

Algorithm Codelet

TieWay59

July 3, 2019

Contents


1 其它	5
1.1 c++ 中处理 2 进制的一些函数.cpp	5
1.2 IO	6
1.2.1 fread.cpp	6
1.2.2 fread2.cpp	7
1.2.3 保留小数.cpp	9
1.2.4 读取整数.cpp	9
1.3 测量程序的运行时间.cpp	9
1.4 转化成二进制.cpp	9
2 几何	10
2.1 2D	10
2.1.1 8 旋转卡壳.cpp	10
2.1.2 PSLG.cpp	12
2.1.3 二维几何模板.cpp	14
2.1.4 二维凸包.cpp	17
2.1.5 判断点是否在多边形内.cpp	17
2.1.6 圆与多边形相交的面积.cpp	18
2.1.7 求圆与直线的交点.cpp	22
2.2 3D	22
2.2.1 三维几何的基本操作.cpp	22
2.2.2 三维几何的模版.cpp	23
2.2.3 三维凸包.cpp	26
2.2.4 维度转换为三维坐标.cpp	28
3 动态规划	28
3.1 1 单调队列.cpp	28
3.2 1 最长上升子序列.cpp	29
3.3 string dp	29
3.3.1 trie+dp.cpp	29
3.4 zhuangyadp	31
3.4.1 1 多米诺骨牌覆盖.cpp	31
3.5 树上的分治	35
3.5.1 1 树的重心.cpp	35
4 图论	36
4.1 DFS	36
4.1.1 1. 无向图的割点和桥.cpp	36
4.1.2 2. 无向图的双连通分量.cpp	39
4.1.3 3 有向图的强联通分量.cpp	40
4.1.4 4 2-sat 问题.cpp	41
4.2 LCA	42
4.2.1 1 DFS+RMQ.cpp	42
4.2.2 2 倍增算法.cpp	43
4.3 Maxflow	45
4.3.1 1 Dinic.cpp	45
4.3.2 2 ISAP.cpp	48
4.3.3 3 MCMF.cpp	50
4.4 二分图	52
4.4.1 1 匈牙利算法.cpp	52
4.4.2 2 KM.cpp	54
4.4.3 3 一般图最大匹配.cpp	55
4.5 最小生成树	57
4.5.1 1 Kruskal 卡鲁斯卡尔算法.cpp	57

4.5.2	2 prim 算法.cpp	58
4.5.3	3 最小限制生成树.cpp	59
4.5.4	4 次小生成树.cpp	62
4.6	最短路	65
4.6.1	1 Dijkstra.cpp	65
4.6.2	2 Bellman-ford.cpp	68
4.6.3	3 floyed.cpp	70
4.6.4	堆优化的有限队列.cpp	71
5	数学	72
5.1	3 FWT 模板.cpp	72
5.2	4 单纯形法.cpp	73
5.3	5. 线性基.cpp	75
5.4	BM.cpp	76
5.5	Combinatorial mathematics	77
5.5.1	康托展开.cpp	77
5.6	FFT	78
5.6.1	FFT.cpp	78
5.6.2	kuangbin.cpp	79
5.6.3	lrj.cpp	81
5.7	Lagrange-poly	83
5.7.1	template.cpp	83
5.8	三分.cpp	84
5.9	博弈	85
5.9.1	2. 威佐夫博弈.cpp	85
5.9.2	3 Nim 积.cpp	86
5.9.3	4 K 倍动态减法.cpp	87
5.9.4	5 海盗分金问题.cpp	88
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	93
5.10	数论	94
5.10.1	1 加法.cpp	94
5.10.2	1 逆元.cpp	94
5.10.3	2 减法.cpp	95
5.10.4	3 乘法.cpp	96
5.10.5	4 除法.cpp	96
5.10.6	5. 蒙哥马利快速模.cpp	97
5.10.7	Euler.cpp	98
5.10.8	lucas , 组合数.cpp	99
5.10.9	miller-rabin-Pollard-rho.cpp	100
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

