

# Algorithm Codelet

TieWay59

October 11, 2019

# Contents

<b>1 其它</b>	<b>3</b>
1.1 c++ 中处理 2 进制的一些函数.cpp	3
1.2 IO	4
1.2.1 fread.cpp	4
1.2.2 fread2.cpp	5
1.2.3 保留小数.cpp	7
1.2.4 读取整数.cpp	7
1.3 测量程序的运行时间.cpp	7
1.4 转化成二进制.cpp	8
<b>2 几何</b>	<b>8</b>
2.1 2D	8
2.1.1 8 旋转卡壳.cpp	8
2.1.2 PSLG.cpp	11
2.1.3 二维几何模板.cpp	13
2.1.4 二维凸包.cpp	15
2.1.5 判断点是否在多边形内.cpp	16
2.1.6 圆与多边形相交的面积.cpp	16
2.1.7 求圆与直线的交点.cpp	20
2.2 3D	21
2.2.1 三维几何的基本操作.cpp	21
2.2.2 三维几何的模版.cpp	22
2.2.3 三维凸包.cpp	24
2.2.4 维度转换为三维坐标.cpp	26
<b>3 动态规划</b>	<b>26</b>
3.1 1 单调队列.cpp	26
3.2 1 最长上升子序列.cpp	27
3.3 string dp	28
3.3.1 trie+dp.cpp	28
3.4 zhuangyadp	30
3.4.1 1 多米诺骨牌覆盖.cpp	30
3.5 树上的分治	34
3.5.1 1 树的重心.cpp	34
<b>4 图论</b>	<b>35</b>
4.1 DFS	35
4.1.1 1. 无向图的割点和桥.cpp	35
4.1.2 2. 无向图的双连通分量.cpp	37
4.1.3 3 有向图的强联通分量.cpp	38
4.1.4 4 2-sat 问题.cpp	40
4.2 LCA	41
4.2.1 1 DFS+RMQ.cpp	41
4.2.2 2 倍增算法.cpp	42
4.3 Maxflow	44
4.3.1 1 Dinic.cpp	44
4.3.2 2 ISAP.cpp	47
4.3.3 3 MCMF.cpp	49
4.4 二分图	51
4.4.1 1 匈牙利算法.cpp	51
4.4.2 2 KM.cpp	53
4.4.3 3 一般图最大匹配.cpp	54
4.5 最小生成树	56
4.5.1 1 Kruskal 卡鲁斯卡尔算法.cpp	56

4.5.2	2 prim 算法.cpp	57
4.5.3	3 最小限制生成树.cpp	58
4.5.4	4 次小生成树.cpp	61
4.6	最短路	64
4.6.1	1 Dijkstra.cpp	64
4.6.2	2 Bellman-ford.cpp	67
4.6.3	3 floyed.cpp	69
4.6.4	堆优化的有限队列.cpp	70
5	数学	71
5.1	3 FWT 模板.cpp	71
5.2	4 单纯形法.cpp	72
5.3	5. 线性基.cpp	74
5.4	BM.cpp	75
5.5	Combinatorial mathematics	77
5.5.1	康托展开.cpp	77
5.6	FFT	77
5.6.1	FFT.cpp	77
5.6.2	kuangbin.cpp	78
5.6.3	lrj.cpp	81
5.7	Lagrange-poly	82
5.7.1	template.cpp	82
5.8	三分.cpp	83
5.9	博弈	85
5.9.1	2. 威佐夫博弈.cpp	85
5.9.2	3 Nim 积.cpp	85
5.9.3	4 K 倍动态减法.cpp	86
5.9.4	5 海盗分金问题.cpp	87
5.9.5	6 Green Hackbush.cpp	88
5.9.6	7 反 nim 博弈.cpp	92
5.9.7	8 超自然数.cpp	92
5.10	数论	93
5.10.1	1 加法.cpp	93
5.10.2	1 逆元.cpp	94
5.10.3	2 减法.cpp	95
5.10.4	3 乘法.cpp	95
5.10.5	4 除法.cpp	96
5.10.6	5. 蒙哥马利快速模.cpp	96
5.10.7	Euler.cpp	98
5.10.8	lucas , 组合数.cpp	99
5.10.9	miller-rabin-Pollard-rho.cpp	99
5.10.10	分段求和.cpp	102
5.10.11	大数.cpp	102
5.10.12	快速数论变换.cpp	108
5.10.13	欧拉函数打表.cpp	109
5.10.14	欧拉筛和埃氏筛.cpp	111
5.10.15	素性检测.cpp	111
5.10.16	素数筛.cpp	113
5.10.17	逆元打表.cpp	114
5.11	矩阵快速幂.cpp	114
5.12	自适应辛普森积分.cpp	115

<b>6</b>	<b>数据结构</b>	<b>115</b>
6.1	CDQ 分治	115
6.1.1	CDQ 分治.cpp	115
6.1.2	CDQ 求动态逆序数.cpp	117
6.1.3	陌上花开 CDQ 三位偏序.cpp	119
6.2	fenkuai	121
6.2.1	区间修改区间查询.cpp	121
6.2.2	区间数的平方.cpp	122
6.2.3	在线查询区间众数.cpp	123
6.3	pbds	125
6.3.1	1 可合并优先队列.cpp	125
6.4	二叉搜索树	126
6.4.1	1 二叉树.cpp	126
6.4.2	2 treap.cpp	127
6.4.3	3 伸展树.cpp	131
6.5	基础数据结构	134
6.5.1	堆.cpp	134
6.6	字符串	134
6.6.1	1 Trie(前缀树).cpp	134
6.6.2	2 KMP.cpp	135
6.6.3	3 AC 自动机.cpp	136
6.6.4	4 KMP-KMP 变形.cpp	138
6.6.5	5 字符串 hash.cpp	139
6.6.6	6 后缀数组.cpp	140
6.7	并查集	141
6.7.1	加权并查集 + 区间合并.cpp	141
6.7.2	并查集.cpp	142
6.8	树状数组	143
6.8.1	1 树状数组模板.cpp	143
6.8.2	2 区间出现两次的数的个数.cpp	143
6.9	线段树	145
6.9.1	1. 区间更新区间查询.cpp	145
6.9.2	2 主席树求第 k 大.cpp	147
6.9.3	2 树套树求动态第 k 大.cpp	148
6.9.4	3 树套树求动态逆序数.cpp	151
6.9.5	4 李超树.cpp	155
6.9.6	5 线段树-区间最小乘积.cpp	157
6.9.7	6 区间加斐波那契数.cpp	159
6.9.8	7 区间加 + 区间乘.cpp	162
<b>7</b>	<b>模拟</b>	<b>165</b>
7.1	1 日期.cpp	165

# 1 其它

## 1.1 c++ 中处理 2 进制的一些函数.cpp

---

```
1  int __builtin_ffs (unsigned int x)
2
3  // Returns one plus the index of the least significant 1-bit of x, or if x is
   ↪ zero, returns zero.
4  // 返回右起第一个‘1’的位置。
5
6  int __builtin_clz (unsigned int x)
7
8  // Returns the number of leading 0-bits in x, starting at the most significant
   ↪ bit position. If x is 0, the result is undefined.
9  // 返回左起第一个‘1’之前 0 的个数。
10
11 int __builtin_ctz (unsigned int x)
12
13 // Returns the number of trailing 0-bits in x, starting at the least significant
   ↪ bit position. If x is 0, the result is undefined.
14 // 返回右起第一个‘1’之后的 0 的个数。
15
16 int __builtin_popcount (unsigned int x)
17
18 // Returns the number of 1-bits in x.
19 // 返回‘1’的个数。
20
21 int __builtin_parity (unsigned int x)
22
23 // Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
24 // 返回‘1’的个数的奇偶性。
25
26 int __builtin_ffsl (unsigned long)
27
28 // Similar to __builtin_ffs, except the argument type is unsigned long.
29
30 int __builtin_clzl (unsigned long)
31
32 // Similar to __builtin_clz, except the argument type is unsigned long.
33
34 int __builtin_ctzl (unsigned long)
35
36 // Similar to __builtin_ctz, except the argument type is unsigned long.
37
38 int __builtin_popcountl (unsigned long)
39
40 // Similar to __builtin_popcount, except the argument type is unsigned long.
41
42 int __builtin_parityl (unsigned long)
43
44 // Similar to __builtin_parity, except the argument type is unsigned long.
45
46 int __builtin_ffsll (unsigned long long)
47
48 // Similar to __builtin_ffs, except the argument type is unsigned long long.
```

```

49
50 int __builtin_clzll (unsigned long long)
51
52 // Similar to __builtin_clz, except the argument type is unsigned long long.
53
54 int __builtin_ctzll (unsigned long long)
55
56 // Similar to __builtin_ctz, except the argument type is unsigned long long.
57
58 int __builtin_popcountll (unsigned long long)
59
60 // Similar to __builtin_popcount, except the argument type is unsigned long long.
61
62 int __builtin_parityll (unsigned long long)
63
64 // Similar to __builtin_parity, except the argument type is unsigned long long.

```

---

## 1.2 IO

### 1.2.1 fread.cpp

```

1 namespace io {
2     const int L = 1 << 20 | 1;
3     char ibuf[L], *iS, *iT, c, obuf[L], *oS = obuf, *oT = obuf + L - 1, qu[55];
4     ↪ int f, qr;
5     #ifdef whzzt
6         #define gc() getchar()
7     #else
8         #define gc() (iS == iT ? (iT = (iS = ibuf) + fread (ibuf, 1, L, stdin), iS ==
9             ↪ iT ? EOF : *iS++) : *iS++)
10    #endif
11    template <class I>
12    inline void gi (I &x) {
13        for (f = 1, c = gc(); c < '0' || c > '9'; c = gc()) if (c == '-') f = -1;
14        for (x = 0; c <= '9' && c >= '0'; c = gc()) x = x * 10 + (c & 15); x *= f;
15    }
16    inline void flush () {
17        fwrite (obuf, 1, oS - obuf, stdout);
18    }
19    inline void putc (char x) {
20        *oS ++ = x;
21        if (oS == oT) flush (), oS = obuf;
22    }
23    template <class I>
24    void print (I x) {
25        if (!x) putc ('0'); if (x < 0) putc ('-'), x = -x;
26        while (x) qu[++ qr] = x % 10 + '0', x /= 10;
27        while (qr) putc (qu[qr --]);
28    }
29    struct io_ff { ~io_ff() { flush(); } } _io_ff_;
30 }
31 using io :: gi;
32 using io :: putc;
33 using io :: print;

```

---

## 1.2.2 fread2.cpp

```
1 namespace IO{
2     #define BUF_SIZE 100000
3     #define OUT_SIZE 100000
4     #define ll long long
5     //fread->read
6
7     bool IOerror=0;
8     inline char nc(){
9         static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=buf+BUF_SIZE;
10        if (p1==pend){
11            p1=buf; pend=buf+fread(buf,1,BUF_SIZE,stdin);
12            if (pend==p1){IOerror=1;return -1;}
13            //{printf("IO error!\n");system("pause");for (;;);exit(0);}
14        }
15        return *p1++;
16    }
17    inline bool blank(char ch){return ch==' '||ch=='\n'||ch=='\r'||ch=='\t';}
18    inline void read(int &x){
19        bool sign=0; char ch=nc(); x=0;
20        for (;blank(ch);ch=nc());
21        if (IOerror)return;
22        if (ch=='-')sign=1,ch=nc();
23        for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
24        if (sign)x=-x;
25    }
26    inline void read(ll &x){
27        bool sign=0; char ch=nc(); x=0;
28        for (;blank(ch);ch=nc());
29        if (IOerror)return;
30        if (ch=='-')sign=1,ch=nc();
31        for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
32        if (sign)x=-x;
33    }
34    inline void read(double &x){
35        bool sign=0; char ch=nc(); x=0;
36        for (;blank(ch);ch=nc());
37        if (IOerror)return;
38        if (ch=='-')sign=1,ch=nc();
39        for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
40        if (ch=='.'){
41            double tmp=1; ch=nc();
42            for (;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(ch-'0');
43        }
44        if (sign)x=-x;
45    }
46    inline void read(char *s){
47        char ch=nc();
48        for (;blank(ch);ch=nc());
49        if (IOerror)return;
50        for (;!blank(ch)&&!IOerror;ch=nc())*s++=ch;
51        *s=0;
52    }
53    inline void read(char &c){
```





```

106 inline void print(int x){Ostream.print(x);}
107 inline void println(int x){Ostream.println(x);}
108 inline void print(char x){Ostream.out(x);}
109 inline void println(char x){Ostream.out(x);Ostream.out('\n');}
110 inline void print(ll x){Ostream.print(x);}
111 inline void println(ll x){Ostream.println(x);}
112 inline void print(double x,int y){Ostream.print(x,y);}
113 inline void println(double x,int y){Ostream.println(x,y);}
114 inline void print(char *s){Ostream.print(s);}
115 inline void println(char *s){Ostream.println(s);}
116 inline void println(){Ostream.out('\n');}
117 inline void flush(){Ostream.flush();}
118 #undef ll
119 #undef OUT_SIZE
120 #undef BUF_SIZE
121 };
122

```

---

### 1.2.3 保留小数.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const double pi = acos(-1.0);
4 int main(void)
5 {
6     for(int i = 0;i < 5; ++i)
7         printf("%.5f\n",i,pi);
8     for(int i = 0;i < 5; ++i)
9         cout<<setiosflags(ios::fixed)<<setprecision(i)<<pi<<endl;
10    return 0;
11 }

```

---

### 1.2.4 读取整数.cpp

```

1 //读取正负整数
2 inline int input(void)
3 {
4     int num = 0;
5     char c;
6     int flag = 0;
7     while((c = getchar()) < '0' || c > '9') flag = c=='-' ? 1:flag;
8     while(c >= '0' && c <= '9')
9         num = num * 10 + c - '0',c = getchar();
10    if(flag) num = -num;
11    return num;
12 }

```

---

## 1.3 测量程序的运行时间.cpp

```

1 clock_t start,end;
2 start = clock();
3 end = clock();

```

---

```

4     dur = double(end - start);
5     printf("Use Time: %f\n", (dur/CLOCKS_PER_SEC));

```

---

## 1.4 转化成二进制.cpp

---

```

1 void To_string_base2(LL n,string &s){
2     while(n > 0){
3         if(n&1)
4             s += "1";
5         else
6             s += "0";
7         n >>= 1;
8     };
9     reverse(s.begin(),s.end());
10 }
11 // nn 是要转化的数, ss 是 string, n 转化成多少位 2 进制
12 void To_string_base2_n(LL nn,string &ss,int n){
13     ss.clear();
14     To_string_base2(nn,ss);
15     while((int)ss.size() < n)
16         ss = "0"+ss;
17 }

```

---

## 2 几何

### 2.1 2D

#### 2.1.1 8 旋转卡壳.cpp

---

```

1 //2017-2018 ACM-ICPC Southwestern European Regional Programming Contest (SWERC
  ↳ 2017)
2 //K      Blowing Candles
3 // 求包含所有点的两条平行线之间的最短距离
4 #include <stdio.h>
5 #include <string.h>
6 #include <stdlib.h>
7 #include <math.h>
8 #define INF 99999999999999.9
9 #define PI acos(-1.0)
10 struct Point
11 {
12     double x, y, dis;
13 }pt[200005], stack[200005], p0;
14 int top, tot;
15 //计算几何距离
16 double Dis(double x1, double y1, double x2, double y2)
17 {
18     return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
19 }
20 //极角比较, 返回-1: p0p1 在 p0p2 的右侧, 返回 0:p0,p1,p2 共线
21 int Cmp_PolarAngel(struct Point p1, struct Point p2, struct Point pb)
22 {
23     double delta=(p1.x-pb.x)*(p2.y-pb.y)-(p2.x-pb.x)*(p1.y-pb.y);
24     if (delta<0.0) return 1;

```

```

25     else if (delta==0.0) return 0;
26     else return -1;
27 }
28 // 判断向量 p2p3 是否对 p1p2 构成左旋
29 bool Is_LeftTurn(struct Point p3, struct Point p2, struct Point p1)
30 {
31     int type=Cmp_PolarAngel(p3, p1, p2);
32     if (type<0) return true;
33     return false;
34 }
35 //先按极角排, 再按距离由小到大排
36 int Cmp(const void*p1, const void*p2)
37 {
38     struct Point*a1=(struct Point*)p1;
39     struct Point*a2=(struct Point*)p2;
40     int type=Cmp_PolarAngel(*a1, *a2, p0);
41     if (type<0) return -1;
42     else if (type==0)
43     {
44         if (a1->dis<a2->dis) return -1;
45         else if (a1->dis==a2->dis) return 0;
46         else return 1;
47     }
48     else return 1;
49 }
50 //求凸包
51 void Hull(int n)
52 {
53     int i, k;
54     p0.x=p0.y=INF;
55     for (i=0;i<n;i++)
56     {
57         scanf("%lf %lf",&pt[i].x, &pt[i].y);
58         if (pt[i].y < p0.y)
59         {
60             p0.y=pt[i].y;
61             p0.x=pt[i].x;
62             k=i;
63         }
64         else if (pt[i].y==p0.y)
65         {
66             if (pt[i].x<p0.x)
67             {
68                 p0.x=pt[i].x;
69                 k=i;
70             }
71         }
72     }
73     pt[k]=pt[0];
74     pt[0]=p0;
75     for (i=1;i<n;i++)
76         pt[i].dis=Dis(pt[i].x,pt[i].y, p0.x,p0.y);
77     qsort(pt+1, n-1, sizeof(struct Point), Cmp);
78     //去掉极角相同的点
79     tot=1;
80     for (i=2;i<n;i++)

```

```

81         if (Cmp_PolarAngel(pt[i], pt[i-1], p0))
82             pt[tot++]=pt[i-1];
83     pt[tot++]=pt[n-1];
84     //求凸包
85     top=1;
86     stack[0]=pt[0];
87     stack[1]=pt[1];
88     for (i=2;i<tot;i++)
89     {
90         while (top>=1 && Is_LeftTurn(pt[i], stack[top], stack[top-1])==false)
91             top--;
92         stack[++top]=pt[i];
93     }
94 }
95 //计算叉积
96 double CrossProduct(struct Point p1, struct Point p2, struct Point p3)
97 {
98     return (p1.x-p3.x)*(p2.y-p3.y)-(p2.x-p3.x)*(p1.y-p3.y);
99 }
100 //卡壳旋转, 求出凸多边形所有对踵点
101 double h1(double a,double b,double c)
102 {
103     double p=(a+b+c)/2.0;
104     return sqrt(p*(p-a)*(p-b)*(p-c));
105 }
106 double dist(Point a,Point b)
107 {
108     return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
109 }
110 void Rotate(struct Point*ch, int n)
111 {
112     int i, p=1;
113     double t1, t2, ans=INF, dif;
114     ch[n]=ch[0];
115     for (i=0;i<n;i++)
116     {
117         //如果下一个点与当前边构成的三角形的面积更大, 则说明此时不构成对踵点
118         while (fabs(CrossProduct(ch[i],ch[i+1],ch[p+1])) >
119             ↪ fabs(CrossProduct(ch[i],ch[i+1],ch[p])))
120             p=(p+1)%n;
121         dif=fabs(CrossProduct(ch[i],ch[i+1],ch[p+1])) -
122             ↪ fabs(CrossProduct(ch[i],ch[i+1],ch[p]));
123         //如果当前点和下一个点分别构成的三角形面积相等, 则说明两条边即为平行线, 对角线两
124         ↪ 端都可能是对踵点
125         ↪ t1=h1(dist(ch[i],ch[i+1]),dist(ch[i+1],ch[p]),dist(ch[p],ch[i]))*2.0/dist(ch[i],ch[i+1]));
126         if (t1<ans)ans=t1;
127     }
128     printf("%.15lf\n",ans);
129 }
130 int main (void)
131 {
132     int n;
133     scanf("%d%d",&n);
134     Hull(n);

```

```

133     Rotate(stack, top+1);
134     return 0;
135 }

```

---

## 2.1.2 PSLG.cpp

---

```

1  typedef vector<Point> Polygon;
2  double PolygonArea(Polygon poly)
3  {
4      double area = 0;
5      int n = poly.size();
6      for(int i = 1; i < n-1; i++)
7          area += Cross(poly[i]-poly[0], poly[(i+1)%n]-poly[0]);
8      return area/2;
9  }
10
11 struct Edge
12 {
13     int from, to; // 起点, 终点, 左边的面编号
14     double ang;
15     Edge(int f, int t, double a):from(f),to(t),ang(a) {}
16 };
17
18 const int maxn = 10000 + 10; // 最大边数
19
20 // 平面直线图 (PSGL) 实现
21 struct PSLG
22 {
23     int n, m, face_cnt; // face_cnt 面数
24     double x[maxn], y[maxn];
25     vector<Edge> edges; // 储存边
26     vector<int> G[maxn]; // 指向边
27     int vis[maxn*2]; // 每条边是否已经访问过
28     int left[maxn*2]; // 左面的编号
29     int prev[maxn*2]; // 相同起点的上一条边 (即顺时针旋转碰到的下一条边) 的编号
30
31     vector<Polygon> faces; // faces 储存面
32     double area[maxn]; // 每个 polygon 的面积
33
34     void init(int n)
35     {
36         this->n = n;
37         for(int i = 0; i < n; i++)
38             G[i].clear();
39         edges.clear();
40         faces.clear();
41     }
42
43     // 有向线段 from->to 的极角
44     double getAngle(int from, int to)
45     {
46         return atan2(y[to]-y[from], x[to]-x[from]);
47     }
48
49     void AddEdge(int from, int to)

```

```

50 {
51     edges.push_back((Edge){ from, to, getAngle(from, to)});
52     edges.push_back((Edge){ to, from, getAngle(to, from)});
53     m = edges.size();
54     G[from].push_back(m-2);
55     G[to].push_back(m-1);
56 }
57
58 // 找出 faces 并计算面积
59 void Build()
60 {
61     for(int u = 0; u < n; u++)
62     {
63         // 给从 u 出发的各条边按极角排序
64         int d = G[u].size();
65         for(int i = 0; i < d; i++)
66             for(int j = i+1; j < d; j++) // 这里偷个懒, 假设从每个点出发的线段不
                ↪ 会太多
67                 if(edges[G[u][i]].ang > edges[G[u][j]].ang)
68                     swap(G[u][i], G[u][j]);
69         for(int i = 0; i < d; i++)
70             prev[G[u][(i+1)%d]] = G[u][i];
71     }
72
73     memset(vis, 0, sizeof(vis));
74     face_cnt = 0;
75     for(int u = 0; u < n; u++)
76         for(int i = 0; i < G[u].size(); i++)
77         {
78             int e = G[u][i];
79             if(!vis[e]) // 逆时针找圈
80             {
81                 face_cnt++;
82                 Polygon poly;
83                 for(;;)
84                 {
85                     vis[e] = 1;
86                     left[e] = face_cnt;
87                     int from = edges[e].from;
88                     poly.push_back(Point(x[from], y[from]));
89                     e = prev[e^1];
90                     if(e == G[u][i])
91                         break;
92                     assert(vis[e] == 0);
93                 }
94                 faces.push_back(poly);
95             }
96         }
97
98     for(int i = 0; i < faces.size(); i++)
99     {
100         area[i] = PolygonArea(faces[i]);
101     }
102 }
103 };

```

### 2.1.3 二维几何模板.cpp

```
1 #include <bits/stdc++.h>
2 #define mem(ar,num) memset(ar,num,sizeof(ar))
3 #define me(ar) memset(ar,0,sizeof(ar))
4 #define lowbit(x) (x&(-x))
5 #define forn(i,n) for(int i = 0;i < n; ++i)
6 using namespace std;
7 typedef long long LL;
8 typedef unsigned long long ULL;
9 const int prime = 999983;
10 const int INF = 0x7FFFFFFF;
11 const LL INFF =0x7FFFFFFFFFFFFFFF;
12 const double pi = acos(-1.0);
13 const double inf = 1e18;
14 const double eps = 1e-10;
15 const LL mod = 1e9 + 7;
16 struct Point
17 {
18     double x,y;
19
20     Point(double x = 0,double y = 0):x(x),y(y) {}
21
22 };
23 typedef Point Vector;
24 Vector operator + (Vector A,Vector B)
25 {
26     return Vector(A.x + B.x,A.y + B.y);
27 }
28 Vector operator - (Vector A,Vector B)
29 {
30     return Vector(A.x-B.x,A.y-B.y);
31 }
32 Vector operator / (Vector A,double p)
33 {
34     return Vector(A.x/p,A.y/p);
35 }
36 Vector operator * (Vector A,double p)
37 {
38     return Vector(A.x*p,A.y*p);
39 }
40 double angle(Vector v)//求向量的角度从 0 到 2*pi
41 {
42     return atan2(v.y,v.x);
43 }
44 int dcmp(double x)
45 {
46     if(fabs(x)<eps)
47         return 0;
48     else
49         return x < 0?-1:1;
50 }
51 bool operator < (const Point &a,const Point &b)
52 {
53     if(dcmp(a.x-b.x)==0)
```

```

54         return a.y<b.y;
55     else
56         return a.x<b.x;
57 }
58
59
60 bool operator == (const Point &a,const Point &b)
61 {
62     return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y);
63 }
64 double Dot(Vector A,Vector B)
65 {
66     return A.x*B.x+A.y*B.y;
67 }
68 double Length(Vector A)
69 {
70     return sqrt(A.x*A.x+A.y*A.y);
71 }
72 double Angle(Vector A,Vector B)
73 {
74     return acos(Dot(A,B)/Length(A)/Length(B));
75 }
76 double Cross(Vector A,Vector B)
77 {
78     return A.x*B.y - A.y*B.x;
79 }
80 double Area2(Point A,Point B,Point C)
81 {
82     return Cross(B-A,C-A);
83 }
84 Vector Rotate(Vector A,double rad)
85 {
86     return Vector (A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
87 }
88 Vector Normal(Vector A)//单位法线
89 {
90     double L = Length(A);
91     return Vector(-A.y/L,A.x/L);
92 }
93 //调用前确保直线有唯一交点, 当且仅当 Cross(v,w) 非 0
94 Point Get_Line_Intersection(Point P,Vector v,Point Q,Vector w)
95 {
96     Vector u = P - Q;
97     double t = Cross(w,u)/Cross(v,w);
98     return P+v*t;
99 }
100 double Distance_To_Line(Point P,Point A,Point B)//点到直线的距离
101 {
102     Vector v1 = B-A,v2 = P-A;
103     return fabs(Cross(v1,v2)/Length(v1));
104 }
105 double Distance_To_Segment(Point P,Point A,Point B)
106 {
107     if(A==B)
108         return Length(P-A);
109     Vector v1 = B-A,v2 = P-A,v3 = P-B;

```



```

110     if(dcmp(Dot(v1,v2))<0)
111         return Length(v1);
112     else if(dcmp(Dot(v1,v3))>0)
113         return Length(v3);
114     else
115         return fabs(Cross(v1,v2))/Length(v1);
116 }
117 Point Get_Line_Projection(Point P,Point A,Point B)//求投影点
118 {
119     Vector v = B- A;
120     return A + v*(Dot(v,P-A)/Dot(v,v));
121 }
122 //线段相交判定 相交不在线段的端点
123 bool Segment_Proper_Intersection(Point a1,Point a2,Point b1,Point b2)
124 {
125     double c1 = Cross(a2-a1,b1-a1),c2 = Cross(a2-a1,b2-a1),
126            c3 = Cross(b2-b1,a2-b1),c4 = Cross(b2-b1,a1-b1);
127     return dcmp(c1)*dcmp(c2)<0&&dcmp(c3)*dcmp(c4)<0;
128 }
129 //判断点是否在线段上 (不包括端点)
130 bool Onsegment(Point p,Point a1,Point a2)
131 {
132     return dcmp(Cross(a1-p,a2-p))==0&&dcmp(Dot(a1-p,a2-p))<0;
133 }

```

---

#### 2.1.4 二维凸包.cpp

---

```

1 //计算凸包,输入点数组 p, 个数为 p, 输出点数组为 ch。函数返回凸包顶点数
2 //输入不能有重复节点
3 //如果精度要求搞需要用 dcmp 判断
4 //如果不希望在边上右点, 需要将 <= 改为 <
5 int ConvexHull(Point *p,int n ,Point *ch)
6 {
7     sort(p,p+n);
8     int m = 0;
9     for(int i = 0;i < n; ++i)
10     {
11         while(m>1&& Cross(ch[m-1]-ch[m-2],p[i]-ch[m-2])<=0) m--;
12         ch[m++] = p[i];
13     }
14     int k = m;
15     for(int i = n-2; i >= 0; --i)
16     {
17         while(m > k&& Cross(ch[m-1]-ch[m-2],p[i]-ch[m-2]) <= 0) m--;
18         ch[m++] = p[i];
19     }
20     if(n > 1) m--;
21     return m;
22 }
23 }

```

---

### 2.1.5 判断点是否在多边形内.cpp

```
1 typedef vector<Point> Polygon;
2 int isPointInPolygon(Point p,Polygon poly)
3 {
4     int n = poly.size();
5     int wn = 0;
6     for(int i = 0;i < n; ++i)
7     {
8         if(Onsegment(p,poly[i],poly[(i+1)%n])) return -1;
9         int k = dcmp(Cross(poly[(i+1)%n]-poly[i],p-poly[i]));
10        int d1 = dcmp(poly[i].y-p.y);
11        int d2 = dcmp(poly[(i+1)%n].y-p.y);
12        if(k>0&&d1 <= 0&&d2 > 0) wn ++;
13        if(k<0&&d2 <= 0&&d1 > 0) wn --;
14    }
15    if(wn != 0) return 1;
16    return 0;
17 }
```

### 2.1.6 圆与多边形相交的面积.cpp

```
1 #include <iostream>
2 #include <cstdio>
3 #include <string>
4 #include <cmath>
5 #include <iomanip>
6 #include <ctime>
7 #include <climits>
8 #include <cstdlib>
9 #include <cstring>
10 #include <algorithm>
11 #include <queue>
12 #include <vector>
13 #include <set>
14 #include <map>
15 using namespace std;
16 typedef unsigned int UI;
17 typedef long long LL;
18 typedef unsigned long long ULL;
19 typedef long double LD;
20 const double pi = acos(-1.0);
21 const double e = exp(1.0);
22 const double eps = 1e-8;
23 const int maxn = 400;
24 double x, y, h;
25 double vx, vy;
26 double R;
27 int n;
28 struct point
29 {
30     double x, y;
31     point(double _x=0.0, double _y=0.0)
32         : x(_x), y(_y) {}
33 }
```

```

33     point operator - (const point & p)
34     {
35         return point(x-p.x, y-p.y);
36     }
37     double sqrx()
38     {
39         return sqrt(x*x+y*y);
40     }
41 } p[maxn];
42
43 double xmult(point & p1, point & p2, point & p0);
44 double distancex(point & p1, point & p2);
45 point intersection(point u1, point u2, point v1, point v2);
46 void intersection_line_circle(point c, double r, point l1, point l2, point & p1,
    ↪ point & p2);
47 point ptoseg(point p, point l1, point l2);
48 double distp(point & a, point & b);
49 double Direct_Triangle_Circle_Area(point a, point b, point o, double r);
50
51
52 double xmult(point & p1, point & p2, point & p0)
53 {
54     return (p1.x-p0.x)*(p2.y-p0.y)-(p1.y-p0.y)*(p2.x-p0.x);
55 }
56
57 double distancex(point & p1, point & p2)
58 {
59     return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
60 }
61
62 point intersection(point u1, point u2, point v1, point v2)
63 {
64     point ret = u1;
65     double t = ((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
66             / ((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
67     ret.x += (u2.x-u1.x)*t;
68     ret.y += (u2.y-u1.y)*t;
69     return ret;
70 }
71
72 void intersection_line_circle(point c, double r, point l1, point l2, point & p1,
    ↪ point & p2)
73 {
74     point p = c;
75     double t;
76     p.x += l1.y-l2.y;
77     p.y += l2.x-l1.x;
78     p = intersection(p, c, l1, l2);
79     t = sqrt(r*r-distancex(p, c)*distancex(p, c))/distancex(l1, l2);
80     p1.x = p.x+(l2.x-l1.x)*t;
81     p1.y = p.y+(l2.y-l1.y)*t;
82     p2.x = p.x-(l2.x-l1.x)*t;
83     p2.y = p.y-(l2.y-l1.y)*t;
84 }
85
86 point ptoseg(point p, point l1, point l2)

```

```

87 {
88     point t = p;
89     t.x += l1.y-l2.y;
90     t.y += l2.x-l1.x;
91     if (xmult(l1, t, p)*xmult(l2, t, p)>eps)
92         return distancex(p, l1)<distancex(p, l2) ? l1 : l2;
93     return intersection(p, t, l1, l2);
94 }
95
96 double distp(point & a, point & b)
97 {
98     return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
99 }
100
101 double Direct_Triangle_Circle_Area(point a, point b, point o, double r)
102 {
103     double sign = 1.0;
104     a = a-o;
105     b = b-o;
106     o = point(0.0, 0.0);
107     if (fabs(xmult(a, b, o)) < eps)
108         return 0.0;
109     if (distp(a, o) > distp(b, o))
110     {
111         swap(a, b);
112         sign = -1.0;
113     }
114     if (distp(a, o) < r*r+eps)
115     {
116         if (distp(b, o) < r*r+eps)
117             return xmult(a, b, o)/2.0*sign;
118         point p1, p2;
119         intersection_line_circle(o, r, a, b, p1, p2);
120         if (distancex(p1, b) > distancex(p2, b))
121             swap(p1, p2);
122         double ret1 = fabs(xmult(a, p1, o));
123         double ret2 = acos((p1.x*b.x+p1.y*b.y)/p1.sqrn()/b.sqrn())*r*r;
124         double ret = (ret1+ret2)/2.0;
125         if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
126             ret = -ret;
127         return ret;
128     }
129     point ins = ptoseg(o, a, b);
130     if (distp(o, ins)>r*r-eps)
131     {
132         double ret = acos((a.x*b.x+a.y*b.y)/a.sqrn()/b.sqrn())*r*r/2.0;
133         if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
134             ret = -ret;
135         return ret;
136     }
137     point p1, p2;
138     intersection_line_circle(o, r, a, b, p1, p2);
139     double cm = r/(distancex(o, a)-r);
140     point m = point((o.x+cm*a.x)/(1+cm), (o.y+cm*a.y)/(1+cm));
141     double cn = r/(distancex(o, b)-r);
142     point n = point((o.x+cn*b.x)/(1+cn), (o.y+cn*b.y)/(1+cn));

```

```

143     double ret1 = acos((m.x*n.x+m.y*n.y)/m.sqrn()/n.sqrn())*r*r;
144     double ret2 =
        ↪  acos((p1.x*p2.x+p1.y*p2.y)/p1.sqrn()/p2.sqrn())*r*r-fabs(xmult(p1, p2,
        ↪  o));
145     double ret = (ret1-ret2)/2.0;
146     if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
147         ret = -ret;
148     return ret;
149 }
150 double Inter(double x,double y,double R,int n,point *area){
151     area[n] = area[0];
152     point temp = point(x, y);
153     double sum = 0;
154     for (int i=0; i<n-1; i++)
155         sum += Direct_Triangle_Circle_Area(area[i], area[i+1], temp, R);
156
157     sum += Direct_Triangle_Circle_Area(area[n-1], area[0], temp, R);
158     return fabs(sum);
159 }
160 double Cross(point A,point B)
161 {
162     return A.x*B.y - A.y*B.x;
163 }
164 int N,M;
165 double PolygonArea (point * p,int n)
166 {
167     double area = 0;
168     for(int i = 1; i < n - 1; ++i)
169     {
170         area += Cross(p[i]-p[0],p[i+1]-p[0]);
171     }
172     return fabs(area/2);
173 }
174
175 int dcmp(double x)
176 {
177     if(fabs(x)<eps)
178         return 0;
179     else
180         return x < 0?-1:1;
181 }
182 double S ;
183 double xi,yi,P,Q;
184 bool check(double R){
185     //      cout<<xi<<" "<<yi<<" "<<P<<" "<<Q<<endl;
186     //      printf("r = %lf Intersect = %lf\n",R,Inter(xi,yi,R,N,p) );
187     //      printf("%lf\n",(1-P/Q)*S);
188     return dcmp(Inter(xi,yi,R,N,p) - (1-P/Q)*S) > 0;
189 }
190 int main()
191 {
192
193     cin>>N;
194     for(int i=0;i< N;i++)
195     {
196         scanf("%lf%lf",&p[i].x,&p[i].y);

```

```

197     }
198
199     S= PolygonArea(p,N);
200     //cout<<S<<endl;
201     cin>>M;
202     for(int i = 0;i < M; ++i){
203
204         scanf("%lf %lf %lf %lf",&xi,&yi,&P,&Q);
205
206         double l = 0,r = 1e6;
207         for(int j = 0;j < 100; ++j){
208             double mid = l+(r-l)/2;
209             if(check(mid))
210                 r = mid;
211             else
212                 l = mid;
213             // printf("%lf %lf\n",l,r);
214         }
215         printf("%.8lf\n",r);
216     }
217
218     return 0;
219 }

```

---

### 2.1.7 求圆与直线的交点.cpp

