(int i=1;i<=n;++i) fir[i]=-1;
    for(int i=1;i<=m;++i)
    {
        u=read(),v=read();
        edge[i].u=u,edge[i].v=v;
        addedge(u,v),addedge(v,u);
    }
    for(int i=1;i<=n;++i)
        if(!dfn[i]) tarjan(i,i);
    for(int i=1;i<=n;++i)
        if(!color[i]) dfs(i,i);
    for(int i=1;i<=n;++i)
    {
        u=color[edge[i].u];
        v=color[edge[i].v];
        if(u==v) continue;
        deg[u]++,deg[v]++;
    }
    for(int i=1;i<=n;++i)
        if(deg[i]==1) cnt++;
    printf("%d",(cnt+1)/2);
    return 0;
}
```

5.6. 点双缩点

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=1e3+10;
const int maxm=1e6+10;
const int inf=1e9+10;
int n,m,u,v,num,ans1,indx,root,T;
int fir[maxn],dfn[maxn],low[maxn],color[maxn],cnt[maxn],cntd[maxn];
```

```
bool node[maxn];
ll ans2;
set<int> s;
struct qvq
{
   int u,v;
}edge[maxm<<1];
struct qwq
{
   int to,nxt;
}e[maxm<<1];
void addedge(int u,int v)
{
   e[++num].to=v;
   e[num].nxt=fir[u];
   fir[u]=num;
}
void tarjan(int u,int f)
{
   int child=0;
   dfn[u]=low[u]=++indx;
   for(int i=fir[u];i;i=e[i].nxt)
   {
      int v=e[i].to;
      if(v==f) continue;
      if(!dfn[v])
      {
         child++;
         tarjan(v,u);
         low[u]=min(low[u],low[v]);
         if(low[v]>=dfn[u]&&u!=root) node[u]=true;
      }
      else low[u]=min(low[u],dfn[v]);
   }
   if(u==root&&child>1) node[u]=true;
}
void dfs(int u,int p)
{
   color[u]=p,cnt[p]++;
   for(int i=fir[u];i;i=e[i].nxt)
   {
      int v=e[i].to;
      if(node[v]&&s.find(v)==s.end()) s.insert(v),cntd[p]++;
      if(color[v]||node[v]) continue;
```

```
            dfs(v,p);
        }
    }
    int main()
    {
        while(~scanf("%d",&m))
        {
            if(!m) break;
            T++;
            n=root=1;
            ans1=num=indx=0,ans2=1;
            for(int i=1;i<=m;++i)
            {
                u=read(),v=read();
                edge[i].u=u,edge[i].v=v;
                n=max(n,max(u,v));
                addedge(u,v);
                addedge(v,u);
            }
            for(int i=1;i<=n;++i)
                color[i]=node[i]=cnt[i]=cntd[i]=low[i]=dfn[i]=0;
            tarjan(1,1);
            for(int i=1;i<=n;++i)
            {
                if(!color[i]&&!node[i])
                {
                    s.clear();
                    dfs(i,i);
                    if(cntd[i]==1)
                    {
                        ans1++;
                        ans2*=cnt[i];
                    }
                    if(cntd[i]==0&&cnt[i]>=2)
                    {
                        ans1+=2;
                        ans2*=cnt[i]*(cnt[i]-1)/2;
                    }
                    if(cntd[i]==0&&cnt[i]==1)
                        ans1++;
                }
                if(node[i]) color[i]=i;
            }
            for(int i=1;i<=n;++i) fir[i]=0;
```

```
        printf("Case %d: %d %lld\n",T,ans1,ans2);
    }
    return 0;
}
```

5.7. 二分图博弈

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=1e3+10;
const int maxm=1e6+10;
int n,m,tot,cnt,ans;
char s[maxn][maxn];
int dir[4][2]={{0,1},{0,-1},{1,0},{-1,0}};
int num[maxn][maxn],belong[maxm],pos[maxm][2],fir[maxm],match[maxm];
bool vis[maxm],tag[maxm];
struct qwq
{
    int to,nxt;
}e[maxm<<1];
void addedge(int u,int v)
{
    e[++tot].to=v;
    e[tot].nxt=fir[u];
    fir[u]=tot;
}
bool dfs1(int u)
{
    for(int i=fir[u];i;i=e[i].nxt)
    {
        int v=e[i].to;
        if(vis[v]||v==match[u]) continue;
        vis[v]=true;
        if(!match[v]||dfs1(match[v]))
        {
            match[v]=u;
            match[u]=v;
            return true;
        }
    }
}
void dfs2(int u)
```

```
{
   for(int i=fir[u];i;i=e[i].nxt)
   {
      int v=e[i].to;
      if(vis[v]) continue;
      vis[v]=true;
      tag[match[v]]=true;
      dfs2(match[v]);
   }
}
int main()
{
   n=read(),m=read();
   for(int i=0;i<n;++i)
      scanf("%s",s[i]);
   for(int i=0;i<n;++i)
   for(int j=0;j<m;++j)
   {
      if(s[i][j]=='#') continue;
      num[i][j]=++cnt;
      pos[cnt][0]=i;
      pos[cnt][1]=j;
      if((i+j)&1) belong[cnt]=1;
      else belong[cnt]=2;
   }
   for(int i=0;i<n;++i)
   for(int j=0;j<m;++j)
   for(int k=0;k<4;++k)
   {
      int x=i+dir[k][0];
      int y=j+dir[k][1];
      if(x<0||x>=n||y<0||y>=m) continue;
      if(s[i][j]=='#'||s[x][y]=='#') continue;
      addedge(num[i][j],num[x][y]);
   }
   for(int i=1;i<=cnt;++i)
   {
      if(belong[i]==2) continue;
      for(int j=1;j<=cnt;++j) vis[j]=false;
      if(dfs1(i)) ans++;
   }
   if(ans*2==cnt){puts("LOSE");return 0;}
   puts("WIN");
   for(int i=1;i<=cnt;++i)
```

```
    {
        for(int j=1;j<=cnt;++j) vis[j]=false;
        if(!match[i]) dfs2(i);
    }
    for(int i=1;i<=cnt;++i)
    {
        if(!match[i]||tag[i])
            printf("%d %d\n",pos[i][0]+1,pos[i][1]+1);
    }
    return 0;
}
```

5.8. 二分图匹配

```
#include<bits/stdc++.h>
#define ll long long
#define maxn 1000+10
#define maxm 1000000+10
using namespace std;
bool vis[maxn];
int n,m,s,u,v,num,ans;
int fir[maxn],match[maxn];
struct qwq
{
    int to,nxt;
}e[maxm];
void addedge(int u,int v)
{
    e[++num].to=v;
    e[num].nxt=fir[u];
    fir[u]=num;
}
bool dfs(int u)
{
    for(int i=fir[u];i;i=e[i].nxt)
    {
        int v=e[i].to;
        if(!vis[v])
        {
            vis[v]=true;
            if(!match[v]||dfs(match[v]))
            {
                match[v]=u;
```

