

模板

Ether Strike

2022 年 10 月 31 日

目录

| | | |
|----------|------------------------|-----------|
| 1 | 数学 | 2 |
| 1.1 | 线性代数 | 2 |
| 1.1.1 | 矩阵乘法 | 2 |
| 1.1.2 | 么半群-证明结合律的工具 | 3 |
| 1.1.3 | 线性基 | 3 |
| 1.1.4 | 矩阵树定理 | 4 |
| 1.2 | gcd 相关 | 8 |
| 1.3 | 数相关 | 9 |
| 1.3.1 | 生成函数 | 9 |
| 1.3.2 | 斯特林数 | 10 |
| 1.3.3 | 卡特兰数 | 10 |
| 1.3.4 | 组合数 | 11 |
| 1.4 | 博弈论 | 11 |
| 1.4.1 | sg 函数 | 11 |
| 1.4.2 | Nash 均衡 | 11 |
| 1.5 | 多项式 | 11 |
| 1.5.1 | FFT | 11 |
| 1.5.2 | 单位根反演 | 14 |
| 1.5.3 | NTT | 14 |
| 1.5.4 | 拆系数 FFT | 23 |
| 1.5.5 | 拉格朗日插值 | 24 |
| 1.6 | 同余 | 28 |
| 1.6.1 | BSGS | 28 |
| 1.6.2 | 原根 | 28 |
| 1.7 | 筛法 | 28 |
| 1.7.1 | 常用积性函数关系 | 28 |
| 1.7.2 | PN 筛 | 28 |
| 1.7.3 | 杜教筛 | 28 |
| 1.7.4 | min25 筛 (待填) | 29 |
| 2 | 数据结构 | 30 |
| 2.1 | 单调队列 | 30 |
| 2.2 | 线段树 | 31 |

| | | |
|----------|-----------------------------|-----------|
| 2.2.1 | 普通线段树 | 31 |
| 2.2.2 | 动态开点线段树 | 32 |
| 2.2.3 | 主席树 | 36 |
| 2.3 | 并查集 | 37 |
| 2.3.1 | 普通并查集 | 37 |
| 2.3.2 | 可删除并查集 | 38 |
| 3 | 字符串 | 39 |
| 3.1 | KMP 算法 | 39 |
| 3.2 | 后缀数组 | 39 |
| 4 | 图论 | 42 |
| 4.1 | 最短路 – dij 的最长路是错的 | 42 |
| 4.2 | 二分图 | 43 |
| 4.2.1 | 二分图最大匹配 | 43 |
| 4.2.2 | 二分图最大权匹配 | 43 |
| 4.3 | 网络流 | 45 |
| 4.4 | 强连通分量 | 49 |
| 4.5 | 点双联通分量 | 50 |
| 5 | 杂项 | 52 |
| 5.1 | 快读快输板子 | 52 |
| 5.2 | 分治 | 53 |
| 5.2.1 | cdq 分治 | 53 |
| 5.2.2 | 树上点分治 | 55 |
| 5.3 | 莫队 | 57 |
| 5.3.1 | 普通莫队 | 57 |
| 5.3.2 | 带修莫队 | 58 |
| 5.3.3 | 回滚莫队 | 60 |
| 5.4 | 悬线法 | 62 |
| 5.5 | n 数码问题 | 63 |
| 5.6 | 高精度板子 | 63 |
| 5.7 | 整体二分 | 67 |
| 5.8 | 背包回退 (待写) | 70 |

1 数学

1.1 线性代数

1.1.1 矩阵乘法

```
1 using Matrix = array<array<int,80>,80>;
2 using Vector = array<int,80>;
3 Vector operator *(Vector a, Matrix b){
4     Vector res={0};
5     for(int i=0; i<m; i++)
6         for(int j=0; j<m; j++)
7             inc(res[j], 111*a[i]*b[i][j]%mod);
8     return res;
9 }
10 Vector operator *(Matrix b, Vector a){
11     Vector res={0};
12     for(int i=0; i<80; i++)
13         for(int j=0; j<80; j++)
14             inc(res[i], 111*a[j]*b[i][j]%mod);
15     return res;
16 }
17 Matrix operator *(Matrix a, Matrix b){
18     Matrix res={0};
19     for(int i=0; i<80; i++)
20         for(int j=0; j<80; j++)
21             for(int k=0; k<80; k++)
22                 inc(res[i][j], 111*a[i][k]*b[k][j]%mod);
23     return res;
24 }
25 Matrix power(Matrix a, ll b){
26     Matrix res={0};
27     for(int i=0; i<80; i++)
28         res[i][i]=1;
29     for(;b>=1,a=a*a)
30         if(b&1)
31             res=res*a;
32     return res;
33 }
34 inline void solve(){
35     int n;
36     cin >> n;
37     Vector now={0};
38     now[0] = 1;
39     a = power(a, n);
```

```

40     now = a * now;
41 }

```

1.1.2 幺半群—证明结合律的工具

满足结合律就可以使用矩阵加速 (如 $\max, +$)

定义: 零元 (O), 加法 ($+$), 乘法 (\times)

矩阵乘法的形式: $c_{i,j} = \sum a_{i,k} \times b_{k,j}$

如果一个运算满足结合律, 当且仅当它满足

1. $O \times A = O$
2. $(a + b) + c = a + (b + c)$
3. $(a \times b) \times c = a \times (b \times c)$
4. $(a + b) \times c = a \times c + b \times c$
5. $a \times (b + c) = a \times b + a \times c$

example: $c_{i,j} = \min\{\max\{a_{i,k}, b_{k,j}\}\}$ 此处 \min 为加法, \max 为乘法, 零元为 ∞

1.1.3 线性基

```

1 struct Linear{
2 //cnt != n说明可以构成0
3     ll p[64]; //根据数字的范围来开
4     int cnt, n;
5     Linear(){cnt = n = 0; for(int i=0; i<64; i++) p[i] = 0;}
6     inline void clear(){cnt = n = 0; for(int i=0; i<64; i++) p[i] = 0;}
7     inline void copy(Lin ear b){
8         for(int i=0; i<64; i++)
9             p[i] = b.p[i];
10    }
11    inline bool insert(ll x){
12        ++n;
13        for(int i=63; i>=0; i--){
14            if(x&(1ll<<i)){
15                if(!p[i]){p[i]=x; cnt++; return 1;}
16                x^=p[i];
17            }
18        }
19        return 0;
20    }
21    inline ll query(ll x){
22        if(cnt != n) --x;
23        if(!x) return 0;
24        ll res = 0;
25        for(int i=0; i<64; i++)

```

```

25         if(p[i]){
26             if(x&1) res^=p[i];
27             x>>=1;
28         }
29         if(x) return -1;
30         return res;
31     }
32     inline ll mx(){// 线性基最大值
33         ll res = 0;
34         for(int i=63; ~i; i--)
35             if(p[i] && (res^p[i]) > res) res^=p[i];
36         return res;
37     }
38 };
39 inline Linear merge(Linear a, Linear b){
40     for(int i=0; i<=63; i++)
41         a.insert(b.p[i]);
42     return a;
43 }

```

1.1.4 矩阵树定理

$O(N^3)$ 求以每个点为根的外向树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define fo(i,j,k) for(int i=(j),end_i=(k);i<=end_i;i++)
4  #define ff(i,j,k) for(int i=(j),end_i=(k);i< end_i;i++)
5  #define fd(i,j,k) for(int i=(j),end_i=(k);i>=end_i;i--)
6  #define all(x) (x).begin(),(x).end()
7  #define cle(x) memset(x,0,sizeof(x))
8  #define lowbit(x) ((x)&-(x))
9  #define VI vector<int>
10 #define ll long long
11 const ll mod=1e9+7;
12 inline ll Add(ll x,ll y){x+=y; return (x<mod)?x:x-mod;}
13 inline ll Dec(ll x,ll y){x-=y; return (x<0)?x+mod:x;}
14 inline ll Mul(ll x,ll y){return x*y%mod;}
15 inline ll Pow(ll x,ll y)
16 {
17     y%=(mod-1);ll ans=1;for(;y;y>>=1,x=x*x%mod)if(y&1) ans=ans*x%mod;
18     return ans;
19 }
20 const int N=505;
21 int n;

```

```

22 struct matrix{
23     ll a[N][N];
24     matrix(){fo(i,0,n) fo(j,0,n) a[i][j]=0;}
25     void clear(int n)
26     {
27         fo(i,0,n) fo(j,0,n) a[i][j]=0;
28     }
29     inline ll* operator [](int x){return a[x];}
30 };
31 inline ll det(matrix a,int n)
32 {
33     ll d=1,inv,tmp;
34     int flag=1;
35     fo(i,1,n)
36     {
37         int k=i;
38         fo(j,i+1,n) if(a[j][i]) {k=j; break;}
39         if(k!=i) {fo(j,i,n) swap(a[k][j],a[i][j]); flag=-flag;}
40         if(!a[i][i]) return 0;
41         inv=Pow(a[i][i],mod-2);
42         fo(j,i+1,n)
43         {
44             tmp=a[j][i]*inv%mod;
45             fo(k,i,n) a[j][k]=Dec(a[j][k],a[i][k]*tmp%mod);
46         }
47         d=d*a[i][i]%mod;
48     }
49     if(flag== -1) d=(mod-d%mod)%mod;
50     return d;
51 }
52 int r(matrix a)
53 {
54     int d=0;
55     ll inv,tmp;
56     fo(i,1,n)
57     {
58         int k=i;
59         fo(j,i+1,n) if(a[j][i]) {k=j; break;}
60         if(k!=i) fo(j,i,n) swap(a[i][j],a[k][j]);
61         if(!a[i][i]) continue;
62         d++;
63         inv=Pow(a[i][i],mod-2);
64         fo(j,i+1,n)
65         {

```

```

66         tmp=a[j][i]*inv%mod;
67         fo(k,i,n) a[j][k]=Dec(a[j][k],a[i][k]*tmp%mod);
68     }
69 }
70 return d;
71 }
72 ll l[N],p[N];
73 inline void solve(matrix a,ll *p)
74 {
75     static ll c[N][N],d[N][N];
76     fo(i,0,n) fo(j,0,n) c[i][j]=d[i][j]=0;
77     auto check = [&](const int &id,ll *g){
78         fo(i,1,n) p[i]=0;
79         p[id]=1;
80         ll tmp;
81         fd(i,n,1)
82             if(l[i])
83                 {
84                     if(!c[i][i])
85                         {
86                             fd(j,i,1) c[i][j]=l[j];
87                             fo(j,1,n) d[i][j]=p[j];
88                             return 0;
89                         }
90                     tmp=Pow(c[i][i],mod-2)*l[i]%mod;
91                     fd(j,i,1) l[j]=Dec(l[j],c[i][j]*tmp%mod);
92                     fo(j,1,n) p[j]=Dec(p[j],d[i][j]*tmp%mod);
93                 }
94         fo(i,1,n) g[i]=p[i];
95         return 1;
96     };
97     fo(i,1,n)
98     {
99         fo(j,1,n)
100             l[j]=a[j][i];
101         if(check(i,p)) return;
102     }
103 }
104 void solve(matrix A)
105 {
106     int rank=r(A);
107     assert(rank!=n);
108     {
109         matrix B,C;

```

```

110     static ll x[N],y[N];
111     C.clear(n);
112     fo(i,1,n) fo(j,1,n) C[j][i]=A[i][j];
113     solve(A,y);
114     solve(C,x);
115     int c=0,r=0;
116     fo(i,1,n) if(y[i]) {c=i; break;}
117     fo(i,1,n) if(x[i]) {r=i; break;}
118     B.clear(n-1);
119     fo(i,1,n)
120         if(i!=r)
121             fo(j,1,n)
122                 if(j!=c)
123                     B[i-(i>r)][j-(j>c)]=A[i][j];
124     ll tmp=det(B,n-1);
125     tmp=((r+c)%2==1)?(mod-tmp)%mod:tmp;
126     tmp=Pow(y[c]*x[r]%mod,mod-2)*tmp%mod;
127     fo(i,1,n)
128         printf("%lld%c",tmp*x[i]%mod*y[i]%mod,(i==n)?'\n':' ');
129     return;
130 }
131 }
132 void solve()
133 {
134     int m,x,y;
135     scanf("%d%d",&n,&m);
136     if(n==1) {printf("1\n"); return;}
137     matrix A;
138     fo(i,1,m)
139     {
140         scanf("%d%d",&x,&y);
141         A[y][y]++;
142         A[x][y]--;
143     }
144     fo(i,1,n) fo(j,1,n) if(A[i][j]<0) A[i][j]+=mod;
145     solve(A);
146 }
147 int main()
148 {
149     int T; scanf("%d",&T);
150     while(T--) solve();
151     return 0;
152 }

```


1.2 gcd 相关