```

1 int getLineCircleIntersection(Point A, Point B, Point C, double r, double& t1,
   ↪ double& t2,vector<Point> &sol){
2     // 初始方程: (A.x + t(B.x - A.x) - C.x)^2 + (A.y + t(B.y - A.y) - C.y)^2 = r^2
3     // 整理得: (at + b)^2 + (ct + d)^2 = r^2
4     double a = B.x - A.x;
5     double b = A.x - C.x;
6     double c = B.y - A.y;
7     double d = A.y - C.y;
8     // 展开得: (a^2 + c^2)t^2 + 2(ab + cd)t + b^2 + d^2 - r^2 = 0, 即 et^2 + ft + g =
   ↪ 0
9     double e = a * a + c * c;
10    double f = 2 * (a * b + c * d);
11    double g = b * b + d * d - r * r;
12    double delta = f * f - 4 * e * g; // 判别式
13    if(dcmp(delta) < 0) return 0; // 相离
14    if(dcmp(delta) == 0){ // 相切
15        t1 = t2 = -f / (2 * e);
16        sol.push_back(A+(B-A)*t1);
17        return 1;
18    }
19    t1 = (-f - sqrt(delta)) / (2 * e);
20    t2 = (-f + sqrt(delta)) / (2 * e);
21    sol.push_back(A+(B-A)*t1);
22    sol.push_back(A+(B-A)*t2);
23    return 2;
24 }

```

---

## 2.2 3D

### 2.2.1 三维几何的基本操作.cpp

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 struct Point3
5 {
6     double x,y,z;
7     Point3(double x = 0,double y = 0,double z = 0):x(x),y(y),z(z) {}
8 };
9 typedef Point3 Vector3;
10
11 Vector3 operator +(Vector3 v1,Vector3 v2)
12 {
13     return Vector3(v1.x+v2.x,v1.y+v2.y,v1.z+v2.z);
14 }
15 Vector3 operator -(Vector3 v1,Vector3 v2)
16 {
17     return Vector3(v1.x-v2.x,v1.y-v2.y,v1.z-v2.z);
18 }
19 Vector3 operator *(Vector3 v,double c)
20 {
21     return Vector3(v.x*c,v.y*c,v.z*c);
22 }
23 Vector3 operator /(Vector3 v,double c)
24 {
25     return Vector3(v.x/c,v.y/c,v.z/c);
26 }
27 double Dot(Vector3 A,Vector3 B)
28 {
29     return A.x*B.x+A.y*B.y+A.z*B.z;
30 }
31 double Length(Vector3 A)
32 {
33     return sqrt(Dot(A,A));
34 }
35 double Angle(Vector3 A,Vector3 B)
36 {
37     return acos(Dot(A,B)/(2*Length(A)*Length(B)));
38 }
39 double DistanceToPlane(const Point3 &p,const Point3 &p0,const Vector3& n)
40 {
41     return fabs(Dot(p-p0,n))/Length(n);
42 }
43 Point3 GetPlaneProjection(const Point3&p,const Point3&p0,const Vector3&n)
44 {
45     return p-n*Dot(p-p0,n);
46 }
47 //直线 p1-p2 到平面 p0-n 的交点。假定交点唯一存在
48 Point3 LinePlaneIntersection(Point3 p1,Point3 p2,Point3 p0,Vector3 n)
49 {
50     Vector3 v= p2 - p1;
51     //     /*if(dcmp(Dot(v,n))==0)
52     //     {
```

```

53 //          if(dcmp(Dot(p1-p0,n))==0)
54 //              直线在平面上
55 //          else
56 //              直线与平面平行
57 //      }
58 //      */
59     double t = Dot(n,p0-p1)/Dot(n,p2-p1);
60     return p1 + v*t;
61 }

```

---

### 2.2.2 三维几何的模版.cpp

```

1  #include <bits/stdc++.h>
2  const double eps = 1e-6;
3  using namespace std;
4
5  struct Point3
6  {
7      double x,y,z;
8      Point3(double x = 0,double y = 0,double z = 0):x(x),y(y),z(z) {}
9  };
10 typedef Point3 Vector3;
11 int dcmp(double d)
12 {
13     if(fabs(d)< eps)
14         return 0;
15     else
16         return d < 0?-1:1;
17 }
18 Vector3 operator +(Vector3 v1,Vector3 v2)
19 {
20     return Vector3(v1.x+v2.x,v1.y+v2.y,v1.z+v2.z);
21 }
22 Vector3 operator -(Vector3 v1,Vector3 v2)
23 {
24     return Vector3(v1.x-v2.x,v1.y-v2.y,v1.z-v2.z);
25 }
26 Vector3 operator *(Vector3 v,double c)
27 {
28     return Vector3(v.x*c,v.y*c,v.z*c);
29 }
30 Vector3 operator /(Vector3 v,double c)
31 {
32     return Vector3(v.x/c,v.y/c,v.z/c);
33 }
34 bool operator ==(Point3 A,Point3 B)
35 {
36     return !dcmp(A.x-B.x)&&!dcmp(A.y-B.y)&&!dcmp(A.z-B.z);
37 }
38 double Dot(Vector3 A,Vector3 B)
39 {
40     return A.x*B.x+A.y*B.y+A.z*B.z;
41 }
42 double Length(Vector3 A)
43 {

```



```

44     return sqrt(Dot(A,A));
45 }
46 double Angle(Vector3 A,Vector3 B)//求两向量的夹角
47 {
48     return acos(Dot(A,B)/(2*Length(A)*Length(B)));
49 }
50 double DistanceToplane(const Point3 &p,const Point3 &p0,const Vector3& n)//
51 {
52     return fabs(Dot(p-p0,n))/Length(n);
53 }
54 Point3 GetPlaneProjection(const Point3&p,const Point3&p0,const Vector3&n)
55 {
56     return p-n*Dot(p-p0,n);
57 }
58 //直线 p1-p2 到平面 p0-n 的交点。假定交点唯一存在
59 Point3 LinePlaneIntetsection(Point3 p1,Point3 p2,Point3 p0,Vector3 n)
60 {
61     Vector3 v= p2 - p1;
62     //     /*if(dcmp(Dot(v,n))==0)
63     //     {
64     //         if(dcmp(Dot(p1-p0,n))==0)
65     //             直线在平面上
66     //         else
67     //             直线与平面平行
68     //     }
69     //     */
70     double t = Dot(n,p0-p1)/Dot(n,p2-p1);
71     return p1 + v*t;
72 }
73 Point3 LinePlaneIntetsection(Point3 p1,Point3 p2,double A,double B,double C,double
↵ D)
74 {
75     Vector3 v = p2-p1;
76     double t =
↵ (A*p1.x+B*p1.y+C*p1.z+D)/(A*(p1.x-p2.x)+B*(p1.y-p2.y)+C*(p1.z-p2.z));
77     return p1 + v*t;
78 }
79 Vector3 Cross(Vector3 A,Vector3 B)
80 {
81     return Vector3(A.y*B.z-A.z*B.y,A.z*B.x-A.x*B.z,A.x*B.y-A.y*B.x);
82 }
83 double Area2(Point3 A,Point3 B,Point3 C)
84 {
85     return Length(Cross(B-A,C-A));
86 }
87 //已知平面的三点, 求出点法式
88 //Vector3 Solven(Point3 A,Point3 B,Point3 C)
89 //{
90 //    return Cross(B-A,C-A);
91 //}
92 //判断一个点是否在三角形内, 可以用面积法
93 bool PointInTri(Point3 P,Point3 A,Point3 B,Point3 C)
94 {
95     double area1 = Area2(P,A,B);
96     double area2 = Area2(P,A,C);
97     double area3 = Area2(P,B,C);

```

```

98     double area4 = Area2(A,B,C);
99     return dcmp(area1+area2+area3-area4)==0;
100 }
101 //判断线段是否与三角形相交
102 bool TriSegIntersection(Point3 P0,Point3 P1,Point3 P2,Point3 A,Point3 B,Point3 &P)
103 {
104     Vector3 n = Cross(P1-P0,P2-P0);
105
106     if(dcmp(Dot(n,B-A))==0)
107         return false;
108
109     double t = Dot(n,P0-A)/Dot(n,B-A);
110     if(dcmp(t) < 0 || dcmp(t-1) > 0)
111         return false;
112     P = A + (B-A) * t;
113     return PointInTri(P,P0,P1,P2);
114 }
115 double DistantceToLine(Point3 P,Point3 A,Point3 B)
116 {
117     return Length(Cross(A-P,B-P))/Length(A-B);
118 }
119 double DistanceToSegment(Point3 P,Point3 A,Point3 B)
120 {
121     if(A==B) return Length(P-A);
122     Vector3 v1 = B - A, v2 = P - A,v3 = P-B;
123     if(dcmp(Dot(v1,v2)) == 0) return Length(v2);
124     if(dcmp(Dot(v1,v3)) > 0) return Length(v3);
125     return Length(Cross(v1,v2))/Length(v1);
126 }
127 double Volume6(Point3 A,Point3 B,Point3 C,Point3 D)
128 {
129     return Dot(D-A,Cross(B-A,C-A));
130 }
131 //
132 int main(void)
133 {
134
135     Point3 A(0,0,0),B(0,100,0),C(100,0,0),D(25,25,0);
136     cout<<PointInTri(D,A,B,C)<<endl;
137     return 0;
138 }

```

---

### 2.2.3 三维凸包.cpp

```

1 struct Face{
2     int v[3];
3     Vector3 normal(Vector *P)
4     {
5         return Cross(P[v[1]]-P[v[0]],P[v[2]]-P[v[0]]);
6     }
7     int cansee(Point *P,int i)const
8     {
9         return Dot(P[i]-P[v[0]],normal(P)) > 0?1 : 0;
10    }
11 };

```

```

12 vector <Face> CH3D(Point3* P,int n)
13 {
14     vector <Face> cur;
15     cur.push_back((Face){0,1,2});
16     cur.push_back((Face){2,1,0});
17     for(int i = 3;i < n; ++i)
18     {
19         vector<Face> next;
20         //计算每条边“左面”的可见性
21         for(int j= 0;j < cur.size(); ++j)
22         {
23             Face &f = cur[j];
24             int res = f.cansee(P,i);
25             if(!res) next.push_back(f);
26             for(int k = 0;k < 3; ++k)
27                 vis[f.v[k]][f.v[(k+1)%3]] = res;
28         }
29         for(int j = 0;j < cur.size(); ++j)
30         {
31             for(int k = 0;k < 3; ++k)
32             {
33                 int a = cur[j].v[k],b = cur[j].v[(k+1)%3];
34                 if(vis[a][b] != vis[b][a]&&vis[a][b])//(a,b) 是分界线, 左边对 P[i]
35                     ↪ 可见
36                     next.push_back((Face){a,b,i});
37             }
38         }
39         cnr = next;
40     }
41     return cur;
42 }
43 double rand01() {return rand() / (double) RAND_MAX;}//0-1 的随机数
44 double randeps() {return (rand01()-0.5) * eps;}
45 Point3 add_noise(Point3 p)
46 {
47     return Point3(p.x + randeps(),p.y+randeps(),p.z+randeps());
48 }
49 //.....
50 struct Face{
51     int v[3];
52     Vector3 normal(Vector *P)
53     {
54         return Cross(P[v[1]]-P[v[0]],P[v[2]]-P[v[0]]);
55     }
56     int cansee(Point *P,int i)const
57     {
58         return Dot(P[i]-P[v[0]],normal(P)) > 0?1 : 0;
59     }
60 };
61 vector <Face> CH3D(Point3* P,int n)
62 {
63     vector <Face> cur;
64     cur.push_back((Face){0,1,2});
65     cur.push_back((Face){2,1,0});
66     for(int i = 3;i < n; ++i)

```

```

67     {
68         vector<Face> next;
69         //计算每条边“左面”的可见性
70         for(int j= 0;j < cur.size(); ++j)
71         {
72             Face &f = cur[j];
73             int res = f.cansee(P,i);
74             if(!res) next.push_back(f);
75             for(int k = 0;k < 3; ++k)
76                 vis[f.v[k]][f.v[(k+1)%3]] = res;
77         }
78         for(int j = 0;j < cur.size(); ++j)
79         {
80             for(int k = 0;k < 3; ++k)
81             {
82                 int a = cur[j].v[k],b = cur[j].v[(k+1)%3];
83                 if(vis[a][b] != vis[b][a]&&vis[a][b])//(a,b) 是分界线, 左边对 P[i]
84                     ↪ 可见
85                     next.push_back((Face){a,b,i});
86             }
87         }
88         cnr = next;
89     }
90     return cur;
91 }
92 double rand01() {return rand() / (double) RAND_MAX;}//0-1 的随机数
93 double randeps() {return (rand01()-0.5) * eps;}
94 Point3 add_noise(Point3 p)
95 {
96     return Point3(p.x + randeps(),p.y+randeps(),p.z+randeps());
97 }

```

#### 2.2.4 维度转换为三维坐标.cpp

```

1 // 经纬度转换为球坐标
2 double torad(double deg)
3 {
4     return deg/180*acos(-1);
5 }
6 void get_coordinate(double R,double lat,double lng,double &x,double &y,double &z)
7 {
8     lat = torad(lat);
9     lng = torad(lng);
10    x = R*cos(lat)*cos(lng);
11    y = R*cos(lat)*sin(lng);
12    z = R*sin(lat);
13 }

```

## 3 动态规划

### 3.1 1 单调队列.cpp

```

1 //https://ac.nowcoder.com/acm/contest/223/C
2 //C      区间区间区间

```

```

3 //$$ v_{l,r} = max(a_i-a_j) (1 <= i,j <= r)$$
4 //$$ \sum_{i=1}^n \sum_{j=1}^n v_{i,j}$$
5 const int maxn = 1e5+100;
6 int a[maxn];
7 int s[maxn]; // 单调栈
8 // 第一遍求在这个区间里面最大
9 int pre[maxn];
10 int nxt[maxn];
11 int main(void)
12 {
13     int T,n;
14     cin>>T;
15     while(T--){
16         scanf("%d",&n);
17         for(int i = 1;i <= n; ++i){
18             scanf("%d",&a[i]);
19         }
20         int t = 0;
21         for(int i = 1;i <= n; ++i){
22             pre[i] = nxt[i] = 0;
23             while(t > 0&&a[i] > a[s[t]]) nxt[s[t]] = i,t--;
24             pre[i] = s[t];
25             s[++t] = i;
26             // cout<<pre[i]<<" ";
27         }
28         while(t > 0)
29             nxt[s[t]] = n+1,t--;
30         LL ans = 0;
31         for(int i = 1;i <= n; ++i){
32             ans += 1ll*a[i]*(nxt[i]-i)*(i-pre[i]);
33         }
34         t = 0;
35         for(int i = 1;i <= n; ++i){
36             pre[i] = nxt[i] = 0;
37             while(t > 0&&a[i] < a[s[t]]) nxt[s[t]] = i,t--;
38             pre[i] = s[t];
39             s[++t] = i;
40         }
41         while(t > 0)
42             nxt[s[t]] = n+1,t--;
43         for(int i = 1;i <= n; ++i){
44             ans -= 1ll*a[i]*(nxt[i]-i)*(i-pre[i]);
45         }
46         printf("%lld\n",ans);
47     }
48
49     return 0;
50 }

```

### 3.2 1 最长上升子序列.cpp

---

```

1 //最长上升子序列 The longest increasing sequence
2
3 template <class It>
4 int n_lisLength(It begin,It end)

```

```

5 {
6     typedef typename iterator_traits<It>::value_type T;
7     T inf = 1<<30;
8     vector<T> best(end-begin,inf);
9     for(It i = begin; i != end; ++i)
10         *lower_bound(best.begin(),best.end(),*i) = *i;
11     return lower_bound(best.begin(),best.end(),inf) - best.begin();
12 }
13

```

---

### 3.3 string dp

#### 3.3.1 trie+dp.cpp

---

```

1  /*
2
3  Margot 有一个 长度为字符串 aa, 给定 nn 个子串,
4  每一个子串一个价值 wi, 从原串中取出一个子串后,
5  原串的左右结合组合成一个新的串,
6  并且得到改子串的价值 wi。问能取到的最大价值
7  */
8  // SWERC 2017 D candy
9  #include<bits/stdc++.h>
10
11 using namespace std;
12 const int maxn = 55;
13 const int maxm = 11000;// 200 个串 200*50 tire 树节点
14
15 inline void up(int &a,int b){
16     a<b?(a=b):0;
17 }
18
19 // tire 树
20 const int maxnode = 4e5+100;
21 const int sigma_size = 26;
22 struct Trie
23 {
24     int ch[maxnode][sigma_size];
25     int val[maxnode];
26     int sz;
27     Trie()
28     {
29         sz = 1;
30         memset(ch[0],0,sizeof(ch[0]));
31         memset(val,-1,sizeof(val));
32     }
33     int idx(char c)
34     {
35         return c-'a';
36     }
37     void insert(char *s,int v)
38     {
39         int u = 0, n = strlen(s);
40         for(int i = 0; i < n; ++i)
41         {

```

```

42         int c = idx(s[i]);
43         if(!ch[u][c])
44         {
45             memset(ch[sz],0,sizeof(ch[sz]));
46             //val[sz] = 0;
47             ch[u][c] = sz++;
48         }
49         u = ch[u][c];
50     }
51     up(val[u], v);
52 }
53 };
54
55 Trie tr;
56
57 int dp[maxn],f[maxn][maxn],g[maxn][maxm];
58 char ar[maxn];
59 char br[maxn];
60 int main(void){
61
62     scanf("%s",ar+1);
63     int n = strlen(ar+1);
64     for(int i = 1;i <= n; ++i)
65         ar[i] -= 'a';
66     int C;
67     scanf("%d",&C);
68     while(C--){
69         int u;
70         scanf("%s %d",br,&u);
71         int nn = strlen(br);
72         tr.insert(br,u);
73         reverse(br,br+nn);
74         tr.insert(br,u);
75     }
76
77     // 初始化
78     // for(int i = 1;i < tr.sz; ++i)
79     //     cout<<tr.val[i]<<" ";
80     // cout<<endl;
81     for(int i = 0;i <= n+1; ++i)
82         for(int j = 0;j <= n+1; ++j)
83             f[i][j] = -1;
84     for(int i = n; i; --i){
85         for(int j = i - 1;j <= n; ++j)
86             for(int k = 0;k < tr.sz; ++k)
87                 g[j][k] = -1;
88         // cout<<tr.sz<<endl;
89         g[i-1][0] = 0;
90         for(int j = i-1;j <= n; ++j){
91             for(int k = 0;k < tr.sz; ++k){
92                 if(~g[j][k]){// 我为人人递推
93                     for(int x = j+1;x <= n; ++x)
94                         if(~f[j+1][x])
95                             up(g[x][k],g[j][k]+f[j+1][x]);
96                     int y = tr.ch[k][(int)ar[j+1]];
97                     // cout<<y<<endl;

```

```

98         if(y != 0){
99             up(g[j+1][y],g[j][k]);
100             if(~tr.val[y]){
101                 // cout<<tr.val[y]<<endl;
102                 up(g[j+1][0],g[j][k]+tr.val[y]);
103             }
104         }
105         if(k == 0)
106             up(f[i][j],g[j][k]);
107     }
108 }
109 }
110 }
111
112
113 // cout<<f[1][n]<<endl;
114 for(int i = 1;i <= n; ++i){
115     dp[i] = dp[i-1];
116     for(int j = 1;j <= i; ++j)
117         if(~f[j][i])
118             up(dp[i],dp[j-1]+f[j][i]);
119 }
120 cout<<dp[n]<<endl;
121
122
123
124
125 return 0;
126 }

```

---

### 3.4 zhuangyadp

#### 3.4.1 1 多米诺骨牌覆盖.cpp

---

```

1  /* 状态压缩 dp+ 矩阵快速幂，用 1*2 的小方块填满 N*M 的矩形 */
2  //1033 骨牌覆盖 V2
3
4  #include<bits/stdc++.h>
5
6  using namespace std;
7  typedef long long LL;
8  const int maxn = 13;
9  const int mod = 1e9+7;
10 int n,m;
11 LL f[12][1<<11];
12 bool in_s[1<<11];
13
14 struct Matrix{
15     #define maxn 100
16     int n,m;
17     Matrix(int nn = 1,int mm = 1):n(nn),m(mm){ memset(a,0,sizeof(a));};
18     long long a[maxn][maxn];
19 };
20 void print(const Matrix &a)
21 {

```



```

22     for(int i = 1; i <= a.n; ++i, cout<<endl)
23         for(int j = 1; j <= a.m; ++j)
24             cout<<a.a[i][j]<<" ";
25 }
26 Matrix operator*(Matrix a, Matrix b)
27 {
28     assert(a.m == b.n);
29     Matrix c(a.n, b.m);
30     for(int i = 1; i <= a.n; ++i)
31     {
32         for(int j = 1; j <= b.m; ++j)
33         {
34             for(int k = 1; k <= a.m; ++k)
35             {
36                 c.a[i][j] += a.a[i][k] * b.a[k][j];
37                 c.a[i][j] %= mod;
38             }
39         }
40     }
41     // print(c);
42     return c;
43 }
44 Matrix B;
45 void solve(int m){
46     for(int i = 0; i < (1<<m); ++i){
47         bool cnt = 0, has_odd = 0;
48         for(int j = 0; j < m; ++j){
49             if(i >>j &1) has_odd |= cnt, cnt = 0;
50             else cnt ^= 1;
51             in_s[i] = has_odd | cnt?0:1;
52         }
53     }
54 }
55
56 // f[0][0] = 1;
57 // for(int i = 1; i <= n; ++i){
58     for(int j = 0; j < (1<<m); ++j){
59         // f[i][j] = 0;
60         for(int k = 0; k < (1<<m); ++k){
61             if((j&k) == 0&& in_s[j|k])
62                 B.a[j+1][k+1] = 1;
63             // f[i][j] += f[i-1][k];
64         }
65     }
66 }
67 // print(B);
68 // cout<<f[n][0]<<endl;
69 }
70
71
72 LL M, N;
73 int main(void){
74     scanf("%lld%lld", &M, &N);
75     B.n = B.m = 1<<N;
76     solve(N);
77     Matrix ans(1, 1<<N);

```

```

78
79     ans.a[1][1] = 1;
80     // print(ans);
81     // cout<<endl;
82     // print(B);
83     while(M > 0){
84         if(M & 1)
85             ans = ans*B;
86         B = B*B;
87         // cout<<endl;
88         // print(B);
89         M >>= 1;
90     }
91     cout<<ans.a[1][1]<<endl;
92
93     return 0;
94 }
95
96 /* 加强版
97 1*1 和 2*1 的小方块
98 SWERC2017 C - Macarons
99 搜索求状态 */
100
101
102 // 矩阵快速幂
103 // 注意修改 maxn 的值，要不然容易 T
104
105 const int maxn = 260;
106 int n;
107 struct Matrix{
108     int n,m;
109     Matrix(int nn = 1,int mm = 1):n(nn),m(mm){ memset(a,0,sizeof(a));};
110     int a[maxn][maxn];
111 };
112 void print(const Matrix &a)
113 {
114     for(int i = 1;i <= a.n; ++i,cout<<endl)
115         for(int j = 1;j <= a.m; ++j)
116             cout<<a.a[i][j]<<" ";
117 }
118 Matrix operator*(Matrix a,Matrix b)
119 {
120     Matrix c(a.n,b.m);
121     for(int i = 1;i <= a.n; ++i)
122     {
123         for(int j = 1;j <= b.m; ++j)
124         {
125             for(int k = 1;k <= a.m; ++k)
126             {
127                 c.a[i][j] = (1ll*c.a[i][j]+1ll*a.a[i][k] * b.a[k][j])%mod;
128             }
129         }
130     }
131     // print(c);
132     return c;
133 }

```

```

134 // 状态压缩
135
136 LL MM[maxn][maxn];
137 LL N,M;
138 // a 代表是 a 的递推, now 代表当前行的状态, nxt 代表下一行的状态
139 void dfs(int a,int now,int nxt){
140     // cout<<a<<endl;
141     int tmpnow = now,tmpnxt = nxt;
142     int one[10],two[10];
143     memset(one,0,sizeof(one));
144     memset(two,0,sizeof(two));
145     int cnt = 0;
146     while(tmpnow > 0){
147         one[cnt++] = tmpnow&1;
148         tmpnow >>= 1;
149     }
150     bool flag = true;
151     for(int i = 0;i < N; ++i){
152         if(!one[i]){
153             flag = false;
154             break;
155         }
156     }
157     if((now & NN) == NN){
158         MM[a][nxt]++;
159         return ;
160     }
161     cnt = 0;
162     while(tmpnxt > 0){
163         two[cnt++] = tmpnxt&1;
164         tmpnxt >>= 1;
165     }
166     for(int i = 0;i < N; ++i){
167         if(!one[i]){
168             dfs(a,now|(1<<i),nxt);
169             dfs(a,now|(1<<i),nxt|(1<<i));
170             if(i + 1 < N&& !one[i+1]){
171                 dfs(a,now|(1<<i)|(1<<(i+1)),nxt);
172             }
173             break;
174         }
175     }
176 }
177
178 int NN;
179 Matrix ans(NN,NN);
180 Matrix B(NN,NN);
181 void solve(){
182     B.n = B.m = ans.n = ans.m = NN;
183     for(int i = 1;i <= NN; ++i){
184         for(int j = 1;j <= NN; ++j)
185         {
186             B.a[i][j] = MM[i-1][j-1];
187         }
188     }
189 }

```

```

190
191     for(int i = 1; i <= NN; ++i) ans.a[i][i] = 1;
192     while(M > 0){
193         if(M & 1)
194             ans = ans*B;
195         B = B*B;
196         M >>= 1;
197     }
198     cout<<ans.a[1][1]<<endl;
199 }
200 int main(void)
201 {
202     scanf("%lld%lld",&N,&M);
203     // cout<<N<<" "<<M<<endl;
204     NN = 1<<N;
205     // cout<<N<<" "<<NN<<endl;
206     for(int i = 0; i < NN; ++i){
207         dfs(i,i,0);
208     }
209     solve();
210     return 0;
211 }

```

---

## 3.5 树上的分治

### 3.5.1 1 树的重心.cpp

---

```

1 // Size[u] 代表以节点 u 为根的子树节点个数
2 // dp[u] 代表去除 u 节点后最大子树的节点个数
3 const int maxn = 2e4+100;
4 vector<int> G[maxn];
5 int dp[maxn];
6 int Size[maxn];
7 int n;
8 int ans;
9 void dfs(int u,int fa){
10     dp[u] = Size[u] = 0;
11     for(int i = 0; i < G[u].size(); ++i){
12         if(fa==G[u][i])continue;
13         dfs(G[u][i],u);
14         // sum += tmp;
15         Size[u] += Size[G[u][i]];
16         dp[u] = max(dp[u],Size[G[u][i]]);
17     }
18     Size[u]++;
19     dp[u] = max(n-Size[u],dp[u]);
20     if(dp[u] < dp[ans]) ans = u;
21 }
22 int main(void)
23 {
24     int T;
25     cin>>T;
26     while(T--){
27         scanf("%d",&n);
28         for(int i = 1; i <= n; ++i) G[i].clear();

```

```

29         for(int i = 1; i <= n-1; ++i){
30             int u,v;
31             scanf("%d%d",&u,&v);
32             G[u].push_back(v);
33             G[v].push_back(u);
34         }
35         ans = 0;
36         dp[0] = INF;
37         dfs(1,-1);
38         printf("%d %d\n",ans,dp[ans]);
39     }
40     return 0;
41 }

```

---

## 4 图论

### 4.1 DFS

#### 4.1.1 1. 无向图的割点和桥.cpp

---

```

1  SPF POJ - 1523
2  // 如果有割点，那么割点与子节点边就是割边
3  int dfs(int u,int fa){
4      int lowu = pre[u] = ++dfs_clock;
5      int child = 0;
6      for(int i = 0; i < G[u].size(); ++i){
7          int v = G[u][i];
8          if(!pre[v]){
9              child++;
10             int lowv = dfs(v,u);
11             lowu = min(lowu,lowv);
12             if(lowv >= pre[u]){
13                 iscut[u]++;
14             }
15         }
16         else if(pre[v] < pre[u] && v != fa){
17             lowu = min(lowu,pre[v]);
18         }
19     }
20     if(fa < 0 && child == 1) iscut[u] = 0;
21     else if(fa < 0 && child >= 2) iscut[u] = child-1;
22     return low[u] = lowu;
23 }
24 如果要输出去掉割点之后的联通分量的个数，需要谈判根的情况
25 #include<iostream>
26 #include<cstdio>
27 #include<cctype>
28 #include<cstring>
29 #include<algorithm>
30 #include<vector>
31 #include<stack>
32 #include<map>
33 #include<queue>
34 #include<cmath>
35 #define mem(ar,num) memset(ar,num,sizeof(ar))

```

```

36 #define me(ar) memset(ar,0,sizeof(ar))
37 #define lowbit(x) (x&(-x))
38 #define Pb push_back
39 #define FI first
40 #define SE second
41 #define rep(i,a,n) for (int i=a;i<n;i++)
42 #define per(i,a,n) for (int i=n-1;i>=a;i--)
43 #define IOS ios::sync_with_stdio(false)
44 #define DEBUG cout<<endl<<"DEBUG"<<endl;
45 using namespace std;
46 typedef long long LL;
47 typedef unsigned long long ULL;
48 const int prime = 999983;
49 const int INF = 0x7FFFFFFF;
50 const LL INFF =0x7FFFFFFFFFFFFFFF;
51 const double pi = acos(-1.0);
52 const double inf = 1e18;
53 const double eps = 1e-6;
54 const LL mod = 1e9 + 7;
55 LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
56 LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
57 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
58 typedef pair<int,int> P;
59 const int maxn = 1000+100;
60 // const int maxm = 1e6+100
61 int pre[maxn];
62 int dfs_clock = 0;
63 vector<int> G[maxn];
64 int iscut[maxn];
65 int low[maxn];
66
67 void init(){
68     dfs_clock = 1;
69     rep(i,1,maxn) G[i].clear();
70     me(iscut);
71     me(low);
72     me(pre);
73 }
74 int dfs(int u,int fa){
75     int lowu = pre[u] = ++dfs_clock;
76     int child = 0;
77     for(int i = 0;i < G[u].size(); ++i){
78         int v = G[u][i];
79         if(!pre[v]){
80             child++;
81             int lowv = dfs(v,u);
82             lowu = min(lowu,lowv);
83             if(lowv >= pre[u]){
84                 iscut[u]++;
85             }
86         }
87         else if(pre[v] < pre[u] && v != fa){
88             lowu = min(lowu,pre[v]);
89         }
90     }
91     if(fa < 0&&child == 1) iscut[u] = 0;

```

```

92     else if(fa < 0&&child >= 2) iscut[u] = child-1;
93     return low[u] = lowu;
94 }
95 // #define Debug
96 int main(void)
97 {
98     #ifdef Debug
99     freopen("input.txt","r",stdin);
100    freopen("output.txt","w+",stdout);
101    #endif
102    int kase = 0;
103    while(1){
104        init();
105        int u,v;
106        int t = 0;
107        while(scanf("%d",&u)==1&&u != 0){
108            t++;
109            scanf("%d",&v);
110            G[u].Pb(v);
111            G[v].Pb(u);
112        }
113        if(t==0)break;
114        // rep(i,1,maxn) if(!G[i].empty()){
115
116        // dfs(i,-1);
117        // break;
118        // }
119        dfs(1,-1);
120        int num = 0;
121        rep(i,1,1001) if(iscut[i]) num++;
122
123        printf("Network #%d\n",++kase);
124        if(num > 0)
125        {
126            rep(i,1,1001) if(iscut[i]){
127                printf("  SPF node %d leaves %d subnets\n",i,iscut[i]+1);
128            }
129        }
130        else
131            printf("  No SPF nodes\n");
132        if(kase) puts("");
133    }
134
135    return 0;
136 }

```

---

#### 4.1.2 2. 无向图的双连通分量.cpp

```

1 // 无向图的点联通分量
2
3 const int maxn= 1000+10;
4 int pre[maxn],iscut[maxn],bccno[maxn],dfs_clock,bcc_cnt;
5 vector<int> G[maxn],bcc[maxn];
6
7 stack<Edge> S;

```

```

8  int dfs(int u,int fa){
9      int lowu = pre[u] = ++dfs_clock;
10     int child = 0;
11     for(int i = 0;i < G[u].size(); ++i){
12         int v = G[u][i];
13         Edge e = (Edge) {u,v};
14         if(!pre[v]){
15             S.push(e);
16             child++;
17             int lowv = dfs(v,u);
18             lowu = min(lowu,lowv);
19             if(lowv >= pre[u]){
20                 iscut[u] = true;
21                 bcc_cnt++;
22                 bcc[bcc_cnt].clear();
23                 for(;;){
24                     Edge x = S.top(); S.pop();
25                     if(bccno[x.u] != bcc_cnt) {bcc[bcc_cnt].push_back(x.u); bccno[x.u] =
26                         ↪ bcc_cnt;}
27                     if(bccno[x.v] != bcc_cnt) {bcc[bcc_cnt].push_back(x.v); bccno[x.v] =
28                         ↪ bcc_cnt;}
29                     if(x.u == u&& x.v == v) break;
30                 }
31             }
32             else if(pre[v] < pre[u]&& v != fa){
33                 S.push(e);lowu = min(pre[v],lowu);
34             }
35         }
36         if(fa < 0&& child == 1) iscut[u] = 0;
37         return lowu;
38     }
39     void find_bcc(int n){
40         memset(pre,0,sizeof(pre));
41         memset(iscut,0,sizeof(iscut));
42         memset(bccno,0,sizeof(bccno));
43         dfs_clock = bcc_cnt = 0;
44         for(int i = 0;i < n; ++i) if(!pre[i]) dfs(i,-1);
45     }
46 }
47
48 //无向图的边-双联通分量
49 // 第一边 dfs 求出所有的割边, 然后第二边 dfs 求出所有边-双联通分量 (不经过割边)

```

---

#### 4.1.3 3 有向图的强联通分量.cpp

---

```

1  // tarjan 算法
2  const int maxn = 2e4+100;
3
4  vector<int> G[maxn];
5  int pre[maxn],lowlink[maxn],sccno[maxn],dfs_clock,scc_cnt;
6  stack<int> S;
7  void dfs(int u){
8      pre[u] = lowlink[u] = ++dfs_clock;

```



```

9      S.push(u);
10     for(int i = 0; i < G[u].size(); ++i){
11         int v = G[u][i];
12         if(!pre[v]){
13             dfs(v);
14             lowlink[u] = min(lowlink[u], lowlink[v]);
15
16         }
17     else if(!sccno[v]){
18         lowlink[u] = min(lowlink[u], pre[v]);
19     }
20 }
21 if(lowlink[u] == pre[u]){
22     scc_cnt++;
23     for(;;){
24         int x = S.top(); S.pop();
25         sccno[x] = scc_cnt;
26         if(x == u) break;
27     }
28 }
29
30 }
31 void find_scc(int n){
32     dfs_clock = scc_cnt = 0;
33     me(sccno), me(pre);
34     rep(i, 0, n) if(!pre[i]) dfs(i);
35 }
36 // kosaraju
37
38
39
40 const int maxn = 2e4+100;
41 vector<int> G[maxn], G2[maxn];
42 vector<int> S;
43 int vis[maxn], sccno[maxn], scc_cnt;
44 void dfs1(int u){
45     if(vis[u]) return ;
46     vis[u] = 1;
47     for(int i = 0; i < G[u].size(); ++i) dfs1(G[u][i]);
48     S.push_back(u);
49 }
50 void dfs2(int u){
51     if(sccno[u]) return ;
52     sccno[u] = scc_cnt;
53     for(int i = 0; i < G2[u].size(); ++i) dfs2(G2[u][i]);
54 }
55 void find_scc(int n){
56     scc_cnt = 0;
57     S.clear();
58     memset(sccno, 0, sizeof(sccno));
59     memset(vis, 0, sizeof(vis));
60     for(int i = 0; i < n; ++i) dfs1(i);
61     for(int i = n-1; i >= 0; --i){
62         if(!sccno[S[i]]) {
63             scc_cnt++;
64             dfs2(S[i]);

```

```

65         }
66     }
67 }

```

---

#### 4.1.4 4 2-sat 问题.cpp

---

```

1  // O(n*m) 复杂度不确定
2
3  const int maxn = 2000 + 10;
4
5  struct TwoSAT {
6      int n;
7      vector<int> G[maxn*2];
8      bool mark[maxn*2];
9      int S[maxn*2], c;
10
11     bool dfs(int x) {
12         if (mark[x^1]) return false;
13         if (mark[x]) return true;
14         mark[x] = true;
15         S[c++] = x;
16         for (int i = 0; i < G[x].size(); i++)
17             if (!dfs(G[x][i])) return false;
18         return true;
19     }
20
21     void init(int n) {
22         this->n = n;
23         for (int i = 0; i < n*2; i++) G[i].clear();
24         memset(mark, 0, sizeof(mark));
25     }
26
27     // x = xval or y = yval
28     void add_clause(int x, int xval, int y, int yval) {
29         x = x * 2 + xval;
30         y = y * 2 + yval;
31         G[x].push_back(y^1); // G[0].Pb(1)
32         G[y].push_back(x^1); // G[1].Pb(0);
33     }
34
35     bool solve() {
36         for(int i = 0; i < n*2; i += 2)
37             if(!mark[i] && !mark[i+1]) {
38                 c = 0;
39                 if(!dfs(i)) {
40                     while(c > 0) mark[S[--c]] = false;
41                     if(!dfs(i+1)) return false;
42                 }
43             }
44         return true;
45     }
46 };