6	数据结构	115
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	130
6.5	基础数据结构	133
6.5.1	堆.cpp	133
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	137
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	142
6.8.1	1 树状数组模板.cpp	142
6.8.2	2 区间出现两次的数的个数.cpp	143
6.9	线段树	144
6.9.1	1. 区间更新区间查询.cpp	144
6.9.2	2 主席树求第 k 大.cpp	146
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	156
6.9.7	6 区间加斐波那契数.cpp	158
6.9.8	7 区间加 + 区间乘.cpp	162
7	模拟	164
7.1	1 日期.cpp	164


1 其它

1.1 c++ 中处理 2 进制的一些函数.cpp

 Built-in Function: `int __builtin_ffs (unsigned int x)`


Returns one plus the index of the least significant 1-bit of x, or if x is zero,
↪ returns zero.

返回右起第一个 '1' 的位置。

 Built-in Function: `int __builtin_clz (unsigned int x)`


Returns the number of leading 0-bits in x, starting at the most significant bit
↪ position. If x is 0, the result is undefined.

返回左起第一个 '1' 之前 0 的个数。

 Built-in Function: `int __builtin_ctz (unsigned int x)`


Returns the number of trailing 0-bits in x, starting at the least significant bit
↪ position. If x is 0, the result is undefined.

返回右起第一个 '1' 之后的 0 的个数。

 Built-in Function: `int __builtin_popcount (unsigned int x)`


Returns the number of 1-bits in x.

返回 '1' 的个数。


 Built-in Function: `int __builtin_parity (unsigned int x)`

Returns the parity of x, i.e. the number of 1-bits in x modulo 2.


返回 '1' 的个数的奇偶性。

 Built-in Function: `int __builtin_ffsl (unsigned long)`


Similar to `__builtin_ffs`, except the argument type is `unsigned long`.

 Built-in Function: `int __builtin_clzl (unsigned long)`


Similar to `__builtin_clz`, except the argument type is `unsigned long`.

 Built-in Function: `int __builtin_ctzl (unsigned long)`


Similar to `__builtin_ctz`, except the argument type is `unsigned long`.

 Built-in Function: `int __builtin_popcountl (unsigned long)`

Similar to `__builtin_popcount`, except the argument type is `unsigned long`.

 Built-in Function: `int __builtin_parityl (unsigned long)`

Similar to `__builtin_parity`, except the argument type is `unsigned long`.

 Built-in Function: `int __builtin_ffsll (unsigned long long)`

Similar to `__builtin_ffs`, except the argument type is `unsigned long long`.

☐ Built-in Function: `int __builtin_clzll (unsigned long long)`

Similar to `__builtin_clz`, except the argument type is `unsigned long long`.

☐ Built-in Function: `int __builtin_ctzll (unsigned long long)`

Similar to `__builtin_ctz`, except the argument type is `unsigned long long`.

☐ Built-in Function: `int __builtin_popcountll (unsigned long long)`

Similar to `__builtin_popcount`, except the argument type is `unsigned long long`.

☐ Built-in Function: `int __builtin_parityll (unsigned long long)`

Similar to `__builtin_parity`, except the argument type is `unsigned long long`.