```
1 //luogu P5656 二元一次不定方程
2 ll exgcd(int a, int b, ll &x, ll &y){
3     if(b == 0){
4         x = 1; y = 0;
5         return a;
6     }
7     ll tx, ty;
8     int d = exgcd(b, a % b, tx, ty);
9     x = ty; y = tx - a / b * ty;
10    return d;
11 }
12
13 inline ll calc(int a, int b){//exCRT
14     //t%M = a t % M = b
15     //Nx+a=My+b=t
16     int N = n<<1, M = m<<1;
17     ll x, y;
18     int d = exgcd(N, M, x, y);
19     int c = b - a;
20     if(c % d != 0) return 2e18;
21     ll lcm = 1ll * N * M / d;
22     x = 1ll * x * (b-a) / d % lcm;
23     ll res = 1ll * x % lcm * N % lcm + a;
24     res = (res + lcm) % lcm;
25     if(res == 0) res = lcm;
26     return res;
27 }
28
29 inline void solve(){
30     int a, b, c;
31     cin >> a >> b >> c;
32     ll x, y;
33     int d = exgcd(a, b, x, y);
34     if(c % d)
35         cout << -1 << ln;
36     else {
37         ll rx = x * c / d, ry = y * c / d;
38         // cout << rx << " " << ry << ln;
39         if(rx <= 0){
40
41             ll tmp = abs(rx) / (b / d) + 1;
42             rx += tmp * b / d;
43             ry -= tmp * a / d;
```

```

44         if(ry <= 0) {
45             cout << rx << " ";
46
47             tmp = abs(ry) / (a / d) + 1;
48             ry += tmp * a / d;
49             rx -= tmp * b / d;
50
51             cout << ry << ln;
52             return;
53         }
54     } else if(ry <= 0){
55         ll tmp = abs(ry) / (a / d) + 1;
56         ry += tmp * a / d;
57         rx -= tmp * b / d;
58         ll tmpy = ry;
59         if(rx <= 0){
60
61             tmp = abs(rx) / (b / d) + 1;
62             rx += tmp * b / d;
63             ry -= tmp * a / d;
64             cout << rx << " " << tmpy << ln;
65
66             return;
67         }
68     }
69     // cout << rx << " " << ry << ln;
70
71     ll L = -rx * d / b, R = ry * d / a;
72     if(rx + L * b / d == 0) L++;
73     if(ry - R * a / d == 0) R--;
74     cout << R - L + 1 << " " << rx + L * b / d << " " << ry - R * a / d
75     //rx <= ry
76
77 }
78 }

```

1.3 数相关

1.3.1 生成函数

1. $e^{dx}[\frac{x^n}{n!}] = \frac{d^n}{n!}$
2. $(e^x - 1)^d[\frac{x^n}{n!}] = \{n \atop d\} d!$
3. $\frac{1}{(x-1)^d}[x^n] = \binom{n+d-1}{d-1}$

1.3.2 斯特林数

第一类斯特林数: 将 p 个不同的物品分成 k 个非空循环排列的方法数

$$S(p, 0) = 0, p \geq 1$$

$$S(p, p) = 1, p \geq 0$$

$$S(p, k) = (p-1)S(p-1, k) + S(p-1, k-1), p-1 \geq k \geq 1$$

生成函数 (可以用分治 fft 加速): $F(x) = \prod_{i=0}^{n-1} (x+i)$

第二类斯特林数

$$S_2(p, 0) = 0, p \geq 1$$

$$S_2(p, p) = 1, p \geq 0$$

$$S_2(p, k) = kS_2(p-1, k) + S_2(p-1, k-1), p-1 \geq k \geq 1$$

恒等式: $n^m = \sum_{k=0}^m S(m, k) \cdot \binom{n}{k} k!$

生成函数 (可以从恒等式二项式反演过来, 可以用 fft 加速): $S(n, k) = \sum_{i=0}^k (-1)^i \binom{k}{i} (k-i)^n$

下降幂公式: $x^n = \sum_{i=0}^n \left\{ \begin{matrix} n \\ i \end{matrix} \right\} x^{\underline{i}}$

| k 个球 | m 个盒子 | 是否允许有空盒子 | 方案数 |
|------|-------|----------|--|
| 各不相同 | 各不相同 | 是 | m^k |
| 各不相同 | 各不相同 | 否 | $m!S_2(k, m)$ |
| 各不相同 | 完全相同 | 是 | $\sum_{i=1}^m S_2(k, i)$ |
| 各不相同 | 完全相同 | 否 | $S_2(k, m)$ |
| 完全相同 | 各不相同 | 是 | $C(m+k-1, k)$ |
| 完全相同 | 各不相同 | 否 | $C(k-1, m-1)$ |
| 完全相同 | 完全相同 | 是 | $\frac{1}{(1-x)(1-x^2)\dots(1-x^m)} [x^k]$ |
| 完全相同 | 完全相同 | 否 | $\frac{x^m}{(1-x)(1-x^2)\dots(1-x^m)} [x^k]$ |

1.3.3 卡特兰数

1. 括号匹配: 你有 n 个左括号, n 个右括号, 问有多少个长度为 $2n$ 的括号序列使得所有的括号都是合法的
2. 进出栈问题有一个栈, 我们有 $2n$ 次操作, n 次进栈, n 次出栈, 问有多少中合法的进出栈序列
3. 312 排列一个长度为 n 的排列 $\{a\}$, 只要满足 $i < j < k$ 且 $a_j < a_k < a_i$ 就称这个排列为 312 排列, 求 n 的全排列中不是 312 排列的排列个数
4. 不相交弦问题在一个圆周上分布着 $2n$ 个点, 两两配对, 并在这两个点之间连一条弦, 要求所得的 $2n$ 条弦彼此不相交的配对方案数
5. 从 $(1, 1)$ 走到 (n, n) 且不穿过对角线有多少条路径

```

1  const int N=2e6+5;
2  const int mod = 20100403;
3  int n, m, fac[N], inv[N], cal[N];
4  inline int power(int a, int b){
5      int res=1;
6      while(b){
7          if(b&1)
8              res=1ll*res*a%mod;
9          b>>=1;

```

```

10     a=1ll*a*a%mod;
11 }
12 return res;
13 }
14 inline int C(int n, int m){
15     return 1ll*fac[n]*inv[m]%mod*inv[n-m]%mod;
16 }
17 inline void pre(int n){
18     fac[0]=1;
19     for(int i=1; i<=n; i++)
20         fac[i]=1ll*fac[i-1]*i%mod;
21     inv[n]=power(fac[n], mod-2);
22     for(int i=n-1; i>=0; i--)
23         inv[i]=1ll*inv[i+1]*(i+1)%mod;
24 }
25 inline void Caterlan(int n){
26     return (C(2*n, n)-C(2*n, n-1)+mod)%mod
27 }

```

1.3.4 组合数

- 链上不相邻地取 k 个数 $C(n-r+1, r)$
- 大小为 n 的环上不相邻的选 r 个点 $C(n-r-1, r-1) + C(n-r, r) = C(n-r, r) \frac{n}{n-r}$

1.4 博弈论

1.4.1 sg 函数

$sg(x) = \text{mex}\{sg(y)\}$ y 为 x 的可达状态点

游戏和的 SG 函数等于各个游戏 SG 函数的异或和

一般是把 sg 函数打印出来找规律

1.4.2 Nash 均衡

对于一个决策，有 p 的概率选择， $(1-p)$ 的概率不选择，设选择的代价为 x ，不选择的代价为 y ，可以得到 $z = px + (1-p)y$ ，是一个喜闻乐见的线性规划式子

- nim – 取最后一个的胜利：先手必胜要求 sg 函数不为 0
- anti-nim – 取最后一个的失败：先手必胜要求 sg 函数不为 0 且存在一个 $sg(x) > 1$ 或所有 $sg(x) = 1$ 且有偶数个游戏

1.5 多项式

1.5.1 FFT

精度可能不够，预处理单位根会好很多

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef unsigned long long ull;
5  typedef pair<int,int> pii;
6  #define pb push_back
7  #define ln '\n'
8
9  //FFT 多项式乘法
10 //n和m表示两个多项式的次数
11 //a[i].x和b[i].x表示第i项
12 //结果保存在a[i].x中， 注意是浮点数， 需要取整输出
13 const int N = 1<<20;
14 // #define double long double
15 const double pi = acos(-1);
16 typedef vector<ll> poly;
17 //根据精度判断double还是long double
18 //long double可能会导致速度变慢
19 //#define double long double
20 struct com{//复数
21     com(double xx=0, double yy=0){x=xx; y=yy;}
22     double x, y;
23     com operator +(com const &b) const{
24         return com(x+b.x, y+b.y);
25     }
26     com operator -(com const &b) const{
27         return com(x-b.x, y-b.y);
28     }
29     com operator *(com const &b) const {
30         return com(x*b.x-y*b.y, x*b.y+b.x*y);
31     }
32     com conj(){return com(x, -y);}
33 }A[N<<1], B[N<<1], rev[N<<1];
34
35 const int L = N<<1;
36 inline void fft_init(){
37     for(int i=0; i<N<<1; i++)
38         rev[i] = com(cos(2*i*pi/L), sin(2*i*pi/L));
39 }
40
41 void fft(com *f, int n, int flag){
42     for(int i=(n>>1), j=1; j<n; j++){
43         if(i<j) swap(f[i], f[j]);
44         int k = (n>>1);

```

```

45         while(i&k){i^=k; k>>=1;}
46         i^=k;// 蝴蝶变换
47     }
48     for(int k=2; k<=n; k<=<1){
49         // com rt(cos(2*pi/k), sin(2*pi/k)*flag);// 单位根
50         //dft && idft
51         int rt = L/k;
52         for(int i=0; i<n; i+=k){
53             int w = 0;
54             for(int j=i; j<(i+(k>>1)); j++){
55                 com u = f[j], v = f[j+(k>>1)] * (flag == 1?rev[w]:r[w]);
56                 f[j] = u+v;
57                 f[j+(k>>1)] = u-v;
58                 w = w + rt;
59             }
60         }
61     }
62     if(flag == -1){//idft
63         for(int i=0; i<n; i++)
64             f[i].x=f[i].x/n;
65     }
66 }
67
68 poly operator*(poly a, poly b){
69     int n = a.size() - 1, m = b.size() - 1;
70     for(m+=n, n=1; n<=m; n<=<1);
71     a.resize(n); b.resize(n);
72     for(int i=0; i<n; i++)
73         A[i] = com(a[i], 0),
74         B[i] = com(b[i], 0);
75     fft(A, n, 1); fft(B, n, 1);
76     for(int i=0; i<n; i++)
77         A[i] = A[i] * B[i];
78     fft(A, n, -1);
79     poly c(m+1);
80     for(int i=0; i<=m; i++){
81         if(A[i].x > 0)
82             c[i] = (ll)(A[i].x + 0.5);
83         else
84             c[i] = (ll)(A[i].x - 0.5);
85     }
86     return c;
87 }
88

```

```

89 int main(){
90     // freopen("0.in","r",stdin);
91     ios::sync_with_stdio(false);
92     cin.tie(0);
93     int n, m;
94     cin >> n >> m;
95     fft_init();
96     poly a(n+1), b(m+1);
97     for(int i=0; i<=n; i++)
98         cin >> a[i];
99     for(int i=0; i<=m; i++)
100         cin >> b[i];
101     a = a * b;
102     for(int i=0; i<=n+m; i++)
103         cout << a[i] << " ";
104 }

```

1.5.2 单位根反演

1. $[n|a] = \frac{1}{n} \sum_{k=0}^n \omega_n^{ak}$
2. $[a = b \pmod n] = [a - b = 0 \pmod n] = [n|(a - b)] = \frac{1}{n} \sum_{i=0}^n -1 \omega_n^{(a-b)i} = \frac{1}{n} \sum_{i=0}^{n-1} \omega_n^{ai} \omega_n^{-bi}$

1.5.3 NTT

跟 FFT 同理，只是用原根 $G^{\frac{P-1}{n}}$ 来代替 w_n^1

```

1 //#pragma GCC optimize(2)
2 //#pragma comment(linker, "/STACK:102400000,102400000")
3 #include<bits/stdc++.h>
4 using namespace std;
5 typedef long long ll;
6 typedef unsigned long long ull;
7 typedef pair<int,int> pii;
8 #define pb push_back
9 #define ln '\n'
10
11 const int g=3;
12 const int mod = 998244353;
13
14 inline void inc(int &a, int b){
15     a+=b;
16     if(a>=mod) a-=mod;
17     if(a<0) a+=mod;
18 }
19 inline void dec(int &a, int b){

```

```

20     a-=b;
21     if(a<0) a+=mod;
22     if(a>=mod) a-=mod;
23 }
24
25 inline int power(int a, int b){
26     int res = 1;
27     for(; b; b>>=1, a=1ll*a*a%mod)
28         if(b&1)
29             res=1ll*res*a%mod;
30     return res;
31 }
32
33 typedef vector<int> poly;
34 void ntt(int *a, int n, int flag){
35     for(int i=(n>>1), j=1; j<n; j++){
36         if(i<j) swap(a[i], a[j]);
37         int k = (n>>1);
38         while(i&k){i^=k; k>>=1;}
39         i^=k;
40     }
41     for(int k=2; k<=n; k<=<1){
42         int rt = power(g, (mod-1)/k);
43         if(flag == -1)
44             rt = power(rt, mod-2);
45         for(int i=0; i<n; i+=k){
46             int del = 1;
47             for(int j=i; j<i+k/2; j++){
48                 int u = a[j], v = 1ll * del * a[j+k/2] % mod;
49                 a[j] = (u + v) % mod;
50                 a[j+k/2] = (u - v + mod) % mod;
51                 del = 1ll * del * rt % mod;
52             }
53         }
54     }
55     if(flag == -1){
56         int inv = power(n, mod-2);
57         for(int i=0; i<n; i++)
58             a[i] = 1ll * a[i] * inv % mod;
59     }
60 }
61
62 poly operator*(poly a, poly b){
63     int m = b.size(), n = a.size();