```

---

## 4.2 LCA

### 4.2.1 1 DFS+RMQ.cpp

```
1  #include<cstdio>
2  #include<cstring>
3  #include<vector>
4  #include<cmath>
5  #include<iostream>
6  using namespace std;
7
8  const int maxn = 40000+100;
9  const int maxlogv = 17;
10 struct Edge{
11     int to,weight;
12     Edge(int t,int w):to(t),weight(w){};
13 };
14 vector<Edge> G[maxn];
15
16 int id[maxn],dis[maxn];
17 int vs[maxn*2],depth[maxn*2];
18 int dp[maxn*2][maxlogv];
19 void dfs(int node,int fa,int d,int &k){
20     id[node] = k;
21     vs[k] = node;
22     depth[k++] = d;
23     // dis[node] = distance;
24     for(int i = 0;i < G[node].size(); ++i){
25         Edge &t = G[node][i];
26         if(t.to == fa) continue;
27         dis[t.to] = dis[node]+t.weight;
28         dfs(t.to,node,d+1,k);
29     }
30     vs[k] = node;
31     depth[k++] = d;
32 }
33
34 void init_rmq(int n){
35
36     for(int i = 0;i < n ; ++i) dp[i][0] = i;
37     for(int j = 1;(1<<j) <= n; ++j){
38         for(int i = 0;i + (1<<j)-1 < n; ++i){
39             if(depth[dp[i][j-1]]< depth[dp[i+(1<<(j-1))][j-1]])
40                 dp[i][j] = dp[i][j-1];
41             else
42                 dp[i][j] = dp[i+(1<<(j-1))][j-1];
43         }
44     }
45 }
46
47 int query(int l,int r){
48     int k = 0;
49     while((1<<(k+1)) <= r-l+1) k++;
50     if(depth[dp[l][k]] < depth[dp[r-(1<<k)+1][k]])
51         return dp[l][k];
52     else
```

```

53         return dp[r-(1<<k)+1][k];
54     }
55     int lca(int u,int v){
56         return vs[query(min(id[u],id[v]),max(id[u],id[v]))];
57     }
58     void init(int n){
59         int k = 0;
60         dfs(0,-1,0,k);
61         init_rmq(2*n-1);
62     }
63     int main(void){
64         int n,m,q;
65         while(~scanf("%d%d",&n,&m)){
66             for(int i = 0;i < n; ++i) G[i].clear();
67             int u,v,w;
68             for(int i = 0;i < m; ++i){
69                 scanf("%d%d%d",&u,&v,&w);
70                 u--,v--;
71                 G[u].push_back(Edge(v,w));
72                 G[v].push_back(Edge(u,w));
73             }
74             init(n);
75             scanf("%d",&q);
76             while(q--){
77                 int u,v;
78                 scanf("%d %d",&u,&v);
79                 u--,v--;
80                 int f = lca(u,v);
81                 printf("%d\n",dis[u]+dis[v]-2*dis[f]);
82             }
83         }
84         return 0;
85     }

```

---

#### 4.2.2 2 倍增算法.cpp

---

```

1 // POJ1330
2 // LCA 的倍增算法
3
4 #include<vector>
5 #include<cstdio>
6 #include<cstring>
7 using namespace std;
8
9 const int maxn = 1e4+100;
10 const int maxlogv = 14;
11 vector<int> G[maxn];
12 int root;
13
14 int parent[maxlogv][maxn];
15 int depth[maxn];
16
17 void dfs(int v,int p,int d){
18     parent[0][v] = p;
19     depth[v] = d;

```

```

20         for(int i = 0; i < G[v].size(); ++i){
21             if(G[v][i] != p){
22                 dfs(G[v][i], v, d+1);
23             }
24         }
25     }
26     void init(int V){
27         dfs(root, -1, 0);
28         for(int k = 0; k+1 < maxlogv; ++k){
29             for(int v = 0; v < V; ++v){
30                 if(parent[k][v] < 0) parent[k+1][v] = -1;
31                 else parent[k+1][v] = parent[k][parent[k][v]];
32             }
33         }
34     }
35 }
36
37 int lca(int u, int v){
38     if(depth[u] > depth[v]) swap(u, v);
39     for(int k = 0; k < maxlogv; ++k){
40         if(((depth[v] - depth[u]) >> k) & 1){
41             v = parent[k][v];
42         }
43     }
44     if(u == v) return u;
45     for(int k = maxlogv-1; k >= 0; --k){
46         if(parent[k][u] != parent[k][v]){
47             u = parent[k][u];
48             v = parent[k][v];
49         }
50     }
51     return parent[0][u];
52 }
53
54 bool OUT[maxn];
55 int main(void)
56 {
57
58     int T;
59     scanf("%d", &T);
60     while(T--){
61         int n;
62         for(int i = 0; i < n; ++i) G[i].clear();
63         memset(OUT, 0, sizeof(OUT));
64         scanf("%d", &n);
65         for(int i = 1; i < n; ++i) {
66             int u, v;
67             scanf("%d %d", &u, &v);
68             u--, v--;
69             G[u].push_back(v);
70             OUT[v] = 1;
71         }
72         for(int i = 0; i < n; ++i) if(!OUT[i]){
73             root = i;
74             break;
75         }

```

```

76         init(n);
77         int u,v;
78         scanf("%d %d",&u,&v);
79         u--,v--;
80         printf("%d\n",lca(u,v)+1);
81     }
82
83     return 0;
84 }

```

---

## 4.3 Maxflow

### 4.3.1 1 Dinic.cpp

---

```

1 // dinic
2 #include <cstdio>//C 语言 io
3 #include <cstring>//以下是 c 语言常用头文件
4 #include <cmath>
5 #include <cstdlib>
6 #include <ctime>
7 #include <cctype>
8 #include <cstring>
9 #include <cmath>
10 #include <iostream>//c++IO
11 #include <sstream>
12 #include <string>
13 #include <list>//c++ 常用容器
14 #include <vector>
15 #include <set>
16 #include <map>
17 #include <queue>
18 #include <stack>
19 #include <algorithm>//c++ 泛型的一些函数
20 #include <functional>//用来提供一些模版
21 #define fo0(i,n) for(int i = 0;i < n; ++i)
22 #define fo1(i,n) for(int i = 1;i <= n; ++i)
23 #define mem(ar,num) memset(ar,num,sizeof(ar))
24 #define me(ar) memset(ar,0,sizeof(ar))
25 #define lowbit(x) (x&(-x))
26 using namespace std;
27 typedef long long LL;
28 typedef unsigned long long ULL;
29 const int prime = 999983;
30 const int INF = 0x7FFFFFFF;
31 const LL INFF =0x7FFFFFFFFFFFFFFF;
32 const double pi = acos(-1.0);
33 const double inf = 1e18;
34 const double eps = 1e-6;
35 const LL mod = 1e9 + 7;
36 const int LEN = 20000+1000;
37 const int maxn = 1e8;
38 struct Edge{
39     int from,to,cap,flow;
40     Edge(int u,int v,int w,int f): from(u),to(v),cap(w),flow(f){}
41 };

```

```

42 struct Dinic{
43     int n,m,s,t;
44     vector<Edge> edges;
45     vector<int> G[LEN];
46     int a[LEN];
47     int vis[LEN];
48     int d[LEN];
49     int cur[LEN]; //好吧就是点，代表该点在一次求增广的过程中搜索到了那条边，意思就是从这条
    ↪ 边往下肯定搜索不到结果了
50 void init(int n)
51 {
52     this->n = n;
53     for(int i = 0;i < n; ++i)
54         G[i].clear();
55     edges.clear();
56 }
57 void Add(int u,int v,int w)
58 {
59     edges.push_back(Edge(u,v,w,0));
60     edges.push_back(Edge(v,u,0,0));
61     m = edges.size();
62     G[u].push_back(m-2);
63     G[v].push_back(m-1);
64 }
65 bool Bfs(void) //分层
66 {
67     me(d);
68     me(vis);
69     d[s] = 0;
70     vis[s] = 1;
71
72     queue<int> Q;
73     Q.push(s);
74     while(!Q.empty())
75     {
76         int q = Q.front();Q.pop();
77
78         for(size_t i = 0;i < G[q].size();++i)
79         {
80             Edge &tmp = edges[G[q][i]];
81             if(!vis[tmp.to]&&tmp.cap>tmp.flow)
82             {
83                 vis[tmp.to] = 1;
84                 d[tmp.to] = d[q] + 1;
85                 Q.push(tmp.to);
86             }
87         }
88     }
89     return vis[t];
90 }
91 int Dfs(int node,int a)
92 {
93
94     if(node == t || a == 0)
95         return a;
96     int flow = 0,f;

```

```

97     for(int &i = cur[node]; i < G[node].size(); ++i)
98     {
99         Edge &tmp = edges[G[node][i]];
100         if(d[tmp.to] == d[node] + 1 && (f = Dfs(tmp.to, min(a, tmp.cap - tmp.flow))) > 0)
101         {
102             flow += f;
103             tmp.flow += f;
104             edges[G[node][i]^1].flow -= f;
105             a -= f;
106             if(a == 0)
107                 break;
108         }
109     }
110     return flow;
111 }
112 int MaxFlow(int s, int t)
113 {
114     this->s = s;
115     this->t = t;
116     int flow = 0;
117     while(Bfs())
118     {
119         me(cur);
120         flow += Dfs(s, maxn);
121     }
122     return flow;
123 }
124 }
125
126 };
127 Dinic dinic;
128 int main()
129 {
130     int N, M, S, T;
131     while(cin >> N >> M)
132     {
133         S = 1, T = N;
134         dinic.init(N);
135         int u, v, w;
136         for(int i = 0; i < M; ++i)
137         {
138             scanf("%d %d %d", &u, &v, &w);
139             dinic.Add(u, v, w);
140         }
141         int ans = 0;
142         ans = dinic.MaxFlow(S, T);
143         printf("%d\n", ans);
144     }
145 }
146
147
148
149
150 return 0;
151 }

```



### 4.3.2 2 ISAP.cpp

```
1 // 点的下标从零开始, 注意初始化
2 #include<cstdio>
3 #include<cstring>
4 #include<queue>
5 #include<vector>
6 #include<algorithm>
7 using namespace std;
8
9 const int maxn = 10000 + 10;
10 const int INF = 1000000000;
11
12 struct Edge {
13     int from, to, cap, flow;
14 };
15
16 bool operator < (const Edge& a, const Edge& b) {
17     return a.from < b.from || (a.from == b.from && a.to < b.to);
18 }
19
20 struct ISAP {
21     int n, m, s, t;
22     vector<Edge> edges;
23     vector<int> G[maxn]; // 邻接表, G[i][j] 表示结点 i 的第 j 条边在 e 数组中的序号
24     bool vis[maxn]; // BFS 使用
25     int d[maxn]; // 从起点到 i 的距离
26     int cur[maxn]; // 当前弧指针
27     int p[maxn]; // 可增广路上的上一条弧
28     int num[maxn]; // 距离标号计数
29
30     void AddEdge(int from, int to, int cap) {
31         edges.push_back((Edge){from, to, cap, 0});
32         edges.push_back((Edge){to, from, 0, 0});
33         m = edges.size();
34         G[from].push_back(m-2);
35         G[to].push_back(m-1);
36     }
37
38     bool BFS() {
39         memset(vis, 0, sizeof(vis));
40         queue<int> Q;
41         Q.push(t);
42         vis[t] = 1;
43         d[t] = 0;
44         while(!Q.empty()) {
45             int x = Q.front(); Q.pop();
46             for(int i = 0; i < G[x].size(); i++) {
47                 Edge& e = edges[G[x][i]^1];
48                 if(!vis[e.from] && e.cap > e.flow) {
49                     vis[e.from] = 1;
50                     d[e.from] = d[x] + 1;
51                     Q.push(e.from);
52                 }
53             }
54         }
55     }
```

```

54     }
55     return vis[s];
56 }
57
58 void init(int n) {
59     this->n = n;
60     for(int i = 0; i < n; i++) G[i].clear();
61     edges.clear();
62 }
63
64
65 int Augment() {
66     int x = t, a = INF;
67     while(x != s) {
68         Edge& e = edges[p[x]];
69         a = min(a, e.cap - e.flow);
70         x = edges[p[x]].from;
71     }
72     x = t;
73     while(x != s) {
74         edges[p[x]].flow += a;
75         edges[p[x]^1].flow -= a;
76         x = edges[p[x]].from;
77     }
78     return a;
79 }
80
81 int Maxflow(int s, int t) {
82     this->s = s; this->t = t;
83     int flow = 0;
84     BFS();
85     memset(num, 0, sizeof(num));
86     for(int i = 0; i < n; i++) num[d[i]]++;
87     int x = s;
88     memset(cur, 0, sizeof(cur));
89     while(d[s] < n) {
90         if(x == t) {
91             flow += Augment();
92
93             x = s;
94         }
95         int ok = 0;
96         for(int i = cur[x]; i < G[x].size(); i++) {
97             Edge& e = edges[G[x][i]];
98             if(e.cap > e.flow && d[x] == d[e.to] + 1) { // Advance
99                 ok = 1;
100                 p[e.to] = G[x][i];
101                 cur[x] = i; // 注意
102                 x = e.to;
103                 break;
104             }
105         }
106         if(!ok) { // Retreat
107             int m = n-1; // 初值注意
108             for(int i = 0; i < G[x].size(); i++) {
109                 Edge& e = edges[G[x][i]];

```

```

110         if(e.cap > e.flow) m = min(m, d[e.to]);
111     }
112     if(--num[d[x]] == 0) break;
113     num[d[x] = m+1]++;
114     cur[x] = 0; // 注意
115     if(x != s) x = edges[p[x]].from;
116 }
117 }
118 return flow;
119 }
120 };
121
122
123 ISAP g;
124
125 int main() {
126
127     int N,M;
128     int S,T;
129     scanf("%d %d",&N,&M);
130     scanf("%d %d",&S,&T);
131     int u,v,w;
132     g.init(N);
133     while(M--){
134         scanf("%d %d %d",&u,&v,&w);
135         u--,v--;
136         g.AddEdge(u,v,w);
137     }
138     printf("%d",g.Maxflow(S-1,T-1));
139
140
141     return 0;
142 }

```

---

### 4.3.3 3 MCMF.cpp

---

```

1 // 最小费用最大流, 下标从 1 开始
2
3 #include <bits/stdc++.h>
4 #define mem(ar,num) memset(ar,num,sizeof(ar))
5 #define me(ar) memset(ar,0,sizeof(ar))
6 #define lowbit(x) (x&(-x))
7 #define Pb push_back
8 #define FI first
9 #define SE second
10 #define For(i,a,b) for(int i = a; i < b; ++i)
11 #define IOS ios::sync_with_stdio(false)
12 using namespace std;
13 typedef long long LL;
14 typedef unsigned long long ULL;
15 const int prime = 999983;
16 const int INF = 1e8;
17 const LL INFF = 0x7FFFFFFFFFFFFFFF;
18 const double pi = acos(-1.0);
19 const double inf = 1e18;

```

```

20 const double eps = 1e-6;
21 const LL mod = 1e9 + 7;
22 LL qpow(LL a, LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
23 LL gcd(LL a, LL b) {return b?gcd(b,a%b):a;}
24 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
25 typedef pair<int,int> P;
26 struct Edge{
27     int from,to,cap,flow,cost;
28 };
29 const int maxn = 5000+100;
30 struct MCMF{
31     int n,m,s,t;
32     vector<Edge> edges;
33     vector<int> G[maxn];
34     int inq[maxn];
35     int d[maxn];
36     int p[maxn];
37     int a[maxn];
38     void init(int n){
39         this->n = n;
40         for(int i = 0;i < n; ++i) G[i].clear();
41         edges.clear();
42     }
43     void AddEdge(int from,int to,int cap,int cost){
44         edges.push_back((Edge){from,to,cap,0,cost});
45         edges.push_back((Edge){to,from,0,0,-cost});
46         int m = edges.size();
47         G[from].push_back(m-2);
48         G[to].push_back(m-1);
49     }
50 }
51 bool BellmanFord(int s,int t,int &flow,int &cost){
52     for(int i = 0;i < n; ++i) d[i] = INF;
53     memset(inq,0,sizeof(inq));
54     d[s] = 0,inq[s] = 1;p[s] = 0,a[s] = INF;
55
56     queue<int> Q;
57     Q.push(s);
58     while(!Q.empty()){
59
60         int u = Q.front(); Q.pop();
61         inq[u] = 0;
62         for(int i = 0;i < G[u].size(); ++i){
63             Edge& e = edges[G[u][i]];
64             if(e.cap > e.flow && d[e.to] > d[u]+e.cost){
65                 d[e.to] = d[u]+e.cost;
66                 p[e.to] = G[u][i];
67                 a[e.to] = min(a[u],e.cap-e.flow);
68                 if(!inq[e.to]) {
69                     Q.push(e.to); inq[e.to] = 1;
70                 }
71             }
72         }
73     }
74
75     if(d[t] == INF) return false;

```

```

76         flow += a[t];
77         cost += d[t]*a[t];
78         int u = t;
79         while(u != s){
80             edges[p[u]].flow += a[t];
81             edges[p[u]^1].flow -= a[t];
82             u = edges[p[u]].from;
83         }
84         return true;
85     }
86     int Mincost(int s,int t,int &flow,int &cost){
87         flow = 0,cost = 0;
88
89         while(BellmanFord(s,t,flow,cost));
90         return cost;
91     }
92 }
93
94 };
95 MCMF mcmf;
96 int main(void)
97 {
98     int n,m,s,t;
99     scanf("%d %d %d %d",&n,&m,&s,&t);
100     int u,v,w,c;
101     mcmf.init(n+1);
102     while(m--){
103         scanf("%d %d %d %d",&u,&v,&w,&c);
104         mcmf.AddEdge(u,v,w,c);
105     }
106     int flow,cost;
107     flow = 0,cost = 0;
108     mcmf.Mincost(s,t,flow,cost);
109     printf("%d %d\n",flow,cost);
110
111
112     return 0;
113 }

```

---

## 4.4 二分图

### 4.4.1 1 匈牙利算法.cpp

```

1  #include <bits/stdc++.h>
2  #define mem(ar,num) memset(ar,num,sizeof(ar))
3  #define me(ar) memset(ar,0,sizeof(ar))
4  #define lowbit(x) (x&(-x))
5  #define Pb push_back
6  #define FI first
7  #define SE second
8  #define For(i,a,b) for(int i = a; i < b; ++i)
9  #define IOS ios::sync_with_stdio(false)
10 using namespace std;
11 typedef long long LL;
12 typedef unsigned long long ULL;

```

```

13 const int prime = 999983;
14 const int INF = 0x7FFFFFFF;
15 const LL INFF =0x7FFFFFFFFFFFFFFF;
16 const double pi = acos(-1.0);
17 const double inf = 1e18;
18 const double eps = 1e-6;
19 const LL mod = 1e9 + 7;
20 LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
21 LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
22 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
23 typedef pair<int,int> P;
24 const int maxn = 1000+10;
25 vector<int> G[maxn];
26 int match[maxn];
27 bool used[maxn];
28 int N,M;
29 bool dfs(int v){
30     used[v] = true;
31     for(int i = 0;i < G[v].size(); ++i){
32         if(used[u]) continue; used[u] = true;
33         int u = G[v][i],w = match[u];
34         if(w < 0 || !used[w]&&dfs(w)){
35             match[v] = u;
36             match[u] = v;
37             return true;
38         }
39     }
40     return false;
41 }
42 int main(void)
43 {
44     scanf("%d %d",&N,&M);
45
46     while(M--){
47         int u,v;
48         scanf("%d %d",&u,&v);
49         G[u].Pb(v);
50         G[v].Pb(u);
51     }
52     int ans = 0;
53     memset(match,-1,sizeof(match));
54     for(int i = 1;i <= N; ++i){
55         if(match[i] < 0){
56             memset(used,0,sizeof(used));
57             if(dfs(i)){
58                 ans++;
59             }
60         }
61     }
62     cout<<ans<<endl;
63     return 0;
64 }

```

---

#### 4.4.2 2 KM.cpp

```
1  const int maxn = 500+5;
2  struct KM{
3      int n;
4      vector<int> G[maxn];
5      int W[maxn][maxn];
6      int Lx[maxn];
7      int Ly[maxn];
8      int Left[maxn];
9      bool S[maxn],T[maxn];
10     void init(int n){
11         this->n = n;
12         for(int i = 1;i <= n; ++i) G[i].clear();
13         memset(W,0,sizeof(W));
14     }
15     void AddEdge(int u,int v,int w){
16         G[u].push_back(v);
17         W[u][v] = w;
18     }
19     bool match(int u){
20         S[u] = true;
21         for(int i =0;i < G[u].size(); ++i){
22             int v = G[u][i];
23             if(Lx[u]+Ly[v] == W[u][v]&&!T[v]){
24                 T[v] = true;
25                 if(Left[v] == -1||match(Left[v])){
26                     Left[v] = u;
27                     return true;
28                 }
29             }
30         }
31         return false;
32     }
33     void update(){
34         int a = INF;
35         for(int u = 0;u < n; ++u)
36             if(S[u])
37                 for(int i = 0;i < G[u].size(); ++i){
38                     int v = G[u][i];
39                     if(!T[v])
40                         a = min(a,Lx[u]+Ly[v]-W[u][v]);
41                 }
42         for(int i = 0;i < n; ++i){
43             if(S[i]) Lx[i] -= a;
44             if(T[i]) Ly[i] += a;
45         }
46     }
47     void solve(){
48         for(int i = 0;i < n; ++i){
49             Lx[i] = *max_element(W[i],W[i]+n);
50             Left[i] = -1;
51             Ly[i] = 0;
52         }
53         for(int u = 0;u < n; ++u){
```

```

54         for(;;){
55             for(int i = 0;i < n; ++i) S[i] = T[i] = 0;
56             if(match(u)) break;
57             else update();
58         }
59     }
60 }
61 };

```

---

#### 4.4.3 3 一般图最大匹配.cpp

---

```

1  #include<cstdio>
2  #include<algorithm>
3  #include<cmath>
4  #include<cstring>
5  #include<vector>
6  #define SF scanf
7  #define PF printf
8  #define MAXN 510
9  using namespace std;
10 int mk[MAXN],fa[MAXN],nxt[MAXN],q[MAXN],vis[MAXN],match[MAXN];
11 int fr,bk,t,n,m;
12 vector<int> a[MAXN];
13 int find(int x){
14     if(fa[x]==x)
15         return x;
16     fa[x]=find(fa[x]);
17     return fa[x];
18 }
19 int LCA(int x,int y){
20     t++;
21     while(1){
22         if(x){
23             x=find(x);
24             if(vis[x]==t)
25                 return x;
26             vis[x]=t;
27             if(match[x])
28                 x=nxt[match[x]];
29             else
30                 x=0;
31         }
32         swap(x,y);
33     }
34 }
35 void Union(int x,int y){
36     if(find(x)!=find(y))
37         fa[fa[x]]=fa[y];
38 }
39 void gr(int a,int p){
40     while(a!=p){
41         int b=match[a];
42         int c=nxt[b];
43         if(find(c)!=p)
44             nxt[c]=b;

```



```

45         if(mk[b]==2){
46             q[++bk]=b;
47             mk[b]=1;
48         }
49         Union(a,b);
50         Union(b,c);
51         a=c;
52     }
53 }
54 void aug(int S){
55     for(int i=1;i<=n;i++){
56         mk[i]=nxt[i]=0;
57         fa[i]=i;
58     }
59     mk[S]=1;
60     fr=bk=0;
61     q[fr]=S;
62     while(fr<=bk){
63         int x=q[fr++];
64         for(int i=0;i<a[x].size();i++){
65             int y=a[x][i];
66             if(match[x]==y)
67                 continue;
68             else if(find(x)==find(y))
69                 continue;
70             else if(mk[y]==2)
71                 continue;
72             else if(mk[y]==1){
73                 int r=LCA(x,y);
74                 if(find(x)!=r)
75                     nxt[x]=y;
76                 if(find(y)!=r)
77                     nxt[y]=x;
78                 gr(x,r);
79                 gr(y,r);
80             }
81             else if(!match[y]){
82                 nxt[y]=x;
83                 for(int u=y;u;){
84                     int v=nxt[u];
85                     int mv=match[v];
86                     match[u]=v;
87                     match[v]=u;
88                     u=mv;
89                 }
90                 return;
91             }
92             else{
93                 nxt[y]=x;
94                 mk[y]=2;
95                 q[++bk]=match[y];
96                 mk[match[y]]=1;
97             }
98         }
99     }
100 }

```

```

101 int main(){
102     SF("%d%d",&n,&m);
103     int u,v;
104     for(int i=1;i<=m;i++){
105         SF("%d%d",&u,&v);
106         a[u].push_back(v);
107         a[v].push_back(u);
108     }
109     for(int i=1;i<=n;i++)
110         if(!match[i])
111             aug(i);
112     int sum=0;
113     for(int i=1;i<=n;i++)
114         if(match[i])
115             sum++;
116     PF("%d\n",sum/2);
117     for(int i=1;i<=n;i++)
118         PF("%d ",match[i]);
119 }

```

---

## 4.5 最小生成树

### 4.5.1 1 Krustal 卡鲁斯卡尔算法.cpp

---

```

1  /*
2  复杂度 E*log(E), 适用于稀疏图
3  https://vjudge.net/problem/HDU-1863
4  */
5
6
7  #include<bits/stdc++.h>
8
9  using namespace std;
10
11  const int maxn = 100+100;
12  struct Edge//边
13  {
14      int from,to,cost;
15      bool operator< ( const Edge & a)
16      {
17          return cost < a.cost;
18      }
19  };
20  Edge edge[maxn];
21  int F[maxn];
22  int Find(int x)//并查集算法
23  {
24      return x == F[x] ? x:F[x] = Find(F[x]);
25  }
26  int main(void)
27  {
28      int N,M;
29      while(cin>>N>>M&&N)// N 代表的是道路数量, M 代表村庄的数量
30      {
31          for(int i = 0; i <= M; ++i)

```

```

32         F[i] = i;
33     for(int i = 0; i < N; ++i)
34     {
35         Edge &t = edge[i];
36         scanf("%d %d %d",&t.from,&t.to,&t.cost);
37     }
38     sort(edge,edge+N);// 对边进行排序
39     int sum = 0;
40     int num = M;
41     for(int i = 0;i < N ; ++i)// 一个个将边加进去
42     {
43         Edge t = edge[i];
44         if(Find(t.from) == Find(t.to))
45             continue;
46         F[Find(t.from)] = F[Find(t.to)];
47         sum += t.cost;
48         num--;
49     }
50     if(num == 1)
51         cout<<sum<<endl;
52     else
53         cout<<"?"<<endl;
54 }
55
56
57 return 0;
58 }

```

---

#### 4.5.2 2 prim 算法.cpp

---

```

1  /*
2  prim 算法是进行加点，使用于稠密图，可以选择用堆或者不用
3  不用堆  $O(V^2)$ ;
4  用堆  $O(E * \log(V))$ ;
5  https://vjudge.net/problem/HDU-1863
6  */
7
8
9  typedef pair<int,int> P;
10 const int LEN = 2e6+100;
11 int Away[LEN];//记录从当前已选结点到 j 节点的路径的最小值
12 bool vis[LEN];
13 int N,M;//N 道路数目, M 村庄个数
14 vector<vector<P> > vec(LEN);
15 int main()
16 {
17     cin>>M>>N;
18
19     int from,to,weight;
20     while(N--)
21     {
22         scanf("%d %d %d",&from,&to,&weight);
23         vec[from].push_back(P(weight,to));
24         vec[to].push_back(P(weight,from));
25     }// 添加边

```

```

26
27
28     for(int i = 2; i <= M; ++i)
29         Away[i] = INF; //初始化 Away 数组
30     Away[1] = 0;
31     int Left = M;
32     int All_cost = 0;
33     priority_queue<P,vector<P>,greater<P> > q; // 小顶堆
34     q.push(P(0,1));
35     while(!q.empty() && Left > 0)
36     {
37         P tmp = q.top(); q.pop();
38         int To = tmp.second;
39         if(vis[To])
40             continue;
41         vis[To] = 1;
42         Left--;
43         All_cost += tmp.first;
44         for(int i = 0; i < vec[To].size(); ++i) // 更新 Away 数组
45         {
46             P &t = vec[To][i];
47             if(!vis[t.second] && Away[t.second] > t.first)
48             {
49                 Away[t.second] = t.first;
50                 q.push(t);
51             }
52         }
53     }
54
55     cout<<All_cost<<endl;
56
57
58
59     return 0;
60 }

```

---

### 4.5.3 3 最小限制生成树.cpp

---

```

1 // 限制某一点的度数不能超过 K
2 #include<cstring>
3 #include<map>
4 #include<cstdio>
5 #include<iostream>
6 #include<algorithm>
7 #include<set>
8 using namespace std;
9 #define me(ar) memset(ar,0,sizeof(ar))
10 const int INF = 1e8;
11 //.....
12 const int LEN = 30;
13 int K;
14 int n,m;
15 struct Edge
16 {
17     int x,y;

```

```

18     int weight;
19     bool operator <(const Edge &a) const
20     {
21         return weight < a.weight;
22     }
23 } edge[LEN*LEN+10]; //邻接表存边,Kruskal 算法要用
24 int dis[LEN][LEN]; //邻接矩阵
25 int sign[LEN][LEN]; //记录那些边已经在生成树里面了
26 int vis[LEN]; //记录是否相连
27 int F[LEN]; //并查集所用
28 int Father[LEN]; //由 i 到 i+1 度限制生成树需要用动态规划求解, 用来状态转移
29 int Best[LEN]; //Best[i] 指的是由当前节点到 park 这些边中最长边是多少
30 int Find(int x) //并查集所用 Find 函数
31 {
32     return x == F[x]?x:F[x] = Find(F[x]);
33 }
34 void Dfs(int x) //Dfs 动态规划记忆化搜索
35 {
36     // vis[x] = 1;
37     for(int i = 1; i <= n; ++i )
38     {
39         if(sign[i][x] & !vis[i]) //如果有边相连并且下一个节点没有被访问
40         {
41             if(x == 0)
42                 Best[i] = -INF; //与 park 直接相连的边不能删除
43
44             else
45                 Best[i] = max(Best[x], dis[x][i]); //状态转移方程
46             Father[i] = x;
47             vis[i] = 1;
48             Dfs(i);
49         }
50     }
51 }
52 void init(){
53     for(int i = 0; i < LEN; ++i)
54         F[i] = i;
55     me(sign); //初始化标记数组
56     me(vis);
57     //初始化邻接矩阵
58     for(int i = 0; i < LEN; ++i)
59         for(int j = 0; j < LEN; ++j)
60             dis[i][j] = INF;
61 }
62 int main(void)
63 {
64     while(cin >> m)
65     {
66         //初始化并查集数组
67         init();
68         n = 0; //用来记录共有多少个节点
69         // set<string> se;
70         map<string, int> ma; //将地点编号
71         ma["Park"] = 0; //将 park 加入节点
72         string s1, s2;
73         int a, b;

```

```

74     int weight = 0;
75     for(int i = 0; i < m; ++i)
76     {
77         cin>>s1>>s2>>weight;
78         if(s1 == "Park" || ma[s1] != 0)
79             a = ma[s1]; //如果节点已编号，则直接使用
80         else
81             a = ma[s1] = ++n; //如果没有编号，编号
82         if(s2 == "Park" || ma[s2] != 0)
83             b = ma[s2];
84         else
85             b = ma[s2] = ++n;
86         dis[a][b] = dis[b][a] = weight;
87         edge[i].x = a;
88         edge[i].y = b;
89         edge[i].weight = weight;
90     }
91     //求最小生成树
92     int ans = 0; //kruskal 算法求最小生成树
93     sort(edge, edge+m);
94     for(int i = 0; i < m; ++i)
95     {
96         int x = edge[i].x;
97         int y = edge[i].y;
98         weight = edge[i].weight;
99         if(x==0 || y==0) //去除掉 park 这个点
100             continue;
101         int xx = Find(x);
102         int yy = Find(y);
103         if(xx!=yy)
104         {
105             F[xx] = F[yy];
106             ans += weight;
107             sign[x][y] = sign[y][x] = 1;
108         }
109     }
110
111
112     cin>>K; //最小 k 度生成树
113     int Min[LEN]; //用来记录每一个最小生成树到 park 点的最小路径
114     for(int i = 0; i < LEN; ++i)
115         Min[i] = INF; //初始化
116     int index[LEN]; //用来记录最小路径的点
117     for(int i = 1; i <= n; ++i)
118     {
119         if(dis[i][0] < Min[Find(i)])
120         {
121             Min[Find(i)] = dis[i][0];
122             index[Find(i)] = i;
123         }
124     }
125     //// cout<<se.size()<<endl;
126     int m = 0; //用来记录除去 park 点即 0 点之后共有多少个连通分量
127     for(int i = 1; i <= n; ++i)
128     {
129         if(Min[i] != INF)

```

```

130     {
131         ans += Min[i];
132         sign[index[i]][0] = sign[0][index[i]] = 1; //将这个最小路径的点与
            ↪ park 相连
133         m++;
134     }
135 }
136 int MMin = ans;
137 for(int i = m + 1; i <= K; ++i) //从 m+1 到 K 求最小 i 度生成树
138 {
139     me(vis);
140     vis[0] = 1;
141     Dfs(0);
142     int select = -1; //select 用来记录选择哪个与 park 点相连是最小的
143     int sum = INF;
144     for(int i = 1; i <= n; ++i)
145     {
146         if(!sign[0][i] && dis[0][i] != INF)
147         {
148             if(dis[i][0] - Best[i] < sum)
149             {
150                 select = i;
151                 sum = dis[i][0] - Best[i];
152             }
153         }
154     }
155     if(select == -1) //如果找不到，就跳出循环
156         break;
157     ans += sum;
158     sign[select][0] = sign[0][select] = 1;
159     MMin = min(MMin, ans);
160     for(int i = select; i != 0; i = Father[i])
161     {
162         if(dis[Father[i]][i] == Best[select])
163         {
164             sign[i][Father[i]] = sign[Father[i]][i] = 0;
165             break;
166         }
167     }
168     cout << ans << endl;
169 }
170 printf("Total miles driven: %d\n", MMin);
171 // cout << MMin << endl;
172 }
173 return 0;
174 }

```

---

#### 4.5.4 4 次小生成树.cpp

---

```

1 #include<iostream>
2 #include<cstdio>
3 #include<cstring>
4 #include<string>
5 #include<algorithm>

```

```

6  #include<cmath>
7  #include<vector>
8  #include<queue>
9  #define ll long long
10 using namespace std;
11
12 int getint()
13 {
14     int i=0,f=1;char c;
15     for(c=getchar();(c<'0' || c>'9')&&c!='-';c=getchar());
16     if(c=='-')f=-1,c=getchar();
17     for(;c>='0'&&c<='9';c=getchar())i=(i<<3)+(i<<1)+c-'0';
18     return i*f;
19 }
20
21 const int N=100005,M=300005;
22 struct node
23 {
24     int x,y,w;
25     inline friend bool operator < (const node &a,const node &b)
26     {
27         return a.w<b.w;
28     }
29 }bian[M];
30 int n,m;
31 int id[N],fa[N][20],mx1[N][20],mx2[N][20],dep[N];
32 int tot,first[N],nxt[N<<1],to[N<<1],w[N<<1];
33 ll totlen,ans;
34 bool chs[M];
35
36 void add(int x,int y,int z)
37 {
38     nxt[++tot]=first[x],first[x]=tot,to[tot]=y,w[tot]=z;
39 }
40
41 int find(int x)
42 {
43     return id[x]==x?x:id[x]=find(id[x]);
44 }
45
46 void kruskal()
47 {
48     for(int i=1;i<=n;i++)id[i]=i;
49     sort(bian+1,bian+m+1);
50     int cnt=0;
51     for(int i=1;i<=m;i++)
52     {
53         int x=find(bian[i].x),y=find(bian[i].y);
54         if(x!=y)
55         {
56             cnt++;
57             totlen+=bian[i].w;
58             chs[i]=true;
59             add(bian[i].x,bian[i].y,bian[i].w);
60             add(bian[i].y,bian[i].x,bian[i].w);
61             id[y]=x;