1.2 IO

1.2.1 fread.cpp

```
namespace io {
    const int L = 1 << 20 | 1;
    char ibuf[L], *iS, *iT, c, obuf[L], *oS = obuf, *oT = obuf + L - 1, qu[55]; int f,
    ↪ qr;
    #ifdef whzzt
        #define gc() getchar()
    #else
        #define gc() (iS == iT ? (iT = (iS = ibuf) + fread (ibuf, 1, L, stdin), iS == iT ?
    ↪ EOF : *iS++) : *iS++)
    #endif
    template <class I>
    inline void gi (I &x) {
        for (f = 1, c = gc(); c < '0' || c > '9'; c = gc()) if (c == '-') f = -1;
        for (x = 0; c <= '9' && c >= '0'; c = gc()) x = x * 10 + (c & 15); x *= f;
    }
    inline void flush () {
        fwrite (obuf, 1, oS - obuf, stdout);
    }
    inline void putc (char x) {
        *oS++ = x;
        if (oS == oT) flush (), oS = obuf;
    }
    template <class I>
    void print (I x) {
        if (!x) putc ('0'); if (x < 0) putc ('-'), x = -x;
        while (x) qu[++qr] = x % 10 + '0', x /= 10;
        while (qr) putc (qu[qr--]);
    }
    struct io_ff { ~io_ff() { flush(); } } _io_ff_;
}
using io :: gi;
using io :: putc;
using io :: print;
```

1.2.2 fread2.cpp

```
namespace IO{
#define BUF_SIZE 100000
#define OUT_SIZE 100000
#define ll long long
//fread->read

bool IOerror=0;
inline char nc(){
    static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=buf+BUF_SIZE;
    if (p1==pend){
        p1=buf; pend=buf+fread(buf,1,BUF_SIZE,stdin);
        if (pend==p1){IOerror=1;return -1;}
        //{printf("IO error!\n");system("pause");for (;;);exit(0);}
    }
    return *p1++;
}
inline bool blank(char ch){return ch==' '||ch=='\n'||ch=='\r'||ch=='\t';}
inline void read(int &x){
    bool sign=0; char ch=nc(); x=0;
    for (;blank(ch);ch=nc());
    if (IOerror)return;
    if (ch=='-')sign=1,ch=nc();
    for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
    if (sign)x=-x;
}
inline void read(ll &x){
    bool sign=0; char ch=nc(); x=0;
    for (;blank(ch);ch=nc());
    if (IOerror)return;
    if (ch=='-')sign=1,ch=nc();
    for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
    if (sign)x=-x;
}
inline void read(double &x){
    bool sign=0; char ch=nc(); x=0;
    for (;blank(ch);ch=nc());
    if (IOerror)return;
    if (ch=='-')sign=1,ch=nc();
    for (;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
    if (ch=='.'){
        double tmp=1; ch=nc();
        for (;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(ch-'0');
    }
    if (sign)x=-x;
}
inline void read(char *s){
    char ch=nc();
    for (;blank(ch);ch=nc());
    if (IOerror)return;
    for (;!blank(ch)&&!IOerror;ch=nc())*s++=ch;
    *s=0;
}
inline void read(char &c){
    for (c=nc();blank(c);c=nc());
}
```

```

    if (IOerror){c=-1;return;}
}
//fwrite->write
struct Ostream_fwrite{
    char *buf,*p1,*pend;
    Ostream_fwrite(){buf=new char[BUF_SIZE];p1=buf;pend=buf+BUF_SIZE;}
    void out(char ch){
        if (p1==pend){
            fwrite(buf,1,BUF_SIZE,stdout);p1=buf;
        }
        *p1++=ch;
    }
    void print(int x){
        static char s[15],*s1;s1=s;
        if (!x)*s1++='0';if (x<0)out('-'),x=-x;
        while(x)*s1++=x%10+'0',x/=10;
        while(s1--!=s)out(*s1);
    }
    void println(int x){
        static char s[15],*s1;s1=s;
        if (!x)*s1++='0';if (x<0)out('-'),x=-x;
        while(x)*s1++=x%10+'0',x/=10;
        while(s1--!=s)out(*s1); out('\n');
    }
    void print(ll x){
        static char s[25],*s1;s1=s;
        if (!x)*s1++='0';if (x<0)out('-'),x=-x;
        while(x)*s1++=x%10+'0',x/=10;
        while(s1--!=s)out(*s1);
    }
    void println(ll x){
        static char s[25],*s1;s1=s;
        if (!x)*s1++='0';if (x<0)out('-'),x=-x;
        while(x)*s1++=x%10+'0',x/=10;
        while(s1--!=s)out(*s1); out('\n');
    }
    void print(double x,int y){
        static ll mul[]={1,10,100,1000,10000,100000,1000000,10000000,100000000,
            ↪ 1000000000,10000000000LL,100000000000LL,1000000000000LL,10000000000000LL,
            ↪ 100000000000000LL,1000000000000000LL,10000000000000000LL,100000000000000000LL,
            if (x<-1e-12)out('-'),x=-x;x*=mul[y];
            ll x1=(ll)floor(x); if (x-floor(x)>=0.5)++x1;
            ll x2=x1/mul[y],x3=x1-x2*mul[y]; print(x2);
            if (y>0){out('.'); for (size_t i=1;i<y&& x3*mul[i]<mul[y];out('0'),++i);
            ↪ print(x3);}
        }
    void println(double x,int y){print(x,y);out('\n');}
    void print(char *s){while (*s)out(*s++);}
    void println(char *s){while (*s)out(*s++);out('\n');}
    void flush(){if (p1!=buf){fwrite(buf,1,p1-buf,stdout);p1=buf;}}
    ~Ostream_fwrite(){flush();}
}Ostream;
inline void print(int x){Ostream.print(x);}
inline void println(int x){Ostream.println(x);}