```



```

64     for(m+=n, n=1; n<=m; n<=1);
65     a.resize(n); b.resize(n);
66     ntt(a.data(), n, 1); ntt(b.data(), n, 1);
67     for(int i=0; i<n; i++)
68         a[i] = 1ll * a[i] * b[i] % mod;
69     ntt(a.data(), n, -1);
70     a.resize(m-1);
71     return a;
72 }
73
74 poly operator*(poly a, int b){
75     int m = a.size();
76     for(int i=0; i<m; i++)
77         a[i] = 1ll * a[i] * b % mod;
78     return a;
79 }
80
81 poly operator*(int a, poly b){
82     int m = b.size();
83     for(int i=0; i<m; i++)
84         b[i] = 1ll * b[i] * a % mod;
85     return b;
86 }
87
88 poly operator+(poly a, poly b){
89     int m = b.size(), n = a.size();
90     a.resize(max(n, m));
91     for(int i=0; i<b.size(); i++)
92         inc(a[i], b[i]);
93     return a;
94 }
95
96 poly operator-(poly a, poly b){
97     int m = b.size(), n = a.size();
98     a.resize(max(n, m));
99     for(int i=0; i<b.size(); i++)
100         dec(a[i], b[i]);
101     return a;
102 }
103
104 void modxk(poly &a, int k){
105     if(a.size() > k) a.resize(k);
106 }
107

```

```

108 poly Inv(poly &f){
109     poly R{power(f[0], mod-2)};
110     for(int i = 1; i<f.size(); i<=1)
111         R = 2*R-R*R*f, modxk(R, i<1);
112     return R;
113 }
114
115 int main(){
116     ios::sync_with_stdio(false);
117     cin.tie(0);
118     int n;
119     cin >> n;
120     poly f(n);
121     for(int i=0; i<n; i++)
122         cin >> f[i];
123     f = Inv(f);
124     // f = inv(f);
125     for(int i=0; i<n; i++)
126         cout << f[i] << " ";
127     cout << ln;
128 }

```

```

1 namespace polybase {
2     constexpr int mod(998244353), G(3), L(1 << 20);
3     //L(20): 长度, 根据题目需要改
4     //g(3): 原根, 根据模数改
5     //mod(998244353): ntt经典模数
6
7     inline void inc(int &x, int y)
8     {
9         x += y;
10        if (x >= mod) x -= mod;
11        if (x < 0) x += mod;
12    }
13
14    inline void dec(int &x, int y)
15    {
16        x -= y;
17        if (x < 0) x += mod;
18        if (x >= mod) x -= mod;
19    }
20
21    int fpow(int x, int k = mod - 2)
22    {
23        int r = 1;

```

```

24     for (; k; k >= 1, x = 1LL * x * x % mod)
25     {
26         if (k & 1) r = 1LL * r * x % mod;
27     }
28     return r;
29 }
30
31 int w[L], _ = []
32 {
33     w[L / 2] = 1;
34     for (int i = L / 2 + 1,
35         x = fpow(G, (mod - 1) / L); i < L; i++)
36         w[i] = 1LL * w[i - 1] * x % mod;
37     for (int i = L / 2 - 1; i >= 0; i--)
38         w[i] = w[i << 1];
39     return 0;
40 }();
41
42 void dft(int *a, int n)
43 {
44     assert((n & n - 1) == 0);
45     for (int k = n >> 1; k; k >>= 1)
46     {
47         for (int i = 0; i < n; i += k << 1)
48         {
49             for (int j = 0; j < k; j++)
50             {
51                 int &x = a[i + j], y = a[i + j + k];
52                 a[i + j + k] = 1LL *
53                     (x - y + mod) * w[k + j] % mod;
54                 inc(x, y);
55             }
56         }
57     }
58 }
59
60 void idft(int *a, int n)
61 {
62     assert((n & n - 1) == 0);
63     for (int k = 1; k < n; k <= 1)
64     {
65         for (int i = 0; i < n; i += k << 1)
66         {
67             for (int j = 0; j < k; j++)

```

```

68         {
69             int x = a[i + j], y = 1LL *
70                 a[i + j + k] * w[k + j] % mod;
71             a[i + j + k] =
72                 x - y < 0 ? x - y + mod : x - y;
73             inc(a[i + j], y);
74         }
75     }
76 }
77 for (int i = 0, inv =
78     mod - (mod - 1) / n; i < n; i++)
79     a[i] = 1LL * a[i] * inv % mod;
80 std::reverse(a + 1, a + n);
81 }
82
83 inline int norm(int n) { return 1 << std::__lg(n * 2 - 1); }
84
85 struct poly : public std::vector<int>
86 {
87     #define T (*this)
88     using std::vector<int>::vector;
89
90     void append(const poly &r)
91     {
92         insert(end(), r.begin(), r.end());
93     }
94
95     int len() const { return size(); }
96
97     poly operator-() const
98     {
99         poly r(T);
100         for (auto &x : r) x = x ? mod - x : 0;
101         return r;
102     }
103
104     poly &operator+=(const poly &r)
105     {
106         if (r.len() > len()) resize(r.len());
107         for (int i = 0; i < r.len(); i++) inc(T[i], r[i]);
108         return T;
109     }
110
111     poly &operator-=(const poly &r)

```

```

112 {
113     if (r.len() > len()) resize(r.len());
114     for (int i = 0; i < r.len(); i++) dec(T[i], r[i]);
115     return T;
116 }
117
118 poly &operator^=(const poly &r)
119 {
120     if (r.len() < len()) resize(r.len());
121     for (int i = 0; i < len(); i++)
122         T[i] = 1LL * T[i] * r[i] % mod;
123     return T;
124 }
125
126 poly &operator*=(int r)
127 {
128     for (int &x : T) x = 1LL * x * r % mod;
129     return T;
130 }
131
132 poly operator+(const poly &r) const { return poly(T) += r; }
133
134 poly operator-(const poly &r) const { return poly(T) -= r; }
135
136 poly operator^(const poly &r) const { return poly(T) ^= r; }
137
138 poly operator*(int r) const { return poly(T) *= r; }
139
140 poly &operator<=<=(int k) { return insert(begin(), k, 0), T; }
141
142 poly operator<<(int r) const { return poly(T) <<= r; }
143
144 poly operator>>(int r) const {
145     return r >= len() ? poly() : poly(begin() + r, end()); }
146
147 poly &operator>>=(int r) { return T = T >> r; }
148
149 poly pre(int k) const {
150     return k < len() ? poly(begin(), begin() + k) : T; }
151
152 friend void dft(poly &a) { dft(a.data(), a.len()); }
153
154 friend void idft(poly &a) { idft(a.data(), a.len()); }
155

```

```

156 friend poly conv(const poly &a, const poly &b, int n)
157 {
158     poly p(a), q;
159     p.resize(n), dft(p);
160     p ^= &a == &b ? p : (q = b, q.resize(n), dft(q), q);
161     idft(p);
162     return p;
163 }
164
165 friend poly operator*(const poly &a, const poly &b)
166 {
167     int len = a.len() + b.len() - 1;
168     if (a.len() <= 16 || b.len() <= 16)
169     {
170         poly c(len);
171         for (int i = 0; i < a.len(); i++)
172             for (int j = 0; j < b.len(); j++)
173                 c[i + j] = (c[i + j]
174                     + 1LL * a[i] * b[j] % mod) % mod;
175         return c;
176     }
177     return conv(a, b, norm(len)).pre(len);
178 }
179
180 poly deriv() const
181 {
182     if (empty()) return poly();
183     poly r(len() - 1);
184     for (int i = 1; i < len(); i++)
185         r[i - 1] = 1LL * i * T[i] % mod;
186     return r;
187 }
188
189 poly integ() const
190 {
191     if (empty()) return poly();
192     poly r(len() + 1);
193     for (int i = 0; i < len(); i++)
194         r[i + 1] = 1LL * fpow(i + 1) * T[i] % mod;
195     return r;
196 }
197
198 poly inv(int m) const
199 {

```

```

200     poly x = {fpow(T[0])};
201     for (int k = 1; k < m; k *= 2)
202     {
203         x.append(-((conv(pre(k * 2),
204             x, k * 2) >> k) * x).pre(k));
205     }
206     return x.pre(m);
207 }
208
209 poly log(int m) const { return (deriv()
210     * inv(m)).integ().pre(m); }
211
212 poly exp(int m) const
213 {
214     poly x = {1};
215     for (int k = 1; k < m; k *= 2)
216     {
217         x.append((x * (pre(k * 2) -
218             x.log(k * 2) >> k)).pre(k));
219     }
220     return x.pre(m);
221 }
222
223 poly sqrt(int m) const
224 {
225     poly x = {1}, y = {1};
226     for (int k = 1; k < m; k *= 2)
227     {
228         x.append(((pre(k * 2)
229             - x * x >> k) * y).pre(k) * (mod + 1 >> 1));
230         if (k * 2 < m)
231         {
232             y.append(-((conv(x.pre(k * 2),
233                 y, k * 2) >> k) * y).pre(k));
234         }
235     }
236     return x.pre(m);
237 }
238
239 poly rev() const { return poly(rbegin(), rend()); }
240
241 poly mulT(poly b) { return T * b.rev() >> b.len() - 1; }
242
243 #undef T

```

```

244     };
245 }

```

1.5.4 拆系数 FFT

```

1  // #define double long double
2  // #define ll __int128
3  //精度不够的时候要开int128和long double
4  //空间要开4倍 多项式长度2n, 扩域最多到4n
5  const int N = 1<<20;
6  typedef vector<ll> poly;
7  typedef complex<double> Complex;
8  const double pi = acos(-1);
9  const int L = 15, MASK = (1<<L) - 1;
10 Complex rev[N+7];
11 void FFT_init(int n){
12     for(int i=0; i<n; i++){
13         rev[i] = Complex(cos(2*i*pi/n), sin(2*i*pi/n));
14     }
15 void FFT(Complex *p, int n){
16     for(int i=1, j=0; i<n-1; i++){
17         for(int s=n; j^=s>>=1, ~j&s;);
18         if(i<j) swap(p[i], p[j]);
19         //依旧是蝴蝶变换, 没有什么不一样
20     }
21     for(int d=0; (1<<d)<n; d++){
22         int m = 1<<d, m2 = m<<1, rm = n >> (d+1);
23         for(int i=0; i<n; i+=m2){
24             for(int j=0; j<m; j++){
25                 Complex &p1 = p[i+j+m], &p2 = p[i+j];
26                 Complex t = rev[rm*j]*p1;
27                 p1 = p2 - t; p2 = p2 + t;
28             }
29         }
30     }
31 Complex A[N+7], B[N+7], C[N+7], D[N+7];
32 //拆系数fft || 任意模数ntt
33 poly operator*(poly &a, poly &b){
34     int n = a.size()-1, m = b.size()-1;
35     for(m+=n, n=1; n<=m; n<<=1);
36     FFT_init(n);
37     a.resize(n); b.resize(n);
38     for(int i=0; i<n; i++){
39         A[i] = Complex(a[i]>>L, a[i]&MASK);

```



```

40         B[i] = Complex(b[i]>>L, b[i]&MASK); // 拆成两部分
41     }
42     FFT(A, n); FFT(B, n);
43     for(int i=0; i<n; i++){
44         int j = (n-i)%n;
45         // conj 表示返回一个 complex 的共轭
46         // 压缩值域, 前一半和后一半
47         Complex da = (A[i] - conj(A[j])) * Complex(0, -0.5), // 实部
48                 db = (A[i] + conj(A[j])) * Complex(0.5, 0), // 虚部
49                 dc = (B[i] - conj(B[j])) * Complex(0, -0.5),
50                 dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
51         C[j] = da*dd+da*dc*Complex(0, 1); // 进行复合
52         D[j] = db*dd+db*dc*Complex(0, 1);
53     }
54     FFT(C, n); FFT(D, n);
55     poly res(n);
56     for(int i=0; i<n; i++){
57         ll da = (ll)(C[i].imag()/n+0.5) % mod,
58             db = (ll)(C[i].real()/n+0.5) % mod,
59             dc = (ll)(D[i].imag()/n+0.5) % mod,
60             dd = (ll)(D[i].real()/n+0.5) % mod;
61         res[i] = (((dd<<(L<<1))%mod + ((db+dc)<<L)%mod)%mod + da) % mod;
62     }
63     res.resize(m+1);
64     return res;
65 }

```

1.5.5 拉格朗日插值

一个 n 次多项式可以由 $n+1$ 个点唯一确定

$$F(x) = \sum_{k=1}^{n+1} a_k x^k$$

$$F(x) \equiv F(a) \pmod{(x-a)}$$

$$F(x) \equiv y_i \pmod{(x-x_i)} \quad i \in [0, n]$$

可以用中国剩余定理求出 $F(x)$, 复杂度为 $O(N^2)$

$$M = \prod (x - x_i), \quad m_i = \prod_{i \neq j} (x - x_j)$$

$$m_i^{-1} = \prod_{i \neq j} (x_i - x_j) \pmod{(x - x_i)}$$

$$F(x) = \sum_{i=1}^{n+1} y_i \cdot \frac{m_i}{m_i^{-1}} \pmod{M}$$

当 x_i 连续的时候，上式可以变成

$$F(x) = \sum_{i=1}^{n+1} y_i \cdot \frac{\prod_{i \neq j} (x - j)}{\prod_{i \neq j} (i - j)} \pmod{M}$$