```



```

62         if(cnt==n-1)break;
63     }
64 }
65 }
66
67 void dfs(int u)
68 {
69     for(int i=1;i<20;i++)fa[u][i]=fa[fa[u][i-1]][i-1];
70     for(int i=1;i<20;i++)mx1[u][i]=max(mx1[u][i-1],mx1[fa[u][i-1]][i-1]);
71     for(int i=1;i<20;i++)
72     {
73         mx2[u][i]=max(mx2[u][i-1],mx2[fa[u][i-1]][i-1]);
74         if(mx1[u][i-1]<mx1[fa[u][i-1]][i-1]&&mx2[u][i]<mx1[u][i-1])
75             mx2[u][i]=mx1[u][i-1];
76         if(mx1[u][i-1]>mx1[fa[u][i-1]][i-1]&&mx1[fa[u][i-1]][i-1]>mx2[u][i])
77             mx2[u][i]=mx1[fa[u][i-1]][i-1];
78     }
79     for(int e=first[u];e=e nxt[e])
80     {
81         int v=to[e];
82         if(v==fa[u][0])continue;
83         fa[v][0]=u;mx1[v][0]=w[e];
84         dep[v]=dep[u]+1;
85         dfs(v);
86     }
87 }
88
89 int Find(int x,int y,int len)
90 {
91     int Mx1=0,Mx2=0;
92     if(dep[x]<dep[y])swap(x,y);
93     int delta=dep[x]-dep[y];
94     for(int i=19;i>=0;i--)
95         if(delta&(1<<i))
96         {
97             if(Mx1>mx1[x][i]&&mx1[x][i]>Mx2)Mx2=mx1[x][i];
98             if(Mx1<mx1[x][i])Mx2=max(Mx1,mx2[x][i]),Mx1=mx1[x][i];
99             x=fa[x][i];
100         }
101     if(x==y)return Mx1==len?Mx2:Mx1;
102     for(int i=19;i>=0;i--)
103         if(fa[x][i]!=fa[y][i])
104         {
105             if(Mx1>mx1[x][i]&&mx1[x][i]>Mx2)Mx2=mx1[x][i];
106             if(Mx1<mx1[x][i])Mx2=max(Mx1,mx2[x][i]),Mx1=mx1[x][i];
107             x=fa[x][i];
108             if(Mx1>mx1[y][i]&&mx1[y][i]>Mx2)Mx2=mx1[y][i];
109             if(Mx1<mx1[y][i])Mx2=max(Mx1,mx2[y][i]),Mx1=mx1[y][i];
110             y=fa[y][i];
111         }
112     if(Mx1>mx1[x][0]&&mx1[x][0]>Mx2)Mx2=mx1[x][0];
113     if(Mx1<mx1[x][0])Mx2=max(Mx1,mx2[x][0]),Mx1=mx1[x][0];
114     x=fa[x][0];
115     if(Mx1>mx1[y][0]&&mx1[y][0]>Mx2)Mx2=mx1[y][0];
116     if(Mx1<mx1[y][0])Mx2=max(Mx1,mx2[y][0]),Mx1=mx1[y][0];
117     y=fa[y][0];

```

```

118     return Mx1==len?Mx2:Mx1;
119 }
120
121 void solve(int e)
122 {
123     int x=bian[e].x,y=bian[e].y,len=bian[e].w;
124     int tmp=Find(x,y,len);
125     ans=min(ans,totlen-tmp+len);
126 }
127
128 int main()
129 {
130     //freopen("lx.in","r",stdin);
131     n=getint(),m=getint();
132     for(int i=1;i<=m;i++)
133     {
134         bian[i].x=getint();
135         bian[i].y=getint();
136         bian[i].w=getint();
137     }
138     kruskal();
139     dfs(1);
140     ans=1e18;
141     for(int i=1;i<=m;i++)
142         if(!chs[i])solve(i);
143     printf("%lld",ans);
144 }

```

---

## 4.6 最短路

### 4.6.1 1 Dijkstra.cpp

---

```

1  #include <bits/stdc++.h>
2  #define mem(ar,num) memset(ar,num,sizeof(ar))
3  #define me(ar) memset(ar,0,sizeof(ar))
4  #define lowbit(x) (x&(-x))
5  #define Pb push_back
6  #define FI first
7  #define SE second
8  #define For(i,a,b) for(int i = a; i < b; ++i)
9  #define IOS ios::sync_with_stdio(false)
10 using namespace std;
11 typedef long long LL;
12 //typedef unsigned long long ULL;
13 //const int prime = 999983;
14 //const int INF = 0x7FFFFFFF;
15 //const LL INFF =0x7FFFFFFFFFFFFFFF;
16 //const double pi = acos(-1.0);
17 //const double inf = 1e18;
18 //const double eps = 1e-6;
19 //const LL mod = 1e9 + 7;
20 //LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return
    ↪ s;}
21 //LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
22 //int dr[2][4] = {1,-1,0,0,0,0,-1,1};

```

```

23 //typedef pair<int,int> P;
24 struct Dijkstra{
25     #define maxn 1234
26     #define INF 123456789
27     int n,m;
28     int s,t;
29
30     int dis[maxn],M[maxn][maxn];
31     bool vis[maxn];
32     void init(){
33         scanf("%d %d %d %d",&n,&m,&s,&t);
34         int u,v,c;
35         for(int i = 1;i <= n; ++i)
36             for(int j = 1;j <= n; ++j)
37                 if(i != j)
38                     M[i][j] = INF;
39         for(int i = 0;i < m; ++i){
40             scanf("%d %d %d",&u,&v,&c);
41             M[u][v] = M[v][u] = min(M[u][v],c);
42         }
43     }
44     void solve(){
45         memset(vis,0,sizeof(vis));
46         fill(dis+1,dis+n+1,INF);
47         dis[s] = 0;
48         for(int i = 1;i <= n; ++i){
49             int x,Min = INF;
50             for(int j = 1;j <= n; ++j){
51                 if(!vis[j]&&dis[j] <= Min)
52                     Min = dis[x=j];
53             }
54             vis[x] = 1;
55
56             for(int j = 1;j <= n; ++j){
57                 if(!vis[j]&&dis[j] > dis[x]+M[x][j])
58                     dis[j] = dis[x]+M[x][j];
59             }
60
61         }
62         printf("%d\n",dis[t]);
63     }
64 };
65 Dijkstra Dij;
66 int main(void)
67 {
68     Dij.init();
69     Dij.solve();
70
71     return 0;
72 }
73 // 加了堆优化的 dij
74
75 #include <bits/stdc++.h>
76 #define mem(ar,num) memset(ar,num,sizeof(ar))
77 #define me(ar) memset(ar,0,sizeof(ar))
78 #define lowbit(x) (x&(-x))

```

```

79 #define Pb push_back
80 #define FI first
81 #define SE second
82 #define For(i,a,b) for(int i = a; i < b; ++i)
83 #define IOS ios::sync_with_stdio(false)
84 using namespace std;
85 typedef long long LL;
86 typedef unsigned long long ULL;
87
88
89 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
90 typedef pair<int,int> P;
91 struct Edge{
92     int u,v,d;
93     Edge(int uu,int vv,int dd):u(uu),v(vv),d(dd){
94     }
95 };
96 struct Dijstra{
97     #define maxn 123456
98     #define INF 123456789
99     int N,M,S,T;
100
101     typedef pair<int,int> P;
102     vector<Edge> edges;
103     vector<int> G[maxn];
104     bool done[maxn];
105     int d[maxn];
106     int p[maxn];
107     void init(){
108         for(int i = 1; i <= N; ++i) G[i].clear();
109         edges.clear();
110         scanf("%d %d %d %d",&N,&M,&S,&T);
111         // cout<<N<<M<<S<<T<<endl;
112         int u,v,w;
113         for(int i = 1; i <= M; ++i){
114             scanf("%d %d %d",&u,&v,&w);
115             AddEdge(u,v,w);
116             AddEdge(v,u,w);
117         }
118
119     }
120     void AddEdge(int u,int v,int d){
121         edges.push_back(Edge(u,v,d));
122         int m = edges.size();
123         G[u].push_back(m-1);
124     }
125     void solve(){
126         priority_queue<P,vector<P>,greater<P>> Q;
127         for(int i = 1; i <= N; ++i) d[i] = INF;
128         d[S] = 0;
129         memset(done,0,sizeof(done));
130         Q.push(P(0,S));
131         while(!Q.empty()){
132             P x = Q.top(); Q.pop();
133             int u = x.second;
134             if(done[u]) continue;

```

```

135         done[u] = true;
136         for(int i = 0; i < G[u].size(); ++i){
137             Edge &e = edges[G[u][i]];
138             if(!done[e.v] && d[e.v] > d[u] + e.d){
139                 d[e.v] = d[u] + e.d;
140                 p[e.v] = G[u][i];
141                 Q.push(P(d[e.v], e.v));
142             }
143         }
144     }
145
146     printf("%d\n", d[T]);
147 }
148 };
149 Dijstra Dij;
150 int main(void)
151 {
152     Dij.init();
153     Dij.solve();
154
155     return 0;
156 }

```

---

#### 4.6.2 2 Bellman-ford.cpp

---

```

1  #include <bits/stdc++.h>
2  #define mem(ar,num) memset(ar,num,sizeof(ar))
3  #define me(ar) memset(ar,0,sizeof(ar))
4  #define lowbit(x) (x&(-x))
5  #define Pb push_back
6  #define FI first
7  #define SE second
8  #define For(i,a,b) for(int i = a; i < b; ++i)
9  #define IOS ios::sync_with_stdio(false)
10 using namespace std;
11 typedef long long LL;
12 typedef unsigned long long ULL;
13 const int prime = 999983;
14 const int INF = 0x7FFFFFFF;
15 const LL INFF = 0x7FFFFFFFFFFFFFFF;
16 const double pi = acos(-1.0);
17 const double inf = 1e18;
18 const double eps = 1e-6;
19 const LL mod = 1e9 + 7;
20 LL qpow(LL a, LL b) {
21     LL s = 1;
22     while(b > 0) {
23         if(b & 1)
24             s = s * a % mod;
25         a = a * a % mod;
26         b >>= 1;
27     }
28     return s;
29 }
30 LL gcd(LL a, LL b) {

```

```

31     return b?gcd(b,a%b):a;
32 }
33 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
34 typedef pair<int,int> P;
35 struct Edge{
36     int from,to,dist;
37     Edge(int u,int v,int d):from(u),to(v),dist(d){
38     }
39 };
40 struct Bellman_ford {
41     #define maxn 1234567
42     bool inq[maxn]; // 用来记录入队次数
43     int cnt[maxn], d[maxn], p[maxn];
44     // cnt 来记录入队次数, 大于 n 就退出, d 用来记录最短距离, p 用来记录路径
45     int n,m;
46     int s,t;
47     vector<Edge> edges;
48     vector<int> G[maxn];
49     void AddEdge(int from,int to,int dist){
50         edges.push_back(Edge(from,to,dist));
51         edges.push_back(Edge(to,from,dist));
52         int m = edges.size();
53         G[from].push_back(m-2);
54         G[to].push_back(m-1);
55     }
56     void init(){
57
58         scanf("%d %d %d %d",&n,&m,&s,&t);
59         int u,v,c;
60         for(int i = 0; i < m; ++i){
61             scanf("%d %d %d",&u,&v,&c);
62             AddEdge(u,v,c);
63         }
64         /// cout<<"test"<<endl;
65     }
66     bool bellman_ford() {
67         queue<int> Q;
68         memset(inq,0,sizeof(inq));
69         memset(cnt,0,sizeof(cnt));
70         for(int i = 1; i <= n; ++i)
71             d[i] = INF;
72         d[s] = 0;
73         inq[s] = true;
74         Q.push(s);
75
76         while(!Q.empty()) {
77             int u = Q.front();
78             Q.pop();
79             inq[u] = false;
80             for(int i = 0; i < G[u].size(); ++i) {
81                 Edge &e = edges[G[u][i]];
82                 if(d[u] < INF&& d[e.to] > d[u]+e.dist) {
83                     d[e.to] = d[u]+e.dist;
84                     p[e.to] = G[u][i];
85                     if(!inq[e.to]) {
86

```

```

87         inq[e.to] = true;
88         if(++cnt[e.to] > n)
89             return false;
90     }
91 }
92 }
93 }
94 printf("%d\n",d[t]);
95
96 }
97 };
98 Bellman_ford bell;
99 int main(void) {
100     bell.init();
101     bell.bellman_ford();
102
103     return 0;
104 }

```

---

#### 4.6.3 3 floyed.cpp

---

```

1 // https://hihocoder.com/problemset/problem/1089?sid=1348128
2 #include <bits/stdc++.h>
3 #define mem(ar,num) memset(ar,num,sizeof(ar))
4 #define me(ar) memset(ar,0,sizeof(ar))
5 #define lowbit(x) (x&(-x))
6 #define Pb push_back
7 #define FI first
8 #define SE second
9 #define For(i,a,b) for(int i = a; i < b; ++i)
10 #define IOS ios::sync_with_stdio(false)
11 using namespace std;
12 typedef long long LL;
13 typedef unsigned long long ULL;
14 const int prime = 999983;
15 const int INF = 0x7FFFFFFF;
16 const LL INFF = 0x7FFFFFFFFFFFFFFF;
17 const double pi = acos(-1.0);
18 const double inf = 1e18;
19 const double eps = 1e-6;
20 const LL mod = 1e9 + 7;
21 LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
22 LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
23 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
24 typedef pair<int,int> P;
25 struct Floyd{
26     // 复杂度 O(n^3)
27     #define maxn 300
28     int d[maxn][maxn];
29     int n,m;
30     void init(void){
31         scanf("%d %d",&n,&m);
32         for(int i = 1;i <= n ;++i)
33             for(int j = 1;j <= n; ++j)
34                 if(i != j)

```

```

35         d[i][j] = INF;
36     int u,v,c;
37     for(int i = 0;i < m; ++i){
38         scanf("%d %d %d",&u,&v,&c);
39         d[u][v] = d[v][u] = min(d[v][u],c);
40     }
41 }
42 void floyd(void){
43     for(int k = 1; k <= n; ++k)
44         for(int i = 1;i <= n ;++i)
45             for(int j = 1;j <= n; ++j)
46                 if(d[i][k] < INF&& d[j][k] < INF)
47                     d[i][j] = min(d[i][j],d[i][k]+d[j][k]);
48 }
49 void print(void){
50     for(int i = 1;i <= n; ++i){
51         for(int j = 1;j <= n; ++j)
52             printf("%d%c",d[i][j], " \n"[j==n]);
53     }
54 }
55
56 };
57 Floyd floyd;
58 int main(void)
59 {
60     floyd.init();
61     floyd.floyd();
62     floyd.print();
63
64
65     return 0;
66 }

```

---

#### 4.6.4 堆优化的有限队列.cpp

---

```

1  #include <cstdio>
2  #include <iostream>
3  #include <algorithm>
4  #include <ext/pb_ds/priority_queue.hpp>
5  #define N 1000010
6  #define M 10000010
7  #define inf 1000000000000000ll
8
9  using namespace std;
10 using namespace __gnu_pbds;
11
12 typedef long long ll;
13 typedef pair<ll,int> pairs;
14 typedef __gnu_pbds::priority_queue<pairs,greater<pairs>,pairing_heap_tag> heap;
15
16 heap Q;
17 heap::point_iterator p[N];
18 int n,m,t,cnt;
19 ll rxa,rxr,rya,ryc,rp;
20 int G[N],vis[N];

```



```

21 ll dis[N];
22 struct edge{
23     int t,nx;
24     ll w;
25 }E[M];
26
27 inline void Insert(int x,int y,ll w){
28     E[++cnt].t=y;E[cnt].nx=G[x];E[cnt].w=w;G[x]=cnt;
29 }
30
31 inline void dijkstra(){
32     for(int i=1;i<=n;i++) dis[i]=inf;
33     dis[1]=0; vis[1]=0; p[1]=Q.push(pairs(0,1));
34     while(!Q.empty()){
35         int x=Q.top().second; Q.pop(); vis[x]=0;
36         for(int i=G[x];i;i=E[i].nx)
37             if(dis[E[i].t]>dis[x]+E[i].w){
38                 dis[E[i].t]=dis[x]+E[i].w;
39                 if(vis[E[i].t]) Q.modify(p[E[i].t],pairs(dis[E[i].t],E[i].t));
40                 else p[E[i].t]=Q.push(pairs(dis[E[i].t],E[i].t)),vis[E[i].t]=1;
41             }
42     }
43 }
44
45 int main(){
46     freopen("1.in","r",stdin);
47     freopen("1.out","w",stdout);
48     scanf("%d%d%d%d%d%d%d",&n,&m,&t,&rxa,&rxs,&rya,&ryc,&rp);
49     ll x=0,y=0,z=0,a,b;
50     for(int i=1;i<=t;i++){
51         x=(x*rxs+ryc)%rp;
52         y=(y*rya+ryc)%rp;
53         a=min(x%n+1,y%n+1);
54         b=max(y%n+1,y%n+1);
55         Insert(a,b,1e8-100*a);
56     }
57     for(int i=1;i<=m-t;i++){
58         scanf("%lld%lld%lld",&x,&y,&a);
59         Insert(x,y,a);
60     }
61     dijkstra();
62     printf("%lld\n",dis[n]);
63 }

```

---

## 5 数学

### 5.1 3 FWT 模板.cpp

---

```

1 // 异或
2 void FWT(int *a,int N,int opt){
3     const int inv2 = qpow(2,mod-2);
4     // j 是区间开始点, i 是区间距离, k 是具体位置, j+k,i+j+k 就是在 a 数组中的坐标
5     for(int i = 1;i < N; i <= 1){
6         for(int p = i<<1,j = 0;j < N; j += p){

```

```

7         for(int k = 0; k < i; ++k){
8             int X = a[j+k], Y = a[i+j+k];
9             a[j+k] = (X+Y)%mod;
10            a[i+j+k] = (X+mod-Y)%mod;
11            if(opt == -1) a[j+k] = 111*a[j+k]*inv2%mod, a[i+j+k] =
                ↪ 111*a[i+j+k]*inv2%mod;
12
13
14            }
15        }
16    }
17 }
18
19 或
20 if(opt == 1) F[i+j+k] = (F[i+j+k]+F[j+k]) %mod;
21 else        F[i+j+k] = (F[i+j+k+mod-F[j+k]) %mod;
22 和
23 if(opt == 1) F[j+k] = (F[j+k]+F[i+j+k]) %mod;
24 else        F[j+k] = (F[j+k] +mod-F[i+j+k])%mod;

```

---

## 5.2 4 单纯形法.cpp

---

```

1 // UVA10498 Happiness!
2 // Rujia Liu
3 #include<cstdio>
4 #include<cstring>
5 #include<algorithm>
6 #include<cassert>
7 using namespace std;
8
9 // 改进单纯性法的实现
10 // 参考: http://en.wikipedia.org/wiki/Simplex\_algorithm
11 // 输入矩阵 a 描述线性规划的标准形式。a 为 m+1 行 n+1 列，其中行 0~m-1 为不等式，行 m 为
    ↪ 目标函数（最大化）。列 0~n-1 为变量 0~n-1 的系数，列 n 为常数项
12 // 第 i 个约束为 a[i][0]*x[0] + a[i][1]*x[1] + ... <= a[i][n]
13 // 目标为 max(a[m][0]*x[0] + a[m][1]*x[1] + ... + a[m][n-1]*x[n-1] - a[m][n])
14 // 注意：变量均有非负约束 x[i] >= 0
15 const int maxm = 500; // 约束数目上限
16 const int maxn = 500; // 变量数目上限
17 const double INF = 1e100;
18 const double eps = 1e-10;
19
20 struct Simplex {
21     int n; // 变量个数
22     int m; // 约束个数
23     double a[maxm][maxn]; // 输入矩阵
24     int B[maxm], N[maxn]; // 算法辅助变量
25
26     void pivot(int r, int c) {
27         swap(N[c], B[r]);
28         a[r][c] = 1 / a[r][c];
29         for(int j = 0; j <= n; j++) if(j != c) a[r][j] *= a[r][c];
30         for(int i = 0; i <= m; i++) if(i != r) {
31             for(int j = 0; j <= n; j++) if(j != c) a[i][j] -= a[i][c] * a[r][j];
32             a[i][c] = -a[i][c] * a[r][c];

```

```

33     }
34 }
35
36 bool feasible() {
37     for(;;) {
38         int r, c;
39         double p = INF;
40         for(int i = 0; i < m; i++) if(a[i][n] < p) p = a[r = i][n];
41         if(p > -eps) return true;
42         p = 0;
43         for(int i = 0; i < n; i++) if(a[r][i] < p) p = a[r][c = i];
44         if(p > -eps) return false;
45         p = a[r][n] / a[r][c];
46         for(int i = r+1; i < m; i++) if(a[i][c] > eps) {
47             double v = a[i][n] / a[i][c];
48             if(v < p) { r = i; p = v; }
49         }
50         pivot(r, c);
51     }
52 }
53
54 // 解有界返回 1, 无解返回 0, 无界返回-1。b[i] 为 x[i] 的值, ret 为目标函数的值
55 int simplex(int n, int m, double x[maxn], double& ret) {
56     this->n = n;
57     this->m = m;
58     for(int i = 0; i < n; i++) N[i] = i;
59     for(int i = 0; i < m; i++) B[i] = n+i;
60     if(!feasible()) return 0;
61     for(;;) {
62         int r, c;
63         double p = 0;
64         for(int i = 0; i < n; i++) if(a[m][i] > p) p = a[m][c = i];
65         if(p < eps) {
66             for(int i = 0; i < n; i++) if(N[i] < n) x[N[i]] = 0;
67             for(int i = 0; i < m; i++) if(B[i] < n) x[B[i]] = a[i][n];
68             ret = -a[m][n];
69             return 1;
70         }
71         p = INF;
72         for(int i = 0; i < m; i++) if(a[i][c] > eps) {
73             double v = a[i][n] / a[i][c];
74             if(v < p) { r = i; p = v; }
75         }
76         if(p == INF) return -1;
77         pivot(r, c);
78     }
79 }
80 };
81
82 /////////////// 题目相关
83 #include<cmath>
84 Simplex solver;
85
86 int main() {
87     int n, m;
88     while(scanf("%d%d", &n, &m) == 2) {

```

```

89     for(int i = 0; i < n; i++) scanf("%lf", &solver.a[m][i]); // 目标函数
90     solver.a[m][n] = 0; // 目标函数常数项
91     for(int i = 0; i < m; i++)
92         for(int j = 0; j < n+1; j++)
93             scanf("%lf", &solver.a[i][j]);
94     double ans, x[maxn];
95     assert(solver.simplex(n, m, x, ans) == 1);
96     ans *= m;
97     printf("Nasa can spend %d taka.\n", (int)floor(ans + 1 - eps));
98 }
99 return 0;
100 }

```

---

### 5.3 5. 线性基.cpp

```

1  #include<bits/stdc++.h>
2  #define reg register
3  using namespace std;
4  typedef long long LL;
5  const int MN=60;
6  LL a[61],tmp[61];
7  bool flag;
8  void ins(LL x){
9      for(reg int i=MN;~i;i--)
10         if(x&(1LL<<i))
11             if(!a[i]){a[i]=x;return;}
12             else x^=a[i];
13     flag=true;
14 }
15 bool check(LL x){
16     for(reg int i=MN;~i;i--)
17         if(x&(1LL<<i))
18             if(!a[i])return false;
19             else x^=a[i];
20     return true;
21 }
22 LL qmax(LL res=0){
23     for(reg int i=MN;~i;i--)
24         res=max(res,res^a[i]);
25     return res;
26 }
27 LL qmin(){
28     if(flag)return 0;
29     for(reg int i=0;i<=MN;i++){
30         if(a[i])return a[i];
31     }
32 }
33 LL query(LL k){
34     reg LL res=0;reg int cnt=0;
35     k-=flag;if(!k)return 0;
36     for(reg int i=0;i<=MN;i++){
37         for(int j=i-1;~j;j--)
38             if(a[i]&(1LL<<j))a[i]^=a[j];
39         if(a[i])tmp[cnt++]=a[i];
40     }
41     if(k>=(1LL<<cnt))return -1;

```

```

41     for(reg int i=0;i<cnt;i++)
42         if(k&(1LL<<i))res^=tmp[i];
43     return res;
44 }
45 int main(){
46     int n;LL x;scanf("%d",&n);
47     for(int i=1;i<=n;i++)scanf("%lld",&x),ins(x);
48     printf("%lld\n",qmax());
49     return 0;
50 }

```

---

## 5.4 BM.cpp

---

```

1 //O(n^2) n 是传入的数
2 //输入的 n 是第几个数
3
4
5 #include<bits/stdc++.h>
6 using namespace std;
7 #define rep(i,a,n) for (int i=a;i<n;i++)
8 #define per(i,a,n) for (int i=n-1;i>=a;i--)
9 #define pb push_back
10 #define mp make_pair
11 #define all(x) (x).begin(),(x).end()
12 #define fi first
13 #define se second
14 #define SZ(x) ((int)(x).size())
15 typedef vector<int> VI;
16 typedef long long ll;
17 typedef pair<int,int> PII;
18 const ll mod=1000000007;
19 ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
20     ↪ for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
21 ll _,n;
22 namespace linear_seq{
23     const int N=10010;
24     ll res[N],base[N],_c[N],_md[N];
25     vector<ll> Md;
26     void mul(ll *a,ll *b,int k)
27     {
28         rep(i,0,k+k) _c[i]=0;
29         rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
30         for (int i=k+k-1;i>=k;i--) if (_c[i])
31             rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
32         rep(i,0,k) a[i]=_c[i];
33     }
34     int solve(ll n,VI a,VI b)
35     {
36         ll ans=0,pnt=0;
37         int k=SZ(a);
38         assert(SZ(a)==SZ(b));
39         rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
40         Md.clear();
41         rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
42         rep(i,0,k) res[i]=base[i]=0;

```

```

42     res[0]=1;
43     while ((1ll<<pnt)<=n) pnt++;
44     for (int p=pnt;p>=0;p--)
45     {
46         mul(res,res,k);
47         if ((n>>p)&1)
48         {
49             for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
50             rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
51         }
52     }
53     rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
54     if (ans<0) ans+=mod;
55     return ans;
56 }
57 VI BM(VI s) {
58     VI C(1,1),B(1,1);
59     int L=0,m=1,b=1;
60     rep(n,0,SZ(s)) {
61         ll d=0;
62         rep(i,0,L+1) d=(d+(1ll)C[i]*s[n-i])%mod;
63         if (d==0) ++m;
64         else if (2*L<=n) {
65             VI T=C;
66             ll c=mod-d*powmod(b,mod-2)%mod;
67             while (SZ(C)<SZ(B)+m) C.pb(0);
68             rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
69             L=n+1-L; B=T; b=d; m=1;
70         } else {
71             ll c=mod-d*powmod(b,mod-2)%mod;
72             while (SZ(C)<SZ(B)+m) C.pb(0);
73             rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
74             ++m;
75         }
76     }
77     return C;
78 }
79 int gao(VI a,ll n){
80     VI c=BM(a);
81     c.erase(c.begin());
82     rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
83     return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
84 }
85 };
86 int main()
87 {
88     int t;
89     scanf("%d",&t);
90     while(t--)
91     {
92         scanf("%lld",&n);
93         vector<int>v
94         ↪ {2,3,4,5,7,9,12,15,19,24,31,40,52,67,86,110,141,181,233,300,386,496,637};
95         // n = v.size();

```

```

95         //
        ↪ v.push_back({2,3,4,5,7,9,12,15,19,24,31,40,52,67,86,110,141,181,233,300,386,496});
        ↪ //至少 8 项, 越多越好。
96     printf("%lld\n",linear_seq::gao(v,n-1)%mod);
97 }
98 }

```

---

## 5.5 Combinatorial mathematics

### 5.5.1 康托展开.cpp

```

1  int cantor(int a[],int n){//cantor 展开,n 表示是 n 位的全排列, a[] 表示全排列的数
2      int ans=0,sum=0;
3      for(int i=1;i<n;i++){
4          for(int j=i+1;j<=n;j++){
5              if(a[j]<a[i])
6                  sum++;
7          ans+=sum*factorial[n-i];//累积
8          sum=0;//计数器归零
9      }
10     return ans+1;
11 }
12
13
14 static const int FAC[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880}; // 阶
    ↪ 乘
15
16 //康托展开逆运算
17 void decantor(int x, int n)
18 {
19     vector<int> v; // 存放当前可选数
20     vector<int> a; // 所求排列组合
21     for(int i=1;i<=n;i++)
22         v.push_back(i);
23     for(int i=n;i>=1;i--){
24         {
25             int r = x % FAC[i-1];
26             int t = x / FAC[i-1];
27             x = r;
28             sort(v.begin(),v.end());// 从小到大排序
29             a.push_back(v[t]); // 剩余数里第 t+1 个数为当前位
30             v.erase(v.begin()+t); // 移除选做当前位的数
31         }
32     }

```

---

## 5.6 FFT

### 5.6.1 FFT.cpp

```

1  const double PI = acos(-1.0);
2  struct Complex
3  {
4      double r,i;
5      Complex(double _r = 0,double _i = 0){
6          r = _r; i = _i;

```

```

7     }
8     Complex operator +(const Complex &b) {
9         return Complex(r+b.r,i+b.i);
10    }
11    Complex operator -(const Complex &b) {
12        return Complex(r-b.r,i-b.i);
13    }
14    Complex operator *(const Complex &b){
15        return Complex(r*b.r-i*b.i,r*b.i+i*b.r);
16    }
17 };
18
19 void FFT(Complex y[],int n ,int on)
20 {
21     for(int i = 0, j = 0; i < n; i++) {
22         if(j > i) swap(y[i], y[j]);
23         int k = n;
24         while(j & (k >>= 1)) j &= ~k;
25         j |= k;
26     }
27     for(int h = 2;h <= n;h <<= 1){
28         Complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
29         for(int j = 0;j < n;j += h){
30             Complex w(1,0);
31             for(int k = j;k < j+h/2;k++){
32                 Complex u = y[k];
33                 Complex t = w*y[k+h/2];
34                 y[k] = u+t;
35                 y[k+h/2] = u-t;
36                 w = w*wn;
37             }
38         }
39     }
40     if(on == -1)
41         for(int i = 0;i < n;i++)
42             y[i].r /= n;
43 }

```

---

### 5.6.2 kuangbin.cpp

```

1  #include <stdio.h>
2  #include <iostream>
3  #include <string.h>
4  #include <algorithm>
5  #include <math.h>
6  using namespace std;
7
8  const double PI = acos(-1.0);
9  struct complex
10 {
11     double r,i;
12     complex(double _r = 0,double _i = 0)
13     {
14         r = _r; i = _i;
15     }

```



```

16     complex operator +(const complex &b)
17     {
18         return complex(r+b.r,i+b.i);
19     }
20     complex operator -(const complex &b)
21     {
22         return complex(r-b.r,i-b.i);
23     }
24     complex operator *(const complex &b)
25     {
26         return complex(r*b.r-i*b.i,r*b.i+i*b.r);
27     }
28 };
29 void change(complex y[],int len)
30 {
31     int i,j,k;
32     for(i = 1, j = len/2;i < len-1;i++)
33     {
34         if(i < j)swap(y[i],y[j]);
35         k = len/2;
36         while( j >= k)
37         {
38             j -= k;
39             k /= 2;
40         }
41         if(j < k)j += k;
42     }
43 }
44 void fft(complex y[],int len,int on)
45 {
46     change(y,len);
47     for(int h = 2;h <= len;h <= 1)
48     {
49         complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
50         for(int j = 0;j < len;j += h)
51         {
52             complex w(1,0);
53             for(int k = j;k < j+h/2;k++)
54             {
55                 complex u = y[k];
56                 complex t = w*y[k+h/2];
57                 y[k] = u+t;
58                 y[k+h/2] = u-t;
59                 w = w*wn;
60             }
61         }
62     }
63     if(on == -1)
64         for(int i = 0;i < len;i++)
65             y[i].r /= len;
66 }
67
68 const int MAXN = 400040;
69 complex x1[MAXN];
70 int a[MAXN/4];
71 long long num[MAXN]; //100000*100000 会超 int

```

```

72 long long sum[MAXN];
73
74 int main()
75 {
76     int T;
77     int n;
78     scanf("%d",&T);
79     while(T--)
80     {
81         scanf("%d",&n);
82         memset(num,0,sizeof(num));
83         for(int i = 0;i < n;i++)
84         {
85             scanf("%d",&a[i]);
86             num[a[i]]++;
87         }
88         sort(a,a+n);
89         int len1 = a[n-1]+1;
90         int len = 1;
91         while( len < 2*len1 )len <= 1;
92         for(int i = 0;i < len1;i++)
93             x1[i] = complex(num[i],0);
94         for(int i = len1;i < len;i++)
95             x1[i] = complex(0,0);
96         fft(x1,len,1);
97         for(int i = 0;i < len;i++)
98             x1[i] = x1[i]*x1[i];
99         fft(x1,len,-1);
100         for(int i = 0;i < len;i++)
101             num[i] = (long long)(x1[i].r+0.5);
102         len = 2*a[n-1];
103         //减掉取两个相同的组合
104         for(int i = 0;i < n;i++)
105             num[a[i]+a[i]]--;
106         //选择的无序, 除以 2
107         for(int i = 1;i <= len;i++)
108         {
109             num[i]/=2;
110         }
111         sum[0] = 0;
112         for(int i = 1;i <= len;i++)
113             sum[i] = sum[i-1]+num[i];
114         long long cnt = 0;
115         for(int i = 0;i < n;i++)
116         {
117             cnt += sum[len]-sum[a[i]];
118             //减掉一个取大, 一个取小的
119             cnt -= (long long)(n-1-i)*i;
120             //减掉一个取本身, 另外一个取其它
121             cnt -= (n-1);
122             //减掉大于它的取两个的组合
123             cnt -= (long long)(n-1-i)*(n-i-2)/2;
124         }
125         //总数
126         long long tot = (long long)n*(n-1)*(n-2)/6;
127         printf("%.7lf\n",(double)cnt/tot);

```

```

128     }
129     return 0;
130 }

```

---

### 5.6.3 lrj.cpp

---

```

1  #include <bits/stdc++.h>
2  #define mem(ar,num) memset(ar,num,sizeof(ar))
3  #define me(ar) memset(ar,0,sizeof(ar))
4  #define lowbit(x) (x&(-x))
5  using namespace std;
6  typedef long long LL;
7  typedef unsigned long long ULL;
8  const int prime = 999983;
9  const int INF = 0x7FFFFFFF;
10 const LL INFF = 0x7FFFFFFFFFFFFFFF;
11 //const double pi = acos(-1.0);
12 const double inf = 1e18;
13 const double eps = 1e-6;
14 const LL mod = 1e9 + 7;
15 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
16 // UVA12298 Super Poker II
17 // Rujia Liu
18
19 const long double PI = acos(0.0) * 2.0;
20
21 typedef complex<double> CD;
22
23 // Cooley-Tukey 的 FFT 算法，迭代实现。inverse = false 时计算逆 FFT
24 inline void FFT(vector<CD> &a, bool inverse) {
25     int n = a.size();
26     // 原地快速 bit reversal
27     for(int i = 0, j = 0; i < n; i++) {
28         if(j > i) swap(a[i], a[j]);
29         int k = n;
30         while(j & (k >>= 1)) j &= ~k;
31         j |= k;
32     }
33
34     double pi = inverse ? -PI : PI;
35     for(int step = 1; step < n; step <= 1) {
36         // 把每相邻两个“step 点 DFT”通过一系列蝴蝶操作合并为一个“2*step 点 DFT”
37         double alpha = pi / step;
38         // 为求高效，我们并不是依次执行各个完整的 DFT 合并，而是枚举下标 k
39         // 对于一个下标 k，执行所有 DFT 合并中该下标对应的蝴蝶操作，即通过 E[k] 和 O[k] 计算
40         // ↪ X[k]
41         // 蝴蝶操作参考：http://en.wikipedia.org/wiki/Butterfly_diagram
42         for(int k = 0; k < step; k++) {
43             // 计算 omega^k。这个方法效率低，但如果用每次乘 omega 的方法递推会有精度问题。
44             // 有更快更精确的递推方法，为了清晰起见这里略去
45             CD omegak = exp(CD(0, alpha*k));
46             for(int Ek = k; Ek < n; Ek += step < 1) { // Ek 是某次 DFT 合并中 E[k] 在原
47                 // ↪ 始序列中的下标
48                 int Ok = Ek + step; // Ok 是该 DFT 合并中 O[k] 在原始序列中的下标
49                 CD t = omegak * a[Ok]; // 蝴蝶操作：x1 * omega^k

```

```

48         a[Ok] = a[Ek] - t; // 蝴蝶操作:  $y_1 = x_0 - t$ 
49         a[Ek] += t;       // 蝴蝶操作:  $y_0 = x_0 + t$ 
50     }
51 }
52 }
53
54 if(inverse)
55     for(int i = 0; i < n; i++) a[i] /= n;
56 }
57
58 // 用 FFT 实现的快速多项式乘法
59 inline vector<double> operator * (const vector<double>& v1, const vector<double>&
    ↪ v2) {
60     int s1 = v1.size(), s2 = v2.size(), S = 2;
61     while(S < s1 + s2) S <<= 1;
62     vector<CD> a(S,0), b(S,0); // 把 FFT 的输入长度补成 2 的幂, 不小于 v1 和 v2 的长度
    ↪ 之和
63     for(int i = 0; i < s1; i++) a[i] = v1[i];
64     FFT(a, false);
65     for(int i = 0; i < s2; i++) b[i] = v2[i];
66     FFT(b, false);
67     for(int i = 0; i < S; i++) a[i] *= b[i];
68     FFT(a, true);
69     vector<double> res(s1 + s2 - 1);
70     for(int i = 0; i < s1 + s2 - 1; i++) res[i] = a[i].real(); // 虚部均为 0
71     return res;
72 }

```

---

## 5.7 Lagrange-poly

### 5.7.1 template.cpp

```

1 // 适用范围, 求 n 次多项式第 x 项的值
2
3
4 namespace polysum {
5     #define rep(i,a,n) for (int i=a;i<n;i++)
6     #define per(i,a,n) for (int i=n-1;i>=a;i--)
7     const int D=1e6+10;
8     ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
9     ll powmod(ll a,ll b){ll
    ↪ res=1;a%=mod;assert(b>=0);for(;b>=>1){if(b&1)res=res*a%mod;a=a*a%mod;}return
    ↪ res;}
10 //.....
11 // 已知 a_i 的 d 次多项式, 求第 n 项
12 ll calcn(int d,ll *a,ll n) { // a[0].. a[d] a[n]
13     if (n<=d) return a[n];
14     p1[0]=p2[0]=1;
15     rep(i,0,d+1) {
16         ll t=(n-i+mod)%mod;
17         p1[i+1]=p1[i]*t%mod;
18     }
19     rep(i,0,d+1) {
20         ll t=(n-d+i+mod)%mod;
21         p2[i+1]=p2[i]*t%mod;