```
            return true;
         }
      }
   }
   return false;
}
int main()
{
   n=read(),m=read(),s=read();
   for(int i=1;i<=s;++i)
   {
      u=read(),v=read();
      if(u<=n&&v<=m) addedge(u,v);
   }
   for(int i=1;i<=n;++i)
   {
      memset(vis,false,sizeof(vis));
      if(dfs(i)) ans++;
   }
   printf("%d",ans);
   return 0;
}
```

5.9. tarjan 求割点

```
#include<bits/stdc++.h>
#define ll long long
#define maxn 100000+10
#define maxm 200000+10
#define inf 0x7fffffffff
using namespace std;
int n,m,num,u,v,indx,cnt;
bool vis[maxn],dote[maxn];
int fir[maxn],low[maxn],dfn[maxn];
struct qwq
{
   int to,nxt;
}e[maxm];
void addedge(int u,int v)
{
   e[++num].to=v;e[num].nxt=fir[u];fir[u]=num;
}
void tarjan(int x,int fa)
```

```
{
   int child=0;
   dfn[x]=low[x]=++indx;
   for(int i=fir[x];i;i=e[i].nxt)
   {
      int tx=e[i].to;
      if(!dfn[tx])
      {
         tarjan(tx,fa);
         low[x]=min(low[x],low[tx]);
         if(low[tx]>=dfn[x]&&x!=fa)
             dote[x]=true;
          if(x==fa) child++;
      }
      else if(vis[tx]) low[x]=min(low[x],dfn[tx]);
   }
   if(child>1&&fa==x) dote[x]=true;
}
int main()
{
   n=read();m=read();
   for(int i=1;i<=m;++i)
       u=read(),v=read(),
      addedge(u,v),addedge(v,u);
   for(int i=1;i<=n;++i)
       if(!dfn[i]) tarjan(i,i);
   for(int i=1;i<=n;++i)
       if(dote[i]) cnt++;
   printf("%d\n",cnt);
   for(int i=1;i<=n;++i)
       if(dote[i]) printf("%d ",i);
   return 0;
}
```

5.10. 网络流

```
#include<bits/stdc++.h>
#define ll long long
#define inf 0x7fffffff
#define maxn 10050
#define maxm 100050
using namespace std;
bool vis[maxn];
```

```
int n,m,s,t,u,v,w,num;
ll ans,d;
int fir[maxn],dis[maxn],cur[maxn];
queue<int> q;
struct qwq
{
   int to,nxt;ll res;
}e[maxm<<1];
void addedge(int u,int v,int w)
{
   e[num].to=v;
   e[num].res=w;
   e[num].nxt=fir[u];
   fir[u]=num++;
}
int bfs()
{
   memset(vis,false,sizeof(vis));
   memset(dis,0,sizeof(dis));
   memcpy(cur,fir,sizeof(fir));
   q.push(s),dis[s]=1,vis[s]=true;
   while(!q.empty())
   {
      int u=q.front();q.pop();
      for(int i=fir[u];~i;i=e[i].nxt)
      {
         int v=e[i].to;
         if(vis[v]||!e[i].res) continue;
         dis[v]=dis[u]+1;
         q.push(v),vis[v]=true;
      }
   }
   return dis[t];
}
int dfs(int u,ll flow)
{
   if(u==t) return flow;
   if(dis[u]>=dis[t]) return 0;
   for(int i=cur[u];~i;i=e[i].nxt)
   {
      cur[u]=i;
      int v=e[i].to;
      if(dis[v]==dis[u]+1&&e[i].res)
      {
```

```
            ll add=dfs(v,min(flow,e[i].res));
            if(!add) continue;
            e[i].res-=add;
            e[i^1].res+=add;
            return add;
        }
    }
    return 0;
}
int main()
{
    memset(fir,-1,sizeof(fir));
    n=read(),m=read(),s=read(),t=read();
    for(int i=1;i<=m;++i)
    {
        u=read(),v=read(),w=read();
        addedge(u,v,w),addedge(v,u,0);
    }
    while(bfs())
        while(d=dfs(s,inf)) ans+=d;
    printf("%lld",ans);
    return 0;
}
```

5.11. 最小费用最大流

```
#include<bits/stdc++.h>
#define maxn 20000
#define maxm 200000
using namespace std;
bool vis[maxn];
int fir[maxn],dis[maxn];
int n,m,u,v,w,c,s,t,num,ans,cost;
const int inf=0x7fffffff;
queue<int> q;
struct qwq
{
    int to,nxt,c,w;
}e[maxm<<1];
bool spfa(int s,int t)
{
    memset(vis,false,sizeof(vis));
    for(int i=1;i<=n;++i) dis[i]=inf;
```

```
    dis[t]=0,vis[t]=true,q.push(t);
    while(!q.empty())
    {
        int u=q.front();q.pop();
        for(int i=fir[u];~i;i=e[i].nxt)
        {
            int v=e[i].to;
            if(e[i^1].c&&dis[v]>dis[u]-e[i].w)
            {
                dis[v]=dis[u]-e[i].w;
                if(!vis[v]) q.push(v),vis[v]=true;
            }
        }
        vis[u]=false;
    }
    return dis[s]<inf;
}
int dfs(int u,int low)
{
    if(u==t){vis[t]=true;return low;}
    int used=0,flow;
    vis[u]=true;
    for(int i=fir[u];~i;i=e[i].nxt)
    {
        int v=e[i].to;
        if(!vis[v]&&e[i].c&&dis[u]-e[i].w==dis[v])
        {
            flow=dfs(v,min(e[i].c,low-used));
            if(flow)
            {
                cost+=flow*e[i].w,e[i].c-=flow;
                e[i^1].c+=flow,used+=flow;
            }
        }
    }
    return used;
}
int lowcost()
{
    int flow=0;
    while(spfa(s,t))
    {
        vis[t]=true;
        while(vis[t])
```

```
        {
            memset(vis,false,sizeof(vis));
            flow+=dfs(s,inf);
        }
    }
    return flow;
}
void addedge(int u,int v,int c,int w)
{
    e[num].to=v;
    e[num].c=c;
    e[num].w=w;
    e[num].nxt=fir[u];
    fir[u]=num++;
}
int main()
{
    memset(fir,-1,sizeof(fir));
    n=read(),m=read(),s=read(),t=read();
    for(int i=1;i<=m;++i)
    {
        u=read(),v=read(),c=read(),w=read();
        addedge(u,v,c,w),addedge(v,u,0,-w);
    }
    printf("%d ",lowcost());
    printf("%d",cost);
    return 0;
}
```

5.12. 严格次小生成树