其中，分子的部分可以变成 $\frac{\prod_{i \neq j} (x - j)}{x - i}$ ，可以预处理

分母的部分可以拆成两段阶乘的乘积 $(i - 1)!(n + 1 - i)!(-1)^{n+1-i}$

可以在 $O(n)$ 的时间内求出 $F(x)$

```

1 // #pragma GCC optimize(2)
2 // #pragma comment(linker, "/STACK:102400000,102400000")
3 #include <bits/stdc++.h>
4 using namespace std;
5 typedef long long ll;
6 typedef unsigned long long ull;
7 typedef pair<int, int> pii;
8 #define pb push_back
9 #define ln '\n'
10
11 const int mod = 998244353;
12 typedef vector<int> poly;
13 inline int power(int a, int b){
14     int res = 1;
15     for(; b; b >>= 1, a = 1ll * a * a % mod)
16         if(b & 1)
17             res = 1ll * res * a % mod;
18     return res;
19 }
20
21 inline int Inv(int x){return power(x, mod - 2);}
22 // 如果 mod 非质数需要换成 exgcd
23
24 poly Lagrange(const poly &x, const poly &y){ // 插系数, x 不连续, y 连续
25     int n = x.size(); // n 个点确定一个 n+1 次多项式
26     poly a(n, 0), b(n + 1, 0), c(n + 1, 0), f(n, 0);
27     for(int i = 0; i < n; i++){
28         int A = 1;
29         for(int j = 0; j < n; j++){
30             if(i != j)
31                 A = 1ll * A * (x[i] - x[j] + mod) % mod;
32             // crt
33             a[i] = 1ll * Inv(A) * y[i] % mod;
34         }
35         b[0] = 1;
36         for(int i = 0; i < n; i++){
37             for(int j = i + 1; j < n; j++){

```

```

38         b[j] = (1ll * b[j] * (mod - x[i]) % mod + b[j-1]) % mod;
39         b[0] = 1ll * b[0] * (mod - x[i]) % mod;
40     }
41     for(int i = 0; i < n; i++){
42         int inv = Inv(mod - x[i]);
43         if(inv){
44             c[0] = 1ll * b[0] * inv % mod;
45             for(int j = 1; j <= n; j++)
46                 c[j] = 1ll * (b[j] + mod - c[j-1]) * inv % mod;
47         } else {
48             for(int j = 0; j < n; j++)
49                 c[j] = b[j + 1];
50         }
51         for(int j = 0; j < n; j++)
52             f[j] = (f[j] + 1ll * a[i] * c[j] % mod) % mod;
53     }
54     return f;
55 }
56
57 inline int calc(int x, const poly &a){
58     int ans = 0;
59     int n = a.size(); --n;
60     for(; ~n; n--)
61         ans = (1ll * ans * x % mod + a[n]) % mod;
62     return ans;
63 }
64
65 int main(){
66     ios::sync_with_stdio(false);
67     cin.tie(0);
68
69 }

```

```

1  // CF622F The Sum of the k-th Powers
2  // 给n, k, 求  $\sigma i^k$ 
3  // 注意到答案是一个k+1次的多项式, 线性筛出1..k+2的值然后插值即可
4  #include <bits/stdc++.h>
5  using namespace std;
6  const int N = 1e6 + 5, mod = 1e9 + 7;
7
8  int n, k, tab[N], p[N], pcnt, f[N], pre[N], suf[N], fac[N], inv[N], ans;
9
10 int qpow(int x, int y) {
11     int ans = 1;
12     for (; y; y >>= 1, x = 1LL * x * x % mod)

```

```

13     if (y & 1) ans = 1LL * ans * x % mod;
14     return ans;
15 }
16
17 void sieve(int lim) {
18     f[1] = 1;
19     for (int i = 2; i <= lim; i++) {
20         if (!tab[i]) {
21             p[++pcnt] = i;
22             f[i] = qpow(i, k);
23         }
24         for (int j = 1; j <= pcnt && 1LL * i * p[j] <= lim; j++) {
25             tab[i * p[j]] = 1;
26             f[i * p[j]] = 1LL * f[i] * f[p[j]] % mod;
27             if (!(i % p[j])) break;
28         }
29     }
30     for (int i = 2; i <= lim; i++) f[i] = (f[i - 1] + f[i]) % mod;
31 }
32
33 int main() {
34     scanf("%d%d", &n, &k);
35     sieve(k + 2);
36     if (n <= k + 2) return printf("%d\n", f[n]) & 0;
37     pre[0] = suf[k + 3] = 1;
38     for (int i = 1; i <= k + 2; i++) pre[i] = 1LL * pre[i - 1] * (n - i) % mod;
39     for (int i = k + 2; i >= 1; i--) suf[i] = 1LL * suf[i + 1] * (n - i) % mod;
40     fac[0] = inv[0] = fac[1] = inv[1] = 1;
41     for (int i = 2; i <= k + 2; i++) {
42         fac[i] = 1LL * fac[i - 1] * i % mod;
43         inv[i] = 1LL * (mod - mod / i) * inv[mod % i] % mod;
44     }
45     for (int i = 2; i <= k + 2; i++) inv[i] = 1LL * inv[i - 1] * inv[i] % mod;
46     for (int i = 1; i <= k + 2; i++) {
47         int P = 1LL * pre[i - 1] * suf[i + 1] % mod;
48         int Q = 1LL * inv[i - 1] * inv[k + 2 - i] % mod;
49         int mul = ((k + 2 - i) & 1) ? -1 : 1;
50         ans = (ans + 1LL * (Q * mul + mod) % mod * P % mod * f[i] % mod) % mod;
51     }
52     printf("%d\n", ans);
53     return 0;
54 }

```

1.6 同余

1.6.1 BSGS

1.6.2 原根

1. 概念及定理

- 阶 (即幂的最小循环节): a 在模 p 意义下的阶是: 最小的整数 k , 使得 $a^k \equiv 1(\text{mod } p)$, 记作 $\delta_p(a)$

性质 1. : 设 $p \geq 1, \gcd(a, p) = 1$, 若 $a^n \equiv 1(\text{mod } p), n > 0$, 则 $\delta_p(a) | n$

性质 2. : 设 $\delta_p(a) | \phi(p)$

推论 1. : 若 p, q 均为奇质数, 且 $q | (a^p - 1)$ 则有 $q | (a - 1)$ 或 $2p | (q - 1)$

推论 2. : 原根 (满足 $\delta_p(g) = \phi(p)$ 的 g , 也即 $g^{\phi(p)} \equiv 1(\text{mod } p)$)

1.7 筛法

1.7.1 常用积性函数关系

0. 若没有声明 $*$ 表示狄利克雷卷积

1. $\epsilon = \mu * I$

2. $id = \varphi * I$

3. $\sum_{i=1}^n \frac{\mu(i)}{i} = \frac{\phi(n)}{n}$

1.7.2 PN 筛

1. $f = g * h$

2. g 的前缀和 G 好求

3. $f(p) = g(p)$, 此时 $h(p) = 0, h(1) = 1$

4. 有用的值即 $h(n) > 1$ 的 n 只有 $O(\sqrt{n})$ 个

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

1.7.3 杜教筛

1. $f * g$

2. 直接暴力是 $O(n^{\frac{3}{4}})$ 的

3. 预处理前 $O(n^{\frac{2}{3}})$ 项复杂度可以到 $O(n^{\frac{2}{3}})$, 这个东西被叫做阈值分治

4. 如果需要多次递归可以先递归一次然后处理出每个需要的元素整除分块对应的根号个元素

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

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

$$\sum_{i=1}^n (f * g)(i) = \sum_{d=1}^n g(d) S(\lfloor \frac{n}{d} \rfloor)$$

$$g(1)S(n) = \sum_{i=1}^n (f * g)(i) - \sum_{d=2}^n g(d) S(\lfloor \frac{n}{d} \rfloor)$$

1.7.4 min25 筛 (待填)

2 数据结构

2.1 单调队列

```
1 //51nod1275 连续子段的差异
2 // 求满足最大值 - 最小值 <= k 的区间有多少个
3 #include <bits/stdc++.h>
4 using namespace std;
5 #define ll long long
6 #define pb push_back
7
8 inline ll read() {
9     ll x = 0, f = 1; char ch = getchar();
10    for(; ch < '0' || ch > '9'; ch = getchar())
11        if(ch == '-') f = -f;
12    for(; ch >= '0' && ch <= '9'; ch = getchar())
13        x = x * 10 + ch - '0';
14    return x * f;
15 }
16
17 #define ln endl
18
19 const int N = 5e4+5;
20 int n, k, a[N], q[2][N], l[2], r[2];
21 ll ans;
22
23 int main(){
24     #ifndef ONLINE_JUDGE
25         freopen("0.txt", "w", stdout);
26     #endif
27     n=read(); k=read();
28     l[1] = l[0] = r[1] = r[0] = 1;
29     for(int i=1; i<=n; i++)
30         a[i] = read();
31     int j = 1;
32     q[1][l[1]] = q[0][l[0]] = 1;
33     for(int i=1; i<=n; i++){//以i为左端点
34         while(l[1] <= r[1] && q[1][l[1]] < i)
35             l[1]++;
36         while(l[0] <= r[0] && q[0][l[0]] < i)
37             l[0]++;
38         //把i往左的点都排掉
39         while(j<=n && a[q[1][l[1]]] - a[q[0][l[0]]] <= k) {
40             ++j;
41             while(l[1] <= r[1] && a[q[1][r[1]]] <= a[j])
```

```

42         --r[1];
43         q[1][++r[1]] = j;
44         while(l[0] <= r[0] && a[q[0][r[0]]] >= a[j])
45             --r[0];
46         q[0][++r[0]] = j;
47     }
48     ans = ans + (j-i);
49     // printf("%d %d\n", i, j);
50 }
51 printf("%lld\n", ans);
52 }

```

2.2 线段树

2.2.1 普通线段树

```

1 //这个N得根据需要改成自己需要的值域大小
2 struct segtree {
3     int l, r;
4     ll mn, mx, sum;
5 }T[N<<2];
6 int tag[N<<2];
7 segtree operator +(segtree ls, segtree rs){
8     segtree rt;
9     rt.l = ls.l; rt.r = rs.r;
10    rt.mx = max(ls.mx, rs.mx);
11    rt.mn = min(ls.mn, rs.mn);
12    rt.sum = ls.sum + rs.sum;
13    return rt;
14 }
15 void pushup(int rt){
16     T[rt] = T[rt<<1] + T[rt<<1|1];
17 }
18 void pushdown(int rt){
19     if(tag[rt]){
20         segtree &ls = T[rt<<1], &rs = T[rt<<1|1];
21         ls.sum += 1ll*tag[rt]*(ls.r-ls.l+1);
22         rs.sum += 1ll*tag[rt]*(rs.r-rs.l+1);
23         ls.mx += tag[rt]; rs.mx += tag[rt];
24         ls.mn += tag[rt]; rs.mn += tag[rt];
25         tag[rt<<1] += tag[rt];
26         tag[rt<<1|1] += tag[rt];
27         tag[rt] = 0;
28     }
29 }

```



```

30 void build(int l, int r, int rt){
31     tag[rt] = 0;
32     if(l == r){
33         T[rt] = (segtree){l, l, 0, 0, 0};
34         return;
35     }
36     int mid = l+r >> 1;
37     build(l, mid, rt<<1);
38     build(mid+1, r, rt<<1|1);
39     pushup(rt);
40 }
41 void add(int L, int R, int v, int rt){
42     int l = T[rt].l, r = T[rt].r;
43     if(L <= l && r <= R)
44         return T[rt].sum += 1ll*(r-l+1)*v, T[rt].mx += v,
45             T[rt].mn += v, tag[rt]+=v, void();
46     int mid = l+r>>1;
47     pushdown(rt);
48     if(L <= mid)
49         add(L, R, v, rt<<1);
50     if(R > mid)
51         add(L, R, v, rt<<1|1);
52     pushup(rt);
53 }
54 segtree query(int L, int R, int rt){
55     int l = T[rt].l, r = T[rt].r;
56     if(L <= l && r <= R)
57         return T[rt];
58     int mid = l+r>>1;
59     pushdown(rt);
60     if(L <= mid && R > mid)
61         return query(L, R, rt<<1) + query(L, R, rt<<1|1);
62     else if(L <= mid)
63         return query(L, R, rt<<1);
64     else
65         return query(L, R, rt<<1|1);
66 }

```

2.2.2 动态开点线段树

```

1  /*
2     维护一段递增的序列
3     一开始是a[i]=i
4     在线段树上二分， 找到最右的一个 <T的数 和 最左的一个 >T的数