```

```

22     }
23     ll ans=0;
24     rep(i,0,d+1) {
25         ll t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
26         if ((d-i)&1) ans=(ans-t+mod)%mod;
27         else ans=(ans+t)%mod;
28     }
29     return ans;
30 }
31 // 初始化, 初始化的时候记得将 D 的值
32 void init(int M) {
33     f[0]=f[1]=g[0]=g[1]=1;
34     rep(i,2,M+5) f[i]=f[i-1]*i%mod;
35     g[M+4]=powmod(f[M+4],mod-2);
36     per(i,1,M+4) g[i]=g[i+1]*(i+1)%mod;
37 }
38 // 已知 a_i, 并且知道 a_i 是 m 次多项式
39 ll polysum(ll m,ll *a,ll n) { // a[0].. a[m] \sum_{i=0}^n a[i]
40     ll b[D];
41     ll b[D];
42     for(int i=0;i<=m;i++) b[i]=a[i];
43     b[m+1]=calcn(m,b,m+1);
44     rep(i,1,m+2) b[i]=(b[i-1]+b[i])%mod;
45     return calcn(m+1,b,n); // m 次多项式的和是 m+1 次多项式
46 }
47
48 ll qpolysum(ll R,ll n,ll *a,ll m) {
49     // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
50     if (R==1) return polysum(n,a,m);
51     a[m+1]=calcn(m,a,m+1);
52     ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;
53     h[0][0]=0;h[0][1]=1;
54     rep(i,1,m+2) {
55         h[i][0]=(h[i-1][0]+a[i-1])*r%mod;
56         h[i][1]=h[i-1][1]*r%mod;
57     }
58     rep(i,0,m+2) {
59         ll t=g[i]*g[m+1-i]%mod;
60         if (i&1)
61             ↪ p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+mod)%mod;
62         else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
63     }
64     c=powmod(p4,mod-2)*(mod-p3)%mod;
65     rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]*c)%mod;
66     rep(i,0,m+2) C[i]=h[i][0];
67     ans=(calcn(m,C,n)*powmod(R,n)-c)%mod;
68     if (ans<0) ans+=mod;
69     return ans;
70 } // polysum::init();

```

## 5.8 三分.cpp

```

1 //1142 : 三分·三分求极值
2 #include <bits/stdc++.h>

```

```

3  #define mem(ar,num) memset(ar,num,sizeof(ar))
4  #define me(ar) memset(ar,0,sizeof(ar))
5  #define lowbit(x) (x&(-x))
6  #define Pb push_back
7  #define FI first
8  #define SE second
9  #define For(i,a,b) for(int i = a; i < b; ++i)
10 #define IOS ios::sync_with_stdio(false)
11 using namespace std;
12 typedef long long LL;
13 typedef unsigned long long ULL;
14 const int prime = 999983;
15 const int INF = 0x7FFFFFFF;
16 const LL INFF = 0x7FFFFFFFFFFFFFFF;
17 const double pi = acos(-1.0);
18 const double inf = 1e18;
19 const double eps = 1e-9;
20 const LL mod = 1e9 + 7;
21 LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
22 LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
23 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
24 typedef pair<int,int> P;
25 double a,b,c,X,Y;
26 double f(double xx){
27     return a*xx*xx+b*xx+c;
28 }
29 double d(double x){
30     double t = a*x*x+b*x+c;
31     return sqrt((X-x)*(X-x)+(t-Y)*(t-Y));
32 }
33 }
34 int main(void)
35 {
36
37     cin>>a>>b>>c>>X>>Y;
38
39     double l,r,lm,rm;
40     l = -200.0,r = 200.0;
41     while(r - l >= eps){
42         lm = (r+l)/2;
43         rm = (r+lm)/2;
44         if(d(rm)<d(lm))
45             l = lm;
46         else
47             r = rm;
48     }
49
50     printf("%.31f\n",d(l));
51
52
53     return 0;
54 }

```

---

## 5.9 博弈

### 5.9.1 2. 威佐夫博弈.cpp

---

```
1 // 威佐夫博弈
2 // 两对石子，只能选择在一堆或者两堆石子里面取相同石子
3 // 打表发现规律，第 k 个必败点,  $a_k = b_k + k$ 
4 //  $a_k = (1 + \sqrt{5})/2 * k$ ，判断就是直接下面的式子了
5 int main(void)
6 {
7     int a, b;
8     while(cin >> a >> b){
9         if(a > b)
10             swap(a, b);
11         int c = floor((b - a) * ((1.0 + sqrt(5.0)) / 2.0));
12         if(a == c)
13             cout << 0 << endl;
14         else
15             cout << 1 << endl;
16     }
17     return 0;
18 }
```

---

### 5.9.2 3 Nim 积.cpp

---

```
1 /* 在一个二维平面中，有 n 个灯亮着并告诉你坐标，
2  每回合需要找到一个矩形，这个矩形 xy 坐标最大的那个角落的点必须是亮着的灯，
3  然后我们把四个角落的灯状态反转，不能操作为败
4  */
5 #include <set>
6 #include <map>
7 #include <stack>
8 #include <cmath>
9 #include <queue>
10 #include <vector>
11 #include <cstdio>
12 #include <cstring>
13 #include <iostream>
14 #include <algorithm>
15 typedef long long ll;
16 const int maxn = 1e6 + 10;
17 const int seed = 131;
18 const ll MOD = 1e9 + 7;
19 const int INF = 0x3f3f3f3f;
20 using namespace std;
21 int m[2][2] = {0, 0, 0, 1};
22 int Nim_Mul_Power(int x, int y){
23     if(x < 2) return m[x][y];
24     int a = 0;
25     for(; ; a++){
26         if(x >= (1 << (1 << a)) && x < (1 << (1 << (a + 1))))
27             break;
28     }
29     int m = 1 << (1 << a);
30     int p = x / m, s = y / m, t = y % m;
```

---

```

31     int d1 = Nim_Mul_Power(p, s);
32     int d2 = Nim_Mul_Power(p, t);
33     return (m * (d1 ^ d2)) ^ Nim_Mul_Power(m / 2, d1);
34 }
35 int Nim_Mul(int x, int y){
36     if(x < y) return Nim_Mul(y, x);
37     if(x < 2) return m[x][y];
38     int a = 0;
39     for(; ; a++){
40         if(x >= (1 << (1 << a)) && x < (1 << (1 << (a + 1))))
41             break;
42     }
43     int m = 1 << (1 << a);
44     int p = x / m, q = x % m, s = y / m, t = y % m;
45     int c1 = Nim_Mul(p, s), c2 = Nim_Mul(p, t) ^ Nim_Mul(q, s), c3 = Nim_Mul(q,
46         ↵ t);
47     return (m * (c1 ^ c2)) ^ c3 ^ Nim_Mul_Power(m / 2, c1);
48 }
49 int main(){
50     int T;
51     scanf("%d", &T);
52     int ans;
53     while(T--){
54         ans = 0;
55         int n, x, y;
56         scanf("%d", &n);
57         while(n--){
58             scanf("%d%d", &x, &y);
59             ans ^= Nim_Mul(x, y);
60         }
61         if(ans)
62             printf("Have a try, lxhgww.\n");
63         else
64             printf("Don't waste your time.\n");
65     }
66     return 0;
67 }

```

---

### 5.9.3 4 K 倍动态减法.cpp

```

1  /*
2  有 n 个石子，先手第一次最多取 n-1 个，之后如果前一个人取 m 个，
3  则下一个人可以取 1 到 k*m 个，取完最后一个为胜，
4  问先手是否会胜，如果会胜输出第一次取几个。
5  */
6  const int maxn = 2e6+100;
7  int a[maxn], b[maxn];
8  int main(void)
9  {
10     int T;
11     cin >> T;
12     for(int kase = 1; kase <= T; ++kase){
13         int n, k;
14         cin >> n >> k;
15         a[0] = 1, b[0] = 1;

```



```

16     int i = 0, j = 0;
17     while(a[i] < n){
18         i++;
19         a[i] = b[i-1]+1;
20         if(a[j+1] * k < a[i]) j++;
21         if(a[j] * k < a[i]) b[i] = b[j]+a[i];
22         else b[i] = a[i];
23
24     }
25     printf("Case %d: ", kase);
26     if(a[i] == n) {
27         puts("lose");
28         continue;
29     }
30     // i--;
31     while(i >= 0){
32         if(n-a[i] > 0)
33             n -= a[i];
34         if(n == a[i]) break;
35         i--;
36     }
37     printf("%d\n", n);
38 }
39
40 return 0;
41 }

```

---

#### 5.9.4 5 海盗分金问题.cpp

---

```

1  /*
2  A Puzzle for Pirates HDU - 1538
3  */
4
5  int solve(int n, int m, int q){
6      if(n <= 2*m+2){
7          if(q == n){
8              return m-(n-1)/2;
9          }
10         else{
11             if(q % 2 == n%2) return 1;
12             else return 0;
13         }
14     }
15     else{
16         if(q <= 2*m+2) return 0;
17         if(n == q)
18         {
19             LL t = 2*m+2;
20             while(t < n)
21                 t = 2*(t-m);
22             if(t == n) return 0;
23             else return -1;
24         }
25         else{
26             LL t = 2*m+2;

```

```

27         while(t < q)
28             t = 2*(t-m);
29         if(t <= n) return 0;
30         else      return -1;
31     }
32 }
33 }
34 int main(void)
35 {
36     int T;
37     cin>>T;
38     while(T--){
39         LL n,m,q;
40         cin>>n>>m>>q;
41         LL ans = solve(n,m,q);
42
43         if(ans == -1) puts("Thrown");
44         else printf("%lld\n",ans);
45     }
46
47     return 0;
48 }
49

```

---

### 5.9.5 6 Green Hackbush.cpp

---

```

1 // N 个点, M 条边
2
3 #include<bits/stdc++.h>
4 using namespace std;
5 #define min(x,y) ((x)<(y))?(x):(y)
6
7 int Cases,N,M;
8 vector< list<int> > G,G2;
9 vector<int> GV;
10 vector<int> visited,from,time_disc,time_up;
11 int DFStime;
12
13 void DFS_Visit(int v){
14     int edges_to_parent=0;
15     visited[v]=1; time_disc[v]=time_up[v]++DFStime;
16     for (list<int>::iterator start=G[v].begin();start!=G[v].end();start++) {
17         if (!visited[*start]) { from[*start]=v; DFS_Visit(*start);
18             ↪ time_up[v]=min(time_up[v],time_up[*start]); }
19         else {
20             if ((*start)!=from[v]) { time_up[v]=min(time_up[v],time_disc[*start]); }
21             else {
22                 if (edges_to_parent) { time_up[v]=min(time_up[v],time_disc[*start]); }
23                 edges_to_parent++;
24             }
25         }
26     }
27 }
28 void FindBridges(void){

```

```

29     time_disc.clear(); time_up.clear(); visited.clear(); from.clear();
30     visited.resize(N+3,0); time_disc.resize(N+3,0); time_up.resize(N+3,0);
31     ↪ from.resize(N+3,0);
32     from[1]=1; DFStime=0;
33     DFS_Visit(1);
34 }
35
36 int IsBridge(int v_lo, int v_high) {
37     if (v_high!=from[v_lo]) return 0;
38     return ( time_disc[v_lo]==time_up[v_lo] );
39 }
40
41 void ContractGraph(void){
42     vector<int> color(N+3,0);
43     int colors=1;
44     color[1]=1;
45
46     list<int> Q;
47     Q.clear(); Q.push_back(1);
48     while (!Q.empty()) {
49         int where=Q.front(); Q.pop_front();
50         for (list<int>::iterator it=G[where].begin(); it!=G[where].end(); it++) if
51             ↪ (!color[*it]) {
52             if (IsBridge(*it,where)) color[*it]=++colors; else color[*it]=color[where];
53             visited[*it]=1; Q.push_back(*it);
54         }
55     }
56
57     G2.clear(); G2.resize(N+3);
58     for (int i=1;i<=N;i++)
59         for (list<int>::iterator it=G[i].begin(); it!=G[i].end(); it++)
60             G2[color[i]].push_back(color[*it]);
61
62 int GrundyValue(int v){
63     int loops=0,gv=0;
64
65     if (GV[v]!=-1) return GV[v]; GV[v]=1000000000;
66
67     for (list<int>::iterator start=G2[v].begin(); start!=G2[v].end(); start++) {
68         if ((*start)==v) loops++; else if (GV[*start]!=1000000000)
69             ↪ gv^=(1+GrundyValue(*start));
70     }
71     loops/=2; if (loops%2) gv^=1;
72     return GV[v]=gv;
73 }
74
75 int main(void){
76     int v1,v2;
77     // freopen("input.txt","r",stdin);
78     // freopen("out.txt","w+",stdout);
79     cin >> Cases;
80     while (Cases--) {
81         // read graph dimensions
82         cin >> N >> M;
83         // read the graph

```

```

82     G.clear(); G.resize(N+3);
83     for (int i=0;i<M;i++) { cin >> v1 >> v2; G[v1].push_back(v2);
84         ↪ G[v2].push_back(v1); }
85     // collapse all circuits in the graph
86     FindBridges();
87     ContractGraph();
88     // compute the SG value
89     GV.clear(); for (int i=0;i<=N;i++) GV.push_back(-1);
90     int result=GrundyValue(1);
91     if (result) cout << "Alice\n"; else cout << "Bob\n"; // cout << result <<
92     ↪ "\n";
93
94     //cout << result << "\n";
95 }
96 return 0;
97 }
98
99 typedef pair<int,int> P;
100 vector<P> edges;
101 // 边连通分量
102 const int maxn = 1000+100;
103 // const int maxm = 1e6+100
104 int pre[maxn];
105 int dfs_clock = 0;
106 vector<int> G[maxn];
107 vector<int> G2[maxn];
108 bool Is[maxn];
109 int low[maxn];
110
111 void init(){
112     dfs_clock = 1;
113     rep(i,1,maxn) G[i].clear(),G2[i].clear();
114     me(low);
115     me(pre);
116     me(Is);
117 }
118 int dfs1(int u,int fa){
119     int lowu = pre[u] = ++dfs_clock;
120     int child = 0;
121     for(int i = 0;i < (int)G[u].size(); ++i){
122         int v = edges[G[u][i]].second;
123         if(!pre[v]){
124             child++;
125             int lowv = dfs1(v,u);
126             lowu = min(lowu,lowv);
127             if(lowv >= pre[u]){
128                 // iscut[u]++;
129                 Is[G[u][i]] = 1;
130             }
131         }
132         else if(pre[v] < pre[u] && v != fa){
133             lowu = min(lowu,pre[v]);
134         }
135     }

```

```

136     return low[u] = lowu;
137 }
138 // #define Debug
139
140 int belong[maxn];
141 int num[maxn];
142
143 void dfs(int u,int be){
144     belong[u] = be;
145     for(int i = 0;i < (int)G[u].size(); ++i){
146         if(Is[G[u][i]])
147             continue;
148         int v = edges[G[u][i]].second;
149         if(!belong[v])
150             dfs(v,be);
151     }
152 }
153 int SG(int u,int fa){
154     int t = 0;
155     for(int i = 0;i < (int)G2[u].size(); ++i){
156         int v = G2[u][i];
157         if(v==fa) continue;
158         t ^= (SG(v,u)+1);
159     }
160     if(num[u]&1) t ^= 1;
161     return t;
162 }
163 int main(void)
164 {
165     int n,m,k;
166     while(cin>>n){
167         int sum = 0;
168         while(n--){
169             init();
170             edges.clear();
171             me(belong);
172             me(num);
173             scanf("%d%d",&m,&k);
174             rep(i,0,k){
175                 int u,v;
176                 scanf("%d%d",&u,&v);
177                 edges.push_back(P(u,v));
178                 edges.push_back(P(v,u));
179                 G[u].push_back(edges.size()-2);
180                 G[v].push_back(edges.size()-1);
181             }
182             dfs1(1,-1);
183
184             int tot = 0;
185             rep(i,1,m+1)
186                 if(!belong[i])
187                     dfs(i,++tot);
188             // dfs(m+1,)
189             for(int i = 0;i < (int)edges.size(); i += 2){
190                 int x = belong[edges[i].first];
191                 int y = belong[edges[i].second];

```

```

192         if(x != y)
193             G2[x].Pb(y),G2[y].Pb(x);
194         else
195             num[x]++;
196     }
197
198     // cout<<SG(1,-1)<<endl;
199     sum ^= SG(1,-1);
200 }
201 if(sum)
202     puts("Sally");
203 else
204     puts("Harry");
205 }
206 return 0;
207 }

```

---

### 5.9.6 7 反 nim 博弈.cpp

```

1  /*
2  先手必胜当且仅当：
3  (1) 所有堆的石子数都为 1 且游戏的 SG 值为 0；
4  (2) 有些堆的石子数大于 1 且游戏的 SG 值不为 0。
5  对于任意一个 Anti-SG 游戏，如果我们规定当局面中所有的单一游戏的 SG 值为 0 时，游戏结束，
6  ↪ 则先手必胜当且仅当：
7  (1) 游戏的 SG 函数不为 0 且游戏中某个单一游戏的 SG 函数大于 1；
8  (2) 游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1。
9  Every-SG 游戏规定，对于还没有结束的单一游戏，游戏者必须
10  对该游戏进行一步决策；
11  Every-SG 游戏的其他规则与普通 SG 游戏相同
12  对于 Every-SG 游戏先手必胜当且仅当单一游戏中最大的 step 为奇数。
13  */

```

---

### 5.9.7 8 超自然数.cpp

```

1  //[POJ-2931]
2  // 超自然数求解不平等博弈问题
3  char ar[100];
4  bool b[100];
5  LL sureal(int n){
6      LL k = 1;
7      k <= 52;
8      for(int i = 0;i < n; ++i){
9          scanf("%s",ar);
10         if(ar[0] == 'W')
11             b[i] = 1;
12         else
13             b[i] = 0;
14     }
15     LL x = 0,i = 0;
16     while(i < n&&b[i] == b[0]){
17         if(b[i]) x += k;
18         else x -= k;
19         i++;

```

```

20     }
21     k >>= 1;
22     while(i < n){
23         if(b[i])
24             x += k;
25         else
26             x -= k;
27         i++;
28         k >>= 1;
29     }
30     return x;
31 }
32 int main(void)
33 {
34     int T;
35     cin>>T;
36     while(T--){
37         int n;
38         char br[100];
39         scanf("%s %d: ",br,&n);
40
41         LL ans1 = 0,ans2 = 0;
42         int a[3];
43         rep(i,0,3)    scanf("%d",&a[i]);
44         rep(i,0,3)    ans1 += surreal(a[i]);
45         rep(i,0,3)    scanf("%d",&a[i]);
46         rep(i,0,3)    ans2 += surreal(a[i]);
47         // cout<<ans1<<" "<<ans2<<endl;
48         printf("%s %d: ",br,n);
49         if(ans1 >= ans2)
50             puts("Yes");
51         else
52             puts("No");
53     }
54
55     return 0;
56 }

```

---

## 5.10 数论

### 5.10.1 1 加法.cpp

---

```

1 string add(string a,string b)
2 {
3     string c;
4     int len1=a.length();
5     int len2=b.length();
6     int len=max(len1,len2);
7     for(int i=len1;i<len;i++)
8         a="0"+a;
9     for(int i=len2;i<len;i++)
10        b="0"+b;
11     int ok=0;
12     for(int i=len-1;i>=0;i--)
13     {

```

```

14     char temp=a[i]+b[i]-'0'+ok;
15     if(temp>'9')
16     {
17         ok=1;
18         temp-=10;
19     }
20     else ok=0;
21     c=temp+c;
22 }
23 if(ok) c="1"+c;
24 return c;
25 }

```

---

### 5.10.2 1 逆元.cpp

---

```

1 // 欧几里得扩展
2 long long ex_gcd(long long a,long long b,long long &x,long long &y)
3 {
4     if(b == 0)
5     {
6         x = 1;
7         y = 0;
8         return a;
9     }
10    long long m = ex_gcd(b,a%b,y,x);
11    y -= a/b * x;
12    return m;
13 }
14 int main()
15 {
16    long long a,b,x,y;
17    cin>>a>>b; //求 a 关于 b 的逆元
18    if(ex_gcd(a,b,x,y)==1)
19        cout<<(x%b+b)%b<<endl;
20    else
21        cout<<"None"<<endl;
22    return 0;
23 }
24 // 费马小定理求逆元
25 qpow(a,p-2,p);
26 // 逆元打表
27
28 int inv[10000];
29 int p;
30 cin>>p;
31 inv[1] = 1;
32 for(int i = 2;i < p; ++i)
33 {
34     inv[i] = (p - p/i*inv[p%i]%p)%p;
35 }
36 for(int i = 1;i < p; ++i)
37     cout<<inv[i]<<" ";
38 cout<<endl;
39 for(int i = 1;i < p; ++i)
40     cout<<i * inv[i] % p<<" ";

```



```

41
42 // 快速阶乘逆元
43
44 const int maxn = 1e5+10;
45 long long fac[maxn], invfac[maxn];
46 void init(int n){
47     fac[0] = 1;
48     for(int i = 1; i <= n; ++i) fac[i] = fac[i-1]*i%mod;
49     invfac[n] = qpow(fac[n], mod-2);
50     for(int i = n-1; i >= 0; --i) invfac[i] = invfac[i+1]*(i+1)%mod;
51 }

```

---

### 5.10.3 2 减法.cpp

```

1 string sub(string a, string b)
2 {
3     string c;
4     bool ok=0;
5     int len1=a.length();
6     int len2=b.length();
7     int len=max(len1, len2);
8     for(int i=len1; i<len; i++)
9         a="0"+a;
10    for(int i=len2; i<len; i++)
11        b="0"+b;
12    if(a<b)
13    {
14        string temp=a;
15        a=b;
16        b=temp;
17        ok=1;
18    }
19    for(int i=len-1; i>=0; i--)
20    {
21        if(a[i]<b[i])
22        {
23            a[i-1]-=1;
24            a[i]+=10;
25        }
26        char temp=a[i]-b[i]+'0';
27        c=temp+c;
28    }
29    int pos=0;
30    while(c[pos]=='0' && pos<len) pos++;
31    if(pos==len) return "0";
32    if(ok) return "-" + c.substr(pos);
33    return c.substr(pos);
34 }

```

---

### 5.10.4 3 乘法.cpp

```

1 string mul(string a, int b)
2 {
3     string c;

```

```

4     char s;
5     int len=a.length();
6     int ok=0;
7     for(int i=len-1;i>=0;i--)
8     {
9         int temp=(a[i]- '0')*b+ok;
10        ok=temp/10;
11        s=temp%10+'0';
12        c=s+c;
13    }
14    while(ok)
15    {
16        s=ok%10+'0';
17        c=s+c;
18        ok/=10;
19    }
20    return c;
21 }

```

---

#### 5.10.5 4 除法.cpp

```

1 string div(string a,int b)
2 {
3     string c;
4     int len=a.length();
5     int ans=0;
6     char s;
7     for(int i=0;i<len;i++)
8     {
9         ans=ans*10+a[i]- '0';
10        s=ans/b+'0';
11        ans%=b;
12        c+=s;
13    }
14    int pos=0;
15    while(pos<len && c[pos]=='0') pos++;
16    if(pos==len) return "0";
17    return c.substr(pos);
18 }

```

---

#### 5.10.6 5. 蒙哥马利快速模.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define rep(i,a,n) for (int i=a;i<n;i++)
4 #define per(i,a,n) for (int i=n-1;i>=a;i--)
5 #define pb push_back
6 #define mp make_pair
7 #define all(x) (x).begin(),(x).end()
8 #define fi first
9 #define se second
10 #define SZ(x) ((int)(x).size())
11 typedef vector<int> VI;
12 typedef long long ll;

```

```

13 typedef pair<int,int> PII;
14 const ll mod=1000000007;
15 ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
    ↪ for(;b>=>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
16 ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
17 // head
18
19 typedef unsigned long long u64;
20 typedef __int128_t i128;
21 typedef __uint128_t u128;
22 int _,k;
23 u64 A0,A1,M0,M1,C,M;
24
25 struct Mod64 {
26     Mod64():n_(0) {}
27     Mod64(u64 n):n_(init(n)) {}
28     static u64 init(u64 w) { return reduce(u128(w) * r2); }
29     static void set_mod(u64 m) {
30         mod=m; assert(mod&1);
31         inv=m; rep(i,0,5) inv*=2-inv*m;
32         r2=-u128(m)%m;
33     }
34     static u64 reduce(u128 x) {
35         u64 y=u64(x>>64)-u64((u128(u64(x)*inv)*mod)>>64);
36         return ll(y)<0?y+mod:y;
37     }
38     Mod64& operator += (Mod64 rhs) { n_+=rhs.n_-mod; if (ll(n_)<0) n_+=mod; return
    ↪ *this; }
39     Mod64 operator + (Mod64 rhs) const { return Mod64(*this)+=rhs; }
40     Mod64& operator -= (Mod64 rhs) { n_-=rhs.n_; if (ll(n_)<0) n_+=mod; return
    ↪ *this; }
41     Mod64 operator - (Mod64 rhs) const { return Mod64(*this)-=rhs; }
42     Mod64& operator *= (Mod64 rhs) { n_ = reduce(u128(n_)*rhs.n_); return *this; }
43     Mod64 operator * (Mod64 rhs) const { return Mod64(*this)*=rhs; }
44     u64 get() const { return reduce(n_); }
45     static u64 mod,inv,r2;
46     u64 n_;
47 };
48 u64 Mod64::mod,Mod64::inv,Mod64::r2;
49
50 u64 pmod(u64 a,u64 b,u64 p) {
51     u64 d=(u64)floor(a*(long double)b/p+0.5);
52     ll ret=a*b-d*p;
53     if (ret<0) ret+=p;
54     return ret;
55 }
56
57
58 void bruteforce() {
59     u64 ans=1;
60     for (int i=0;i<=k;i++) {
61         ans=pmod(ans,A0,M);
62         u64 A2=pmod(M0,A1,M)+pmod(M1,A0,M)+C;
63         while (A2>=M) A2-=M;
64         A0=A1; A1=A2;
65     }

```



```

36         phi[i*Prime[j]] = phi[i]*Prime[j];
37         break;
38     }
39     else{
40         phi[i*Prime[j]] = phi[i]*(Prime[j]-1);
41     }
42 }
43 }
44
45 }
46 ```

```

---

### 5.10.8 lucas , 组合数.cpp

```

1  LL qpow(LL a,LL b,LL m){
2      LL ans = 1;
3      a %= m;
4      while(b > 0){
5          if(b&1)
6              ans = ans*a%m;
7              a = a*a%m;
8              b >>= 1;
9      }
10     return ans;
11 }
12 LL C(LL n,LL m,LL p){
13     if(m > n) return 0;
14     LL tmp1 = 1,tmp2 = 1;
15     m = min(n-m,m);
16     for(LL i = 1;i <= m; ++i){
17         tmp1 = tmp1*(n-m+i)%p;
18         tmp2 = tmp2*i%p;
19     }
20     return tmp1*qpow(tmp2,p-2,p)%p;
21 }
22 LL lucas(LL n, LL m, LL p){
23     if(m == 0)
24         return 1;
25     return lucas(n/p,m/p,p)*C(n%p,m%p,p)%p;
26 }

```

---

### 5.10.9 miller-rabin-Pollard-rho.cpp

```

1  // 可以对一个  $2^{63}$  的素数进行判断。
2
3  可以分解比较大的数的因子。
4
5  #include<stdio.h>
6  #include<string.h>
7  #include<iostream>
8  #include<math.h>
9  #include<stdlib.h>
10 #include<time.h>
11 using namespace std;

```

```

12
13
14 typedef long long LL;
15 #define maxn 10000
16
17 LL factor[maxn];
18 int tot;
19 const int S=20;
20 LL muti_mod(LL a,LL b,LL c){ //返回 (a*b) mod c,a,b,c<2^63
21     a%=c;
22     b%=c;
23     LL ret=0;
24     while (b){
25         if (b&1){
26             ret+=a;
27             if (ret>=c) ret-=c;
28         }
29         a<<=1;
30         if (a>=c) a-=c;
31         b>>=1;
32     }
33     return ret;
34 }
35
36 LL pow_mod(LL x,LL n,LL mod){ //返回 x^n mod c , 非递归版
37     if (n==1) return x%mod;
38     int bit[90],k=0;
39     while (n){
40         bit[k++]=n&1;
41         n>>=1;
42     }
43     LL ret=1;
44     for (k=k-1;k>=0;k--){
45         ret=muti_mod(ret,ret,mod);
46         if (bit[k]==1) ret=muti_mod(ret,x,mod);
47     }
48     return ret;
49 }
50
51 bool check(LL a,LL n,LL x,LL t){ //以 a 为基, n-1=x*2^t, 检验 n 是不是合数
52     LL ret=pow_mod(a,x,n),last=ret;
53     for (int i=1;i<=t;i++){
54         ret=muti_mod(ret,ret,n);
55         if (ret==1 && last!=1 && last!=n-1) return 1;
56         last=ret;
57     }
58     if (ret!=1) return 1;
59     return 0;
60 }
61
62 bool Miller_Rabin(LL n){
63     LL x=n-1,t=0;
64     while ((x&1)==0) x>>=1,t++;
65     bool flag=1;
66     if (t>=1 && (x&1)==1){
67         for (int k=0;k<S;k++){

```

```

68         LL a=rand()%(n-1)+1;
69         if (check(a,n,x,t)) {flag=1;break;}
70         flag=0;
71     }
72 }
73 if (!flag || n==2) return 0;
74 return 1;
75 }
76
77 LL gcd(LL a,LL b){
78     if (a==0) return 1;
79     if (a<0) return gcd(-a,b);
80     while (b){
81         LL t=a%b; a=b; b=t;
82     }
83     return a;
84 }
85
86 LL Pollard_rho(LL x,LL c){
87     LL i=1,x0=rand()%x,y=x0,k=2;
88     while (1){
89         i++;
90         x0=(muti_mod(x0,x0,x)+c)%x;
91         LL d=gcd(y-x0,x);
92         if (d!=1 && d!=x){
93             return d;
94         }
95         if (y==x0) return x;
96         if (i==k){
97             y=x0;
98             k+=k;
99         }
100     }
101 }
102
103 void findfac(LL n){                //递归进行质因数分解 N
104     if (!Miller_Rabin(n)){
105         factor[tot++] = n;
106         return;
107     }
108     LL p=n;
109     while (p>=n) p=Pollard_rho(p,rand() % (n-1) +1);
110     findfac(p);
111     findfac(n/p);
112 }
113
114 int main()
115 {
116     // srand(time(NULL)); //POJ 上 G++ 要去掉这句话
117     int T;
118     scanf("%d",&T);
119     long long n;
120     while(T--){
121         {
122             scanf("%I64d",&n);
123             if (!Miller_Rabin(n)) {printf("Prime\n"); continue; }

```

```

124         tot = 0;
125         findfac(n);
126         long long ans=factor[0];
127         for(int i=1;i<tot;i++)
128             if(factor[i]<ans)ans=factor[i];
129         printf("%I64d\n",ans);
130     }
131     return 0;
132 }

```

---

#### 5.10.10 分段求和.cpp

---

```

1  int main(void)
2  {
3      std::ios::sync_with_stdio(false);
4      int T;
5      cin>>T;
6      int Kase = 0;
7      while(T--)
8      {
9          LL n;
10         cin>>n;
11         int m = (int)sqrt(n);
12         LL ans = 0;
13         for(LL i = 1;i < m; ++i)
14         {
15             ans += n/i;
16             ans += (LL)i*(n/i - n/(i+1));
17         }
18         ans += n/m;
19         ans += m*(n/m-m);
20         printf("Case %d: %lld\n",++Kase,ans);
21     }

```

---

#### 5.10.11 大数.cpp

---

```

1  #include<iostream>
2  #include<string>
3  #include<iomanip>
4  #include<algorithm>
5  using namespace std;
6
7  #define MAXN 9999
8  #define MAXSIZE 10
9  #define DLEN 4
10
11  class BigNum
12  {
13  private:
14      int a[500];    //可以控制大数的位数
15      int len;        //大数长度
16  public:
17      BigNum(){ len = 1;memset(a,0,sizeof(a)); } //构造函数
18      BigNum(const int);    //将一个 int 类型的变量转化为大数

```



```

19     BigNum(const char*);           //将一个字符串类型的变量转化为大数
20     BigNum(const BigNum &);       //拷贝构造函数
21     BigNum &operator=(const BigNum &); //重载赋值运算符，大数之间进行赋值运算
22
23     friend istream& operator>>(istream&, BigNum&); //重载输入运算符
24     friend ostream& operator<<(ostream&, BigNum&); //重载输出运算符
25
26     BigNum operator+(const BigNum &) const; //重载加法运算符，两个大数之间的相
    ↪ 加运算
27     BigNum operator-(const BigNum &) const; //重载减法运算符，两个大数之间的相
    ↪ 减运算
28     BigNum operator*(const BigNum &) const; //重载乘法运算符，两个大数之间的相
    ↪ 乘运算
29     BigNum operator/(const int &) const; //重载除法运算符，大数对一个整数进
    ↪ 行相除运算
30
31     BigNum operator^(const int &) const; //大数的 n 次方运算
32     int operator%(const int &) const; //大数对一个 int 类型的变量进行取模
    ↪ 运算
33     bool operator>(const BigNum & T) const; //大数和另一个大数的大小比较
34     bool operator>(const int & t) const; //大数和一个 int 类型的变量的大小
    ↪ 比较
35
36     void print(); //输出大数
37 };
38 BigNum::BigNum(const int b) //将一个 int 类型的变量转化为大数
39 {
40     int c,d = b;
41     len = 0;
42     memset(a,0,sizeof(a));
43     while(d > MAXN)
44     {
45         c = d - (d / (MAXN + 1)) * (MAXN + 1);
46         d = d / (MAXN + 1);
47         a[len++] = c;
48     }
49     a[len++] = d;
50 }
51 BigNum::BigNum(const char*s) //将一个字符串类型的变量转化为大数
52 {
53     int t,k,index,l,i;
54     memset(a,0,sizeof(a));
55     l=strlen(s);
56     len=l/DLEN;
57     if(l%DLEN)
58         len++;
59     index=0;
60     for(i=l-1;i>=0;i-=DLEN)
61     {
62         t=0;
63         k=i-DLEN+1;
64         if(k<0)
65             k=0;
66         for(int j=k;j<=i;j++)
67             t=t*10+s[j]-'0';
68         a[index++]=t;

```

```

69     }
70 }
71 BigNum::BigNum(const BigNum & T) : len(T.len) //拷贝构造函数
72 {
73     int i;
74     memset(a,0,sizeof(a));
75     for(i = 0 ; i < len ; i++)
76         a[i] = T.a[i];
77 }
78 BigNum & BigNum::operator=(const BigNum & n) //重载赋值运算符，大数之间进行赋值运算
79 {
80     int i;
81     len = n.len;
82     memset(a,0,sizeof(a));
83     for(i = 0 ; i < len ; i++)
84         a[i] = n.a[i];
85     return *this;
86 }
87 istream& operator>>(istream & in, BigNum & b) //重载输入运算符
88 {
89     char ch[MAXSIZE*4];
90     int i = -1;
91     in>>ch;
92     int l=strlen(ch);
93     int count=0,sum=0;
94     for(i=l-1;i>=0;)
95     {
96         sum = 0;
97         int t=1;
98         for(int j=0;j<4&& i>=0;j++,i--,t*=10)
99         {
100             sum+=(ch[i]-'0')*t;
101         }
102         b.a[count]=sum;
103         count++;
104     }
105     b.len =count++;
106     return in;
107 }
108 }
109 ostream& operator<<(ostream& out, BigNum& b) //重载输出运算符
110 {
111     int i;
112     cout << b.a[b.len - 1];
113     for(i = b.len - 2 ; i >= 0 ; i--)
114     {
115         cout.width(DLEN);
116         cout.fill('0');
117         cout << b.a[i];
118     }
119     return out;
120 }
121
122 BigNum BigNum::operator+(const BigNum & T) const //两个大数之间的相加运算
123 {
124     BigNum t(*this);

```

```

125     int i, big;           //位数
126     big = T.len > len ? T.len : len;
127     for(i = 0 ; i < big ; i++)
128     {
129         t.a[i] += T.a[i];
130         if(t.a[i] > MAXN)
131         {
132             t.a[i + 1]++;
133             t.a[i] -= MAXN + 1;
134         }
135     }
136     if(t.a[big] != 0)
137         t.len = big + 1;
138     else
139         t.len = big;
140     return t;
141 }
142 BigNum BigNum::operator-(const BigNum & T) const //两个大数之间的相减运算
143 {
144     int i, j, big;
145     bool flag;
146     BigNum t1, t2;
147     if(*this > T)
148     {
149         t1 = *this;
150         t2 = T;
151         flag = 0;
152     }
153     else
154     {
155         t1 = T;
156         t2 = *this;
157         flag = 1;
158     }
159     big = t1.len;
160     for(i = 0 ; i < big ; i++)
161     {
162         if(t1.a[i] < t2.a[i])
163         {
164             j = i + 1;
165             while(t1.a[j] == 0)
166                 j++;
167             t1.a[j--]--;
168             while(j > i)
169                 t1.a[j--] += MAXN;
170             t1.a[i] += MAXN + 1 - t2.a[i];
171         }
172         else
173             t1.a[i] -= t2.a[i];
174     }
175     t1.len = big;
176     while(t1.a[t1.len - 1] == 0 && t1.len > 1)
177     {
178         t1.len--;
179         big--;
180     }

```