```
#include<bits/stdc++.h>
#define LL long long
using namespace std;
const int maxn=1e5+10;
const int maxm=3e5+10;
const LL inf=1e18;
int n,m,num,indx,maxedge,subedge;
int
fa[maxn],fir[maxn],dep[maxn],sz[maxn],wson[maxn],head[maxn],node[maxn],df
n[maxn],a[maxn];
int ll[maxn<<2],rr[maxn<<2],maxi[maxn<<2],sub[maxn<<2];
LL mini,ans=inf;
```

```
struct qwq
{
   int to,nxt,val;
}e[maxn<<1];
struct qvq
{
   int u,v,w;
   bool tag;
}edge[maxm];
bool cmp(qvq x,qvq y)
{
   return x.w<y.w;
}
void addedge(int u,int v,int w)
{
   e[++num].to=v;
   e[num].val=w;
   e[num].nxt=fir[u];
   fir[u]=num;
}
int find(int x)
{
   return fa[x]=fa[x]==x?x:find(fa[x]);
}
void dfs1(int u,int f)
{
   sz[u]=1,dep[u]=dep[f]+1,fa[u]=f;
   for(int i=fir[u];i;i=e[i].nxt)
   {
      int v=e[i].to;
      if(v==f) continue;
      dfs1(v,u);
      a[v]=e[i].val;
      sz[u]+=sz[v];
      if(sz[wson[u]]<sz[v]) wson[u]=v;
   }
}
void dfs2(int u,int top)
{

   dfn[u]=++indx,node[indx]=u,head[u]=top;
   if(wson[u]) dfs2(wson[u],top);
   for(int i=fir[u];i;i=e[i].nxt)
   {
```

```
        int v=e[i].to;
        if(v==fa[u]||v==wson[u]) continue;
        dfs2(v,v);
    }
}
void build(int l,int r,int num)
{
    ll[num]=l,rr[num]=r;
    if(l==r)
    {
        maxi[num]=a[node[l]];
        sub[num]=-1;
        return;
    }
    int mid=(l+r)>>1;
    build(l,mid,num<<1);
    build(mid+1,r,num<<1|1);
    maxi[num]=max(maxi[num<<1],maxi[num<<1|1]);
    if(maxi[num]==maxi[num<<1]&&maxi[num]==maxi[num<<1|1])
        sub[num]=max(sub[num<<1],sub[num<<1|1]);
    else if(maxi[num]==maxi[num<<1])
        sub[num]=max(sub[num<<1],maxi[num<<1|1]);
    else if(maxi[num]==maxi[num<<1|1])
        sub[num]=max(maxi[num<<1],sub[num<<1|1]);
}
void query(int l,int r,int num,int &maxedge,int &subedge)
{
    if(ll[num]>r||rr[num]<l) return;
    if(l<=ll[num]&&rr[num]<=r)
    {
        maxedge=max(maxedge,maxi[num]);
        if(maxedge==maxi[num])
            subedge=max(subedge,sub[num]);
        else subedge=max(subedge,maxi[num]);
        return;
    }
    query(l,r,num<<1,maxedge,subedge);
    query(l,r,num<<1|1,maxedge,subedge);
}
void jump(int u,int v,int &maxedge,int &subedge)
{
    while(head[u]!=head[v])
    {
        if(dep[head[u]]<dep[head[v]]) swap(u,v);
```

```
        query(dfn[head[u]],dfn[u],1,maxedge,subedge);
        u=fa[head[u]];
    }
    if(dfn[u]>dfn[v]) swap(u,v);
    if(u!=v) query(dfn[u]+1,dfn[v],1,maxedge,subedge);
}
int main()
{
    n=read(),m=read();
    for(int i=1;i<=n;++i) fa[i]=i;
    for(int i=1;i<=m;++i)
    edge[i].u=read(),edge[i].v=read(),edge[i].w=read(),edge[i].tag=false;
    sort(edge+1,edge+1+m,cmp);
    for(int i=1;i<=m;++i)
    {
        int u=find(edge[i].u),v=find(edge[i].v);
        if(u==v) continue;
        mini+=edge[i].w;
        fa[u]=v,edge[i].tag=true;
        addedge(edge[i].u,edge[i].v,edge[i].w);
        addedge(edge[i].v,edge[i].u,edge[i].w);
    }
    dfs1(1,0);dfs2(1,1);
    build(1,n,1);
    for(int i=1;i<=m;++i)
    {
        if(edge[i].tag) continue;
        maxedge=subedge=-1;
        jump(edge[i].u,edge[i].v,maxedge,subedge);
        if(maxedge==edge[i].w&&subedge!=-1)
            ans=min(ans,mini-subedge+edge[i].w);
        else if(maxedge!=edge[i].w)
            ans=min(ans,mini-maxedge+edge[i].w);
    }
    printf("%lld",ans);
    return 0;
}
```

5.13. 最大独立集

```
/*
对于左边每个未匹配点跑增广路打 tag
左边没 tag 和右边有 tag 的构成最小点覆盖
```

```
左边有 tag 和右边没 tag 的构成最大独立集
*/
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=5e3+10;
int n,ans,num;
ll a[maxn];
int fir[maxn],match[maxn];
bool vis[maxn],flag[maxn];
struct qwq
{
   int to,nxt;
}e[maxn*maxn];
void addedge(int u,int v)
{
   e[++num].to=v;
   e[num].nxt=fir[u];
   fir[u]=num;
}
int count(ll x)
{
   int ans=0;
   while(x)
   {
      if(x&1) ans++;
      x>>=1;
   }
   return ans;
}
bool dfs1(int u)
{
   for(int i=fir[u];i;i=e[i].nxt)
   {
      int v=e[i].to;
      if(!vis[v])
      {
         vis[v]=true;
         if(!match[v]||dfs1(match[v]))
         {
            match[v]=u;
            return true;
         }
      }
```