```

```

5      找数需要的点最多 2e5+2e5 个
6      所以总共最多需要 6e5 个点
7      区间修改
8      单点查询需要的节点最多 2e5 个
9      空间开 ki*4 即可
10     单个点最多需要 logN 的深度， 也就是
11 */
12 #include<bits/stdc++.h>
13 using namespace std;
14
15 #define ll long long
16
17 inline ll read() {
18     ll x = 0, f = 1; char ch = getchar();
19     for(; ch < '0' || ch > '9'; ch = getchar())
20         if(ch == '-') f = -f;
21     for(; ch >= '0' && ch <= '9'; ch = getchar())
22         x = x * 10 + ch - '0';
23     return x * f;
24 }
25
26 inline void chkmin( int &a, int b ) { if(a > b) a = b; }
27
28 inline void chkmax( int &a, int b ) { if(a < b) a = b; }
29
30 #define _ read()
31
32 #define ln endl
33
34 const int N=6e5+5;
35
36 int cnt;
37 struct node{
38     int l, r, lson, rson, mn, mx, lzy;
39 }T[N<5];
40
41 void up(int rt){
42     T[rt].mn=min(T[T[rt].lson].mn, T[T[rt].rson].mn);
43     T[rt].mx=max(T[T[rt].lson].mx, T[T[rt].rson].mx);
44 }
45
46 void down(int rt){
47     // if(T[rt].lzy){
48     if(!T[rt].lson)

```

```

49         T[rt].lson=++cnt,
50         T[cnt]=(node){T[rt].l, T[rt].l+T[rt].r>>1, 0, 0, T[rt].l, T[rt].l+T[rt].r>>1, 0};
51     if(!T[rt].rson)
52         T[rt].rson=++cnt,
53         T[cnt]=(node){(T[rt].l+T[rt].r>>1)+1, T[rt].r, 0, 0, (T[rt].l+T[rt].r>>1)+1, T[rt].r, 0};
54     T[T[rt].lson].mx+=T[rt].lzy;
55     T[T[rt].rson].mx+=T[rt].lzy;
56     T[T[rt].lson].mn+=T[rt].lzy;
57     T[T[rt].rson].mn+=T[rt].lzy;
58     T[T[rt].lson].lzy+=T[rt].lzy;
59     T[T[rt].rson].lzy+=T[rt].lzy;
60     T[rt].lzy=0;
61     // }
62 }
63
64 void add(int L, int R, int v, int l, int r, int rt){// 区间加法
65     if(!rt)
66         T[rt=++cnt]=(node){l, r, 0, 0, l, r, 0};
67     if(L<=l&&R<=r){
68         T[rt].mx+=v, T[rt].mn+=v, T[rt].lzy+=v;
69         return;
70     }
71     int mid=l+r>>1;
72     down(rt);
73     if(L<=mid)
74         add(L, R, v, l, mid, T[rt].lson);
75     if(R>mid)
76         add(L, R, v, mid+1, r, T[rt].rson);
77     up(rt);
78 }
79
80 int queryl(int x, int l, int r, int rt){// 二分找到=r的点
81     if(l==r)
82         return l-(T[rt].mn>=x);
83     int mid=l+r>>1;
84     down(rt);
85     if(T[T[rt].rson].mn<x)// 右子树中有 <x的点
86         return queryl(x, mid+1, r, T[rt].rson);
87     else
88         return queryl(x, l, mid, T[rt].lson);
89 }
90
91 int queryr(int x, int l, int r, int rt){
92     if(l==r)

```

```

93         return l+(T[rt].mx<=x); // 如果最右边的比x小的点都 >=x
94     int mid=l+r>>1;
95     down(rt);
96     if(T[T[rt].lson].mx>x) // 左子树中有 >x的点
97         return queryr(x, l, mid, T[rt].lson);
98     else
99         return queryr(x, mid+1, r, T[rt].rson);
100 }
101
102 int query(int x, int l, int r, int rt){
103     if(l==r)
104         return T[rt].mx;
105     int mid=l+r>>1;
106     down(rt);
107     if(x<=mid)
108         return query(x, l, mid, T[rt].lson);
109     else
110         return query(x, mid+1, r, T[rt].rson);
111 }
112
113
114 int main(){
115     #ifndef ONLINE_JUDGE
116         freopen("0.out","w", stdout);
117     #endif
118     int n=read();
119     T[cnt=1]=(node){0, (int)1e9, 0, 0, 0, (int)1e9, 0};
120     int lastans=0;
121     while(n--){
122         int Ti=read();
123         int L=queryl(Ti, 0, (int)1e9, 1), R=queryr(Ti, 0, (int)1e9, 1);
124         // printf("%d %d\n", L, R);
125         if(L>=0)
126             add(0, L, 1, 0, (int)1e9, 1);
127         if(R<=(int)1e9)
128             add(R, (int)1e9, -1, 0, (int)1e9, 1);
129         int m=read();
130         while(m--){
131             int x=(read()+lastans)%((int)1e9+1); //x=x+lastans;
132             // printf("%d ", x);
133             printf("%d\n", lastans=query(x, 0, (int)1e9, 1));
134         }
135     }
136 }

```

```

137  /*
138     值域是0..1e9
139  */

```

2.2.3 主席树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  #define ll long long
5
6  inline ll read() {
7      ll x = 0, f = 1; char ch = getchar();
8      for(; ch < '0' || ch > '9'; ch = getchar())
9          if(ch == '-') f = -f;
10     for(; ch >= '0' && ch <= '9'; ch = getchar())
11         x = x * 10 + ch - '0';
12     return x * f;
13 }
14
15 inline void chkmin( int &a, int b ) { if(a > b) a = b; }
16
17 inline void chkmax( int &a, int b ) { if(a < b) a = b; }
18
19 #define _ read()
20
21 #define ln endl
22
23 const int N = 2e5 + 5;
24 int n, m, T[N];
25 int ls[N << 5], rs[N << 5], siz[N << 5], cnt;
26
27 inline void insert( int v, int l, int r, int x, int &y)
28 {
29     y = ++cnt; siz[y] = siz[x] + 1;
30     if(l == r) return;
31     int mid = (l + r) >> 1;
32     ls[y] = ls[x]; rs[y] = rs[x];
33     if(v <= mid)
34         insert(v, l, mid, ls[x], ls[y]);
35     else
36         insert(v, mid + 1, r, rs[x], rs[y]);
37 }
38

```

```

39 inline int ask( int k, int l, int r, int x, int y)
40 {
41     if(k > siz[y] - siz[x] + 1)
42         return 0;
43     if(l == r)
44         return l;
45     int mid = (l + r) >> 1;
46     if(siz[ls[y]] - siz[ls[x]] >= k)
47         return ask(k, l, mid, ls[x], ls[y]);
48     else
49         return ask(k - siz[ls[y]] + siz[ls[x]], mid + 1, r, rs[x], rs[y]);
50 }
51
52 int main()
53 {
54     n = _; m = _;
55     for( int i = 1, x; i <= n; i++ )
56         x = _, insert(x, -1000000000, 1000000000, T[i - 1], T[i]);
57     while(m--)
58     {
59         int l = _, r = _, k = _;
60         printf("%d\n", ask(k, -1000000000, 1000000000, T[l - 1], T[r]));
61     }
62 }

```

2.3 并查集

2.3.1 普通并查集

```

1 struct DSU{
2     vector<int>fa,siz;
3     int tot;
4     DSU(int n){
5         fa.resize(n+1);
6         siz.resize(n+1);
7         for(int i=1; i<=n; i++)
8             fa[i] = i, siz[i] = 1;
9         tot = n;
10    }
11    int find(int x){return fa[x]==x?x:fa[x] = find(fa[x]);}
12    bool merge(int x, int y){
13        x = find(x); y = find(y);
14        if(x == y) return false;
15        if(siz[x] > siz[y])
16            swap(x, y);

```

```

17         fa[x] = y;
18         siz[y] += siz[x];
19         tot--;
20         return true;
21     }
22 };

```

2.3.2 可删除并查集

```

1 // 动态维护无向图的连通性
2 int f[N], siz[N];
3 int top;
4 pair<int,int> e[10*N];
5
6 int find(int x){
7     if(f[x] == x) return f[x];
8     int par = find(f[x]);
9     return par;
10 }
11
12 void add(int x, int y){
13     int fx = find(x), fy = find(y);
14     if(fx != fy){
15         if(siz[fy] > siz[fx])
16             siz[fy] += siz[fx], f[fx] = fy, e[++top] = {fy, fx};
17         else
18             siz[fx] += siz[fy], f[fy] = fx, e[++top] = {fx, fy};
19         // 最后一条边是谁往谁连的
20         res--;
21     } else
22         e[++top] = {0, 0}; // 已经联通那么实际上不需要加到图里
23 }
24
25 void del(){
26     // 删除最后一条边， 回复到上一次的状态
27     auto now = e[top--];
28     if(now.first) res++;
29     siz[now.first] -= siz[now.second];
30     f[now.second] = now.second;
31 }

```

3 字符串

3.1 KMP 算法

```
1 inline void getfail(char *s){
2     // 求fail函数
3     int n = strlen(s+1);
4     for(int i=0; i<=n; i++){
5         fail[i] = 0;
6         for(int i = 2, j = 0; i <= n; i++){
7             while(j && s[j+1] != s[i])
8                 j = fail[j];
9             if(s[j+1] == s[i])
10                 ++j;
11             fail[i] = j;
12         }
13 }
```

3.2 后缀数组

后缀排序

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5
6 inline ll read() {
7     ll x = 0, f = 1; char ch = getchar();
8     for(; ch < '0' || ch > '9'; ch = getchar())
9         if(ch == '-') f = -f;
10    for(; ch >= '0' && ch <= '9'; ch = getchar())
11        x = x * 10 + ch - '0';
12    return x * f;
13 }
14
15 inline void chkmin( int &a, int b ) { if(a > b) a = b; }
16
17 inline void chkmax( int &a, int b ) { if(a < b) a = b; }
18
19 #define _ read()
20
21 #define ln endl
22
23 const int N=1e6+5;
24 char s[N];
```



```

25 int n, cnt[N], sa[N], sa2[N], rk[N<<1], rk2[N<<1];
26
27 inline void getSA(){
28     for(int i=1; i<=n; i++)
29         rk[i]=s[i], sa[i]=i;
30     int m=max(n, 256);
31     for(int i=1; i<=m; i++) cnt[i]=0;
32     for(int i=1; i<=n; i++) cnt[rk[sa[i]]]++;
33     for(int i=1; i<=m; i++)
34         cnt[i]+=cnt[i-1];
35     for(int i=n; i; i--)
36         sa2[cnt[rk[sa[i]]]--]=sa[i];
37     for(int i=1; i<=n; i++)
38         sa[i]=sa2[i], sa2[i]=0;
39     for(int w=1; w<=n; w<=1){
40         for(int i=1; i<=m; i++) cnt[i]=0;
41         for(int i=1; i<=n; i++) cnt[rk[sa[i]+w]]++;
42         for(int i=1; i<=m; i++)
43             cnt[i]+=cnt[i-1];
44         for(int i=n; i; i--)
45             sa2[cnt[rk[sa[i]+w]]--]=sa[i];
46         for(int i=1; i<=n; i++)
47             sa[i]=sa2[i], sa2[i]=0;
48         for(int i=1; i<=m; i++) cnt[i]=0;
49         for(int i=1; i<=n; i++) cnt[rk[sa[i]]]++;
50         for(int i=1; i<=m; i++)
51             cnt[i]+=cnt[i-1];
52         for(int i=n; i; i--)
53             sa2[cnt[rk[sa[i]]]--]=sa[i];
54         for(int i=1; i<=n; i++)
55             sa[i]=sa2[i], sa2[i]=0;
56         int tot=0;
57         for(int i=1; i<=n; i++){
58             if(rk[sa[i]]==rk[sa[i-1]]&&rk[sa[i]+w]==rk[sa[i-1]+w])
59                 rk2[sa[i]]=tot;
60             else
61                 rk2[sa[i]]=++tot;
62         }
63         for(int i=1; i<=n; i++)
64             rk[i]=rk2[i], rk2[i]=0;
65     }
66     for(int i=1; i<=n; i++)
67         cout<<sa[i]<<" ";
68     cout<<ln;

```

```
69 }  
70  
71 int main(){  
72     scanf("%s", s+1); n=strlen(s+1);  
73     getSA();  
74 }
```

4 图论

4.1 最短路 – dij 的最长路是错的

```
1 struct dijkstra{
2     #define N 100005
3     #define M 500005
4     struct edge{
5         int to, nxt;
6         double w;
7     }e[M];
8
9     int head[N], cnt;
10    dijkstra(){clear();}
11
12    inline void clear(){
13        memset(head, 0, sizeof(head));
14        cnt = 0;
15    }
16
17    inline void insert(int u, int v, double w){
18        e[++cnt] = (edge){v, head[u], w};
19        head[u] = cnt;
20    }
21
22    struct node{
23        double dis;
24        int id;
25        bool operator <(const node &b) const{
26            return dis>b.dis;
27        }
28    };
29    priority_queue<node> q;
30    double dis[N];
31    int vis[N], n;
32
33    inline void dij(int st){
34        for(int i=0; i<=n; i++)
35            dis[i] = 2e9, vis[i] = 0;
36        dis[st] = 0;
37        q.push((node){0, st});
38        while(!q.empty()){
39            node now = q.top(); q.pop();
40            if(vis[now.id])
41                continue;
```

```

42         vis[now.id] = 1;
43         int id = now.id;
44         for(int i=head[id]; i; i=e[i].nxt)
45             if(dis[e[i].to] > now.dis + e[i].w)
46                 dis[e[i].to] = now.dis + e[i].w,
47                 q.push((node){dis[e[i].to], e[i].to});
48     }
49 }
50 };