```



```

inline void print(char x){Ostream.out(x);}
inline void println(char x){Ostream.out(x);Ostream.out('\n');}
inline void print(ll x){Ostream.print(x);}
inline void println(ll x){Ostream.println(x);}
inline void print(double x,int y){Ostream.print(x,y);}
inline void println(double x,int y){Ostream.println(x,y);}
inline void print(char *s){Ostream.print(s);}
inline void println(char *s){Ostream.println(s);}
inline void println(){Ostream.out('\n');}
inline void flush(){Ostream.flush();}
#undef ll
#undef OUT_SIZE
#undef BUF_SIZE
};

```

1.2.3 保留小数.cpp

```

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

```

1.2.4 读取整数.cpp

```

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

```

1.3 测量程序的运行时间.cpp

```

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

```

1.4 转化成二进制.cpp

```

void To_string_base2(LL n,string &s){
    while(n > 0){

```

```

    if(n&1)
        s += "1";
    else
        s += "0";
    n >>= 1;
};
reverse(s.begin(),s.end());
}
// nn 是要转化的数, ss 是 string, n 转化成多少位 2 进制
void To_string_base2_n(LL nn,string &ss,int n){
    ss.clear();
    To_string_base2(nn,ss);
    while((int)ss.size() < n)
        ss = "0"+ss;
}

```

2 几何

2.1 2D

2.1.1 8 旋转卡壳.cpp

```

//2017-2018 ACM-ICPC Southwestern European Regional Programming Contest (SWERC 2017)
//K      Blowing Candles
// 求包含所有点的两条平行线之间的最短距离
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#define INF 9999999999999.9
#define PI acos(-1.0)
struct Point
{
    double x, y, dis;
}pt[200005], stack[200005], p0;
int top, tot;
//计算几何距离
double Dis(double x1, double y1, double x2, double y2)
{
    return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}
//极角比较, 返回-1: p0p1 在 p0p2 的右侧, 返回 0:p0,p1,p2 共线
int Cmp_PolarAngel(struct Point p1, struct Point p2, struct Point pb)
{
    double delta=(p1.x-pb.x)*(p2.y-pb.y)-(p2.x-pb.x)*(p1.y-pb.y);
    if (delta<0.0) return 1;
    else if (delta==0.0) return 0;
    else return -1;
}
// 判断向量 p2p3 是否对 p1p2 构成左旋
bool Is_LeftTurn(struct Point p3, struct Point p2, struct Point p1)
{
    int type=Cmp_PolarAngel(p3, p1, p2);
    if (type<0) return true;
    return false;
}