```
    }
    return false;
}
void dfs2(int u)
{
    vis[u]=true;
    for(int i=fir[u];i;i=e[i].nxt)
    {
        int v=e[i].to;
        if(vis[v]||match[v]==u||!match[v]) continue;
        vis[v]=true;
        if(match[v]) dfs2(match[v]);
    }
}

int main()
{
    n=read();
    for(int i=1;i<=n;++i) a[i]=read();
    for(int i=1;i<=n;++i)
    for(int j=i+1;j<=n;++j)
    {
        if(count(a[i]^a[j])!=1) continue;
        if((count(a[i])&1)==1) addedge(i,j);
        else addedge(j,i);
    }
    for(int i=1;i<=n;++i)
    {
        if((count(a[i])&1)==0) continue;
        for(int j=1;j<=n;++j)
            vis[j]=false;
        if(dfs1(i)) ans++,flag[i]=true;
    }
    for(int i=1;i<=n;++i) vis[i]=false;
    for(int i=1;i<=n;++i)
        if(((count(a[i])&1)==1)&&!flag[i]) dfs2(i);
    printf("%d\n",n-ans);
    for(int i=1;i<=n;++i)
    {
        if(((count(a[i])&1)==1)&&vis[i]) printf("%d ",a[i]);
        if(((count(a[i])&1)==0)&&!vis[i]) printf("%d ",a[i]);
    }
    return 0;
}
```

5.14. 最小路径覆盖

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=300+10;
const int maxm=6e3+10;
const int inf=1e9;
int n,m,u,v,num,s,t,d,ans;
bool vis[maxn];
int dis[maxn],cur[maxn],fir[maxn],match[maxn],deg[maxn];
queue<int> q;
struct qwq
{
   int to,nxt,res;
}e[maxm<<1];
void addedge(int u,int v,int w)
{
   e[num].to=v;
   e[num].res=w;
   e[num].nxt=fir[u];
   fir[u]=num++;
}
bool bfs()
{
   for(int i=s;i<=t;++i)
      dis[i]=0,vis[i]=false,cur[i]=fir[i];
   dis[s]=1;
   vis[s]=true;
   q.push(s);
   while(!q.empty())
   {
      int u=q.front();q.pop();
      for(int i=fir[u];~i;i=e[i].nxt)
      {
         int v=e[i].to;
         if(vis[v]||!e[i].res) continue;
         dis[v]=dis[u]+1;
         vis[v]=true;
         q.push(v);
      }
   }
   return vis[t];
}
int dfs(int u,int flow)
```

```
{
   if(u==t) return flow;
   if(dis[u]>=dis[t]) return 0;
   for(int i=cur[u];~i;i=e[i].nxt)
   {
      cur[u]=i;
      int v=e[i].to;
      if(dis[v]!=dis[u]+1||!e[i].res) continue;
      int add=dfs(v,min(flow,e[i].res));
      e[i].res-=add,e[i^1].res+=add;
      if(add) return add;
   }
   return 0;
}
int main()
{
   n=read(),m=read();
   s=0,t=2*n+1;
   for(int i=s;i<=t;++i) fir[i]=-1;
   for(int i=1;i<=m;++i)
   {
      u=read(),v=read();
      addedge(u,v+n,1);
      addedge(v+n,u,0);
   }
   for(int i=1;i<=n;++i)
   {
      addedge(s,i,1),addedge(i,s,0);
      addedge(i+n,t,1),addedge(t,i+n,0);
   }
   while(bfs())
      while(d=dfs(s,inf)) ans+=d;
   for(int i=1;i<=n;++i)
   {
      vis[i]=false;
      for(int j=fir[i];~j;j=e[j].nxt)
      {
         if(!e[j].res&&(j&1)==0)
         {
            match[i]=e[j].to-n;
            deg[match[i]]++;
            vis[i]=true;
         }
      }
```

```
    }
    for(int i=1;i<=n;++i)
    {
        if(vis[i]&&!deg[i])
        {
            u=i;
            printf("%d ",u);
            while(match[u])
                u=match[u],printf("%d ",u);
            puts("");
        }
    }
    printf("%d",n-ans);
    return 0;
}
```

6. 动态规划

6.1. 01 背包

```
#include<bits/stdc++.h>
#define MAXN 100001
using namespace std;
long long w[MAXN],c[MAXN];
long long dp[MAXN];
long long n,m;
int main()
{
    scanf("%d%d",&m,&n);
    for(int i=1;i<=n;i++) scanf("%d%d",&w[i],&c[i]);
    for(int i=1;i<=n;i++)
        for(int j=m;j>=w[i];j--)
            dp[j]=max(dp[j],dp[j-w[i]]+c[i]);
    printf("%d",dp[m]);
    return 0;
}
```

6.2. 背包容斥

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

```
#define ll long long
using namespace std;
const int maxs=1e5+10;
int n,s,p,c[5],d[5];
ll ans,f[maxs];
int main()
{
   for(int i=1;i<=4;++i) c[i]=read();
   n=read();f[0]=1;
   for(int i=1;i<=4;++i)
   for(int j=c[i];j<maxs;++j)
      f[j]+=f[j-c[i]];
   for(int i=1;i<=n;++i)
   {
      for(int j=1;j<=4;++j) d[j]=read();
      s=read();ans=0;
      for(int j=0;j<=15;++j)
      {
         int tmp=s;p=1;
         for(int k=1;k<=4;++k)
            if((j>>(k-1))&1) p*=-1,tmp-=(d[k]+1)*c[k];
         if(tmp>=0) ans+=p*f[tmp];
      }
      printf("%lld\n",ans);
   }
   return 0;
}
```

6.3. 单调队列优化 DP

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=2e5+10;
const ll inf=-1e18;
ll ans=inf,f[2][maxn];
ll n,m,head,tail,d,k;
ll a[maxn],b[maxn],t[maxn],q[maxn];
int main()
{
   n=read(),m=read(),d=read();
   for(int i=1;i<=m;++i)
      a[i]=read(),b[i]=read(),t[i]=read();
```

```
    for(int i=1,o=0;i<=m;++i,o^=1)
    {
        head=1,tail=0,k=1;
        for(int j=1;j<=n;++j)
        {
            int T=t[i]-t[i-1];
            for(;k<=min(n,j+T*d);++k)
            {
                while(head<=tail&&f[o^1][q[tail]]<=f[o^1][k]) tail--;
                q[++tail]=k;
            }
            while(head<=tail&&q[head]<j-T*d) head++;
            f[o][j]=f[o^1][q[head]]+b[i]-abs(a[i]-j);
            if(i==m) ans=max(ans,f[o][j]);
        }
    }
    printf("%lld",ans);
    return 0;
}
```

6.4. 单调队列优化多重背包

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn=1e6+10;
int n,W,head,tail,k,ans;
int v[maxn],w[maxn],m[maxn],q[maxn],f[maxn];
int main()
{
    n=read(),W=read();
    for(int i=1;i<=n;++i)
        v[i]=read(),w[i]=read(),m[i]=read();
    for(int i=1;i<=n;++i)//枚举物品
    for(int res=0;res<w[i];++res)//枚举余数
    {
        head=1,tail=0,k=(W-res)/w[i];
        for(int j=(W-res)/w[i];j>=0;--j)//枚举倍数
        {
            for(;k>=max(j-m[i],0);--k)
            {
                while(head<=tail&&f[k*w[i]+res]-k*v[i]
                      >=f[q[tail]*w[i]+res]-q[tail]*v[i])
```

```
                tail--;
            q[++tail]=k;
         }
         while(head<=tail&&q[head]>=j) head++;
         if(head<=tail) f[j*w[i]+res]=max(f[j*w[i]+res],
                        f[q[head]*w[i]+res]+(j-q[head])*v[i]);
      }
   }
   for(int i=0;i<=W;++i) ans=max(ans,f[i]);
   printf("%d",ans);
   return 0;
}
```