```

181         if(flag)
182             t1.a[big-1]=0-t1.a[big-1];
183         return t1;
184     }
185
186     BigNum BigNum::operator*(const BigNum & T) const    //两个大数之间的相乘运算
187     {
188         BigNum ret;
189         int i,j,up;
190         int temp,temp1;
191         for(i = 0 ; i < len ; i++)
192         {
193             up = 0;
194             for(j = 0 ; j < T.len ; j++)
195             {
196                 temp = a[i] * T.a[j] + ret.a[i + j] + up;
197                 if(temp > MAXN)
198                 {
199                     temp1 = temp - temp / (MAXN + 1) * (MAXN + 1);
200                     up = temp / (MAXN + 1);
201                     ret.a[i + j] = temp1;
202                 }
203                 else
204                 {
205                     up = 0;
206                     ret.a[i + j] = temp;
207                 }
208             }
209             if(up != 0)
210                 ret.a[i + j] = up;
211         }
212         ret.len = i + j;
213         while(ret.a[ret.len - 1] == 0 && ret.len > 1)
214             ret.len--;
215         return ret;
216     }
217     BigNum BigNum::operator/(const int & b) const    //大数对一个整数进行相除运算
218     {
219         BigNum ret;
220         int i,down = 0;
221         for(i = len - 1 ; i >= 0 ; i--)
222         {
223             ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
224             down = a[i] + down * (MAXN + 1) - ret.a[i] * b;
225         }
226         ret.len = len;
227         while(ret.a[ret.len - 1] == 0 && ret.len > 1)
228             ret.len--;
229         return ret;
230     }
231     int BigNum::operator%(const int & b) const    //大数对一个 int 类型的变量进行取模运
232     ↪ 算
233     {
234         int i,d=0;
235         for (i = len-1; i>=0; i--)

```

```

236         d = ((d * (MAXN+1))% b + a[i])% b;
237     }
238     return d;
239 }
240 BigNum BigNum::operator^(const int & n) const    //大数的 n 次方运算
241 {
242     BigNum t,ret(1);
243     int i;
244     if(n<0)
245         exit(-1);
246     if(n==0)
247         return 1;
248     if(n==1)
249         return *this;
250     int m=n;
251     while(m>1)
252     {
253         t=*this;
254         for( i=1;i<=m;i++)
255         {
256             t=t*t;
257         }
258         m-=i;
259         ret=ret*t;
260         if(m==1)
261             ret=ret*(t);
262     }
263     return ret;
264 }
265 bool BigNum::operator>(const BigNum & T) const    //大数和另一个大数的大小比较
266 {
267     int ln;
268     if(len > T.len)
269         return true;
270     else if(len == T.len)
271     {
272         ln = len - 1;
273         while(a[ln] == T.a[ln] && ln >= 0)
274             ln--;
275         if(ln >= 0 && a[ln] > T.a[ln])
276             return true;
277         else
278             return false;
279     }
280     else
281         return false;
282 }
283 bool BigNum::operator >(const int & t) const    //大数和一个 int 类型的变量的大小比
↪ 较
284 {
285     BigNum b(t);
286     return *this>b;
287 }
288
289 void BigNum::print()    //输出大数
290 {

```

```

291     int i;
292     cout << a[len - 1];
293     for(i = len - 2 ; i >= 0 ; i--)
294     {
295         cout.width(DLEN);
296         cout.fill('0');
297         cout << a[i];
298     }
299     cout << endl;
300 }
301 int main(void)
302 {
303     int i,n;
304     BigNum x[101];    //定义大数的对象数组
305     x[0]=1;
306     for(i=1;i<101;i++)
307         x[i]=x[i-1]*(4*i-2)/(i+1);
308     while(scanf("%d",&n)==1 && n!=-1)
309     {
310         x[n].print();
311     }
312 }

```

---

#### 5.10.12 快速数论变换.cpp

```

1  const int mod = 998244353;
2  LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
3  const int g = 3;    //原根
4  LL quick_mod(LL a,LL b)
5  {
6      LL ans=1;
7      for(;b;b/=2)
8      {
9          if(b&1)
10             ans=ans*a%mod;
11             a=a*a%mod;
12     }
13     return ans;
14 }
15 int rev(int x,int r)    //蝴蝶操作
16 {
17     int ans=0;
18     for(int i=0; i<r; i++)
19     {
20         if(x&(1<<i))
21         {
22             ans+=1<<(r-i-1);
23         }
24     }
25     return ans;
26 }
27 void NTT(int n, LL A[],int on) // 长度为 N (2 的次数)
28 {
29     int r=0;
30     for(;; r++)

```

```

31     {
32         if((1<<r)==n)
33             break;
34     }
35     for(int i=0; i<n; i++)
36     {
37         int tmp=rev(i,r);
38         if(i<tmp)
39             swap(A[i],A[tmp]);
40     }
41     for(int s=1; s<=r; s++)
42     {
43         int m=1<<s;
44         LL wn=quick_mod(g,(mod-1)/m);
45         for(int k=0; k<n; k+=m)
46         {
47             LL w=1;
48             for(int j=0; j<m/2; j++)
49             {
50                 LL t,u;
51                 t=w*(A[k+j+m/2]%mod)%mod;
52                 u=A[k+j]%mod;
53                 A[k+j]=(u+t)%mod;
54                 A[k+j+m/2]=((u-t)%mod+mod)%mod;
55                 w=w*wn%mod;
56             }
57         }
58     }
59     if(on== -1)
60     {
61         for(int i=1; i<n/2; i++)
62             swap(A[i],A[n-i]);
63         LL inv=quick_mod(n,mod-2);
64         for(int i=0; i<n; i++)
65             A[i]=A[i]%mod*inv%mod;
66     }
67 }
68

```

---

### 5.10.13 欧拉函数打表.cpp

```

1  求任意一个数的欧拉函数值
2
3  ```.cpp
4  long long Euler(long long num)
5  {
6      long long temp=num;
7      for(long long i=2; i*i<=num; i++)
8          if(num%i==0)
9          {
10             while(num%i==0)
11                 num=num/i;
12             temp=temp/i*(i-1);
13         }
14     if(num!=1)

```

```

15     temp=temp/num*(num-1);
16     return temp;
17 }
18
19 ```
20
21 ##### 欧拉函数打表
22 O(nlog(n))
23 ```cpp
24
25 const int maxn = 1e6+100;
26 int phi[maxn],Prime[maxn];
27
28 void init2(int n){
29     for(int i = 1;i <= n; ++i) phi[i] = i;
30     for(int i = 2;i <= n; ++i){
31         if(i == phi[i]){
32             for(int j = i; j <= n; j += i) phi[j] = phi[j]/i*(i-1);
33         }
34     }
35 }
36 ```
37 线性筛 O(n)
38 ```cpp
39 const int maxn = 1e6+100;
40 bool check[maxn];
41 int phi[maxn],Prime[maxn];
42 void init(int MAXN){
43     int N = maxn-1;
44     memset(check,false,sizeof(check));
45     phi[1] = 1;
46     int tot = 0;
47     for(int i = 2;i <= N; ++i){
48         if(!check[i]){
49             Prime[tot++] = i;
50             phi[i] = i-1;
51         }
52         for(int j = 0;j < tot; ++j){
53             if(i*Prime[j] > N) break;
54             check[i*Prime[j]] = true;
55             if(i%Prime[j] == 0){
56                 phi[i*Prime[j]] = phi[i]*Prime[j];
57                 break;
58             }
59             else{
60                 phi[i*Prime[j]] = phi[i]*(Prime[j]-1);
61             }
62         }
63     }
64 }
65 ```
66

```



#### 5.10.14 欧拉筛和埃氏筛.cpp

---

```
1 void Era_s(void){
2     check[1] = 1;
3     tot = 1;
4     for(int i = 2;i < maxn; ++i){
5         if(!check[i]){
6             Prime[tot++] = i;
7             for(int j = i+i;j < maxn; ++j) check[j] = 1;
8         }
9     }
10 }
11 void Euler_s(void){
12     check[1] = 1;
13     tot = 1;
14     int n = 1e6;
15     for(int i = 2;i <= n; ++i){
16         if(!check[i]) Prime[tot++] = i;
17         for(int j = 1;j < tot; ++j){
18             if(i*Prime[j] > n) break;
19             check[i*Prime[j]] = 1;
20             if(i % Prime[j] == 0) break;
21         }
22     }
23 }
```

---

#### 5.10.15 素性检测.cpp

---

```
1 #include<bits/stdc++.h>
2
3 using namespace std;
4 //typedef long long LL;
5 const int LEN = 1e6+1;
6 bool vis[LEN];
7 //int prime[LEN];
8 int Prime[LEN];
9 int cnt = 1;
10 typedef unsigned long long LL;
11
12 LL modular_multi(LL x,LL y,LL mo) {
13     LL t;
14     x%=mo;
15     for(t=0;y;x=(x<<1)%mo,y>>=1)
16         if (y&1)
17             t=(t+x)%mo;
18     return t;
19 }
20
21 LL modular_exp(LL num,LL t,LL mo) {
22     LL ret=1,temp=num%mo;
23     for(;t;t>>=1,temp=modular_multi(temp,temp,mo))
24         if (t&1)
25             ret=modular_multi(ret,temp,mo);
26     return ret;
27 }
```

```

27 }
28
29 bool miller_rabin(LL n) {
30     if (n==2 || n==7 || n==61)
31         return true;
32     if (n==1 || (n&1)==0)
33         return false;
34     int t=0,num[3]={2,7,61}; //2,7,61 对 unsigned int 内的所有数够用了, 最小不能判
        ↪ 断的数为 4 759 123 141; 用 2,3,7,61 在 10^16 内唯一不能判断的数是 46 856
        ↪ 248 225 981
35     LL a,x,y,u=n-1;
36     while((u&1)==0)
37         t++,u>>=1;
38     for(int i=0;i<3;i++) {
39         a=num[i];
40         x=modular_exp(a,u,n);
41         for(int j=0;j<t;j++) {
42             y=modular_multi(x,x,n);
43             if (y==1&&x!=1&&x!=n-1)
44                 return false;
45             //其中用到定理, 如果对模 n 存在 1 的非平凡平方根, 则 n 是合数。
46             //如果一个数 x 满足方程  $x^2 \equiv 1 \pmod{n}$ , 但 x 不等于对模 n 来说 1 的两个‘平
                ↪ 凡’平方根: 1 或 -1, 则 x 是对模 n 来说 1 的非平凡平方根
47                 x=y;
48         }
49         if (x!=1) //根据费马小定理, 若 n 是素数, 有  $a^{(n-1)} \equiv 1 \pmod{n}$ . 因此 n
            ↪ 不可能是素数
50             return false;
51     }
52     return true;
53 }
54 void init(void)
55 {
56     int n = LEN -1;
57     for(int i = 2; i <= n; ++i)
58     {
59         if(!vis[i])
60         {
61             Prime[cnt++] = i;
62             for(LL j = (LL)i * i; j <= n; j += i)
63                 vis[j] = 1;
64         }
65     }
66 }
67 bool isPrime(LL n)
68 {
69     if(n < 1e6)
70     {
71         for(LL i = 1; i < cnt&&Prime[i] < n; ++i)
72         {
73             if(n % Prime[i] == 0)
74                 return false;
75         }
76         return true;
77     }
78     else

```

```

79         return miller_rabin(n);
80     }
81
82     int main(void)
83     {
84         init();
85
86         int T;
87         cin>>T;
88         while(T-->0)
89         {
90             LL n;
91             cin>>n;
92             if(isPrime(n))
93                 cout<<"Yes"<<endl;
94             else
95                 cout<<"No"<<endl;
96         }
97
98         return 0;
99     }

```

---

#### 5.10.16 素数筛.cpp

```

1  ~~~
2  Eratosthenes 筛法 (埃拉托斯特尼筛法)
3  const int maxn = 1e6+10;
4  bool check[maxn];
5  int Prime[maxn];
6  int tot = 1;
7  void Eratosthenes(void){
8      const int n = maxn - 1;
9      memset(check,0,sizeof(check));
10     for(int i = 2;i < n; ++i){
11         if(!check[i]){
12             Prime[tot++] = i;
13             for(int j = i+i;j < n;j += i) check[j] = 1;
14         }
15     }
16 }
17 ~~~
18 欧拉筛
19 ~~~
20 const int maxn = 1e6+10;
21 bool check[maxn];
22 int Prime[maxn];
23 int tot = 1;
24 void Euler_shai(void){
25     int n = maxn-1;
26     memset(check,0,sizeof(check));
27     for(int i = 2;i <= n; ++i){
28         if(!check[i]){
29             Prime[tot++] = i;
30         }

```

```

31         for(int j = 1;j < tot; ++j){
32             if(i*Prime[j] > n) break;
33             check[i*Prime[j]] =1 ;
34             if(i % Prime[j]==0) break;
35         }
36     }
37 }
38
39 ```

```

---

#### 5.10.17 逆元打表.cpp

```

1  int inv[10000];
2  int p;
3  cin>>p;
4  inv[1] = 1;
5  for(int i = 2;i < p; ++i)
6  {
7      inv[i] = (p - p/i*inv[p%i]%p)%p;
8  }
9  for(int i = 1;i < p; ++i)
10     cout<<inv[i]<<" ";
11     cout<<endl;
12     for(int i = 1;i < p; ++i)
13         cout<<i * inv[i] % p<<" ";

```

---

#### 5.11 矩阵快速幂.cpp

```

1  // 注意修改 maxn 的值，要不然容易 T
2  // 注意 maxn 值过大，栈可能会不够
3  const int maxn = 100;
4  int n;
5  struct Matrix{
6      int n,m;
7      Matrix(int nn = 1,int mm = 1):n(nn),m(mm){ memset(a,0,sizeof(a));};
8      long long a[maxn][maxn];
9  };
10 // void print(const Matrix &a)
11 // {
12 //     for(int i = 1;i <= a.n; ++i,cout<<endl)
13 //         for(int j = 1;j <= a.m; ++j)
14 //             cout<<a.a[i][j]<<" ";
15 // }
16 Matrix operator*(Matrix a,Matrix b)
17 {
18     Matrix c(a.n,b.m);
19     for(int i = 1;i <= a.n; ++i)
20     {
21         for(int j = 1;j <= b.m; ++j)
22         {
23             for(int k = 1;k <= a.m; ++k)
24             {
25                 c.a[i][j] += a.a[i][k] * b.a[k][j];
26                 c.a[i][j] %= mod;

```

```

27         }
28     }
29 }
30 //     print(c);
31     return c;
32 }

```

---

## 5.12 自适应辛普森积分.cpp

```

1 double F(double x)
2 {
3     //Simpson 公式用到的函数
4 }
5 double simpson(double a, double b)//三点 Simpson 法, 这里要求 F 是一个全局函数
6 {
7     double c = a + (b - a) / 2;
8     return (F(a) + 4 * F(c) + F(b))*(b - a) / 6;
9 }
10 double asr(double a, double b, double eps, double A)//自适应 Simpson 公式 (递归过
    ↪ 程)。已知整个区间 [a,b] 上的三点 Simpson 值 A
11 {
12     double c = a + (b - a) / 2;
13     double L = simpson(a, c), R = simpson(c, b);
14     if (fabs(L + R - A) <= 15 * eps)return L + R + (L + R - A) / 15.0;
15     return asr(a, c, eps / 2, L) + asr(c, b, eps / 2, R);
16 }
17 double asr(double a, double b, double eps)//自适应 Simpson 公式 (主过程)
18 {
19     return asr(a, b, eps, simpson(a, b));
20 }

```

---

## 6 数据结构

### 6.1 CDQ 分治

#### 6.1.1 CDQ 分治.cpp

```

1 // CDQ 解决 单点修改, 区间查询
2 /*
3
4 */
5 const int maxn = 5e6+100;
6
7 struct node{
8     int type,id;
9     LL val;
10    bool operator <(const node &a) const
11    {
12        if(a.id != id) return id < a.id;
13        return type < a.type;
14    }
15 };
16
17 node A[maxn],B[maxn];

```

```

18 LL ans[maxn];
19
20
21
22 void CDQ(int L,int R){
23     // cout<<L<<" "<<R<<endl;
24     if(L == R) return ;
25     int M = (L+R)>>1;
26     CDQ(L,M),CDQ(M+1,R);
27     int t1 = L,t2 = M+1;
28     LL sum = 0;
29     for(int i = L;i <= R; ++i){
30         if((t1 <= M && A[t1] < A[t2]) || t2 > R){
31             if(A[t1].type == 1) sum += A[t1].val;
32             B[i] = A[t1++];
33         }
34         else{
35             if(A[t2].type == 2) ans[A[t2].val] -= sum;
36             else if(A[t2].type == 3) ans[A[t2].val] += sum;
37             B[i] = A[t2++];
38         }
39     }
40 }
41
42 for(int i = L;i <= R; ++i) A[i] = B[i];
43 }
44 int main(void)
45 {
46     int n,q;
47     cin>>n>>q;
48     int tot = 0;
49     for(int i = 1;i <= n; ++i){
50         scanf("%lld",&A[i].val);
51         A[i].type = 1;
52         A[i].id = i;
53     }
54     tot = n;
55     int sz = 0;
56     rep(i,0,q){
57         int type;
58         scanf("%d",&type);
59         if(type ==1){
60             A[++tot].type = 1;
61             scanf("%d%lld",&A[tot].id,&A[tot].val);
62         }
63         else{
64             int l,r;
65             scanf("%d%d",&l,&r);
66             A[++tot].type = 2,A[tot].id = l-1,A[tot].val = ++sz;
67             A[++tot].type = 3,A[tot].id = r, A[tot].val = sz;
68         }
69     }
70     CDQ(1,tot);
71     rep(i,1,sz+1){
72         printf("%lld\n",ans[i]);
73     }

```

```

74
75     return 0;
76 }

```

---

### 6.1.2 CDQ 求动态逆序数.cpp

---

```

1  #include <bits/stdc++.h>
2  #define mem(ar,num) memset(ar,num,sizeof(ar))
3  #define me(ar) memset(ar,0,sizeof(ar))
4  #define lowbit(x) (x&(-x))
5  #define Pb push_back
6  #define FI first
7  #define SE second
8  #define rep(i,a,n) for (int i=a;i<n;i++)
9  #define per(i,a,n) for (int i=n-1;i>=a;i--)
10 #define IOS ios::sync_with_stdio(false)
11 #define DEBUG cout<<endl<<"DEBUG"<<endl;
12 using namespace std;
13 typedef long long LL;
14 typedef unsigned long long ULL;
15 const int prime = 999983;
16 const int INF = 0x7FFFFFFF;
17 const LL INFF =0x7FFFFFFFFFFFFFFF;
18 const double pi = acos(-1.0);
19 const double inf = 1e18;
20 const double eps = 1e-6;
21 const LL mod = 1e9 + 7;
22 LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
23 LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
24 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
25 typedef pair<int,int> P;
26
27 const int maxn = 2e5+100;
28 int n,m;
29 int a[maxn];
30 int del[maxn];
31 int id[maxn];
32 bool del2[maxn];
33 struct nd{
34     int id,val;
35 };
36 bool operator <(const nd &a,const nd &b){
37     return a.id < b.id;
38 }
39 bool operator >(const nd &a,const nd &b){
40     return !(a < b);
41 }
42 nd A[maxn],B[maxn];
43 LL ans[maxn];
44 LL tree[maxn];
45 void Add(int x,int y){
46     while(x <= n)
47     {
48         tree[x] += y;
49         x += lowbit(x);

```

```

50     }
51 }
52 LL Sum(int x){
53     LL sum = 0;
54     while(x > 0){
55         sum += tree[x];
56         x -= lowbit(x);
57     }
58     return sum;
59 }
60 void CDQ(int L,int R){
61     // DEBUG;
62     if(L == R) return ;
63     int M = (L+R)>>1;
64     CDQ(L,M),CDQ(M+1,R);
65     int t1 = L,t2 = M+1;
66     for(int i = L;i <= R; ++i){
67         if((t1 <= M&&A[t1] < A[t2])||t2 > R){
68             Add(A[t1].val,1);
69             B[i] = A[t1++];
70         }
71         else{
72             ans[id[A[t2].val]] += Sum(n)-Sum(A[t2].val);
73             B[i] = A[t2++];
74         }
75     }
76     for(int i = L;i <= M; ++i)
77         Add(A[i].val,-1);
78     t1 = M,t2 = R;
79     for(int i = R;i >= L; --i){
80         if((t1 >= L&&A[t1] > A[t2])||t2 <= M){
81             Add(A[t1].val,1);
82             t1--;
83             // B[i] = A[t2++];?
84         }
85         else{
86             ans[id[A[t2].val]] += Sum(A[t2].val);
87             t2--;
88         }
89     }
90     for(int i = L;i <= M; ++i)
91         Add(A[i].val,-1);
92     for(int i = L;i <= R; ++i)
93         A[i] = B[i];
94 }
95 LL ans2[maxn];
96 int sign[maxn];
97 int main(void)
98 {
99
100     // cout<<maxn*maxn/2<<endl;
101     // freopen("input.txt","r",stdin);
102     // freopen("output.txt","w",stdout);
103     scanf("%d%d",&n,&m);
104     // int s;
105     for(int i = 1;i <= n; ++i){

```



```

106         scanf("%d",&a[i]);
107         id[a[i]] = i;
108     }
109     for(int i = 1;i <= m;++i){
110         scanf("%d",&del[i]);
111         del2[id[del[i]]] = 1;
112     }
113     // DEBUG;
114     int cnt = 0;
115     for(int i = 1;i <= n; ++i){
116         if(!del2[i])
117             {
118                 A[++cnt].id = i,A[cnt].val = a[i];
119                 sign[cnt] = a[i];
120             }
121     }
122     for(int i = m; i >= 1; --i){
123         // A[++cnt].op = 1,A[cnt].id = id[del[i]],A[cnt].val = a[i];
124         A[++cnt].id = id[del[i]],A[cnt].val = del[i];
125         sign[cnt] = del[i];
126     }
127     CDQ(1,n);
128     LL sum = 0;
129     for(int i = 1;i <= n; ++i){
130         sum += ans[id[sign[i]]];
131         ans2[i] = sum;
132     }
133     for(int i = n;i >= n-m+1; --i){
134         printf("%lld\n",ans2[i]);
135     }
136
137     return 0;
138 }

```

---

### 6.1.3 陌上花开 CDQ 三位偏序.cpp

```

1  #include <cstdio>
2  #include <algorithm>
3  #include <iostream>
4  using namespace std;
5
6  const int N = 200005;
7  int w,q,c[500005];
8  struct nd {int op,x1,y1,x2,y2,z,id,ans;}a[N],b[N];
9  bool cmp(const nd &a, const nd &b) {return a.x1<b.x1 || (a.x1==b.x1&&a.op<b.op);}
10
11 int lowbit(int x) {return x & -x;}
12 void Add(int x, int y) {while(x <= w) c[x] += y, x += lowbit(x);}
13 int Sum(int x) {
14     int r = 0;
15     while(x) r += c[x], x -= lowbit(x);
16     return r;
17 }
18 struct node{
19     int x,y,z,id,num;

```

```

20 }Node[N],Node2[N];
21 bool operator<(const node &a,const node &b){
22     return a.z < b.z||(a.z == b.z &&a.y < b.y)||((a.z == b.z && a.y == b.y&&a.x <
    ↪ b.x);
23 }
24 bool operator==(const node &a,const node&b){
25     return a.x == b.x && a.y == b.y&&a.z == b.z;
26 }
27 void CDQ(int l, int r) {
28     if(l == r) return;
29
30     // printf("%d %d\n",l,r);
31     int m = (l+r) >> 1, cnt = 0;
32     CDQ(l,m),CDQ(m+1,r);
33     for(int i = l; i <= m; i++) if(a[i].op == 1) b[cnt++] = a[i];
34     for(int i = m+1; i <= r; i++) if(a[i].op == 2) {
35         b[cnt++] = a[i];
36         b[cnt++] = a[i];
37         b[cnt-2].x1--, b[cnt-1].x1=a[i].x2,
38         b[cnt-1].op = 3;
39     }
40     sort(b, b+cnt, cmp);
41     for(int i = 0; i < cnt; i++)
42         if(b[i].op == 1) Add(b[i].y1, b[i].z);
43     else if(b[i].op == 2) a[b[i].id].ans -= Sum(b[i].y2)-Sum(b[i].y1-1);
44     else a[b[i].id].ans += Sum(b[i].y2)-Sum(b[i].y1-1);
45     for(int i = 0; i < cnt; i++)
46         if(b[i].op == 1) Add(b[i].y1, -b[i].z);
47 }
48 int ans[N];
49 int main() {
50     // freopen("locust.in","r",stdin);
51     // freopen("locust.out","w",stdout);
52     scanf("%d%d",&q,&w);
53     for(int i = 1;i <= q; ++i)
54         scanf("%d%d%d",&Node2[i].x,&Node2[i].y,&Node2[i].z),Node2[i].id = i;
55     // DEBUG;
56     // cout<<"1"<<endl;
57     int qq = q;
58     sort(Node2+1,Node2+q+1);
59     int cnt = 1;
60     Node[cnt] = Node2[1];
61     Node[cnt].num = 1;
62     for(int i = 2;i <= q; ++i){
63         if(Node2[i] == Node2[i-1])
64             Node[cnt].num++;
65         else
66             Node[++cnt] = Node2[i],Node[cnt].num = 1;
67
68     }
69     q = cnt;
70
71     for(int i = 1; i <= q; i++) {
72         Node[i].id = i;
73         a[2*i-1].op = 2; a[2*i-1].x1 = 1,a[2*i-1].y1 = 1,a[2*i-1].x2 =
    ↪ Node[i].x,a[2*i-1].y2 = Node[i].y;

```

```

74     a[2*i].op = 1;a[2*i].x1 = Node[i].x,a[2*i].y1 = Node[i].y,a[2*i].z =
       ↪ Node[i].num;
75
76     a[2*i-1].id = a[2*i].id =Node[i].id;
77 }
78 // puts("DEBUG");
79 CDQ(1, 2*q);
80
81 for(int i = 1; i <= q; i++) ans[a[i].ans+Node[i].num-1] += Node[i].num;
82 // cout<<endl;
83 // for(int i = 1;i <= q; ++i) cout<<a[i].ans<<endl;
84 // cout<<endl;
85 for(int i = 0; i < qq; ++i) printf("%d\n",ans[i]);
86 return 0;
87 }

```

---

## 6.2 fenkuai

### 6.2.1 区间修改区间查询.cpp

---

```

1  const int maxn = 100010;
2  LL a[maxn],add[maxn],sum[maxn];
3  int pos[maxn],R[maxn],L[maxn];
4  int n,m,t;
5  void change(int l,int r,LL d){
6      int p = pos[l],q = pos[r];
7      if(p == q){
8          for(int i = l;i <= r; ++i) a[i] += d;
9          sum[p] += (r-l+1)*d;
10     }
11     else{
12         for(int i = p+1;i <= q-1; ++i) add[i] += d;
13         for(int i = l;i <= R[p];++i)
14             a[i] += d;
15         sum[p] += (R[p]-l+1)*d;
16         for(int i = L[q];i <= r; ++i)
17             a[i] += d;
18         sum[q] += (r-L[q]+1)*d;
19     }
20 }
21 LL ask(int l,int r){
22     LL ans = 0;
23     int p = pos[l],q = pos[r];
24     if(p == q){
25         for(int i = l;i <= r; ++i)
26             ans += a[i];
27         ans += (r-l+1)*add[p];
28     }
29     else{
30         for(int i = p+1;i <= q-1; ++i)
31             ans += sum[i]+add[i]*(R[i]-L[i]+1);
32         for(int i = l;i <= R[p]; ++i)
33             ans += a[i];
34         ans += add[p]*(R[p]-l+1);
35         for(int i = L[q];i <= r; ++i)

```

```

36         ans += a[i];
37         ans += add[q]*(r-L[q]+1);
38     }
39     return ans;
40 }
41 int main(void){
42
43     cin>>n>>m;
44     for(int i = 1;i <= n; ++i) scanf("%lld",&a[i]);
45     LL t = sqrt(n);
46     for(int i = 1;i <= t; ++i){
47         L[i] = (i-1)*sqrt(n)+1;
48         R[i] = i*sqrt(n);
49     }
50     if(R[t] < n) t++,L[t] = R[t-1]+1,R[t] = n;
51     // cout<<t<<endl;
52     for(int i = 1;i <= t; ++i){
53         for(int j = L[i];j <= R[i]; ++j){
54             pos[j] = i;
55             sum[i] += a[j];
56         }
57     }
58     while(m--){
59         char op[3];
60         int l,r,x;
61         scanf("%s%d%d",op,&l,&r);
62         if(op[0] == 'C'){
63             scanf("%d",&x);
64             change(l,r,x);
65         }
66         else
67             printf("%lld\n",ask(l,r));
68     }
69     return 0;
70 }

```

---

### 6.2.2 区间数的平方.cpp

---

```

1  const int maxn = 50000+10;
2  int n,m,k;
3  int pos[maxn];
4  int a[maxn];
5  int num[maxn];
6  LL Ans[maxn];
7  int L[maxn],R[maxn];
8  struct Query{
9      int l,r,id;
10 };
11 Query q[maxn];
12 bool cmp1 (const Query &a,const Query &b){
13     return a.l < b.l || (a.l == b.l && a.r < b.r);
14 }
15 bool cmp2(const Query &a,const Query &b){
16     return a.r < b.r;
17 }

```

```

18
19 void work(int x, LL &ans, int d){
20     ans -= 1ll*num[x]*num[x];
21     num[x] += d;
22     ans += 1ll*num[x]*num[x];
23 }
24 int main(){
25     cin>>n>>m>>k;
26     rep(i,1,n+1) scanf("%d",&a[i]);
27     rep(i,1,m+1){
28         scanf("%d%d",&q[i].l,&q[i].r);
29         q[i].id = i;
30     }
31     int t = sqrt(m);
32     for(int i = 1; i <= t; ++i){
33         L[i] = (i-1)*t;
34         R[i] = i*t;
35     }
36     if(R[t] < m){
37         L[t+1] = R[t]+1;
38         R[++t] = m;
39     }
40     sort(q+1,q+m+1,cmp1);
41     for(int i = 1; i <= t; ++i){
42         sort(q+L[i],q+R[i]+1,cmp2);
43         LL ans = 0;
44         me(num);
45         int l = q[L[i]].l, r = q[L[i]].r;
46         rep(i,l,r+1) work(a[i],ans,1);
47         Ans[q[L[i]].id] = ans;
48         for(int j = L[i]+1; j <= R[i]; ++j){
49             // l = L[j].l, r = L[j].r;
50             while(l < q[j].l) work(a[l++],ans,-1);
51             while(l > q[j].l) work(a[--l],ans,1);
52             while(r < q[j].r) work(a[++r],ans,1);
53             while(r > q[j].r) work(a[r--],ans,-1);
54             Ans[q[j].id] = ans;
55         }
56     }
57     rep(i,1,m+1)
58         printf("%lld\n",Ans[i]);
59     return 0;
60 }

```

---

### 6.2.3 在线查询区间众数.cpp

```

1 const int N = 40006, T = 37;
2 int a[N], b[N], L[N], R[N], pos[N];
3 int c[T][T][N], f[T][T][2], now[2];
4 inline void work(int x, int y, int num){
5     ++c[x][y][num];
6     if(c[x][y][num] > now[0] || (c[x][y][num] == now[0] && num < now[1])){
7         now[0] = c[x][y][num];
8         now[1] = num;
9     }

```

```

10 }
11 int ask(int l,int r){
12     int p = pos[l],q = pos[r];
13     int x = 0,y = 0;
14     if(p+1 <= q-1){
15         x = p+1;
16         y = q-1;
17     }
18     memcpy(now,f[x][y],sizeof(now));
19     if(p == q){
20         rep(i,l,r+1) work(x,y,a[i]);
21         rep(i,l,r+1) --c[x][y][a[i]];
22     }
23     else{
24         rep(i,l,R[p]+1) work(x,y,a[i]);
25         rep(i,L[q],r+1) work(x,y,a[i]);
26         rep(i,l,R[p]+1) --c[x][y][a[i]];
27         rep(i,L[q],r+1) --c[x][y][a[i]];
28     }
29     return b[now[1]];
30 }
31 int main(void){
32     // freopen("input.txt","r",stdin);
33
34     // freopen("output1.txt","w+",stdout);
35     int n,m;cin>>n>>m;
36     rep(i,1,n+1) scanf("%d",&a[i]);
37     memcpy(b,a,sizeof(a));
38     sort(b+1,b+n+1);
39     int tot = unique(b+1,b+n+1)-(b+1);
40     rep(i,1,n+1) a[i] = lower_bound(b+1,b+tot+1,a[i])-b;
41     int t = pow((double)n,(double)1/3);
42     int len = t?n/t:n;
43     for(int i = 1;i <= t; ++i){
44         L[i] = (i-1)*len+1;
45         R[i] = i*len;
46     }
47     if(R[t] < n){
48         L[t+1] = R[t]+1;
49         R[++t] = n;
50     }
51     rep(i,1,t+1)
52         rep(j,L[i],R[i]+1)
53             pos[j] = i;
54
55     me(c),me(f);
56     rep(i,1,t+1){
57         rep(j,i,t+1){
58             rep(k,L[i],R[j]+1)
59                 ++c[i][j][a[k]];
60             rep(k,1,tot+1)
61                 if(c[i][j][k] > f[i][j][0]){
62                     f[i][j][0] = c[i][j][k];
63                     f[i][j][1] = k;
64                 }
65         }

```

```

66     }
67     int x = 0;
68     while(m--){
69         int l,r;scanf("%d%d",&l,&r);
70         l = (l+x-1)%n+1;
71         r = (r+x-1)%n+1;
72         if(l > r) swap(l,r);
73         printf("%d\n",x = ask(l,r));
74     }
75
76
77     return 0;
78 }

```

---

## 6.3 pbds

### 6.3.1 1 可合并优先队列.cpp

---

```

1 // pbds zoj2334 合并 logn
2
3 #include<bits/stdc++.h>
4 #include<ext/pb_ds/priority_queue.hpp>
5
6 using namespace std;
7 using namespace __gnu_pbds;
8 typedef pair<int,int> P;
9 typedef __gnu_pbds::priority_queue<int> Heap;
10
11 const int maxn = 1e5+10;
12 Heap heap[maxn];
13
14 int F[maxn];
15
16 int Find(int x){
17     return x == F[x]?x:F[x] = Find(F[x]);
18 }
19 int main(void){
20     int N,M;
21     while(cin>>N){
22         for(int i = 1;i <= N; ++i){
23             int a;
24             scanf("%d",&a);
25             heap[i].clear();
26             heap[i].push(a);
27             F[i] = i;
28         }
29         cin>>M;
30         int a,b;
31         for(int i = 1;i <= M; ++i){
32             scanf("%d%d",&a,&b);
33             int fa = Find(a);
34             int fb = Find(b);
35             if(fa == fb){
36                 puts("-1");
37             }

```

```

38         continue;
39     }
40     // cout<<fa<<" "<<fb<<endl;
41     F[fb] = fa;
42     int t;
43     t = heap[fa].top(), heap[fa].pop(), t/=2, heap[fa].push(t);
44     t = heap[fb].top(), heap[fb].pop(), t/=2, heap[fb].push(t);
45     heap[fa].join(heap[fb]);
46     printf("%d\n", heap[fa].top());
47 }
48 }
49 return 0;
50 }

```

---

## 6.4 二叉搜索树

### 6.4.1 1 二叉树.cpp

---

```

1 // 通过中序遍历和后序遍历建立二叉树
2 //https://vjudge.net/problem/UVA-548
3
4
5 #include<bits/stdc++.h>
6
7 using namespace std;
8 const int maxn = 1e5+10;
9 const int INF = 1e8;
10 int in_order[maxn], post_order[maxn], l[maxn], r[maxn];
11 int n;
12 int read_order(int *a)
13 {
14     string s;
15     if(!getline(cin,s)) return false;
16     stringstream ss(s);
17     n = 0;
18     int v;
19     while(ss >> v)
20         a[n++] = v;
21     return n > 0;
22 }
23 int build_tree(int L1, int R1, int L2, int R2)
24 {
25     if(L1 > R1)
26         return 0;
27     int root = post_order[R2];
28     int p = L1;
29     while(in_order[p] != root)
30         p++;
31     int cnt = p-L1;
32     l[root] = build_tree(L1, p-1, L2, L2+cnt-1);
33     r[root] = build_tree(p+1, R1, L2+cnt, R2-1);
34     return root;
35 }
36 int best, bestsum;
37 void dfs(int a, int b)

```