```

4.2 二分图

4.2.1 二分图最大匹配

```

1  const int N=1e4+5;
2  const int M=1e6+N;
3  int n;
4  int vis[M], from[M];
5  vector<int> v;
6  vector<std::vector<int>> e(N);
7  bool dfs(int x){
8      for(int y:e[x])
9          if(!vis[y]){
10             vis[y]=1;
11             if(!from[y]||dfs(from[y])){
12                 from[y]=x;
13                 return true;
14             }
15         }
16     return false;
17 }

```

4.2.2 二分图最大权匹配

```

1  const int N = 505;
2  const int inf = 2e9+233;
3
4  int n, nl, nr, m, x, y, z;
5  int g[N][N];
6
7  //nl: 左部图点数
8  //nr: 右部图点数
9  //g: 二分图边权
10 namespace KM{

```

```

11     int left[N], right[N];
12     int visl[N], visr[N];
13     int lx[N], ly[N], slack[N];
14
15     bool augment(int x){
16         visl[x] = 1;
17         for(int y = 1; y <= n; y++){
18             if(visr[y]) continue;
19             else {
20                 int slk = lx[x] + ly[y] - g[x][y];
21                 if(!slk){
22                     visr[y] = 1;
23                     if(!right[y] || augment(right[y])){
24                         right[y] = x; left[x] = y;
25                         return 1;
26                     }
27                 } else
28                     slack[y] = min(slack[y], slk);
29             }
30         }
31         return 0;
32     }
33
34     void solve(){
35         for(int i=1; i<=n; i++){
36             for(int j=1; j<=n; j++){
37                 ly[j] = max(ly[j], g[i][j]);
38             }
39             for(int i=1; i<=n; i++){
40                 for(int j=1; j<=n; j++){
41                     visl[j] = visr[j] = 0, slack[j] = inf;
42                     if(augment(i)) continue;
43                     while(1){
44                         int d = inf, x;
45                         for(int j = 1; j <= n; j++){
46                             if(!visr[j]) d = min(d, slack[j]);
47                         }
48                         for(int j = 1; j <= n; j++){
49                             if(!visr[j]){
50                                 ly[j] -= d;
51                                 slack[j] -= d;
52                                 if(!slack[j]) x = j;
53                             }
54                             if(!visl[j])
55                                 lx[j] += d;
56                         }
57                     }
58                 }
59             }
60         }
61     }

```

```

55
56         if(!right[x]) break;
57         visr[x] = 1; visl[right[x]] = 1;
58         x = right[x];
59         for(int y = 1; y <= n; y++)
60             slack[y] = min(slack[y],
61                             lx[x] + ly[y] - g[x][y]);
62     }
63
64     for(int j=1; j<=n; j++)
65         visl[j] = visr[j] = 0;
66     augment(i);
67 }
68 }
69
70 void answer(){
71     ll ans = 0;
72     for(int i=1; i<=n; i++)
73         ans += lx[i] + ly[i];
74     cout << -ans << ln;
75 }
76 }

```

4.3 网络流

最大流效率 $O(N^2M)$, 在二分图上复杂度可以达到 $O(M\sqrt{N})$

```

1 //要先算好边需要多少, 点需要多少
2 //dinic复杂度为n^2m
3 template<typename T>
4 struct Dinic{
5     int s, t, tot;
6     T F;
7     const T INF = 2e9;
8     vector<int> head, d, lst;
9     struct Edge{int to, nxt; T w;};
10    vector<Edge> e;
11    Dinic(int n, int m, int st, int ed){
12        head.assign(n+1, -1);
13        e.resize(m<<1);
14        d.resize(n+1);
15        s = st; t = ed; tot = 0; F = 0;
16    }
17    inline int add(int u, int v, T w){
18        e[tot]=(Edge){v, head[u], w};

```

```

19         head[u] = tot++;
20         if(u == s) F+=w;
21         return tot-1;
22     }
23     inline int insert(int u, int v, T w){
24         int tmp = add(u, v, w);
25         add(v, u, 0);
26         return tmp;
27     }
28     inline bool bfs(){
29         d.assign(d.size(), 0);
30         lst = head;
31         queue<int> q;
32         q.push(s); d[s] = 1;
33         while(!q.empty()){
34             int u = q.front(); q.pop();
35             for(int i=head[u]; ~i; i=e[i].nxt){
36                 int v = e[i].to;
37                 if(!d[v] && e[i].w > 0){
38                     d[v] = d[u] + 1;
39                     q.push(v);
40                 }
41             }
42         }
43         return d[t] > 0;
44     }
45     int dfs(int x, int mn){
46         if(x == t) return mn;
47         for(int &i = lst[x]; ~i; i=e[i].nxt){
48             if(d[e[i].to] == d[x]+1 && e[i].w){
49                 int val = dfs(e[i].to, min(e[i].w, mn));
50                 if(val > 0){
51                     e[i].w -= val;
52                     e[i^1].w += val;
53                     return val;
54                 }
55             }
56         }
57         return 0;
58     }
59     inline T work(){
60         T ans = 0;
61         while(bfs())
62             while(int d = dfs(s, INF))

```

```

63         ans+=d;
64     return ans;
65 }
66 inline bool check(){//判满流
67     return work() == F;
68 }
69 };

```

费用流 (EK) 效率 $O(NM^2F)$

```

1  template<typename T>
2  struct EK{
3      const T INF = 2e18;
4      T cost;
5      struct Edge{int to, nxt, w, f;};
6      vector<Edge>e;
7      vector<bool> vis;
8      vector<int> head, lst;
9      vector<T> dis;
10     int tot, s, t;
11     EK(int n, int m, int st, int ed){
12         head.assign(n+1, -1);
13         e.resize(m<<1);
14         dis.resize(n+1); vis.resize(n+1);
15         s = st; t = ed; tot = 0; cost = 0;
16     }
17     inline int add(int u, int v, int w, int f){
18         e[tot]=(Edge){v, head[u], w, f};
19         head[u] = tot++;
20         return tot-1;
21     }
22     inline int insert(int u, int v, int w, int f){
23         int tmp = add(u, v, w, f);
24         add(v, u, -w, 0);
25         return tmp;
26     }
27     inline bool spfa(){
28         queue<int> q;
29         dis.assign(dis.size(), INF);
30         vis.assign(vis.size(), false);
31         dis[s] = 0; vis[s] = 1;
32         lst = head;
33         q.push(s);
34         while(!q.empty()){
35             int u = q.front(); q.pop();
36             for(int i = head[u]; ~i; i = e[i].nxt){

```



```

37         int v = e[i].to;
38         if(e[i].f > 0 && dis[v] > dis[u] + e[i].w){
39             dis[v] = dis[u] + e[i].w;
40             if(!vis[v]){
41                 vis[v] = 1;
42                 q.push(v);
43             }
44         }
45     }
46     vis[u] = 0;
47 }
48 return dis[t] != INF;
49 }
50 int dfs(int u, int f){
51     if(u == t || !f) return f;
52     vis[u] = 1;
53     int ans = 0;
54     for(int &i=lst[u]; ~i; i=e[i].nxt){
55         int v = e[i].to;
56         if(e[i].f && dis[v] == dis[u] + e[i].w && !vis[v]){
57             int val = dfs(v, min(e[i].f, f - ans));
58             if(val){
59                 e[i].f -= val;
60                 e[i^1].f += val;
61                 cost += val * e[i].w;
62                 ans += val;
63                 if(ans == f) break;
64             }
65         }
66     }
67     vis[u] = 0;
68     return ans;
69 }
70 inline array<T,2> work(){
71     T ans = 0;
72     while(spfa()){
73         T d = dfs(s, INF);
74         ans += d;
75     }
76     return {ans, cost};
77 }
78 };

```

4.4 强连通分量

```
1 struct SCC{
2 //SCC(n) n: 点数
3     vector<vector<pii>>>e;
4     stack<int>sta;
5     vector<int>id,dfn,low,ins,siz;
6     vector<ll>val;
7     int tot, scc, n;
8     SCC(int nn){
9         n = nn;
10        e.resize(n+1); siz.resize(n+1);
11        id.resize(n+1); dfn.resize(n+1);
12        low.resize(n+1); ins.resize(n+1); val.resize(n+1);
13        scc = tot = 0;
14    }
15    void tarjan(int u){
16        dfn[u] = low[u] = ++tot;
17        sta.push(u); ins[u] = 1;
18        for(pii E: e[u]){
19            int v = E.first;
20            if(!dfn[v]){
21                tarjan(v);
22                low[u] = min(low[u], low[v]);
23            } else if(ins[v]){
24                low[u] = min(low[u], dfn[v]);
25            }
26        }
27        if(dfn[u] == low[u]){
28            ++scc;
29            while(sta.top() != u){
30                int v = sta.top(); sta.pop();
31                id[v] = scc;
32                siz[scc]++;
33                ins[v] = 0;
34            }
35            ins[u] = 0;
36            id[u] = scc;
37            siz[scc]++;
38            sta.pop();
39        }
40    }
41    inline void remake(){
42        for(int i=1; i<=n; i++)
43            if(!dfn[i]) tarjan(i);
```

```

44     vector<vector<pii>> g(n+1);
45     for(int u=1; u<=n; u++)
46         for(pii E: e[u]){
47             int v = E.first, w = E.second;
48             if(id[u] != id[v])
49                 g[id[u]].pb({id[v], w}); // 边权
50             else
51                 val[id[u]] += w; // 点权
52         }
53     e = g;
54 }
55 };

```

4.5 点双联通分量

边双只需要 $\text{low}[v] > \text{dfn}[u]$ 即可，然后用强连通分量的弹栈方式

```

1  struct DCC{
2
3      vector<vector<int>>>e;
4      vector<vector<int>>>g;
5      vector<int> dfn, low;
6      stack<int> sta;
7      int vcc, tot, n;
8      //vcc: n+1~2n 方点 可以建Block Forest
9      DCC(int nn){
10         n = nn;
11         e.resize(n+1); g.resize(n<<1|1);
12         dfn.resize(n+1); low.resize(n+1);
13         tot = 0;
14         vcc = n;
15     }
16     void tarjan(int u, int fa){
17         low[u] = dfn[u] = ++tot; sta.push(u);
18         for(int v: e[u]){
19             if(v == fa) continue;
20             if(!dfn[v]){
21                 tarjan(v, u);
22                 low[u] = min(low[u], low[v]);
23                 if(low[v] >= dfn[u]){ //u是割点
24                     vcc++; //vertice double connected components
25                     while(sta.top() != v){
26                         int x = sta.top(); sta.pop();
27                         g[vcc].pb(x); g[x].pb(vcc);
28                     }

```

```

29         sta.pop();
30         g[vcc].pb(v); g[v].pb(vcc);
31         g[vcc].pb(u); g[u].pb(vcc);
32     }
33     } else // 双联通不需要考虑横叉边
34         low[u] = min(low[u], dfn[v]);
35     }
36 }
37 inline int work(){
38     int tot = 0;
39     for(int i=1; i<=n; i++){
40         if(!dfn[i]){
41             if(!tot)
42                 tarjan(i, 0);
43             ++tot;
44         }
45     }
46     return tot;
47 }
48 };

```

5 杂项

5.1 快读快输板子

```
1 inline ll read() {
2     ll x = 0, f = 1; char ch = getchar();
3     for(; ch < '0' || ch > '9'; ch = getchar())
4         if(ch == '-') f = -f;
5     for(; ch >= '0' && ch <= '9'; ch = getchar())
6         x = x * 10 + ch - '0';
7     return x * f;
8 }
9 inline void print(ll x){
10     if(!x)
11         return;
12     print(x/10);
13     putchar(x%10+'0');
14 }
15 inline void write(ll x){
16     if(x<0) putchar('-'), x=-x;
17     if(!x) putchar('0');
18     print(x);
19 }
20 inline void writeln(ll x){
21     write(x); putchar('\n');
22 }
23
24 //用的时候记得改<<里的内容
25 struct FastIO{//不可与scanf,printf,getchar,putchar,gets,puts,cin,cout混用
26 private:
27     static const int BUFSIZE=1e5;
28     char buf[BUFSIZE];int pos,len;//读入buffer(缓冲器)以及读入指针
29     int wpos;char wbuf[BUFSIZE];//输出指针以及输出buffer
30     char a[21];//储存输出的数字
31 #define gc() (pos == len && (len=(pos=0)
32     +fread(buf,1,BUFSIZE,stdin),!len)?EOF:buf[pos++])
33 #define pc(c) (wpos == BUFSIZE ?
34     fwrite(wbuf,1,BUFSIZE,stdout),wpos=0,
35     wbuf[wpos++]=c:wbuf[wpos++]=c)
36 public:
37     FastIO():wpos(0),pos(0),len(0){}
38     ~FastIO(){if(wpos)fwrite(wbuf,1,wpos,stdout),wpos=0;}
39     inline char getc(){return gc();};//读取char
40     inline void putc(char c){pc(c);};//输出字符
41     inline long long rd(){//读取long long
```

```

42     long long x=0;char c=gc();bool f=0;
43     for(;c<'0' || c>'9';c=gc())f|=c=='-';
44     for(;c>='0'&& c<='9';c=gc())x=(x<<3)+(x<<1)+(c^48);
45     return f?~x+1:x;
46 }
47 template<typename T>inline bool read(T &x){// 多测读整数 while(io.read(n))work()
48     x=0;char c=gc();bool f=0;
49     for(;c<'0' || c>'9';c=gc()){if(c==EOF)return false;f|=c=='-';}
50     for(;c>='0'&& c<='9';c=gc())x=(x<<3)+(x<<1)+(c^48);
51     if(f)x=~x+1;return true;
52 }
53 template<typename T>inline void wt(T x){// 输出整数
54     if(x<0)pc('-'),x=-x;short h=0;
55     for(a[++h]='0'+x%10,x/=10;x/x/=10)a[++h]='0'+x%10;
56     while(h)pc(a[h--]);
57 }
58 template<typename T>inline void wtl(T x){wt(x);pc('\n');};//write line 输出整数并
59 template<typename T>inline void wtb(T x){wt(x);pc(' ');};//write blank 输出整数并
60 inline int gets(char *s){
61     int l=0;char c=gc();for(;c<=' ';c=gc());
62     for(;c>' ';c=gc())s[l++]=c;s[l]=0;
63     return l;
64 }
65 inline void puts(const char *s){
66     const char *p=s;while(*p)pc(*p++);
67 }// 输出字符串 (不带换行)
68 template<typename T>inline FastIO & operator >> (T &a){
69     return read(a),*this;
70 }//io>>a>>b; 只能输入整数
71 template<typename T>inline FastIO & operator << (T a){
72     return wtl(a),*this;
73 }//io<<a<<b; 输出整数并带有回车
74 }io;// 本地测试出入结束后请输入一次 ctrl Z