6.5. 矩阵加速

```
#include<bits/stdc++.h>
#define maxn 4
#define mo 1000000007
using namespace std;
long long t,n;
struct matrix
{
    long long m[maxn][maxn];
    void clear()
    {
       for(int i=1;i<maxn;++i)
       for(int j=1;j<maxn;++j)
           m[i][j]=0;
    }
}a,b,e;
matrix mul(matrix a,matrix b)
{
   matrix c;c.clear();
   for(int i=1;i<maxn;++i)
   for(int j=1;j<maxn;++j)
   for(int k=1;k<maxn;++k)
       c.m[i][j]=(c.m[i][j]+((a.m[i][k]%mo)*(b.m[k][j]%mo))%mo)%mo;
   return c;
}
matrix poww(matrix a,long long b)
{
   matrix base=a,ans=e;
   while(b)
```

```
    {
        if(b&1)ans=mul(base,ans);
        base=mul(base,base);
        b>>=1;
    }
    return ans;
}
int main()
{
    t=read();
    while(t--)
    {
        n=read();
        a.clear();b.clear();e.clear();
        a.m[1][1]=a.m[1][2]=a.m[1][3]=1;
        b.m[1][1]=b.m[1][3]=b.m[2][1]=b.m[3][2]=1;
        e.m[1][1]=e.m[2][2]=e.m[3][3]=1;
        matrix ans=mul(poww(b,n-1),a);
        printf("%lld\n",ans.m[1][1]);
    }
    return 0;
}
```

6.6. 树的直径

```
#include<bits/stdc++.h>
#define maxn 10000+10
using namespace std;
int u,v,w,flag,ans,num;
int dis[maxn],fir[maxn];
struct qwq
{
    int nxt,to,val;
}e[maxn<<1];
void addedge(int u,int v,int w)
{
    e[++num].to=v;
    e[num].val=w;
    e[num].nxt=fir[u];
    fir[u]=num;
}
void dfs(int u,int f)
{
```

```
    if(dis[u]>ans)
        ans=dis[u],flag=u;
    for(int i=fir[u];i;i=e[i].nxt)
    {
        int v=e[i].to;
        if(v==f) continue;
        dis[v]=dis[u]+e[i].val;
        dfs(v,u);
    }
}
int main()
{
    while(~scanf("%d%d%d",&u,&v,&w))
        addedge(u,v,w),addedge(v,u,w);
    dfs(1,0);
    dis[flag]=0,dfs(flag,0);
    printf("%d",ans);
    return 0;
}
```

6.7. 树的重心

```
#include<bits/stdc++.h>
#define LL long long
#define maxn 100010
#define maxm 100010
#define inf 0x7fffffff
using namespace std;
int T,n,num,u,v,ans,flag;
int fir[maxn],sz[maxn];
struct qwq
{
    int nxt,to;
}e[maxm<<1];
void addedge(int u,int v)
{
    e[++num].to=v;
    e[num].nxt=fir[u];
    fir[u]=num;
}
int dfs(int u,int f)
{
    int maxi=0;sz[u]=1;
    for(int i=fir[u];i;i=e[i].nxt)
```

```
    {
        int v=e[i].to;
        if(v==f) continue;
        sz[u]+=dfs(v,u);
        maxi=max(maxi,sz[v]);
    }
    maxi=max(maxi,n-sz[u]);
    if(maxi<ans)
        flag=u,ans=maxi;
    return sz[u];
}
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        ans=inf,num=0;
        memset(fir,0,sizeof(fir));
        for(int i=1;i<n;++i)
        {
            scanf("%d%d",&u,&v);
            addedge(u,v);addedge(v,u);
        }
        dfs(1,0);
        printf("%d %d\n",flag,ans);
    }
    return 0;
}
```

6.8. 树上背包

```
#include<bits/stdc++.h>
#define inf 1e9
#define maxn 310
using namespace std;
int n,m,fa,num;
int s[maxn],fir[maxn],sz[maxn],dp[maxn][maxn],tmp[maxn];
struct qwq
{
    int to,nxt;
}e[maxn];
void addedge(int u,int v)
```

```
{
   e[++num].to=v;
   e[num].nxt=fir[u];
   fir[u]=num;
}
void dfs(int u)
{
   sz[u]=1;
   dp[u][1]=s[u];
   for(int i=fir[u];i;i=e[i].nxt)
   {
      int v=e[i].to;
      dfs(v);sz[u]+=sz[v];
      memcpy(tmp,dp[u],sizeof(dp[u]));
      for(int j=1;j<=sz[u];++j)
       for(int k=0;k<j;++k)
         dp[u][j]=max(dp[u][j],dp[v][k]+tmp[j-k]);
   }
}
int main()
{
   n=read();m=min(read(),n);
   for(int i=1;i<=n;++i)
   {
      fa=read(),s[i]=read();
      addedge(fa,i);
   }
   dfs(0);
   printf("%d",dp[0][m+1]);
   return 0;
}
```

6.9. 数位 DP(不要 62)

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=11;
int A,B,a[maxn],dp[maxn][maxn];
int work(int x)
{
   int ans=0,len=0;
   while(x)
   {
```

```
        a[++len]=x%10;
        x/=10;
    }
    for(int i=1;i<len;++i)
    for(int j=1;j<=9;++j)
        ans+=dp[i][j];
    for(int i=1;i<a[len];++i)
        ans+=dp[len][i];
    if(a[len]!=4)
    {
        for(int i=len-1;i>=1;--i)
        {
            for(int j=0;j<a[i];++j)
                if(!(a[i+1]==6&&j==2)) ans+=dp[i][j];
            if(a[i]==4||(a[i+1]==6&&a[i]==2)) break;
        }
    }
    return ans;
}
int main()
{
    for(int i=0;i<=9;++i)
        if(i!=4) dp[1][i]=1;
    for(int i=2;i<=10;++i)
    for(int j=0;j<=9;++j)
    for(int k=0;k<=9;++k)
        if(j!=4&&!(j==6&&k==2)) dp[i][j]+=dp[i-1][k];
    while(~scanf("%d%d",&A,&B))
    {
        if(!A&&!B) break;
        printf("%d\n",work(B+1)-work(A));
    }
    return 0;
}
```

7. 计算几何