```

//先按极角排，再按距离由小到大排

```
int Cmp(const void*p1, const void*p2)
{
    struct Point*a1=(struct Point*)p1;
    struct Point*a2=(struct Point*)p2;
    int type=Cmp_PolarAngel(*a1, *a2, p0);
    if (type<0) return -1;
    else if (type==0)
    {
        if (a1->dis<a2->dis) return -1;
        else if (a1->dis==a2->dis) return 0;
        else return 1;
    }
    else return 1;
}
```

//求凸包

```
void Hull(int n)
{
    int i, k;
    p0.x=p0.y=INF;
    for (i=0;i<n;i++)
    {
        scanf("%lf %lf",&pt[i].x, &pt[i].y);
        if (pt[i].y < p0.y)
        {
            p0.y=pt[i].y;
            p0.x=pt[i].x;
            k=i;
        }
        else if (pt[i].y==p0.y)
        {
            if (pt[i].x<p0.x)
            {
                p0.x=pt[i].x;
                k=i;
            }
        }
    }
    pt[k]=pt[0];
    pt[0]=p0;
    for (i=1;i<n;i++)
        pt[i].dis=Dis(pt[i].x,pt[i].y, p0.x,p0.y);
    qsort(pt+1, n-1, sizeof(struct Point), Cmp);
    //去掉极角相同的点
    tot=1;
    for (i=2;i<n;i++)
        if (Cmp_PolarAngel(pt[i], pt[i-1], p0))
            pt[tot++]=pt[i-1];
    pt[tot++]=pt[n-1];
    //求凸包
    top=1;
    stack[0]=pt[0];
    stack[1]=pt[1];
    for (i=2;i<tot;i++)
    {
        while (top>=1 && Is_LeftTurn(pt[i], stack[top], stack[top-1])==false)
```

```

        top--;
        stack[++top]=pt[i];
    }
}
//计算叉积
double CrossProduct(struct Point p1, struct Point p2, struct Point p3)
{
    return (p1.x-p3.x)*(p2.y-p3.y)-(p2.x-p3.x)*(p1.y-p3.y);
}
//卡壳旋转，求出凸多边形所有对踵点
double hl(double a,double b,double c)
{
    double p=(a+b+c)/2.0;
    return sqrt(p*(p-a)*(p-b)*(p-c));
}
double dist(Point a,Point b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
void Rotate(struct Point*ch, int n)
{
    int i, p=1;
    double t1, t2, ans=INF, dif;
    ch[n]=ch[0];
    for (i=0;i<n;i++)
    {
        //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
        while (fabs(CrossProduct(ch[i],ch[i+1],ch[p+1])) >
            ↪ fabs(CrossProduct(ch[i],ch[i+1],ch[p])))
            p=(p+1)%n;
        dif=fabs(CrossProduct(ch[i],ch[i+1],ch[p+1])) -
            ↪ fabs(CrossProduct(ch[i],ch[i+1],ch[p]));
        //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，对角线两端
        ↪ 都可能是对踵点

        ↪ t1=hl(dist(ch[i],ch[i+1]),dist(ch[i+1],ch[p]),dist(ch[p],ch[i]))*2.0/dist(ch[i],ch[
        //printf(">>%lf\n",dist(ch[i],ch[i+1]));
        if (t1<ans)ans=t1;
    }
    printf("%.15lf\n",ans);
}
int main (void)
{
    int n;
    scanf("%d",&n);
    Hull(n);
    Rotate(stack, top+1);
    return 0;
}