```

38 {
39     if(!l[a] && !r[a])
40     {
41         b += a;
42         if(bestsum > b || (bestsum == b && best > a))
43         {
44             best = a;
45             bestsum = b;
46         }
47     }
48     if(l[a]) dfs(l[a], b+a);
49     if(r[a]) dfs(r[a], b+a);
50 }
51
52
53 int main(void)
54 {
55     while(read_order(in_order))
56     {
57         read_order(post_order);
58         build_tree(0, n-1, 0, n-1);
59         // cout<<0<<endl;
60         bestsum = INF;
61         dfs(post_order[n-1], 0);
62         cout<<best<<endl;
63     }
64
65     return 0;
66 }

```

---

#### 6.4.2 2 treap.cpp

---

```

1 // UVA LA 5031
2 /*
3 给定 n 个节点 m 条边的无向图，每个节点都有一个整数权值。
4 D X 删除 ID 为 x 的边
5 Q X K 计算与节点 x 连通的节点中权值第 k 大的数
6 C X K 把节点 x 的权值改为 v
7
8
9 */
10
11
12 #include <bits/stdc++.h>
13 #define mem(ar,num) memset(ar,num,sizeof(ar))
14 #define me(ar) memset(ar,0,sizeof(ar))
15 #define lowbit(x) (x&(-x))
16 #define Pb push_back
17 #define FI first
18 #define SE second
19 #define rep(i,a,n) for (int i=a;i<n;i++)
20 #define per(i,a,n) for (int i=n-1;i>=a;i--)
21 #define IOS ios::sync_with_stdio(false)
22 #define DEBUG cout<<endl<<"DEBUG"<<endl;
23 using namespace std;

```

```

24 typedef long long LL;
25 typedef unsigned long long ULL;
26 const int prime = 999983;
27 const int INF = 0x7FFFFFFF;
28 const LL INFF = 0x7FFFFFFFFFFFFFFF;
29 const double pi = acos(-1.0);
30 const double inf = 1e18;
31 const double eps = 1e-6;
32 const LL mod = 1e9 + 7;
33 LL qpow(LL a, LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
34 LL gcd(LL a, LL b) {return b?gcd(b,a%b):a;}
35 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
36 typedef pair<int,int> P;
37 struct Node{
38     Node *ch[2]; // 左右子树
39     int r; // 随机优先值
40     int v; // 值
41     int s; // 节点总数
42
43     Node(int v):v(v){ch[0] = ch[1] = NULL; r = rand(); s = 1;}
44     int cmp(int x) {
45         if(x==v) return -1;
46         return x < v?0:1;
47     }
48
49     void maintain(){
50         s = 1;
51         if(ch[0] != NULL) s += ch[0]->s;
52         if(ch[1] != NULL) s += ch[1]->s;
53     }
54 };
55
56 void rotate(Node * &o, int d){
57     Node *k = o->ch[d^1]; o->ch[d^1] = k->ch[d]; k->ch[d] = o;
58     o->maintain(); k->maintain(); o = k;
59 }
60
61
62 void insert(Node * &o, int x){
63     if(o == NULL) o = new Node(x);
64     else{
65         int d = (x < o->v?0:1);
66         insert(o->ch[d], x);
67         if(o->ch[d]->r > o->r) rotate(o, d^1);
68     }
69     o->maintain();
70 }
71
72 void remove(Node * &o, int x){
73     int d = o->cmp(x);
74     // int ret = 0;
75     if(d == -1){
76         Node *u = o;
77         if(o->ch[0] != NULL && o->ch[1] != NULL){
78             int d2 = (o->ch[0]->r > o->ch[1]->r?1:0);
79             rotate(o, d2); remove(o->ch[d2], x);
80         }
81     }
82 }

```

```

80         else{
81             if(o->ch[0] == NULL) o = o->ch[1];
82             else o = o->ch[0];
83             delete u;
84         }
85     } else
86         remove(o->ch[d],x);
87     if(o != NULL) o->maintain();
88 }
89 const int maxc = 5e5+10;
90 struct Command{
91     char type;
92     int x,p;
93 } commands[maxc];
94
95 const int maxn = 2e4+10;
96 const int maxm = 6e4+10;
97 int n,m,weight[maxn],from[maxm],to[maxm],removed[maxm];
98 // 并查集相关
99 int pa[maxn];
100 int findset(int x){ return pa[x] != x?pa[x] = findset(pa[x]) : x;}
101 // 名次数相关
102 Node *root[maxn]; // Treap;
103 int kth(Node *o,int k){
104     if(o == NULL || k <= 0 || k > o->s) return 0;
105     int s = (o->ch[1] == NULL?0:o->ch[1]->s);
106     if(k == s+1) return o->v;
107     else if(k <= s) return kth(o->ch[1],k);
108     else return kth(o->ch[0],k-s-1);
109 }
110 void mergeto(Node* &src,Node * &dest){
111     if(src->ch[0] != NULL) mergeto(src->ch[0],dest);
112     if(src->ch[1] != NULL) mergeto(src->ch[1],dest);
113     insert(dest,src->v);
114     delete src;
115     src = NULL;
116 }
117 void removetree(Node *&x){
118     if(x->ch[0] != NULL) removetree(x->ch[0]);
119     if(x->ch[1] != NULL) removetree(x->ch[1]);
120     delete x;
121     x = NULL;
122 }
123
124 void add_edge(int x){
125     int u = findset(from[x]), v = findset(to[x]);
126     if(u != v){
127         if(root[u]->s < root[v]->s){ pa[u] = v;
128             ⇨ mergeto(root[u],root[v]);}
129         else {pa[v] = u; mergeto(root[v],root[u]);}
130     }
131 }
132
133 int query_cnt;
134 long long query_tot;

```

```

135 void query(int x,int k){
136     query_cnt++;
137     query_tot += kth(root[findset(x)],k);
138
139 }
140
141 void change_weight(int x,int v){
142     int u = findset(x);
143     remove(root[u],weight[x]);
144     insert(root[u],v);
145     weight[x] = v;
146 }
147
148 int main(void){
149     int kase = 0;
150     while(scanf("%d%d",&n,&m) == 2&& n){
151         rep(i,1,n+1) scanf("%d",&weight[i]);
152         rep(i,1,m+1) scanf("%d%d",&from[i],&to[i]);
153         me(removed);
154         int c = 0;
155         for(;;){
156             char type;
157             int x,p = 0,v = 0;
158             scanf(" %c",&type);
159             if(type == 'E') break;
160             scanf("%d",&x);
161             if(type == 'D') removed[x] = 1;
162             if(type == 'Q') scanf("%d",&p);
163             if(type == 'C') {
164                 scanf("%d",&v);
165                 p = weight[x];
166                 weight[x] = v;
167             }
168             commands[c++] = (Command){type,x,p};
169         }
170         rep(i,1,n+1) {
171             pa[i] = i; if(root[i] != NULL) removetree(root[i]);
172             root[i] = new Node(weight[i]);
173         }
174         rep(i,1,m+1) if(!removed[i]) add_edge(i);
175         // 反向操作
176         query_tot = query_cnt = 0;
177         per(i,0,c){
178             if(commands[i].type == 'D') add_edge(commands[i].x);
179             if(commands[i].type == 'Q')
180                 ↪ query(commands[i].x,commands[i].p);
181             if(commands[i].type == 'C')
182                 ↪ change_weight(commands[i].x,commands[i].p);
183         }
184         printf("Case %d: %.6lf\n", ++kase, query_tot / (double)query_cnt);
185     }
186 }

```

### 6.4.3 3 伸展树.cpp

```
1  /*
2  UVA 11922
3  序列反转 (a,b)
4
5
6  */
7  #include <bits/stdc++.h>
8  #define mem(ar,num) memset(ar,num,sizeof(ar))
9  #define me(ar) memset(ar,0,sizeof(ar))
10 #define lowbit(x) (x&(-x))
11 #define Pb push_back
12 #define FI first
13 #define SE second
14 #define rep(i,a,n) for (int i=a;i<n;i++)
15 #define per(i,a,n) for (int i=n-1;i>=a;i--)
16 #define IOS ios::sync_with_stdio(false)
17 #define DEBUG cout<<endl<<"DEBUG"<<endl;
18 using namespace std;
19 typedef long long LL;
20 typedef unsigned long long ULL;
21 const int prime = 999983;
22 const int INF = 0x7FFFFFFF;
23 const LL INFF = 0x7FFFFFFFFFFFFFFF;
24 const double pi = acos(-1.0);
25 const double inf = 1e18;
26 const double eps = 1e-6;
27 const LL mod = 1e9 + 7;
28 LL qpow(LL a,LL b){LL s=1;while(b>0){if(b&1)s=s*a%mod;a=a*a%mod;b>>=1;}return s;}
29 LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
30 int dr[2][4] = {1,-1,0,0,0,0,-1,1};
31 typedef pair<int,int> P;
32 struct Node{
33     Node *ch[2];
34     int s;
35     int flip;
36     int v;
37     int cmp(int k) const {
38         int d = k-ch[0]->s;
39         if(d == 1) return -1;
40         return d <= 0?0:1;
41     }
42     void maintain(){
43         s = ch[0]->s+ch[1]->s+1;
44     }
45     void pushdown(){
46         if(flip){
47             flip = 0;
48             swap(ch[0],ch[1]);
49             ch[0]->flip = !ch[0]->flip;
50             ch[1]->flip = !ch[1]->flip;
51         }
52     }
53 };
```

```

54 Node *null = new Node();
55 void rotate(Node *&o,int d){
56     Node *k = o->ch[d^1];
57     o->ch[d^1] = k->ch[d];
58     k->ch[d] = o;
59     o->maintain(); k->maintain(); o = k;
60
61 }
62
63 void splay(Node * &o,int k){
64     // cout<<1<<endl;
65     o->pushdown();
66     int d = o->cmp(k);
67     if(d == 1) k -= o->ch[0]->s + 1;
68     // DEBUG;
69     if(d != -1){
70         Node *p = o->ch[d];
71         p->pushdown();
72         int d2 = p->cmp(k);
73         int k2 = (d2==0?k:k-p->ch[0]->s-1);
74         // cout<<k2<<endl;
75         if(d2 != -1){
76             splay(p->ch[d2],k2);
77             if(d == d2) rotate(o,d^1);
78             else rotate(o->ch[d],d);
79         }
80         rotate(o,d^1);
81     }
82 }
83 Node * Merge(Node *left,Node*right){
84     splay(left,left->s);
85     left->ch[1] = right;
86     left->maintain();
87     return left;
88 }
89
90 void split(Node *o,int k,Node * &left,Node *&right){
91     splay(o,k);
92     left = o;
93     right = o->ch[1];
94     o->ch[1] = null;
95     left->maintain();
96 }
97 const int maxn = 1e5+10;
98 struct SplaySequence{
99     int n;
100     Node seq[maxn];
101     Node *root;
102     Node *build(int sz){
103         if(!sz) return null;
104         Node *L = build(sz/2);
105         Node *o = &seq[++n];
106         o->v = n;
107         o->ch[0] = L;
108         o->ch[1] = build(sz-sz/2-1);
109         o->flip = o->s = 0;

```

```

110         o->maintain();
111         return o;
112     }
113     void init(int sz){
114         n = 0;
115         null->s = 0;
116         root = build(sz);
117     }
118 };
119 vector<int> ans;
120 void print(Node *o){
121     if(o!=null){
122         o->pushdown();
123         print(o->ch[0]);
124         ans.push_back(o->v);
125         print(o->ch[1]);
126     }
127 }
128 void debug(Node *o){
129     if(o!=null){
130         o->pushdown();
131         debug(o->ch[0]);
132         printf("%d ",o->v-1);
133         debug(o->ch[1]);
134     }
135 }
136 SplaySequence ss;
137 int main(void)
138 {
139     int n,m;
140     scanf("%d%d",&n,&m);
141     // cout<<n<<" "<<m<<endl;
142     ss.init(n+1);
143
144
145     while(m--){
146         int a,b;
147         scanf("%d %d",&a,&b);
148         // cout<<a<<" "<<b<<endl;
149         Node *left,*mid,*right,*o;
150         split(ss.root,a,left,o);
151         // DEBUG;
152         split(o,b-a+1,mid,right);
153         mid->flip ^= 1;
154         ss.root = Merge(Merge(left,right),mid);
155     }
156     print(ss.root);
157     for(int i = 1; i < ans.size(); i++)
158         printf("%d\n",ans[i]-1);
159     return 0;
160 }

```

---

## 6.5 基础数据结构

### 6.5.1 堆.cpp

---

```
1 // 堆的插入和删除操作
2
3 void Insert(int vv)
4 {
5     int t = sz++;
6     h[t] = vv;
7     while(t > 1)
8     {
9         if(h[t] < h[t/2])
10         {
11             swap(h[t],h[t/2]);
12             t /= 2;
13         }
14         else break;
15     }
16 }
17 int Down(int i)
18 {
19     int t;
20     while(i * 2 <= n)
21     {
22         if(h[i] > h[2*i])
23             t = 2*i;
24         else
25             t = i;
26         if(i*2+1 <= n&&h[i*2+1] < h[t])
27             t = i*2+1;
28         if(i == t)
29             break;
30         swap(h[t],h[i]);
31         i = t;
32     }
33 }
```

---

## 6.6 字符串

### 6.6.1 1 Trie(前缀树).cpp

---

```
1 const int maxnode = 4e5+100;
2 const int sigma_size = 26;
3 struct Trie
4 {
5     int ch[maxnode][sigma_size];
6     int val[maxnode];
7     int sz;
8     Trie()
9     {
10         sz = 1;
11         memset(ch[0],0,sizeof(ch[0]));
12     }
13     int idx(char c)
```

---



```

14     {
15         return c-'a';
16     }
17 void init(void)
18 {
19     memset(ch,0,sizeof(ch));
20     memset(val,0,sizeof(val));
21 }
22 void insert(char *s,int v)
23 {
24     int u = 0, n = strlen(s);
25     for(int i = 0; i < n; ++i)
26     {
27         int c = idx(s[i]);
28         if(!ch[u][c])
29         {
30             memset(ch[sz],0,sizeof(ch[sz]));
31             val[sz] = 0;
32             ch[u][c] = sz++;
33         }
34         u = ch[u][c];
35     }
36     val[u] = v;
37 }
38 int query(char *s,int t)
39 {
40     int sum = 0;
41     int u = 0,n = strlen(s);
42     for(int i = 0; i < n; ++i)
43     {
44         int c = idx(s[i]);
45         if(ch[u][c])
46         {
47             if(val[ch[u][c]])
48                 sum = (sum+ans[i+t+1]) % mod;
49         }
50         else
51             return sum;
52         u = ch[u][c];
53     }
54     return sum;
55 }
56
57 };

```

---

## 6.6.2 2 KMP.cpp

```

1 #include <bits/stdc++.h>
2 #define mem(ar,num) memset(ar,num,sizeof(ar))
3 #define me(ar) memset(ar,0,sizeof(ar))
4 #define lowbit(x) (x&(-x))
5 using namespace std;
6 typedef long long LL;
7 typedef unsigned long long ULL;
8 const int prime = 999983;

```

```

9  const int    INF = 0x7FFFFFFF;
10 const LL     INFF =0x7FFFFFFFFFFFFFFF;
11 const double pi = acos(-1.0);
12 const double inf = 1e18;
13 const double eps = 1e-6;
14 const LL mod = 20071027 ;
15 int f[1100];
16 char ch[100];
17 void getFail(char *P,int *f)
18 {
19     int m = strlen(P);
20     f[0] = 0,f[1] = 0;
21     for(int i = 1;i < m; ++i)
22     {
23         int j = f[i];
24         while(j && P[i] != P[j]) j = f[j];
25         f[i+1] = P[i] == P[j] ? j + 1: 0;
26     }
27 }
28
29 //Allinone
30 void find(char * T,char * P,int* f)
31 {
32     int n = strlen(T),m = strlen(P);
33     getFail(P,f);
34     int j = 0;
35     for(int i = 0;i < n; ++i)
36     {
37         while(j&&P[j] != T[i]) j = f[j];
38         if(P[j] == T[i]) j++;
39         if(j == m) printf("%d\n",i-m+1);
40     }
41 }
42
43 int main(void)
44 {
45     cin>>ch;
46     getFail(ch,f);
47     printf("%d",f[strlen(ch)-1]);
48
49     return 0;
50 }

```

---

### 6.6.3 3 AC 自动机.cpp

---

```

1  const int SIGMA_SIZE = 26;
2  const int MAXNODE = 11000;
3  const int MAXS = 150 + 10;
4
5
6  struct AhoCorasickAutomata {
7      int ch[MAXNODE][SIGMA_SIZE];
8      int f[MAXNODE]; // fail 函数
9      int val[MAXNODE]; // 每个字符串的结尾结点都有一个非 0 的 val
10     int last[MAXNODE]; // 输出链表的下一个结点

```

```

11  int sz;
12
13  void init() {
14      sz = 1;
15      memset(ch[0], 0, sizeof(ch[0]));
16  }
17
18  // 字符 c 的编号
19  int idx(char c) {
20      return c - 'a';
21  }
22
23  // 插入字符串。v 必须非 0
24  void insert(char *s, int v) {
25      int u = 0, n = strlen(s);
26      for(int i = 0; i < n; i++) {
27          int c = idx(s[i]);
28          if(!ch[u][c]) {
29              memset(ch[sz], 0, sizeof(ch[sz]));
30              val[sz] = 0;
31              ch[u][c] = sz++;
32          }
33          u = ch[u][c];
34      }
35      val[u] = v;
36  }
37
38  // 递归打印以结点 j 结尾的所有字符串
39  void print(int j) {
40      if(j) {
41          print(last[j]);
42      }
43  }
44
45  // 在 T 中找模板
46  int find(char* T) {
47      int n = strlen(T);
48      int j = 0; // 当前结点编号, 初始为根结点
49      for(int i = 0; i < n; i++) { // 文本串当前指针
50          int c = idx(T[i]);
51          while(j && !ch[j][c]) j = f[j]; // 顺着细边走, 直到可以匹配
52          j = ch[j][c];
53          if(val[j]) print(j);
54          else if(last[j]) print(last[j]); // 找到了!
55      }
56  }
57
58  // 计算 fail 函数
59  void getFail() {
60      queue<int> q;
61      f[0] = 0;
62      // 初始化队列
63      for(int c = 0; c < SIGMA_SIZE; c++) {
64          int u = ch[0][c];
65          if(u) { f[u] = 0; q.push(u); last[u] = 0; }
66      }

```

```

67 // 按 BFS 顺序计算 fail
68 while(!q.empty()) {
69     int r = q.front(); q.pop();
70     for(int c = 0; c < SIGMA_SIZE; c++) {
71         int u = ch[r][c];
72         if(!u) continue;
73         q.push(u);
74         int v = f[r];
75         while(v && !ch[v][c]) v = f[v];
76         f[u] = ch[v][c];
77         last[u] = val[f[u]] ? f[u] : last[f[u]];
78     }
79 }
80 }
81
82 };

```

---

#### 6.6.4 4 KMP-KMP 变形.cpp

---

```

1 //https://www.nowcoder.com/acm/contest/119/E
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int N=200010;
7 int a[N],b[N];
8 int x[N],y[N],nxt[N];
9
10 void kmp_pre(int x[],int m,int nxt[])
11 {
12     int i,j;
13     j=nxt[0]=-1;
14     i=0;
15     while(i<m) {
16         while(-1!=j && (x[i]!=x[j]&&x[j]!=-1))j=nxt[j];
17         nxt[++i]=++j;
18     }
19 }
20
21 int KMP_Count(int x[],int m,int y[],int n)
22 {
23     // for (int i=0;i<n;i++) {
24     //     printf("%d ",y[i]);
25     // }
26     // puts("");
27     // for (int i=0;i<m;i++) {
28     //     printf("%d ",x[i]);
29     // }
30     // puts("");
31     int i,j;
32     int ans=0;
33     kmp_pre(x,m,nxt);
34     i=j=0;
35     while(i<n) {
36         while(-1!=j && !(y[i]==x[j]||(x[j]==-1&&(y[i]==-1||j-y[i]<0)))) j=nxt[j];

```

```

37         i++;
38         j++;
39         if(j>=m) {
40             ans++;
41             j=nxt[j];
42         }
43     }
44     return ans;
45 }
46
47 int main()
48 {
49     int n,m,k;
50     scanf("%d%d",&n,&k);
51     memset(x,-1,sizeof(x));
52     memset(y,-1,sizeof(y));
53     map<int,int> pre;
54     for (int i=0;i<n;i++) {
55         scanf("%d",&a[i]);
56         auto pos=pre.find(a[i]);
57         if (pos!=pre.end()) {
58             y[i]=i-pos->second;
59         }
60         pre[a[i]]=i;
61     }
62     scanf("%d",&m);
63     pre.clear();
64     for (int i=0;i<m;i++) {
65         scanf("%d",&b[i]);
66         auto pos=pre.find(b[i]);
67         if (pos!=pre.end()) {
68             x[i]=i-pos->second;
69         }
70         pre[b[i]]=i;
71     }
72     printf("%d\n",KMP_Count(x,m,y,n));
73     return 0;
74 }

```

---

### 6.6.5 5 字符串 hash.cpp

```

1 // 字符串 hash, 查找在字符串中至少出现 k 次的最长字符串
2 #include<cstdio>
3 #include<cstring>
4 #include<algorithm>
5 using namespace std;
6
7 const int maxn = 40000+10;
8 const int x = 123;
9 int n,m,pos;
10
11 unsigned long long H[maxn],xp[maxn];
12
13 unsigned long long Hash[maxn];
14 int Rank[maxn];

```

```

15
16 int cmp(const int &a,const int &b){
17     return Hash[a] < Hash[b] || (Hash[a] == Hash[b] &&a < b );
18 }
19
20 int possible(int L){
21     int c = 0;
22     pos = -1;
23     for(int i = 0;i < n-L+1; ++i){
24         Rank[i] = i;
25         Hash[i] = H[i]-H[i+L]*xp[L];
26
27     }
28     sort(Rank,Rank+n-L+1,cmp);
29     for(int i = 0;i < n-L+1; ++i){
30         if(i == 0||Hash[Rank[i]] != Hash[Rank[i-1]]) c = 0;
31         if(++c >= m) pos = max(pos,Rank[i]);
32     }
33     return pos >= 0;
34 }
35
36 char s[maxn];
37 int main(void)
38 {
39     while((scanf("%d",&m)) == 1&&m){
40         scanf("%s",s);
41         n = strlen(s);
42         H[n] = 0;
43         for(int i = n-1;i >= 0; i--) H[i] = H[i+1]*x+(s[i]-'a');
44         xp[0] = 1;
45         for(int i = 1;i <= n; ++i) xp[i] = xp[i-1]*x;
46         if(!possible(1)) printf("none\n");
47     else{
48         int L = 1,R = n;
49         while(R >= L){
50             int M = (R+L)/2;
51             if(possible(M)) L = M+1;
52             else R = M-1;
53         }
54         possible(R);
55         printf("%d %d\n",R,pos);
56     }
57 }
58
59 return 0;
60 }

```

---

### 6.6.6 6 后缀数组.cpp

```

1 const int maxn = 1e6 + 10;
2
3 struct SuffixArray {
4     int s[maxn];      // 原始字符数组（最后一个字符应必须是 0，而前面的字符必须非 0）
5     int sa[maxn];     // 后缀数组
6     int rank[maxn];   // 名次数组。rank[0] 一定是 n-1，即最后一个字符

```

```

7  int height[maxn]; // height 数组
8  int t[maxn], t2[maxn], c[maxn]; // 辅助数组
9  int n; // 字符个数
10
11 void clear() { n = 0; memset(sa, 0, sizeof(sa)); }
12
13 // m 为最大字符值加 1。调用之前需设置好 s 和 n
14 void build_sa(int m) {
15     int i, *x = t, *y = t2;
16     for(i = 0; i < m; i++) c[i] = 0;
17     for(i = 0; i < n; i++) c[x[i] = s[i]]++;
18     for(i = 1; i < m; i++) c[i] += c[i-1];
19     for(i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
20     for(int k = 1; k <= n; k <= 1) {
21         int p = 0;
22         for(i = n-k; i < n; i++) y[p++] = i;
23         for(i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i]-k;
24         for(i = 0; i < m; i++) c[i] = 0;
25         for(i = 0; i < n; i++) c[x[y[i]]]++;
26         for(i = 0; i < m; i++) c[i] += c[i-1];
27         for(i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
28         swap(x, y);
29         p = 1; x[sa[0]] = 0;
30         for(i = 1; i < n; i++)
31             x[sa[i]] = y[sa[i-1]]==y[sa[i]] && y[sa[i-1]+k]==y[sa[i]+k] ? p-1 : p++;
32         if(p >= n) break;
33         m = p;
34     }
35 }
36
37 void build_height() {
38     int i, j, k = 0;
39     for(i = 0; i < n; i++) rank[sa[i]] = i;
40     for(i = 0; i < n; i++) {
41         if(k) k--;
42         int j = sa[rank[i]-1];
43         while(s[i+k] == s[j+k]) k++;
44         height[rank[i]] = k;
45     }
46 }
47 };

```

---

## 6.7 并查集

### 6.7.1 加权并查集 + 区间合并.cpp

---

```

1  const int LEN = 234567;
2  int F[LEN];
3  int val[LEN];
4  int Find(int x){
5      int k = F[x];
6      if(x!=k){
7          F[x] = Find(k);
8          val[x] += val[k];
9      }

```

```

10     return F[x];
11 }
12 int main(void)
13 {
14     int N,M;
15     while(cin>>N>>M) {
16         for(int i = 0;i <= N; ++i){
17             F[i] = i;
18             val[i] = 0;
19         }
20         int a,b,c;
21         int Count = 0;
22         while(M--){
23             scanf("%d %d %d",&a,&b,&c);
24             a--;
25             int x1 = Find(a);
26             int y1 = Find(b);
27             if(x1==y1&&c+val[a]!=val[b])
28                 ++Count;
29             else if(x1<y1) {
30                 F[y1] = x1;
31                 val[y1] = c+val[a]-val[b];
32             }
33             else if(x1>y1){
34                 F[x1] = y1;
35                 val[x1] = val[b]-val[a]-c;
36             }
37         }
38         cout<<Count<<endl;
39     }
40     return 0;
41 }

```

---

### 6.7.2 并查集.cpp

```

1 //http://acm.hdu.edu.cn/showproblem.php?pid=1232
2
3 #include <iostream>
4 #include <cstdio>
5 #include <set>
6 #include <cstring>
7 using namespace std;
8 const int LEN = 1000+5;
9 int N,M;
10 int ar[LEN];
11 int Find(int x)//并查集之 find 函数
12 {
13     return x==ar[x]?x:ar[x]=Find(ar[x]);
14 }
15 int main()
16 {
17
18     while(cin>>N&&N)
19     {
20         cin>>M;

```



```

21     for(int i = 1;i <= N; ++i)
22         ar[i] = i;
23     while(M--)
24     {
25         int a,b;
26         scanf("%d %d",&a,&b);
27         if(Find(a)!=Find(b))//如果不在一个集合，合并
28         {
29             ar[Find(a)] = Find(b);
30         }
31     }
32     int Count=0;
33     for(int i = 1;i <= N; ++i)
34         if(Find(ar[i]) == i)
35             Count++;
36     cout<<Count-1<<endl;
37
38 }
39 return 0;
40 }

```

---

## 6.8 树状数组

### 6.8.1 1 树状数组模板.cpp

```

1 void Add(int x,int p)//
2 {
3     while(x<=N)
4     {
5         tree[x] += p;
6         x += lowbit(x);
7     }
8 }
9 int Query(int x)
10 {
11     int sum = 0;
12     while(x)
13     {
14         sum += tree[x];
15         x -= lowbit(x);
16     }
17     return sum;
18 }

```

---

### 6.8.2 2 区间出现两次的数的个数.cpp

```

1 //..... 离线树状数组
2 int n,m;
3 const int LEN = 2e5+100;
4 int tree[LEN];//树状数组
5 int ans[LEN];//答案数组
6 int ar[LEN];
7 int last[LEN];//last[i] 上一个与 ar[i] 相等的元素的位置
8 map<int,int> ma;//存储每一个数对应的最后的位置

```

```

9  struct Q
10 {
11     int l,r,ID;
12 };
13 Q q[LEN];
14 bool operator <(const Q &a,const Q &b)
15 {
16     return a.r < b.r;
17 }
18 void modify(int x,int d)
19 {
20     while(x <= n)
21     {
22         tree[x] += d;
23         x += lowbit(x);
24     }
25 }
26 int Query(int x)
27 {
28     int sum = 0;
29     while(x>0)
30     {
31         sum += tree[x];
32         x -= lowbit(x);
33     }
34     return sum;
35 }
36
37 int main()
38 {
39
40     cin>>n>>m;
41
42     for(int i = 1; i <= n; ++i)
43     {
44         scanf("%d",&ar[i]);
45         last[i] = ma[ar[i]];
46         ma[ar[i]] = i;
47     }
48     for(int i = 1; i <= m; ++i)
49     {
50         scanf("%d %d",&q[i].l,&q[i].r);
51         q[i].ID = i;
52     }
53     sort(q+1,q+m+1);
54     int index = 1;
55     /* 树状数组的目的是进行快速求和，我们可以假设求和的数组是 C*/
56     for(int i = 1; i <= n; ++i)
57     {
58         if(last[i]!=0)
59             modify(last[i],1);//将上一个与这个元素相同的元素的位置 +1，代表有一组
60         int p = last[last[i]];
61         if(p != 0)
62         {
63             modify(p,-2);/* 如果有三个或者多个该元素，则需要-2，把 +1 抵消，并且把之前
        ↪ p 和 last[i] 这个组合抵消 */

```

```

64         int pp = last[p];
65         if(pp != 0)//消除-2 的影响
66             modify(pp,1);
67     }
68     // 分析后得知 c[i] 只有三种可能的值,0,-1,1,
69
70
71     while(index <= m&&q[index].r == i)
72     {
73
74         ans[q[index].ID] = Query(i) - Query(q[index].l-1);/* 这个时候
75         ↪ Query(i) 就代表从 1 到 i 有多少个恰好两次的不同数,Query(q[index].l-1)
76         ↪ 则不是 */
77         index ++;
78     }
79 }
80 for(int i = 1; i <= m; ++i)
81     printf("%d\n",ans[i]);
82 return 0;
83 }

```

---

## 6.9 线段树

### 6.9.1 1. 区间更新区间查询.cpp

---

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define lson (o << 1)
4  #define rson (o << 1|1)
5  const int maxn = 1e5+10;
6  const int INF = 1e9;
7  typedef long long LL;
8  struct Tree{
9      LL min,max,sum,add;
10 };
11 Tree tree[maxn<<2];
12 LL a[maxn];
13 void pushup(int o,int l,int r){
14     tree[o].min = min(tree[lson].min,tree[rson].max);
15     tree[o].max = max(tree[lson].max,tree[rson].max);
16     tree[o].sum = tree[lson].sum + tree[rson].sum;
17 }
18 void pushdown(int o,int l,int r){
19     int m = (l+r)>>1;
20     if(tree[o].add){
21         tree[lson].add += tree[o].add;
22         tree[lson].sum += (m-l+1)*tree[o].add;
23         tree[lson].min += tree[o].add;
24         tree[lson].max += tree[o].add;
25
26         tree[rson].add += tree[o].add;
27         tree[rson].sum += (r-m)*tree[o].add;
28         tree[rson].min += tree[o].add;
29         tree[rson].max += tree[o].add;
30         tree[o].add = 0;

```

```

31     }
32 }
33 void up(Tree & a, Tree b){
34     a.min = min(a.min, b.min);
35     a.max = max(a.max, b.max);
36     a.sum += b.sum;
37 }
38 void build(int o, int l, int r){
39     // cout<<l<<" "<<r<<endl;
40     tree[o].add = 0;
41     if(l == r)
42     {
43         tree[o].min = tree[o].max = tree[o].sum = a[l];
44         // cout<<l <<" "<<a[l]<<endl;
45     }
46     else{
47         int m = (l+r)>>1;
48         build(lson, l, m);
49         build(rson, m+1, r);
50         pushup(o, l, r);
51     }
52 }
53 void Update(int o, int l, int r, int L, int R, int v){
54     if(L <= l && R >= r){
55         tree[o].add += v;
56         tree[o].sum += (r-l+1)*v;
57         tree[o].max += v;
58         tree[o].min += v;
59         return ;
60     }
61     pushdown(o, l, r);
62     int m = (l+r)/2;
63     if(L <= m)
64         Update(lson, l, m, L, R, v);
65     if(R > m)
66         Update(rson, m+1, r, L, R, v);
67     pushup(o, l, r);
68 }
69 Tree Query(int o, int l, int r, int L, int R){
70
71     if(L <= l && R >= r)
72     {
73         return tree[o];
74     }
75     Tree tmp;
76     tmp.min = INF, tmp.max = -INF, tmp.sum = 0;
77     pushdown(o, l, r);
78     int m = (l+r)>>1;
79     if(L <= m)
80         up(tmp, Query(lson, l, m, L, R));
81     if(R > m)
82         up(tmp, Query(rson, m+1, r, L, R));
83     // cout<<tmp.sum<<endl;
84     return tmp;
85 }
86 int main(void){

```

```

87
88     int N,Q;cin>>N>>Q;
89     for(int i =1;i <= N; ++i)
90         scanf("%lld",&a[i]);
91     build(1,1,N);
92     // cout<<Query(1,1,N,1,1).sum<<endl;
93     while(Q--){
94         LL c,x,y,v;
95         scanf("%lld%lld%lld",&c,&x,&y);
96         if(c == 1){
97             scanf("%lld",&v);
98             Update(1,1,N,x,y,v);
99         }
100        else{
101            printf("%lld\n",Query(1,1,N,x,y).sum);
102        }
103    }
104
105
106    return 0;
107 }

```

---

## 6.9.2 2 主席树求第 k 大.cpp

```

1 // 主席树求第 k 大
2 // 先离散，后可持续化建树
3 // poj 2104
4
5 #include <bits/stdc++.h>
6 #define me(ar) memset(ar,0,sizeof(ar))
7 #define rep(i,a,n) for (int i=a;i<n;i++)
8 using namespace std;
9 const int maxn = 1e5+10;
10 int sum[maxn<<5],L[maxn<<5],R[maxn<<5];
11 int rt[maxn];
12 int a[maxn],Hash[maxn];
13 int tot = 0;
14 int build(int l,int r){
15     int rt = (++tot);
16     sum[rt] = 0;
17     if(l < r){
18         int m = (l+r) >> 1;
19         L[rt] = build(l,m);
20         R[rt] = build(m+1,r);
21     }
22     return rt;
23 }
24
25 int update(int pre,int l,int r,int x){
26     int rt = (++tot);
27     L[rt] = L[pre],R[rt] = R[pre],sum[rt] = sum[pre]+1;
28     if(l < r){
29         int m = (l+r)>>1;
30         if(x <= m)
31             L[rt] = update(L[pre],l,m,x);

```

```

32         else
33             R[rt] = update(R[pre],m+1,r,x);
34     }
35     return rt;
36 }
37 int query(int u,int v,int l,int r,int k){
38     if(l >= r) return r;
39     int num = sum[L[v]]-sum[L[u]];
40     int m = (l+r)>>1;
41     if(num >= k)
42         return query(L[u],L[v],l,m,k);
43     return query(R[u],R[v],m+1,r,k-num);
44 }
45 int main(void)
46 {
47
48     int T;
49     scanf("%d",&T);
50     while(T--){
51         tot = 0;
52         int n,m;
53         scanf("%d%d",&n,&m);
54         // map<int,int> ma;
55         rep(i,1,n+1){scanf("%d",&a[i]);Hash[i] = a[i];}
56         sort(Hash+1,Hash+1+n);
57         int id = unique(Hash+1,Hash+1+n) - Hash-1;
58         rt[0] = build(1,id);
59         rep(i,1,n+1){
60             int x = lower_bound(Hash+1,Hash+id+1,a[i]) - Hash;
61             rt[i] = update(rt[i-1],1,id,x);
62         }
63         rep(i,0,m){
64             int l,r,k;
65             scanf("%d%d%d",&l,&r,&k);
66             int ans = query(rt[l-1],rt[r],1,id,k);
67             printf("%d\n",Hash[ans]);
68         }
69     }
70
71     return 0;
72 }

```

---

### 6.9.3 2 树套树求动态第 k 大.cpp

---

```

1  /*
2  ZOJ
3  Dynamic Rankings ZOJ - 2112
4  动态第 k 大数
5  */
6  //lowbit 自己写
7  #define lson l,m
8  #define rson m+1,r
9  const int N = 60006;
10 int a[N],Hash[N];
11 int T[N],L[N<<5],R[N<<5],sum[N<<5];

```

```

12 int S[N];
13 int n,m,tot;
14 struct node{
15     int l,r,k;
16     bool Q;
17 }op[10005];
18
19 int build(int l,int r){
20     int rt = (++tot);
21     sum[rt] = 0;
22     if(l != r){
23         int m = (l+r)>>1;
24         L[rt] = build(lson);
25         R[rt] = build(rson);
26     }
27     return rt;
28 }
29
30 int update(int pre,int l,int r,int x,int val){
31     int rt = (++tot);
32     L[rt] = L[pre],R[rt] = R[pre],sum[rt] = sum[pre]+val;
33     if(l < r){
34         int m = (l+r)>>1;
35         if(x <= m)
36             L[rt] = update(L[pre],lson,x,val);
37         else
38             R[rt] = update(R[pre],rson,x,val);
39     }
40     return rt;
41 }
42 int use[N];
43 void add(int x,int pos,int val){
44     while(x <= n){
45         S[x] = update(S[x],1,m,pos,val);
46         x += lowbit(x);
47     }
48 }
49 int Sum(int x){
50     int ret = 0;
51     while(x > 0){
52         ret += sum[L[use[x]]];
53         x -= lowbit(x);
54     }
55     return ret;
56 }
57
58 int query(int u,int v,int lr,int rr,int l,int r,int k){
59     if(l >= r)
60         return l;
61     int m = (l+r)>>1;
62     int tmp = Sum(v)-Sum(u)+sum[L[rr]]-sum[L[lr]];
63     if(tmp >= k){
64         for(int i = u;i;i -= lowbit(i))
65             use[i] = L[use[i]];
66         for(int i = v;i;i -= lowbit(i))
67             use[i] = L[use[i]];