```
#include<bits/stdc++.h>
#define MAXN 1001
using namespace std;
const double eps = 1e-8;
const double pi = acos(-1.0);
```

```
int sgn(double x)
{
   if(fabs(x) < eps) return 0;//认为是 0
   if(x < 0) return -1;//负数
   else return 1;//正数
}
inline double sqr(double x){return x*x;}
struct Point
{
   double x,y;
   Point(){}
   Point(double X,double Y){x=X;y=Y;}
   void input(){scanf("%lf%lf",&x,&y);}
   void output(){printf("%.2f%.2f",x,y);}
   bool operator==(Point b) const//判断是否相等
   {return sgn(x-b.x)==0&&sgn(y-b.y)==0;}
   bool operator<(Point b) const
   {return sgn(x-b.x)==0? sgn(y-b.y)<0:x<b.x;}
   Point operator+(const Point &b) const
   {return Point(x+b.x,y+b.y);}//加
   Point operator-(const Point &b) const
   {return Point(x-b.x,y-b.y);}//减
   Point operator*(const double &k) const
   {return Point(x*k,y*k);}//乘
   Point operator/(const double &k) const
   {return Point(x/k,y/k);}//除
   double operator^(const Point &b) const
   {return x*b.y-y*b.x;}//叉积
   double operator*(const Point &b) const
   {return x*b.x+y*b.y;}//点积
   double lenth(){return hypot(x,y);}//返回长度
   double lenth2(){return x*x+y*y;}//返回长度的平方
   double distance(Point p){return hypot(x-p.x,y-p.y);}//长度
   double rad(Point a,Point b)//计算 pa 和 pb 的夹角
   {//用法：p.rad(a,b)
      Point p=*this;
      return fabs(atan2(fabs((a-p)^(b-p)),(a-p)*(b-p)));
   }
   Point trunc(double r)//向量长度变为 r
   {
      double l=lenth();
      if(!sgn(l)) return *this;
      r/=l;
      return Point(x*r,y*r);
```

```
    }
    Point rotleft(){return Point(-y,x);}//逆时针旋转 90 度
    Point rotright(){return Point(y,-x);}//顺时针旋转 90 度
    Point rotate(Point p,double angle)//绕着 p 点逆时针旋转 angle
    {//用法：pos.rotate(p,angle_rad)
        Point v=(*this)-p;
        double c=cos(angle),s=sin(angle);
        return Point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
    }
};
struct Line
{
    Point s,e;
    Line(){}
    Line(Point S,Point E){s=S;e=E;}
    bool operator==(Line v){return (s==v.s)&&(e==v.e);}
    Line(Point p,double angle)
    {//一个点和倾斜角确定直线,0<=angle<pi
        s=p;
        if(sgn(angle-pi/2)==0){e=(s+Point(0,1));}
        else{e=(s+Point(1,tan(angle)));}
    }
    Line(double a,double b,double c)
    {//ax+by+c=0 确定直线
        if(sgn(a)==0){s=Point(0,-c/b);e=Point(1,-c/b);}
        else if(sgn(b)==0){s=Point(-c/a,0);e=Point(-c/a,1);}
        else{s=Point(0,-c/b);e=Point(1,(-c-a)/b);}
    }
    void input(){s.input();e.input();}
    void adjust(){if(e<s)swap(s,e);}
    double length(){return s.distance(e);}//线段长度
    double angle()
    {//返回直线倾斜角 0<=angle<pi
        double k=atan2(e.y-s.y,e.x-s.x);
        if(sgn(k)<0) k+=pi;
        if(sgn(k-pi)==0) k-=pi;
        return k;
    }
    int relation(Point p)//点线关系
    {//1 左侧 2 右侧 3 线上
        int c=sgn((p-s)^(e-s));
        if(c<0) return 1;
        else if(c>0) return 2;
        else return 3;
```

```
}
bool parallel(Line v)
{//判断两向量平行(对应直线平行或重合)
   return sgn((e-s)^(v.e-v.s))==0;
}
int linecrossline(Line v)//两直线关系
{//0 平行 1 重合 2 相交
   if((*this).parallel(v)) return v.relation(s)==3;
   return 2;
}
Point crosspoint(Line v)
{//求两直线的交点
   double a1=(v.e-v.s)^(s-v.s);
   double a2=(v.e-v.s)^(e-v.s);
   return Point((s.x*a2-e.x*a1)/(a2-a1),(s.y*a2-e.y*a1)/(a2-a1));
}
bool ifcrossline(Line v)
{//判断两线段是否相交
   if(!(max(s.x,e.x)+eps>=min(v.s.x,v.e.x)
      &&min(s.x,e.x)<=max(v.s.x,v.e.x)+eps))
      return 0;
   if(!(max(s.y,e.y)+eps>=min(v.s.y,v.e.y)
      &&min(s.y,e.y)<=max(v.s.y,v.e.y)+eps))
      return 0;
   if(((e-s)^(v.s-s))*((e-s)^(v.e-s))>eps)
      return 0;
   if(((v.e-v.s)^(s-v.s))*((v.e-v.s)^(e-v.s))>eps)
      return 0;
   return 1;
}
double dispointtoline(Point p)
{//点到直线的距离
   return fabs((p-s)^(e-s))/length();
}
double dispointtoseg(Point p)
{//点到线段的距离
   if(sgn((p-s)*(e-s))<0||sgn((p-e)*(s-e))<0)
      return min(p.distance(s),p.distance(e));
   return dispointtoline(p);
}
double dissegtoseg(Line v)
{//返回线段到线段的距离
   return min(min(dispointtoseg(v.s),dispointtoseg(v.e)),
   min(v.dispointtoseg(s),v.dispointtoseg(e)));
```