```

2.1.2 PSLG.cpp

```

typedef vector<Point> Polygon;
double PolygonArea(Polygon poly)
{
    double area = 0;

```

```

    int n = poly.size();
    for(int i = 1; i < n-1; i++)
        area += Cross(poly[i]-poly[0], poly[(i+1)%n]-poly[0]);
    return area/2;
}

struct Edge
{
    int from, to; // 起点, 终点, 左边的面编号
    double ang;
    Edge(int f, int t, double a):from(f), to(t), ang(a) {}
};

const int maxn = 10000 + 10; // 最大边数

// 平面直线图 (PSGL) 实现
struct PSGL
{
    int n, m, face_cnt; // face_cnt 面数
    double x[maxn], y[maxn];
    vector<Edge> edges; // 储存边
    vector<int> G[maxn]; // 指向边
    int vis[maxn*2]; // 每条边是否已经访问过
    int left[maxn*2]; // 左面的编号
    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

    vector<Polygon> faces; // faces 储存面
    double area[maxn]; // 每个 polygon 的面积

    void init(int n)
    {
        this->n = n;
        for(int i = 0; i < n; i++)
            G[i].clear();
        edges.clear();
        faces.clear();
    }

    // 有向线段 from->to 的极角
    double getAngle(int from, int to)
    {
        return atan2(y[to]-y[from], x[to]-x[from]);
    }

    void AddEdge(int from, int to)
    {
        edges.push_back((Edge){ from, to, getAngle(from, to)});
        edges.push_back((Edge){ to, from, getAngle(to, from)});
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    // 找出 faces 并计算面积
    void Build()
    {

```

```

for(int u = 0; u < n; u++)
{
    // 给从 u 出发的各条边按极角排序
    int d = G[u].size();
    for(int i = 0; i < d; i++)
        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
            if(edges[G[u][i]].ang > edges[G[u][j]].ang)
                swap(G[u][i], G[u][j]);
    for(int i = 0; i < d; i++)
        prev[G[u][(i+1)%d]] = G[u][i];
}

memset(vis, 0, sizeof(vis));
face_cnt = 0;
for(int u = 0; u < n; u++)
    for(int i = 0; i < G[u].size(); i++)
    {
        int e = G[u][i];
        if(!vis[e]) // 逆时针找圈
        {
            face_cnt++;
            Polygon poly;
            for(;;)
            {
                vis[e] = 1;
                left[e] = face_cnt;
                int from = edges[e].from;
                poly.push_back(Point(x[from], y[from]));
                e = prev[e^1];
                if(e == G[u][i])
                    break;
                assert(vis[e] == 0);
            }
            faces.push_back(poly);
        }
    }

for(int i = 0; i < faces.size(); i++)
{
    area[i] = PolygonArea(faces[i]);
}
}
};

```

2.1.3 二维几何模板.cpp

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define forn(i,n) for(int i = 0; i < n; ++i)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;

```

```

const int    INF = 0x7FFFFFFF;
const LL     INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-10;
const LL     mod = 1e9 + 7;
struct Point
{
    double x,y;

    Point(double x = 0,double y = 0):x(x),y(y) {}

};
typedef Point Vector;
Vector operator + (Vector A,Vector B)
{
    return Vector(A.x + B.x,A.y + B.y);
}
Vector operator - (Vector A,Vector B)
{
    return Vector(A.x-B.x,A.y-B.y);
}
Vector operator / (Vector A,double p)
{
    return Vector(A.x/p,A.y/p);
}
Vector operator * (Vector A,double p)
{
    return Vector(A.x*p,A.y*p);
}
double angle(Vector v)//求向量的角度从 0 到 2*pi
{
    return atan2(v.y,v.x);
}
int dcmp(double x)
{
    if(fabs(x)<eps)
        return 0;
    else
        return x < 0?-1:1;
}
bool operator < (const Point &a,const Point &b)
{
    if(dcmp(a.x-b.x)==0)
        return a.y<b.y;
    else
        return a.x<b.x;
}

bool operator == (const Point &a,const Point &b)
{
    return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y);
}
double Dot(Vector A,Vector B)
{

```