```

5.2 分治

5.2.1 cdq 分治

最长上升子序列的数量

```

1 // 最长上升子序列计数
2 // 转化为偏序问题
3 #include<bits/stdc++.h>
4 using namespace std;
5 #define ll long long
6 #define pii pair<int,int>

```

```

7  #define pb push_back
8  #define ln '\n'
9
10 const int N = 1e5+5;
11 const int mod = 1e9+7;
12 int n, a[N], id[N];
13 pii f[N];
14
15 pii calc (pii a, pii b){
16     if(a.first < b.first) return b;
17     if(a.first > b.first) return a;
18     a.second += b.second;
19     if(a.second >= mod)
20         a.second -= mod;
21     return a;
22 };
23
24 void work(int l, int r){
25     if(l == r)
26         return;
27     int mid = l+r>>1;
28     work(l, mid);
29     for(int i=l; i<=r; i++){
30         id[i] = i;
31         sort(id+l, id+r+1, [](int x, int y){
32             return a[x] == a[y] ? x > y : a[x] < a[y];
33         });
34         pair<int,int> now = {0,0};
35
36         for(int i=l; i<=r; i++){
37             if(id[i] <= mid)
38                 now = calc(now, f[id[i]]);
39             else{
40                 auto tmp = now;
41                 tmp.first++;
42                 f[id[i]] = calc(f[id[i]], tmp);
43             }
44         }
45
46         work(mid+1, r);
47     }
48
49 int main(){
50

```

```

51     ios::sync_with_stdio(false);
52     cin.tie(0);
53
54     cin >> n;
55     for(int i=1; i<=n; i++)
56         cin >> a[i];
57     for(int i=1; i<=n; i++)
58         f[i] = {1, 1}; // 最长, 数量
59
60     work(1, n);
61     pair<int,int> ans = {0, 0};
62     for(int i=1; i<=n; i++)
63         ans = calc(ans, f[i]);
64
65     cout << ans.second << ln;
66 }

```

5.2.2 树上点分治

luogu3806 给定一棵树，询问树上距离为 k 的点是否存在

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int,int> pii;
5  #define pb push_back
6  #define ln '\n'
7
8  const int N = 1e4+5;
9  const int M = 1e7+5;
10
11 vector<pii> e[N];
12
13 int n, m;
14 int a[N];
15 int val[N], vis[N], sz[N], sum, rt;
16 // 作为重心的价值, 以及该点是否删了, 子树的大小
17 int res[M]; // 该长度的链是否出现
18 int ans[N]; // 各个询问的答案是否出现
19
20 void getroot(int u, int fa){
21     sz[u] = 1; val[u] = 0;
22     for(auto [v, w]: e[u]){
23         if(vis[v] || v == fa)
24             continue;

```



```

25         getroot(v, u);
26         sz[u] = sz[u] + sz[v];
27         val[u] = max(val[u], sz[v]);
28     }
29     val[u] = max(val[u], sum - sz[u]);
30     if(!rt || val[u] < val[rt])
31         rt = u;
32 }
33
34 vector<int> vec;
35
36 void dfs(int u, int fa, int dist){
37     if(dist > 10000000) return;
38     vec.pb(dist);
39     for(auto [v, w]: e[u])
40         if(!vis[v] && v!=fa)
41             dfs(v, u, dist+w);
42 }
43
44 void calc(int u, int fa){
45     vector<int> p;
46     for(auto [v, w]: e[u]){
47         if(vis[v] || v == fa)
48             continue;
49         vec.clear();
50         dfs(v, u, w);
51         for(auto len: vec){
52             for(int i=1; i<=m; i++)
53                 if(a[i]-len>=0&&res[a[i] - len])
54                     ans[i] = 1;
55         }
56         for(auto len: vec){
57             res[len]=1;
58             p.pb(len);
59         }
60     }
61     for(auto len: p)
62         res[len] = 0;
63 }
64
65 void work(int u, int fa){
66     vis[u] = 1;
67     calc(u, fa);
68     for(auto [v, w]: e[u]){

```

```

69         if(vis[v] || v == fa)
70             continue;
71         sum = sz[v]; rt = 0;
72         getroot(v, u); // 找子树内的重心
73         work(rt, u);
74     }
75 }
76
77 int main(){
78     ios::sync_with_stdio(false);
79     cin.tie(0);
80
81     cin >> n >> m;
82     for(int i=1; i<n; i++){
83         int u, v, w;
84         cin >> u >> v >> w;
85         e[u].pb({v, w});
86         e[v].pb({u, w});
87     }
88
89     for(int i=1; i<=m; i++)
90         cin >> a[i];
91
92     res[0] = 1;
93     rt = 0; sum = n;
94     getroot(1, -1);
95     // cout << "root:= " << rt << ln;
96     work(rt, -1);
97
98     for(int i=1; i<=m; i++)
99         cout << (ans[i]?"AYE":"NAY") << ln;
100
101 }

```

5.3 莫队

5.3.1 普通莫队

小 z 的袜子: n 双袜子, 查询在 [l,r] 中抽到两只同色袜子的概率

```

1 #include<cstdio>
2 #include<cmath>
3 #include<algorithm>
4 #define ll long long
5 struct pos{int l,r,id;}q[50005];
6 int n,m,a[50005],vis[50005],be[50005];

```

```

7 ll ans[50005],Ans[50005];
8 inline ll gcd(ll a,ll b){ return b==0?a:gcd(b,a%b); }
9 bool cmp(pos a,pos b){return be[a.l]==be[b.l]?a.r<b.r:be[a.l]<be[b.l];}
10 inline int read() {
11     int x=0,f=1; char ch=getchar();
12     while(ch<'0' || ch>'9'){if(ch=='-') f=-1;ch=getchar();}
13     while(ch>='0' && ch<='9') x=x*10+ch-'0',ch=getchar();
14     return x*f;
15 }
16 int main() {
17     n=read(); m=read(); int siz=sqrt(n-1)+1;
18     for(int i=1;i<=n;i++) a[i]=read(),be[i]=(i-1)/siz+1;//scanf("%d",&a[i]);
19     for(int i=1;i<=m;i++) q[i].l=read(),q[i].r=read(),q[i].id=i;
20     std::sort(q+1,q+m+1,cmp);
21     int l=1,r=0;ll res=0;
22     for(int i=1;i<=m;i++) {
23         while(l<q[i].l) res=res-(1ll*(vis[a[l]]*(vis[a[l]]-1)/2-1ll*(vis[a[l]]-1));
24         while(r>q[i].r) res=res-(1ll*(vis[a[r]]*(vis[a[r]]-1)/2-1ll*(vis[a[r]]-1));
25         while(l>q[i].l) vis[a[--l]]++,res=res+(1ll*(vis[a[l]]*(vis[a[l]]-1)/2-1ll*(vis[a[l]]-1));
26         while(r<q[i].r) vis[a[++r]]++,res=res+(1ll*(vis[a[r]]*(vis[a[r]]-1)/2-1ll*(vis[a[r]]-1));
27         ans[q[i].id]=res; Ans[q[i].id]=1ll*(q[i].r-q[i].l+1)*(q[i].r+q[i].l);
28         if(l==r) ans[q[i].id]=0;
29     }
30     for(int i=1;i<=m;i++) {
31         if(ans[i]==0) {puts("0/1");continue;}
32         ll G=gcd(ans[i],Ans[i]);
33         printf("%lld/%lld\n",ans[i]/G,Ans[i]/G);
34     }
35 }

```

5.3.2 带修莫队

数颜色：查询 $l..r$ 中有多少不同颜色的画笔，把第 i 支画笔换成颜色 p

```

1 #include<cstdio>
2 #include<cmath>
3 #include<algorithm>
4 using namespace std;
5 using std::sort;
6 const int N=5e4+5,M=1e6+5;
7 struct node{int l,r,id,cur;}q[N];
8 struct Node{int x,c,lst;}b[N];
9 int n,m,a[N],bl[N],siz,cnt,tot;
10 int l,r,ans,vis[M],Ans[N],lst[N];
11 inline void add(int x) {

```

```

12         if(++vis[a[x]]==1)++ans;
13     }
14     inline void del(int x) {
15         if(--vis[a[x]]==0)--ans;
16     }
17     inline void upd(int x,int v) {
18         if(l<=x&&x<=r) del(x),a[x]=v,add(x);
19         else a[x]=v;
20     }
21     bool cmp(node a,node b) {
22         if(bl[a.l]!=bl[b.l])
23             return bl[a.l]<bl[b.l];
24         if(bl[a.r]!=bl[b.r])
25             return bl[a.r]<bl[b.r];
26         return a.id<b.id;
27     }
28     int main() {
29         scanf("%d%d",&n,&m);
30         siz=pow(n,0.6666);
31         for(int i=1;i<=n;i++)scanf("%d",&a[i]),lst[i]=a[i],bl[i]=(i-1)/siz+1;
32         for(int i=1;i<=m;i++) {
33             char s[10];
34             scanf("%s",s);
35             if(s[0]=='Q') {
36                 int l,r;
37                 scanf("%d%d",&l,&r);
38                 ++cnt;
39                 q[cnt]=(node){l,r,cnt,tot}; //q[i].cur 表示位于第几次修改之后
40             }else {
41                 int x,c;
42                 scanf("%d%d",&x,&c);
43                 b[++tot]=(Node){x,c,lst[x]}; lst[x]=c;
44             }
45         }
46         l=1; r=0; ans=0;
47         int now=0;
48         sort(q+1,q+cnt+1,cmp);
49         for(int i=1;i<=cnt;i++) {
50             while(q[i].cur<now) { // 多加了
51                 upd(b[now].x,b[now].lst);
52                 now--;
53             }
54             while(q[i].cur>now) { // 少加了
55                 now++;

```

```

56         upd(b[now].x,b[now].c);
57     }
58     while(l<q[i].l) del(l++);
59     while(l>q[i].l) add(--l);
60     while(r<q[i].r) add(++r);
61     while(r>q[i].r) del(r--);
62     Ans[q[i].id]=ans;
63 }
64 for(int i=1;i<=cnt;i++)
65     printf("%d\n",Ans[i]);
66 }

```

5.3.3 回滚莫队

只有单侧加没有减的莫队

```

1 // 只有加法的莫队
2 //https://codeforces.com/contest/1514/problem/D
3 #include<bits/stdc++.h>
4 using namespace std;
5
6 #define ll long long
7
8 inline ll read() {
9     ll x = 0, f = 1; char ch = getchar();
10    for(; ch < '0' || ch>'9'; ch = getchar())
11        if(ch == '-') f = -f;
12    for(; ch >= '0' && ch <= '9'; ch = getchar())
13        x = x * 10 + ch - '0';
14    return x * f;
15 }
16
17 inline void chkmin( int &a, int b ) { if(a > b) a = b; }
18
19 inline void chkmax( int &a, int b ) { if(a < b) a = b; }
20
21 #define _ read()
22
23 #define ln endl
24
25 const int N=3e5+5;
26 int n, m, siz, v[N], bl[N], res[N];
27 int l, r, R[1005], ans[N], cnt[N], tot[N];
28 vector<int> mo;
29 struct Mo{int l, r, id;}a[N];

```