```
    }
    Point lineprog(Point p)
    {//返回点 p 在直线上的投影
        return s+(((e-s)*((e-s)*(p-s)))/((e-s).lenth2()));
    }
    Point symmetrypoint(Point p)
    {//返回点 p 关于直线的对称点
        Point q=lineprog(p);
        return Point(2*q.x-p.x,2*q.y-p.y);
    }
};
struct Circle
{
    Point p;//圆心
    double r;//半径
    Circle(){}
    Circle(Point P,double R){p=P;r=R;}
    Circle(double x,double y,double R)
    {p=Point(x,y);r=R;}
    Circle(Point a,Point b,Point c)
    {//三角形的外接圆
        Line u=Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
        Line v=Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
        p=u.crosspoint(v);
        r=p.distance(a);
    }
    Circle(Point a,Point b,Point c,bool t)//t 仅作区分
    {//三角形的内切圆
        Line u,v;double m,n;
        m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);
        u.s=a;v.s=b;
        u.e=u.s+Point(cos((n+m)/2),sin((n+m)/2));
        m=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);
        v.e=v.s+Point(cos((n+m)/2),sin((n+m)/2));
        p=u.crosspoint(v);
        r=Line(a,b).dispointtoseg(p);
    }
    void input(){p.input();scanf("%lf",&r);}
    void output(){printf("%.2lf %.2lf %.2lf\n",p.x,p.y,r);}
    bool operator==(Circle v)
    {return (p==v.p)&&sgn(r-v.r)==0;}
    bool operator<(Circle v) const
    {return ((p<v.p)||((p==v.p)&&sgn(r-v.r)<0));}
    double area(){return pi*r*r;}//面积
```

```
double circumference(){return 2*pi*r;}//周长
int relation(Point b)//点和圆的关系
{//0 圆外 1 圆上 2 圆内
   double dst=b.distance(p);
   if(sgn(dst-r)<0) return 2;
   else if(sgn(dst-r)==0) return 1;
   return 0;
}
int relationseg(Line v)//线段和圆的关系
{//0 圆外 1 相切 2 圆内
   double dst=v.dispointtoseg(p);
   if(sgn(dst-r)<0) return 2;
   else if(sgn(dst-r)==0) return 1;
   return 0;
}
int relationline(Line v)//直线和圆的关系
{//0 圆外 1 相切 2 圆内
   double dst=v.dispointtoline(p);
   if(sgn(dst-r)<0) return 2;
   else if(sgn(dst-r)==0) return 1;
   return 0;
}
int relationcircle(Circle v)//两圆的关系
{//5 相离 4 外切 3 相交 2 内切 1 内含
   double d=p.distance(v.p);
   if(sgn(d-r-v.r)>0) return 5;
   if(sgn(d-r-v.r)==0) return 4;
   double l=fabs(r-v.r);
   if(sgn(d-r-v.r)<0&&sgn(d-l)>0) return 3;
   if(sgn(d-l)==0) return 2;
   if(sgn(d-l)<0) return 1;
}
int pointcrosscircle(Circle v,Point &p1,Point &p2)
{//求两个圆的交点个数
   int rel=relationcircle(v);
   if(rel==1||rel==5) return 0;
   double d=p.distance(v.p);
   double l=(d*d+r*r-v.r*v.r)/(2*d);
   double h=sqrt(r*r-l*l);
   Point tmp=p+(v.p-p).trunc(l);
   p1=tmp+((v.p-p).rotleft().trunc(h));
   p2=tmp+((v.p-p).rotright().trunc(h));
   if(rel==2||rel==4) return 1;
   return 2;
```

```
    }
    int pointcrossline(Line v,Point &p1,Point &p2)
    {//求直线和圆的交点个数
        if(!(*this).relationline(v)) return 0;
        Point a=v.lineprog(p);
        double d=v.dispointtoline(p);
        d=sqrt(r*r-d*d);
        if(sgn(d)==0){p1=a;p2=a;return 1;}
        p1=a+(v.e-v.s).trunc(d);
        p2=a-(v.e-v.s).trunc(d);
        return 2;
    }
    int gercircle(Point a,Point b,double r1,Circle &c1,Circle &c2)
    {//得到过 a,b 两点,半径为 r1 的两个圆
        Circle x(a,r1),y(b,r1);
        int t=x.pointcrosscircle(y,c1.p,c2.p);
        if(!t) return 0;
        c1.r=c2.r=r1;
        return t;
    }
    int getcircle(Line u,Point q,double r1,Circle &c1,Circle &c2)
    {//得到与直线 u 相切,过点 q,半径为 r1 的圆
        double dis=u.dispointtoline(q);
        if(sgn(dis-r1*2)>0) return 0;
        if(sgn(dis) == 0)
        {
            c1.p=q+((u.e-u.s).rotleft().trunc(r1));
            c2.p=q+((u.e-u.s).rotright().trunc(r1));
            c1.r=c2.r=r1;return 2;
        }
        Line
u1=Line((u.s+(u.e-u.s).rotleft().trunc(r1)),(u.e+(u.e-u.s).rotleft().trun
c(r1)));
        Line
u2=Line((u.s+(u.e-u.s).rotright().trunc(r1)),(u.e+(u.e-u.s).rotright().tr
unc(r1)));
        Circle cc=Circle(q,r1);Point p1,p2;
        if(!cc.pointcrossline(u1,p1,p2)) cc.pointcrossline(u2,p1,p2);
        c1=Circle(p1,r1);
        if(p1==p2){c2=c1;return 1;}
        c2=Circle(p2,r1);return 2;
    }
    int getcircle(Line u,Line v,double r1,Circle &c1,Circle &c2,Circle
&c3,Circle &c4)
```

```
   {//同时与直线 u,v 相切,半径为 r1 的圆
      if(u.parallel(v))return 0;
      Line
u1=Line(u.s+(u.e-u.s).rotleft().trunc(r1),u.e+(u.e-u.s).rotleft().trunc(r
1));
      Line
u2=Line(u.s+(u.e-u.s).rotright().trunc(r1),u.e+(u.e-u.s).rotright().trunc
(r1));
      Line
v1=Line(v.s+(v.e-v.s).rotleft().trunc(r1),v.e+(v.e-v.s).rotleft().trunc(r
1));
      Line
v2=Line(v.s+(v.e-v.s).rotright().trunc(r1),v.e+(v.e-v.s).rotright().trunc
(r1));
      c1.r=c2.r=c3.r=c4.r=r1;
      c1.p=u1.crosspoint(v1);
      c2.p=u1.crosspoint(v2);
      c3.p=u2.crosspoint(v1);
      c4.p=u2.crosspoint(v2);
      return 4;
   }
   int getcircle(Circle cx,Circle cy,double r1,Circle &c1,Circle &c2)
   {//同时与不相交圆 cx,cy 相切，半径为 r1 的圆
      Circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
      int t=x.pointcrosscircle(y,c1.p,c2.p);
      if(!t) return 0;
      c1.r=c2.r=r1;return t;
   }
   int tangentline(Point q,Line &u,Line &v)
   {//过一点作圆的两条切线
      int x=relation(q);
      if(x==2)return 0;
      if(x==1)
      {u=Line(q,q+(q-p).rotleft());v=u;return 1;}
      double d=p.distance(q);
      double l=r*r/d;
      double h=sqrt(r*r-l*l);
      u=Line(q,p+((q-p).trunc(l)+(q-p).rotleft().trunc(h)));
      v=Line(q,p+((q-p).trunc(l)+(q-p).rotright().trunc(h)));
      return 2;
   }
   double areacircle(Circle v)
   {//求两圆相交的面积
      int rel=relationcircle(v);
```