```

    return A.x*B.x+A.y*B.y;
}
double Length(Vector A)
{
    return sqrt(A.x*A.x+A.y*A.y);
}
double Angle(Vector A,Vector B)
{
    return acos(Dot(A,B)/Length(A)/Length(B));
}
double Cross(Vector A,Vector B)
{
    return A.x*B.y - A.y*B.x;
}
double Area2(Point A,Point B,Point C)
{
    return Cross(B-A,C-A);
}
Vector Rotate(Vector A,double rad)
{
    return Vector (A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
}
Vector Normal(Vector A)//单位法线
{
    double L = Length(A);
    return Vector(-A.y/L,A.x/L);
}
//调用前确保直线有唯一交点, 当且仅当  $Cross(v,w)$  非 0
Point Get_Line_Intersection(Point P,Vector v,Point Q,Vector w)
{
    Vector u = P - Q;
    double t = Cross(w,u)/Cross(v,w);
    return P+v*t;
}
double Distance_To_Line(Point P,Point A,Point B)//点到直线的距离
{
    Vector v1 = B-A,v2 = P-A;
    return fabs(Cross(v1,v2)/Length(v1));
}
double Distance_To_Segment(Point P,Point A,Point B)
{
    if(A==B)
        return Length(P-A);
    Vector v1 = B-A,v2 = P-A,v3 = P-B;
    if(dcmp(Dot(v1,v2))<0)
        return Length(v1);
    else if(dcmp(Dot(v1,v3))>0)
        return Length(v3);
    else
        return fabs(Cross(v1,v2))/Length(v1);
}
Point Get_Line_Projection(Point P,Point A,Point B)//求投影点
{
    Vector v = B - A;
    return A + v*(Dot(v,P-A)/Dot(v,v));
}

```


//线段相交判定 相交不在线段的端点

```
bool Segment_Proper_Intersection(Point a1,Point a2,Point b1,Point b2)
{
    double c1 = Cross(a2-a1,b1-a1),c2 = Cross(a2-a1,b2-a1),
           c3 = Cross(b2-b1,a2-b1),c4 = Cross(b2-b1,a1-b1);
    return dcmp(c1)*dcmp(c2)<0&&dcmp(c3)*dcmp(c4)<0;
}
//判断点是否在线段上 (不包括端点)
bool Onsegment(Point p,Point a1,Point a2)
{
    return dcmp(Cross(a1-p,a2-p))==0&&dcmp(Dot(a1-p,a2-p))<0;
}
```

2.1.4 二维凸包.cpp

//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

//输入不能有重复节点

//如果精度要求搞需要用 $dcmp$ 判断

//如果不希望在边上右点, 需要将 \leq 改为 $<$

```
int ConvexHull(Point *p,int n ,Point *ch)
{
    sort(p,p+n);
    int m = 0;
    for(int i = 0;i < n; ++i)
    {
        while(m>1&& Cross(ch[m-1]-ch[m-2],p[i]-ch[m-2])<=0) m--;
        ch[m++] = p[i];
    }
    int k = m;
    for(int i = n-2; i >= 0; --i)
    {
        while(m > k&& Cross(ch[m-1]-ch[m-2],p[i]-ch[m-2]) <= 0) m--;
        ch[m++] = p[i];
    }
    if(n > 1) m--;
    return m;
}
```

2.1.5 判断点是否在多边形内.cpp

typedef vector<Point> Polygon;

int isPointInPolygon(Point p,Polygon poly)

```
{
    int n = poly.size();
    int wn = 0;
    for(int i = 0;i < n; ++i)
    {
        if(Onsegment(p,poly[i],poly[(i+1)%n])) return -1;
        int k = dcmp(Cross(poly[(i+1)%n]-poly[i],p-poly[i]));
        int d1 = dcmp(poly[i].y-p.y);
        int d2 = dcmp(poly[(i+1)%n].y-p.y);
        if(k>0&&d1 <= 0&&d2 > 0) wn ++;
        if(k<0&&d2 <= 0&&d1 > 0) wn --;
    }
    if(wn != 0) return 1;
}
```