```

30 inline bool cmp(Mo a, Mo b){return bl[a.l]==bl[b.l]?a.r<b.r:bl[a.l]<bl[b.l];}
31
32 inline void solve(int i){
33     while(l<R[i]+1) cnt[v[l]]--, l++;
34     while(r>R[i]) cnt[v[r]]--, r--;
35     while(r<R[i]) ++r, cnt[v[r]]++;
36     int tmp=0, res=0;
37     for(int j=0; j<mo.size(); j++){
38         int x=mo[j];
39         while(r<a[x].r){
40             r++; cnt[v[r]]++;
41             res=max(res, cnt[v[r]]);
42         }
43         tmp=res;
44         while(l>a[x].l){
45             --l; cnt[v[l]]++;
46             tmp=max(tmp, cnt[v[l]]);
47         }
48         ans[a[x].id]=max(2*tmp-(a[x].r-a[x].l+1), 1);
49         while(l<R[i]+1) cnt[v[l]]--, ++l;
50     }
51     mo.clear();
52 }
53
54 int main(){
55     n=read(); m=read(); siz=sqrt(n);
56     for(int i=1; i<=n; i++) {
57         v[i]=read(), bl[i]=(i-1)/siz+1;
58         R[bl[i]]=max(R[bl[i]], i);
59     }
60     for(int i=1; i<=m; i++)
61         a[i].l=read(), a[i].r=read(), a[i].id=i;
62     sort(a+1, a+m+1, cmp);
63     l=1; r=0;
64     int lst=0;
65     for(int i=1; i<=m; i++){
66         if(bl[a[i].l]==bl[a[i].r]){
67             int tmp=0;
68             for(int j=a[i].l; j<=a[i].r; j++){
69                 tot[v[j]]++;
70                 if(tot[v[j]]>tmp)
71                     tmp=tot[v[j]];
72             }
73             ans[a[i].id]=max(2*tmp-(a[i].r-a[i].l+1), 1);

```

```

74         for(int j=a[i].l; j<=a[i].r; j++)
75             tot[v[j]]--;
76     }else{
77         if(bl[a[i].l]==lst);
78         else if(mo.size()) solve(lst);
79         mo.push_back(i); lst=bl[a[i].l];
80     }
81 }
82 if(mo.size())
83     solve(lst);
84 for(int i=1; i<=m; i++)
85     printf("%d\n", ans[i]);
86 }

```

5.4 悬线法

```

1 //51nod最大全1子矩阵
2 //悬线法， 实际上就是单调栈优化dp
3 int ans = 0;
4 for(int i=1; i<=n; i++){
5     int top = 0;
6     for(int j=1; j<=m; j++){
7         a[i][j] = read();
8         if(!a[i][j])
9             U[i][j] = 0;
10        else
11            U[i][j] = U[i-1][j] + 1;
12        while(top&&U[i][j]<=U[i][s[top]])
13            top--;
14        L[i][j] = s[top] + 1;
15        s[++top] = j;
16    }
17    s[++top] = m+1;
18    for(int j=m; j; j--){
19        while(top&&U[i][j]<=U[i][s[top]])
20            top--;
21        R[i][j] = s[top] - 1;
22        s[++top] = j;
23    }
24    for(int j=1; j<=m; j++)
25        ans = max(ans, U[i][j]*(R[i][j]-L[i][j]+1));
26 }
27 printf("%d\n", ans);

```

5.5 n 数码问题

对于一个 $n \times m$ 的矩阵，有一个空格，问能否将 $n \times m - 1$ 个元素排序

记一个矩阵的权值为 $F(A) = (\text{状态 } A \text{ 忽略空格时求出的逆序对数量} + \text{状态 } A \text{ 将空格移动到状态 } B \text{ 所需的行数} \times (\text{A 的列数} + 1))$

如果 $F(A) = F(B)$ 那么两个矩阵可以相互转化

5.6 高精度板子

```
1  const int MAXN = 200; // 字符大小， 没压位
2  struct BigInt
3  {
4      int len, s[MAXN];
5      BigInt () // 初始化
6      {
7          memset(s, 0, sizeof(s));
8          len = 1;
9      }
10     BigInt (int num) { *this = num; }
11     BigInt (const char *num) { *this = num; } // 让this指针指向当前字符串
12     BigInt operator = (const int num)
13     {
14         char s[MAXN];
15         sprintf(s, "%d", num); // sprintf函数将整型映到字符串中
16         *this = s;
17         return *this; // 再将字符串转到下面字符串转化的函数中
18     }
19     BigInt operator = (const char *num)
20     {
21         for(int i = 0; num[i] == '0'; num++) ; // 去前导0
22         len = strlen(num);
23         for(int i = 0; i < len; i++) s[i] = num[len-i-1] - '0'; // 反着存
24         return *this;
25     }
26     BigInt operator + (const BigInt &b) const // 对应位相加，最为简单
27     {
28         BigInt c;
29         c.len = 0;
30         for(int i = 0, g = 0; g || i < max(len, b.len); i++)
31         {
32             int x = g;
33             if(i < len) x += s[i];
34             if(i < b.len) x += b.s[i];
35             c.s[c.len++] = x % 10; // 关于加法进位
36             g = x / 10;
```



```

37     }
38     return c;
39 }
40 BigInt operator += (const BigInt &b) // 如上文所说，此类运算符皆如此重载
41 {
42     *this = *this + b;
43     return *this;
44 }
45 void clean() // 由于接下来的运算不能确定结果的长度，先大而估之然后再查
46 {
47     while(len > 1 && !s[len-1]) len--; // 首位部分 '0' 故删除该部分长度
48 }
49 BigInt operator * (const BigInt &b) // 乘法重载在于列竖式，再将竖式中的数转为抽象
50 {
51     BigInt c;
52     c.len = len + b.len;
53     for(int i = 0; i < len; i++)
54     {
55         for(int j = 0; j < b.len; j++)
56         {
57             c.s[i+j] += s[i] * b.s[j]; // 不妨列个竖式看一看
58         }
59     }
60     for(int i = 0; i < c.len; i++) // 关于进位，与加法意同
61     {
62         c.s[i+1] += c.s[i]/10;
63         c.s[i] %= 10;
64     }
65     c.clean(); // 我们估的位数是a+b的长度和，但可能比它小 (1*1 = 1)
66     return c;
67 }
68 BigInt operator *= (const BigInt &b)
69 {
70     *this = *this * b;
71     return *this;
72 }
73 BigInt operator - (const BigInt &b) // 对应位相减，加法的进位改为借1
74 { // 不考虑负数
75     BigInt c;
76     c.len = 0;
77     for(int i = 0, g = 0; i < len; i++)
78     {
79         int x = s[i] - g;
80         if(i < b.len) x -= b.s[i]; // 可能长度不等

```

```

81         if(x >= 0) g = 0; //是否向上移位借1
82         else
83         {
84             g = 1;
85             x += 10;
86         }
87         c.s[c.len++] = x;
88     }
89     c.clean();
90     return c;
91 }
92 BigInt operator -= (const BigInt &b)
93 {
94     *this = *this - b;
95     return *this;
96 }
97 BigInt operator / (const BigInt &b) //运用除是减的本质，不停地减，直到小于被减
98 {
99     BigInt c, f = 0; //可能会在使用减法时出现高精度运算
100    for(int i = len-1; i >= 0; i--) //正常顺序，从最高位开始
101    {
102        f = f*10; //上面位的剩余到下一位*10
103        f.s[0] = s[i]; //加上当前位
104        while(f >= b)
105        {
106            f -= b;
107            c.s[i]++;
108        }
109    }
110    c.len = len; //估最长位
111    c.clean();
112    return c;
113 }
114 BigInt operator /= (const BigInt &b)
115 {
116     *this = *this / b;
117     return *this;
118 }
119 BigInt operator % (const BigInt &b) //取模就是除完剩下的
120 {
121     BigInt r = *this / b;
122     r = *this - r*b;
123     r.clean();
124     return r;

```

```

125     }
126     BigInt operator %= (const BigInt &b)
127     {
128         *this = *this % b;
129         return *this;
130     }
131     bool operator < (const BigInt &b) // 字符串比较原理
132     {
133         if(len != b.len) return len < b.len;
134         for(int i = len-1; i != -1; i--)
135         {
136             if(s[i] != b.s[i]) return s[i] < b.s[i];
137         }
138         return false;
139     }
140     bool operator > (const BigInt &b) // 同理
141     {
142         if(len != b.len) return len > b.len;
143         for(int i = len-1; i != -1; i--)
144         {
145             if(s[i] != b.s[i]) return s[i] > b.s[i];
146         }
147         return false;
148     }
149     bool operator == (const BigInt &b)
150     {
151         return !(*this > b) && !(*this < b);
152     }
153     bool operator != (const BigInt &b)
154     {
155         return !(*this == b);
156     }
157     bool operator <= (const BigInt &b)
158     {
159         return *this < b || *this == b;
160     }
161     bool operator >= (const BigInt &b)
162     {
163         return *this > b || *this == b;
164     }
165     string str() const // 将结果转化为字符串（用于输出）
166     {
167         string res = "";
168         for(int i = 0; i < len; i++) res = char(s[i]+'0')+res;

```

```

169         return res;
170     }
171 };
172
173 istream& operator >> (istream &in, BigInt &x) // 重载输入流
174 {
175     string s;
176     in >> s;
177     x = s.c_str(); //string 转化为 char[]
178     return in;
179 }
180
181 ostream& operator << (ostream &out, const BigInt &x) // 重载输出流
182 {
183     out << x.str();
184     return out;
185 }

```

5.7 整体二分

k 大数查询, 1. 将 c 加入编号 [l,r] 的集合种, 2. 查询 [l, r] 集合的并集中第 k 大的数是多少

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  #define pb push_back
5  #define ln '\n'
6
7  inline ll read() {
8      ll x = 0, f = 1; char ch = getchar();
9      for(; ch < '0' || ch > '9'; ch = getchar())
10         if(ch == '-') f = -f;
11     for(; ch >= '0' && ch <= '9'; ch = getchar())
12         x = x * 10 + ch - '0';
13     return x * f;
14 }
15
16 const int N = 5e4+5;
17 struct node{int l, r; ll sum, lzy;}tree[N<<2];
18
19 void pushup(int rt){
20     tree[rt].sum = tree[rt<<1].sum + tree[rt<<1|1].sum;
21 }
22
23 void pushdown(int rt){

```

```

24     int l = tree[rt].l, r = tree[rt].r;
25     if(tree[rt].lzy){
26         int mid = l + r >> 1;
27         tree[rt<<1].lzy += tree[rt].lzy;
28         tree[rt<<1|1].lzy += tree[rt].lzy;
29         tree[rt<<1].sum += 1ll * tree[rt].lzy * (mid - l + 1);
30         tree[rt<<1|1].sum += 1ll * tree[rt].lzy * (r - mid);
31         tree[rt].lzy = 0;
32     }
33 }
34
35 void build(int l, int r, int rt){
36     tree[rt].l = l; tree[rt].r = r;
37     if(l == r)
38         return;
39     int mid=l+r>>1;
40     build(l, mid, rt<<1);
41     build(mid+1, r, rt<<1|1);
42 }
43
44 void modify(int L, int R, int v, int l, int r, int rt){
45     if(L <= l && r <= R){
46         tree[rt].sum += 1ll * (r - l + 1) * v;
47         tree[rt].lzy += v;
48         return;
49     }
50     int mid = l + r >> 1;
51     pushdown(rt);
52     if(L <= mid)
53         modify(L, R, v, l, mid, rt<<1);
54     if(R > mid)
55         modify(L, R, v, mid+1, r, rt<<1|1);
56     pushup(rt);
57 }
58
59 ll query(int L, int R, int l, int r, int rt){
60     if(L <= l && r <= R)
61         return tree[rt].sum;
62     int mid = l + r >> 1;
63     ll ans = 0;
64     pushdown(rt);
65     if(L<=mid)
66         ans += query(L, R, l, mid, rt<<1);
67     if(R>mid)

```

```

68         ans += query(L, R, mid+1, r, rt<<1|1);
69     return ans;
70 }
71
72 int n, m;
73 int opt[N], lef[N], rig[N], ans[N];
74 ll c[N];
75
76 void work(int l, int r, vector<int> now){
77     if(l>r) return;
78     if(l==r){
79         for(auto i: now)
80             if(opt[i] == 2)
81                 ans[i] = 1;
82         return;
83     }
84     int mid = l+r>>1;
85     vector<int> L, R;
86     for(auto i: now)
87         if(opt[i] == 1){
88             if(c[i] > mid)
89                 R.pb(i), modify(lef[i], rig[i], 1, 1, n, 1);
90             else
91                 L.pb(i);
92         } else {
93             ll tmp = query(lef[i], rig[i], 1, n, 1);
94             if(tmp >= c[i])//>mid的有>=k个, 说明答案在mid+1..r
95                 R.pb(i);
96             else
97                 c[i] -= tmp,
98                 L.pb(i);
99         }
100     // now.clear();
101     for(auto i: R)
102         if(opt[i] == 1)
103             modify(lef[i], rig[i], -1, 1, n, 1);
104     if(L.size())
105         work(l, mid, L);
106     if(R.size())
107         work(mid+1, r, R);
108     //对>=mid的修改+1, <的不用管-->没贡献
109     //对于修改, 如果c>=mid
110 }
111

```

```

112 int main(){
113     #ifndef ONLINE_JUDGE
114         freopen("0.txt", "w", stdout);
115     #endif
116     n=read(); m=read();
117     build(1, n, 1);
118     std::vector<int> now;
119     for(int i=1; i<=m; i++){
120         opt[i] = read();
121         lef[i]=read();
122         rig[i]=read();
123         c[i]=read();
124         if(lef[i] > rig[i]) swap(lef[i], rig[i]);
125         now.pb(i);
126     }
127     work(0, n, now);
128     for(int i=1; i<=m; i++)
129         if(opt[i] == 2) printf("%d\n", ans[i]);
130 }

```

5.8 背包回退 (待写)