```
        if(rel>=4)return 0.0;
        if(rel<=2)return min(area(),v.area());
        double d=p.distance(v.p);
        double hf=(r+v.r+d)/2.0;
        double ss=2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
        double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
        a1=a1*r*r;
        double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
        a2=a2*v.r*v.r;
        return a1+a2-ss;
    }
    double areatriangle(Point a,Point b)
    {//求圆和三角形 pab 的相交面积
        if(sgn((p-a)^(p-b))==0) return 0.0;
        Point q[5];int len=0;q[len++]=a;
        Line l(a,b);Point p1,p2;
        if(pointcrossline(l,q[1],q[2])==2)
        {
            if(sgn((a-q[1])*(b-q[1]))<0) q[len++]=q[1];
            if(sgn((a-q[2])*(b-q[2]))<0) q[len++]=q[2];
        }
        q[len++]=b;
        if(len==4&&sgn((q[0]-q[1])*(q[2]-q[1]))>0) swap(q[1],q[2]);
        double res=0;
        for(int i=0;i<len-1;i++)
        {
            if(relation(q[i])==0||relation(q[i+1])==0)
            {double arg=p.rad(q[i],q[i+1]);res+=r*r*arg/2.0;}
            else{res+=fabs((q[i]-p)^(q[i+1]-p))/2.0;}
        }
        return res;
    }
};
struct Polygon
{
    int n;
    Point p[MAXN];
    Line l[MAXN];
    void input(int N)
    {
        n=N;
        for(int i=0;i<n;i++) p[i].input();
    }
    void add(Point q){p[n++]=q;}
```

```
    void getline()
    {
        for(int i=0;i<n;i++) l[i]=Line(p[i],p[(i+1)%n]);
    }
    struct cmp
    {
        Point p;
        cmp(const Point &p0){p=p0;}
        bool operator()(const Point &aa,const Point &bb)
        {
            Point a=aa,b=bb;
            int d=sgn((a-p)^(b-p));
            if(d==0)
            {return sgn(a.distance(p)-b.distance(p))<0;}
            return d>0;
        }
    };
    void norm()
    {//进行极角排序
        Point mi=p[0];
        for(int i=1;i<n;i++) mi=min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    void getconvex(Polygon &convex)
    {//得到凸包 0~n-1
        sort(p,p+n);
        convex.n=n;
        for(int i=0;i<min(n,2);i++){convex.p[i]=p[i];}
        if(convex.n==2&&(convex.p[0]==convex.p[1])) convex.n--;//特判
        if(n<=2) return;
        int &top=convex.n;
        top=1;
        for(int i=2;i<n;i++)
        {
            while(top&&sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0)
top--;
            convex.p[++top]=p[i];
        }
        int temp=top;
        convex.p[++top]=p[n-2];
        for(int i=n-3;i>=0;i--)
        {

    while(top!=temp&&sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i]))<=0)
```

```
top--;
            convex.p[++top]=p[i];
        }
        if(convex.n==2&&(convex.p[0]==convex.p[1])) convex.n--;//特判
        convex.norm();//`原来得到的是顺时针的点，排序后逆时针`
    }
    bool isconvex()
    {//判断是不是凸的
        bool s[3];
        memset(s,false,sizeof(s));
        for(int i=0;i<n;i++)
        {
            int j=(i+1)%n;
            int k=(j+1)%n;
            s[sgn((p[j]-p[i])^(p[k]-p[i]))+1]=true;
            if(s[0]&&s[2]) return false;
        }
        return true;
    }
    int relationpoint(Point q)//判断点和任意多边形的关系
    {//3 点上 2 边上 1 内部 0 外部
        for(int i=0;i<n;i++){if(p[i]==q) return 3;}
        getline();
        for(int i=0;i<n;i++){if(l[i].relation(q)==3) return 2;}
        int cnt=0;
        for(int i=0;i<n;i++)
        {
            int j=(i+1)%n;
            int k=sgn((q-p[j])^(p[i]-p[j]));
            int u=sgn(p[i].y-q.y);
            int v=sgn(p[j].y-q.y);
            if(k>0&&u<0&&v>=0) cnt++;
            if(k<0&&v<0&&u>=0) cnt--;
        }
        return cnt!=0;
    }
    void convexcut(Line u,Polygon &po)
    {//用直线 u 切割凸多边形左侧
        int &top=po.n;
        top=0;
        for(int i=0;i<n;i++){
            int d1=sgn((u.e-u.s)^(p[i]-u.s));
            int d2=sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
            if(d1>=0) po.p[top++]=p[i];
```

```
        if(d1*d2<0)
po.p[top++]=u.crosspoint(Line(p[i],p[(i+1)%n]));
    }
}
double getcircumference()
{//得到周长
    double sum=0;
    for(int i=0;i<n;i++){sum+=p[i].distance(p[(i+1)%n]);}
    return sum;
}
double getarea()
{//得到面积
    double sum=0;
    for(int i=0;i<n;i++){sum+=(p[i]^p[(i+1)%n]);}
    return fabs(sum)/2;
}
Point getbarycentre()
{//得到重心
    Point ret(0,0);
    double area=0;
    for(int i=1;i<n-1;i++)
    {
        double tmp=(p[i]-p[0])^(p[i+1]-p[0]);
        if(sgn(tmp)==0)continue;
        area+=tmp;
        ret.x+=(p[0].x+p[i].x+p[i+1].x)/3*tmp;
        ret.y+=(p[0].y+p[i].y+p[i+1].y)/3*tmp;
    }
    if(sgn(area)) ret=ret/area;
    return ret;
}
double areacircle(Circle c)
{//多边形和圆交的面积
    double ans=0;
    for(int i=0;i<n;i++)
    {
        int j=(i+1)%n;
        if(sgn((p[j]-c.p)^(p[i]-c.p))>= 0)
            ans+=c.areatriangle(p[i],p[j]);
        else ans-=c.areatriangle(p[i],p[j]);
    }
    return fabs(ans);
}
int relationcircle(Circle c)//多边形和圆关系
```

```
    {//2 圆完全在里面 1 内切 0 其他
        getline();
        int x=2;
        if(relationpoint(c.p)!=1) return 0;
        for(int i=0;i<n;i++)
        {
            if(c.relationseg(l[i])==2) return 0;
            if(c.relationseg(l[i])==1) x=1;
        }
        return x;
    }
};
int main()
{
    return 0;
}
```