```

```

68     return query(u,v,L[lr],L[rr],lson,k);
69 }
70 else{
71     for(int i = u;i ;i -= lowbit(i))
72         use[i] = R[use[i]];
73     for(int i = v;i ;i -= lowbit(i))
74         use[i] = R[use[i]];
75     return query(u,v,R[lr],R[rr],rson,k-tmp);
76 }
77
78 }
79
80 void modify(int x,int p,int d){
81     while(x <= n){
82         S[x] = update(S[x],1,m,p,d);
83         x += lowbit(x);
84     }
85 }
86 int main(){
87     int t;
88     scanf("%d",&t);
89     while(t--){
90         int q;
91         scanf("%d%d",&n,&q);
92         tot = 0;
93         m = 0;
94         for(int i = 1;i <= n; ++i)
95         {
96             scanf("%d",&a[i]);
97             Hash[++m] = a[i];
98         }
99         for(int i = 0;i < q; ++i){
100             char s[10];
101             scanf("%s",s);
102             if(s[0] == 'Q'){
103                 scanf("%d%d%d",&op[i].l,&op[i].r,&op[i].k);
104                 op[i].Q = 1;
105             }
106             else{
107                 scanf("%d%d",&op[i].l,&op[i].r);
108                 op[i].Q = 0;
109                 Hash[++m] = op[i].r;
110             }
111         }
112         sort(Hash+1,Hash+1+m);
113         int mm = unique(Hash+1,Hash+1+m)-Hash-1;
114         m = mm;
115         T[0] = build(1,m);
116         for(int i = 1;i <= n; ++i)
117             T[i] = update(T[i-1],1,m,lower_bound(Hash+1,Hash+1+m,a[i])-Hash,1);
118         // DEBUG;
119
120         for(int i = 1;i <= n; ++i)
121             S[i] = T[0];
122         for(int i = 0;i < q; ++i){
123             // DEBUG;

```



```

124     if(op[i].Q){
125
126         // cout<<op[i].l<<" "<<op[i].r<<" "<<endl;
127         for(int j = op[i].l-1;j;j -= lowbit(j))
128             use[j] = S[j];
129         for(int j = op[i].r ;j;j -= lowbit(j))
130             use[j] = S[j];
131         // DEBUG;
132
133         ↪ printf("%d\n",Hash[query(op[i].l-1,op[i].r,T[op[i].l-1],T[op[i].r],1,m,op[i].k)
134     }
135     else{
136         modify(op[i].l,lower_bound(Hash+1,Hash+1+m,a[op[i].l])-Hash,-1);
137         modify(op[i].l,lower_bound(Hash+1,Hash+1+m,op[i].r)-Hash,1);
138         a[op[i].l] = op[i].r;
139     }
140
141 }
142 }
143 return 0;
144 }
145
146 /*
147 2
148 5 3
149 3 2 1 4 7
150 Q 1 4 3
151 C 2 6
152 Q 2 5 3
153 5 3
154 3 2 1 4 7
155 Q 1 4 3
156 C 2 6
157 Q 2 5 3
158 */

```

---

#### 6.9.4 3 树套树求动态逆序数.cpp

```

1 //数据范围 1-n 的全排列
2 #include<bits/stdc++.h>
3 #define inf 0x7fffffff
4 #define N 100005
5 #define M 5000005
6 using namespace std;
7 typedef long long ll;
8 ll ans;
9 int n,m,sz,a[100],b[100],val[N],pos[N],a1[N],a2[N];
10 int c[N*10],rt[N],ls[M],rs[M],sumv[M];
11 inline int lowbit(int x){return x&(-x);}
12 inline int ask(int x){
13     int ans=0;
14     for(int i=x;i;i-=lowbit(i))ans+=c[i];
15     return ans;
16 }

```

```

17 void change(int &o,int l,int r,int q){
18     if(!o)o=++sz;sumv[o]++;
19     if(l==r)return;
20     int mid=(l+r)>>1;
21     if(q<=mid)change(ls[o],l,mid,q);
22     else change(rs[o],mid+1,r,q);
23 }
24 int querysub(int x,int y,int v){
25     int cnta=0,cntb=0;int ans=0;x--;
26     for(int i=x;i; i-=lowbit(i))a[++cnta]=rt[i];
27     for(int i=y;i; i-=lowbit(i))b[++cntb]=rt[i];
28     int l=1,r=n;
29     while(l!=r){
30         int mid=(l+r)>>1;
31         if(v<=mid){
32             for(int i=1;i<=cnta;i++)ans-=sumv[rs[a[i]]];
33             for(int i=1;i<=cntb;i++)ans+=sumv[rs[b[i]]];
34             for(int i=1;i<=cnta;i++)a[i]=ls[a[i]];
35             for(int i=1;i<=cntb;i++)b[i]=ls[b[i]];
36             r=mid;
37         }
38         else{
39             for(int i=1;i<=cnta;i++)a[i]=rs[a[i]];
40             for(int i=1;i<=cntb;i++)b[i]=rs[b[i]];
41             l=mid+1;
42         }
43     }
44     return ans;
45 }
46 int querypre(int x,int y,int v){
47     int cnta=0,cntb=0,ans=0;x--;
48     for(int i=x;i; i-=lowbit(i))a[++cnta]=rt[i];
49     for(int i=y;i; i-=lowbit(i))b[++cntb]=rt[i];
50     int l=1,r=n;
51     while(l!=r){
52         int mid=(l+r)>>1;
53         if(v>mid){
54             for(int i=1;i<=cnta;i++)ans-=sumv[ls[a[i]]];
55             for(int i=1;i<=cntb;i++)ans+=sumv[ls[b[i]]];
56             for(int i=1;i<=cnta;i++)a[i]=rs[a[i]];
57             for(int i=1;i<=cntb;i++)b[i]=rs[b[i]];
58             l=mid+1;
59         }
60         else{
61             for(int i=1;i<=cnta;i++)a[i]=ls[a[i]];
62             for(int i=1;i<=cntb;i++)b[i]=ls[b[i]];
63             r=mid;
64         }
65     }
66     return ans;
67 }
68 inline int read(){
69     int f=1,x=0;char ch;
70     do{ch=getchar();if(ch=='-')f=-1;}while(ch<'0' || ch>'9');
71     do{x=x*10+ch-'0';ch=getchar();}while(ch>='0'&&ch<='9');
72     return f*x;

```

```

73 }
74 int main(){
75     n=read();m=read();
76     for(int i=1;i<=n;i++){
77         val[i]=read();pos[val[i]]=i;
78         a1[i]=ask(n)-ask(val[i]);
79         ans+=a1[i];
80         for(int j=val[i];j<=n;j+=lowbit(j))c[j]++;
81     }
82     memset(c,0,sizeof(c));
83     for(int i=n;i;i--){
84         a2[i]=ask(val[i]-1);
85         for(int j=val[i];j<=n;j+=lowbit(j))c[j]++;
86     }
87     for(int i=1;i<=m;i++){
88         printf("%lld\n",ans);
89         int x=read();x=pos[x];
90         ans-=(a1[x]+a2[x]-querysub(1,x-1,val[x])-querypre(x+1,n,val[x]));
91         for(int j=x;j<=n;j+=lowbit(j))change(rt[j],1,n,val[x]);
92     }
93     return 0;
94 }
95
96 // 对于 100% 的数据,  $n \leq 40000$ ,  $m \leq n/2$ , 且保证第二行  $n$  个数互不相同, 第三行  $m$  个数互不相同。
97 #include<iostream>
98 #include<cstdio>
99 #include<cstdlib>
100 #include<algorithm>
101 #include<cstring>
102 #include<queue>
103 #include<vector>
104 #define ll long long
105 const int maxn=100000+9999;
106 using namespace std;
107 int n,m,num[maxn],H[maxn],Q[maxn],cnt,root[maxn*50],t[maxn],pos[maxn];
108 int A[100],B[100];
109 ll ans;
110 int LO(int x){return x&-x;}
111 int qsum(int x){
112     int tmp=0;
113     for(int i=x;i;i-=LO(i))
114         tmp+=t[i];
115     return tmp;
116 }
117 int read(){
118     int an=0,f=1;
119     char ch=getchar();
120     while(ch<'0' || ch>'9'){if(ch=='-')f=-1;ch=getchar();}
121     while('0'<=ch&&ch<='9'){an=an*10+ch-'0';ch=getchar();}
122     return an*f;
123 }
124 struct saber{
125     int r,l,sum;
126 }T[maxn*50];
127 int askmore(int x,int y,int wi){
128     int cnt1,cnt2,tmp=0;cnt1=cnt2=0;

```

```

129     for(int i=x;i;i-=LO(i))cnt1++,A[cnt1]=root[i];
130     for(int i=y;i;i-=LO(i))cnt2++,B[cnt2]=root[i];
131     int l=1,r=n;
132     while(l!=r){
133         int mid=(l+r)>>1;
134         if(wi<=mid){
135             for(int i=1;i<=cnt1;i++)tmp-=T[ T[ A[i] ].r ].sum;
136             for(int i=1;i<=cnt2;i++)tmp+=T[ T[ B[i] ].r ].sum;
137             for(int i=1;i<=cnt1;i++)A[i]=T[ A[i] ].l;
138             for(int i=1;i<=cnt2;i++)B[i]=T[ B[i] ].l;
139             r=mid;
140         }
141         else {
142             for(int i=1;i<=cnt1;i++)A[i]=T[ A[i] ].r;
143             for(int i=1;i<=cnt2;i++)B[i]=T[ B[i] ].r;
144             l=mid+1;
145         }
146     }
147     return tmp;
148 }
149 int askless(int x,int y,int wi){
150     int cnt1,cnt2,tmp=0;
151     cnt1=cnt2=0;x--;
152     for(int i=x;i;i-=LO(i))cnt1++,A[cnt1]=root[i];
153     for(int i=y;i;i-=LO(i))cnt2++,B[cnt2]=root[i];
154     int l=1,r=n;
155     while(l!=r){
156         int mid=(l+r)>>1;
157         if(wi>mid){
158             for(int i=1;i<=cnt1;i++)tmp-=T[ T[ A[i] ].l ].sum;
159             for(int i=1;i<=cnt2;i++)tmp+=T[ T[ B[i] ].l ].sum;
160             for(int i=1;i<=cnt1;i++)A[i]=T[ A[i] ].r;
161             for(int i=1;i<=cnt2;i++)B[i]=T[ B[i] ].r;
162             l=mid+1;
163         }
164         else {
165             for(int i=1;i<=cnt1;i++)A[i]=T[ A[i] ].l;
166             for(int i=1;i<=cnt2;i++)B[i]=T[ B[i] ].l;
167             r=mid;
168         }
169     }
170     return tmp;
171 }
172 void add(int &y,int l,int r,int wi){
173     if(!y)cnt++,y=cnt;
174     T[y].sum++;
175     if(l==r)return ;
176     int mid=(l+r)>>1;
177     if(wi<=mid)add(T[y].l,l,mid,wi);
178     else add(T[y].r,mid+1,r,wi);
179 }
180 struct da{
181     int wi,i;
182 }data[maxn];
183 bool cmp1(da x,da y){
184     return x.wi<y.wi;

```

```

185 }
186 bool cmp2(da x,da y){
187     return x.i<y.i;
188 }
189 void prepare(){
190     n=read();m=read();
191     for(int i=1;i<=n;i++){
192         data[i].wi=read();
193         data[i].i=i;
194     }
195     sort(data+1,data+1+n,cmp1);
196     for(int i=1;i<=n;i++){
197         data[i].wi=i;
198     }
199     sort(data+1,data+1+n,cmp2);
200     for(int i=1;i<=n;i++)
201         num[i]=data[i].wi;
202 }
203 int main(){
204     prepare();
205     for(int i=1;i<=n;i++){
206         Q[i]=qsum(n)-qsum(num[i]); //Q 在 i 这个点前面比 it 大的数贡献
207         ans+=Q[i];
208         for(int j=num[i];j<=n;j+=LO(j)){
209             t[j]++;
210         }
211     }
212     memset(t,0,sizeof(t));
213     for(int i=n;i;i--){
214         H[i]=qsum(num[i]-1);
215         for(int j=num[i];j<=n;j+=LO(j))
216             t[j]++;
217     }
218     printf("%lld ",ans);
219     while(m){m--;
220     int x=read();
221         ans-=(H[x]+Q[x]-askmore(0,x-1,num[x])-askless(x+1,n,num[x])) );
222         for(int j=x;j<=n;j+=LO(j))add(root[j],1,n,num[x]);
223     printf("%lld ",ans);
224     }
225     return 0;
226 }

```

#### 6.9.5 4 李超树.cpp

```

1 // 对于  $y = a*x+b$ ; 这  $n$  个不同的直线, 查询在某个点的最大的  $y$  值
2
3 // 每一个节点存的是当前节点取最大值的线段的 ID // 查询的时候从根到子节点都查询值, 取其中的
  ↪ 最大值
4 // 插入点的时候
5 // 更新节点的规则就是如果插入直线比当前直线更优, 那么说明原本直线对某区间的最优答案没有贡
  ↪ 献, 这个时候它就可以舍弃
6 // 共有四种情况
7 // 插入直线的斜率大于节点存的斜率,

```

```

8 //如果插入直线的值比原来的节点直线在这个地方的值大，当前值更新为插入直线，用原来节点值更新
   ↪ l,mid
9 //如果插入直线的值小，那么用插入直线更新 mid+1, r;
10 // 如果插入直线的斜率小于节点存的斜率
11 // 如果插入直线的值比原来的节点直线在这个地方的值大，当前值更新为插入直线，用原来节点值更
   ↪ 新 mid+1,r
12 // 如果插入直线的值小，那么用插入直线更新 l, mid+1;
13
14
15 #include <bits/stdc++.h>
16 using namespace std;
17 const int N = 5e5+10;
18 int n,m,tree[N*4];
19 double a[N*2],b[N*2];
20 int cmp(int x,int y,int pos){
21     return a[x] + (pos-1)*b[x] > a[y] +(pos-1)*b[y];
22 }
23 void update(int o,int l,int r,int x){
24     if(l == r){
25         if(cmp(x,tree[o],l))
26             tree[o] = x;
27         return ;
28     }
29     int mid = (l+r)/2;
30     if(b[x] > b[tree[o]]){
31         if(cmp(x,tree[o],mid)){
32             update(o<<1,l,mid,tree[o]),tree[o] = x;
33         }
34         else
35             update(o<<1|1,mid+1,r,x);
36     }
37     if(b[x] < b[tree[o]]){
38         if(cmp(x,tree[o],mid)){
39             update(o<<1|1,mid+1,r,tree[o]),tree[o] = x;
40         }
41         else
42             update(o<<1,l,mid,x);
43     }
44 }
45 }
46 double cal(int k,int x){
47     return a[k] + (x-1)*b[k];
48 }
49 double query(int o,int l,int r,int x){
50     if(l==r) return cal(tree[o],x);
51     int mid = (l+r)/2;
52     double ans = cal(tree[o],x);
53     if(x <= mid) ans = max(ans,query(o<<1,l,mid,x));
54     else
55         ans = max(ans,query(o<<1|1,mid+1,r,x));
56     return ans;
57 }
58 int main(void)
59 {
60     scanf("%d",&n);
61     for(int i = 1;i <=n; ++i){

```

```

62         char s[20];
63         scanf("%s",s);
64         if(s[0] == 'P'){
65             m++;
66             scanf("%lf%lf",&a[m],&b[m]);
67             update(1,1,N,m);
68         }
69         else{
70             int x;
71             scanf("%d",&x);
72             double t = query(1,1,N,x);
73             int k = t;
74             printf("%d\n",k/100);
75         }
76     }
77
78
79     return 0;
80 }

```

---

#### 6.9.6 5 线段树-区间最小乘积.cpp

---

```

1  // 单点更新，区间查询
2
3
4  #include <bits/stdc++.h>
5  #define me(ar) memset(ar,0,sizeof(ar))
6  using namespace std;
7  const int INF = 100000;
8  const int maxn = 1e6+10;
9  const int maxnode = 4*maxn;
10 int ql,qr;
11 int _p,_v;
12 struct T{
13     int a,b,c,d;
14     T(int aa = -INF,int bb = -INF,int cc = INF,int dd =
15         ↪ INF):a(aa),b(bb),c(cc),d(dd){
16     };
17     T up(T x,T y)
18     {
19         int a[4] = {x.a,x.b,y.a,y.b};
20         sort(a,a+4);
21         x.a = a[3];
22         x.b = a[2];
23         int b[4] = {x.c,x.d,y.c,y.d};
24         sort(b,b+4);
25         x.c = b[0];
26         x.d = b[1];
27         return x;
28     }
29     T vv[maxnode];
30     T a[maxn];
31     void build(int o,int l,int r)
32     {

```

```

33     int m = (r+1)>>1;
34     if(1 == r) vv[o] = a[1];
35     else
36     {
37         build(o*2,1,m);
38         build(o*2+1,m+1,r);
39         vv[o] = up(vv[o*2],vv[o*2+1]);
40     }
41 }
42 void update(int o,int l,int r)
43 {
44     if(1 == r) vv[o] = T(_v,-INF,_v,INF);
45     else
46     {
47         int m = (r+1)>>1;
48         if(_p <= m)
49             update(o*2,1,m);
50         else
51             update(o*2+1,m+1,r);
52         vv[o] = up(vv[o*2],vv[o*2+1]);
53     }
54 }
55 T query(int o,int l,int r)
56 {
57
58     if(l >= ql&&r <= qr)
59         return vv[o];
60     int m = 1+(r-1)/2;
61     T ans;
62     if(ql <= m&&m < qr)
63         ans = up(query(o*2,1,m),query(o*2+1,m+1,r));
64     else if(ql <= m)
65         ans = query(o*2,1,m);
66     else if( m < qr)
67         ans = query(o*2+1,m+1,r);
68     return ans;
69 }
70
71 int main(void)
72 {
73     int N,Q;
74     while(scanf("%d",&N) != EOF&&N)
75     {
76         for(int i = 1;i <= N; ++i)
77         {
78             int aa;
79             scanf("%d",&aa);
80             a[i] = T(aa,-INF,aa,INF);
81         }
82         build(1,1,N);
83         cin>>Q;
84         while(Q--)
85         {
86             int op;
87             scanf("%d",&op);
88             if(op == 1)

```



```

89         {
90             scanf("%d %d",&_p,&_v);
91             update(1,1,N);
92         }
93     else
94     {
95         scanf("%d %d",&q1,&q2);
96         T ans = query(1,1,N);
97         long long an =
98             ↪ min(ans.a*ans.b,min(ans.a*ans.c,ans.c*ans.d));
99         printf("%lld\n",an);
100     }
101 }
102 }
103
104 return 0;
105 }

```

---

### 6.9.7 6 区间加斐波那契数.cpp

```

1 //CodeForces 446C DZY Loves Fibonacci Numbers
2
3
4 #include <cstdio>
5
6 const int maxn=300000;
7 const long long mod=1e9+9;
8
9 struct fenv {
10     long long tree[maxn+10];
11     void add(int i, long long d) {
12         for (;i<maxn+10;i|=(i+1)) tree[i]=tree[i]+d;
13     }
14     long long get(int i) {
15         long long ans=0;
16         for (;i>=0; i=(i&(i+1))-1) ans+=tree[i];
17         return ans%mod;
18     }
19 };
20
21 fenv t1, t2, t3;
22 long long fb[maxn+10], s[maxn+10];
23 int n, m, a, t, l, r;
24 char ss[20];
25
26 inline long long getfb(int i) {
27     if (i>0) return fb[i];
28     else if (i%2) return fb[-i];
29     else return mod-fb[-i];
30 }
31
32 inline int geti() {
33     char ch=getchar();
34     while (ch<'0' || ch>'9') ch=getchar();

```

```

35     int ans=0;
36     while (ch>='0'&&ch<='9') ans=(ans*10+ch-'0'), ch=getchar();
37     return ans;
38 }
39
40 inline void puti(int i) {
41     int j=0;
42     while (i) ss[j]=(i%10)+'0', j++, i/=10;
43     for (j--; j>=0; j--) putchar(ss[j]);
44     putchar('\n');
45 }
46
47 int main() {
48     fb[1]=fb[2]=1;
49     for (int i=3; i<maxn+10; i++) fb[i]=(fb[i-1]+fb[i-2])%mod;
50     n=geti(), m=geti();
51     for (int i=1, sum=0; i<=n; i++) a=geti(), sum=(sum+a)%mod, s[i]=sum;
52     for (int i=0; i<m; i++) {
53         t=geti(), l=geti(), r=geti();
54         if (t==1) {
55             long long c=getfb(2-1), d=getfb(3-1);
56             t1.add(1, c);
57             t2.add(1, d);
58             t3.add(1, -1);
59             t1.add(r, -c);
60             t2.add(r, -d);
61             t3.add(r, fb[r-1+3]);
62         } else {
63             puti((int)
64                 ↪ (((t3.get(r)+t1.get(r)*fb[r]+t2.get(r)*fb[r+1]-t3.get(1-1)-t1.get(1-1)*fb[1
65             }
66     return 0;
67 }
68 // #include <bits/stdc++.h>
69 #define eps 1e-6
70 #define LL long long
71 #define pii pair<int, int>
72 #define pb push_back
73 #define mp make_pair
74 // #pragma comment(linker, "/STACK:1024000000,1024000000")
75 using namespace std;
76
77 const int MAXN = 1500000;
78 const int MOD = 1e9+9;
79 LL bas = 276601605;
80 LL q1 = 691504013;
81 LL q2 = 308495997;
82 LL mul1[MAXN], mul2[MAXN];
83 int c[MAXN];
84 LL s[MAXN];
85
86 struct Node {
87     LL a, b, sum;
88 } node[MAXN];
89 int n, k;

```

```

90
91 void init(int m) {
92     mul1[0] = mul2[0] = 1;
93     for (int i = 1; i <= m; i++) {
94         mul1[i] = mul1[i-1] * q1 % MOD;
95         mul2[i] = mul2[i-1] * q2 % MOD;
96     }
97 }
98 void build(int o, int l, int r) {
99     node[o].a = node[o].b = node[o].sum = 0;
100    if (l == r) return;
101    int m = (l+r) >> 1;
102    build(o<<1, l, m);
103    build((o<<1)+1, m+1, r);
104 }
105 void push_down(int o, int l, int r) {
106     LL aa = node[o].a, bb = node[o].b;
107     if (!aa && !bb) return;
108     int lc = o << 1, rc = (o<<1)|1, mid = (l+r) >> 1;
109     int len1 = mid-l+1, len2 = r - mid;
110
111     node[lc].a = (node[lc].a+aa) % MOD;
112     node[lc].b = (node[lc].b+bb) % MOD;
113     node[lc].sum = (node[lc].sum+aa*(mul1[len1+2]-mul1[2])) % MOD;
114     node[lc].sum = (node[lc].sum-bb*(mul2[len1+2]-mul2[2])) % MOD;
115
116     node[rc].a = (node[rc].a+aa*mul1[len1]) % MOD;
117     node[rc].b = (node[rc].b+bb*mul2[len1]) % MOD;
118     node[rc].sum = (node[rc].sum +
119         ↪ aa*mul1[len1]%MOD*(mul1[len2+2]-mul1[2])%MOD) % MOD;
120     node[rc].sum = (node[rc].sum -
121         ↪ bb*mul2[len1]%MOD*(mul2[len2+2]-mul2[2])%MOD) % MOD;
122
123     node[o].a = node[o].b = 0;
124 }
125 void push_up(int o) {
126     node[o].sum = (node[o<<1].sum+node[(o<<1)|1].sum) % MOD;
127 }
128 LL query(int o, int l, int r, int ql, int qr) {
129     if (l == ql && r == qr)
130         return node[o].sum;
131     push_down(o, l, r);
132     int mid = (l+r) >> 1;
133     if (qr <= mid)
134         return query(o<<1, l, mid, ql, qr);
135     else if (ql > mid)
136         return query((o<<1)|1, mid+1, r, ql, qr);
137     else
138         return (query(o<<1, l, mid, ql, mid)+query((o<<1)|1, mid+1, r,
139             ↪ mid+1, qr)) % MOD;
140 }
141 void update(int o, int l, int r, int ql, int qr, LL x, LL y) {
142     if (l == ql && r == qr) {
143         node[o].a = (node[o].a+x) % MOD;
144         node[o].b = (node[o].b+y) % MOD;
145         node[o].sum = (node[o].sum+x*(mul1[r-l+3]-mul1[2])) % MOD;

```

```

143         node[o].sum = (node[o].sum-y*(mul2[r-l+3]-mul2[2])) % MOD;
144         return;
145     }
146     push_down(o, l, r);
147     int mid = (l+r) >> 1;
148     if (qr <= mid)
149         update(o<<1, l, mid, ql, qr, x, y);
150     else if (ql > mid)
151         update((o<<1)|1, mid+1, r, ql, qr, x, y);
152     else {
153         int len = mid - ql + 1;
154         update(o<<1, l, mid, ql, mid, x, y);
155         update((o<<1)|1, mid+1, r, mid+1, qr, x*mul1[len]%MOD,
156             ↪ y*mul2[len]%MOD);
157     }
158     push_up(o);
159 }
160 int main()
161 {
162     //freopen("input.txt", "r", stdin);
163     scanf("%d%d", &n, &k);
164     for (int i = 1; i <= n; i++) {
165         scanf("%d", &c[i]);
166         s[i] = s[i-1] + c[i];
167     }
168     init(301000);
169     build(1, 1, n);
170     for (int i = 1; i <= k; i++) {
171         int op, l, r;
172         scanf("%d%d%d", &op, &l, &r);
173         if (op == 1)
174             update(1, 1, n, l, r, 1, 1);
175         else {
176             LL ans = (bas*query(1, 1, n, l, r)%MOD+s[r]-s[l-1]) % MOD;
177             if (ans < 0) ans += MOD;
178             printf("%I64d\n", ans);
179         }
180     }
181     return 0;
182 }
183

```

---

### 6.9.8 7 区间加 + 区间乘.cpp

---

```

1 //洛谷 P3373
2 const int maxn = 100000+10;
3 LL n,m,mod;
4 LL sumv[maxn<<2],addv[maxn<<2],mulv[maxn<<2];
5 LL a[maxn];
6 #define lc (o<<1)
7 #define rc (o<<1|1)
8 void maintain(int o,int l,int r){
9     sumv[o] = sumv[lc]+sumv[rc];
10    sumv[o] %= mod;

```

```

11 }
12 void pushdown(int o,int l,int r){
13     int m = (l+r)>>1;
14     if(mulv[o] != 1){
15         sumv[lc] = sumv[lc]*mulv[o]%mod,sumv[rc] = sumv[rc]* mulv[o]%mod;
16         addv[lc] = addv[lc] *mulv[o]%mod,addv[rc] = addv[rc] *
            ↪ mulv[o]%mod;
17         mulv[lc] = (mulv[lc]*mulv[o])%mod,mulv[rc] = (mulv[rc]*
            ↪ mulv[o]%mod);
18         mulv[o] = 1;
19     }
20     if(addv[o]){
21         sumv[lc] = (sumv[lc]+addv[o]*(m-l+1))%mod;
22         addv[lc] = (addv[lc]+addv[o])%mod;
23         sumv[rc] = (sumv[rc]+addv[o]*(r-m))%mod;
24         addv[rc] = (addv[rc]+addv[o])%mod;
25         addv[o] = 0;
26     }
27 }
28
29
30 void build(int o,int l,int r){
31
32     if(l == r){
33         sumv[o] = a[l];
34         addv[o] = 0;
35         mulv[o] = 1;
36         return ;
37     }
38     int m = (l+r)>>1;
39     build(lc,l,m);
40     build(rc,m+1,r);
41     // sumv[o] =
42     addv[o] = 0,mulv[o] = 1;
43     maintain(o,l,r);
44 }
45 int op;
46 void update(int o,int l,int r,int L,int R,LL v){
47     if(L <= l &&R >= r){
48         if(op == 2){
49             sumv[o] = (sumv[o]+v*(r-l+1))%mod;
50             addv[o] += v;
51         }
52         else{
53             sumv[o] = (sumv[o]*v)%mod;
54             addv[o] = (addv[o]*v)%mod;
55             mulv[o] = (mulv[o]*v)%mod;
56         }
57     }
58     else{
59         int m = (l+r)>>1;
60         pushdown(o,l,r);
61         if(L <= m)
62             update(lc,l,m,L,R,v);
63         if(R > m)
64             update(rc,m+1,r,L,R,v);

```

```

65         maintain(o,l,r);
66     }
67
68 }
69 LL _sum;
70 void query(int o,int l,int r,int L,int R){
71     if(L <= l && R >= r){
72         _sum += sumv[o];
73         _sum %= mod;
74         return ;
75     }
76     pushdown(o,l,r);
77     int m = (l+r)>>1;
78     if(L <= m)
79         query(lc,l,m,L,R);
80     if(R > m)
81         query(rc,m+1,r,L,R);
82     // pushup()
83 }
84
85
86
87 int main(void){
88     cin>>n>>m>>mod;
89     for(int i = 1;i <= n; ++i)
90         scanf("%lld",&a[i]);
91     build(1,1,n);
92     // _sum = 0;
93     // query(1,1,n,1,n);
94     // cout<<_sum<<endl;
95     for(int i = 1;i <= m; ++i){
96         int x,y,v;
97         scanf("%d%d%d",&op,&x,&y);
98         if(op == 1 || op == 2){
99             scanf("%d",&v);
100             update(1,1,n,x,y,v);
101         }
102         else{
103             _sum = 0;
104             query(1,1,n,x,y);
105             _sum %= mod;
106             printf("%lld\n",_sum);
107         }
108     }
109
110
111     return 0;
112 }

```

---

## 7 模拟

### 7.1 1 日期.cpp

```
1 1 计算日期差
2
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 bool isLeapYear(int year)
7 {
8     return ((year%4==0 && year%100!=0) || year%400==0);
9 }
10 // 以公元 1 年 1 月 1 日为基准, 计算经过的日期
11 int getDays(int year, int month, int day)
12 {
13     int m[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
14     if(isLeapYear(year))
15         m[2]++;
16     int result = 0;
17     for(int i = 1; i < year; i++)
18     {
19         result += 365;
20         if(isLeapYear(i))
21             result ++;
22     }
23     for(int i = 1; i < month; i++)
24     {
25         result += m[i];
26     }
27     result += day;
28
29     return result;
30 }
31 int dayDis (int year1, int month1, int day1,
32             int year2, int month2, int day2)
33 {
34     return abs(getDays(year2, month2, day2) - getDays(year1, month1, day1));
35 }
36
37 int main(void)
38 {
39     printf("%d\n", dayDis(2012, 9, 1, 2018, 3, 25));
40
41     return 0;
42 }
43 2 计算某一天星期几
44 int cal1(int y, int m, int d)
45 {
46     if(m==1 || m==2)
47         m+=12, y--;
48     int w=(d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
49     return ++w;
50 }
51 int cal2(int y, int m, int d)
```

```

52 {
53     if(m==1 || m==2)
54         m+=12,y--;
55     int c=y/100,ty=y%100;
56     int w=ty+ty/4+c/4-2*c+26*(m+1)/10+d-1;
57     return w%7==0?7:(w+7)%7;
58 }
59 3 计算从2000 01 01 到9999 12 31 之间任意日期之间日期表示有多少个9
60 #include<bits/stdc++.h>
61
62 using namespace std;
63
64
65 int year,month,day;
66 int a1,b1,c1,a2,b2,c2;
67
68 const int maxn = 1e4+100;
69 int a[maxn];
70 int c[maxn]; // 代表当前年所有的 9
71 // int mon[30] = {0,2,2,2,}
72 int run(int y){
73     return y%400 == 0 || (y%4==0&&y%100!=0);
74 }
75 int wanyue(int t,int y){
76     if(t == 2) return 2+run(y);
77     if(t == 9) return 3+30;
78     return 3;
79 }
80 int wanyear(int t){
81     int num = 0;
82     int tt = t;
83     while(tt > 0){
84         if(tt % 10 == 9) num++;
85         tt /= 10;
86     }
87     a[t] = num;
88     int tmp = run(t);
89     return num*(365+tmp)+65+tmp;
90 }
91 int mo[20] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
92 int Howmuchday(int y,int t){
93     if(t==2){
94         return run(y)+28;
95     }
96     return mo[t];
97 }
98 int subday(int a,int b){
99     int sum = 0;
100     for(int i = a;i <= b; ++i)
101         if(i%10 == 9)
102             sum++;
103     return sum;
104 }
105 int numsubday(int a,int b){
106     return b-a+1;

```



```

107 }
108
109 int numsubday(int y,int b1,int c1,int b2,int c2){
110     int num = 0;
111     if(b1 == b2)
112         return numsubday(c1,c2);
113     for(int i = b1+1;i < b2; ++i)
114         num += mo[i]+(i==2&&run(y));
115     num += numsubday(c1,Howmuchday(y,b1));
116     num += numsubday(1,c2);
117     return num;
118 }
119 int FF(int t){
120     int num = 0;
121     int tt = t;
122     while(tt > 0){
123         if(tt % 10 == 9) num++;
124         tt /= 10;
125     }
126     return num;
127 }
128 int submonth(int y,int b1,int c1,int b2,int c2){
129     if(b1 == b2)
130         return subday(c1,c2)+(c2-c1+1)*FF(b1);
131     int sum = 0;
132     for(int i = b1+1;i < b2; ++i)
133         sum += wanyue(i,y);
134
135     sum += subday(c1,Howmuchday(y,b1))+FF(b1)*(Howmuchday(y,b1)-c1+1);
136     // cout<<sum<<endl;
137     sum += subday(1,c2)+FF(b2)*(c2);
138     return sum;
139 }
140
141 int subyear(int a1,int b1,int c1,int a2,int b2,int c2){
142     if(a1 == a2)
143         return numsubday(a1,b1,c1,b2,c2)*a[a1] + submonth(a1,b1,c1,b2,c2);
144     int ans = 0;
145     ans += c[a2-1]-c[a1];
146     ans += numsubday(a1,b1,c1,12,31)*a[a1];
147     ans += numsubday(a2,1,1,b2,c2)*a[a2];
148     return ans + submonth(a1,b1,c1,12,31)+submonth(a2,1,1,b2,c2);
149 }
150
151 int main(void){
152
153     for(int i = 2000;i < maxn; ++i){
154         c[i] = wanyear(i);
155         c[i] += c[i-1];
156     }
157     int T;
158     cin>>T;
159     while(T--){
160         scanf("%d%d%d %d%d%d",&a1,&b1,&c1,&a2,&b2,&c2);
161         int ans = subyear(a1,b1,c1,a2,b2,c2);
162         printf("%d\n",ans);

```

```

163     }
164     return 0;
165 }
166 // 同上
167 #include <stdio.h>
168 #include <string.h>
169
170 int sum[10005][15][35],pre[10005][15][35];
171 int mon[15] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
172
173
174 int leap(int x)
175 {
176     if (x % 400 == 0) return 1;
177     if (x % 100 == 0) return 0;
178     if (x % 4 == 0) return 1;
179
180     return 0;
181 }
182
183 int check(int y,int m,int d)
184 {
185     int num = 0;
186
187     while (y)
188     {
189         y % 10 == 9 ? ++num : num += 0;
190         y /= 10;
191     }
192
193     while (m)
194     {
195         m % 10 == 9 ? ++num : num += 0;
196         m /= 10;
197     }
198
199     while (d)
200     {
201         d % 10 == 9 ? ++num : num += 0;
202         d /= 10;
203     }
204
205     return num;
206 }
207
208 void init(int y1,int m1,int d1,int y2,int m2,int d2)
209 {
210     int tmp = 0;
211
212
213     while (y1 != y2 || m1 != m2 || d1 != d2)
214     {
215         mon[2] = leap(y1) + 28;
216
217         pre[y1][m1][d1] = tmp;//tmp 是到前一个日期显示的 9 的数量。
218

```

```

219         tmp += check(y1,m1,d1);
220
221         sum[y1][m1][d1] = tmp;//现在的日期显示的 9 的数量
222
223         if (++d1 > mon[m1])
224         {
225             d1 = 1;
226
227             if (++m1 > 12)
228             {
229                 m1 = 1;
230                 mon[2] = 28 + leap(++y1);
231             }
232         }
233     }
234 }
235
236 int main()
237 {
238     int t;
239
240     scanf("%d",&t);
241
242     init(2000,1,1,10000,1,1);
243
244     while (t--)
245     {
246         int y1,m1,d1,y2,m2,d2;
247
248         scanf("%d%d%d%d%d%d",&y1,&m1,&d1,&y2,&m2,&d2);
249
250         printf("%d\n",sum[y2][m2][d2] - pre[y1][m1][d1]); //结束日期减去开始日期之前
        ↪ 的那天，因为开始日期也要算的。
251     }
252
253     return 0;
254 }

```

---