```

    return 0;
}

```

2.1.6 圆与多边形相交的面积.cpp

```

#include <iostream>
#include <cstdio>
#include <string>
#include <cmath>
#include <iomanip>
#include <ctime>
#include <climits>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <queue>
#include <vector>
#include <set>
#include <map>
using namespace std;
typedef unsigned int UI;
typedef long long LL;
typedef unsigned long long ULL;
typedef long double LD;
const double pi = acos(-1.0);
const double e = exp(1.0);
const double eps = 1e-8;
const int maxn = 400;
double x, y, h;
double vx, vy;
double R;
int n;
struct point
{
    double x, y;
    point(double _x=0.0, double _y=0.0)
        : x(_x), y(_y) {}
    point operator - (const point & p)
    {
        return point(x-p.x, y-p.y);
    }
    double sqrx()
    {
        return sqrt(x*x+y*y);
    }
} p[maxn];

double xmult(point & p1, point & p2, point & p0);
double distancex(point & p1, point & p2);
point intersection(point u1, point u2, point v1, point v2);
void intersection_line_circle(point c, double r, point l1, point l2, point & p1, point
↪ & p2);
point ptoseg(point p, point l1, point l2);
double distp(point & a, point & b);
double Direct_Triangle_Circle_Area(point a, point b, point o, double r);

```

```

double xmult(point & p1, point & p2, point & p0)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p1.y-p0.y)*(p2.x-p0.x);
}

double distancex(point & p1, point & p2)
{
    return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}

point intersection(point u1, point u2, point v1, point v2)
{
    point ret = u1;
    double t = ((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
               / ((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
    ret.x += (u2.x-u1.x)*t;
    ret.y += (u2.y-u1.y)*t;
    return ret;
}

void intersection_line_circle(point c, double r, point l1, point l2, point & p1, point
↪ & p2)
{
    point p = c;
    double t;
    p.x += l1.y-l2.y;
    p.y += l2.x-l1.x;
    p = intersection(p, c, l1, l2);
    t = sqrt(r*r-distancex(p, c)*distancex(p, c))/distancex(l1, l2);
    p1.x = p.x+(l2.x-l1.x)*t;
    p1.y = p.y+(l2.y-l1.y)*t;
    p2.x = p.x-(l2.x-l1.x)*t;
    p2.y = p.y-(l2.y-l1.y)*t;
}

point ptoseg(point p, point l1, point l2)
{
    point t = p;
    t.x += l1.y-l2.y;
    t.y += l2.x-l1.x;
    if (xmult(l1, t, p)*xmult(l2, t, p)>eps)
        return distancex(p, l1)<distancex(p, l2) ? l1 : l2;
    return intersection(p, t, l1, l2);
}

double distp(point & a, point & b)
{
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}

double Direct_Triangle_Circle_Area(point a, point b, point o, double r)
{
    double sign = 1.0;
    a = a-o;
    b = b-o;

```

```

o = point(0.0, 0.0);
if (fabs(xmult(a, b, o)) < eps)
    return 0.0;
if (distp(a, o) > distp(b, o))
{
    swap(a, b);
    sign = -1.0;
}
if (distp(a, o) < r*r+eps)
{
    if (distp(b, o) < r*r+eps)
        return xmult(a, b, o)/2.0*sign;
    point p1, p2;
    intersection_line_circle(o, r, a, b, p1, p2);
    if (distancex(p1, b) > distancex(p2, b))
        swap(p1, p2);
    double ret1 = fabs(xmult(a, p1, o));
    double ret2 = acos((p1.x*b.x+p1.y*b.y)/p1.sqr()/b.sqr())*r*r;
    double ret = (ret1+ret2)/2.0;
    if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
        ret = -ret;
    return ret;
}
point ins = ptoseg(o, a, b);
if (distp(o, ins)>r*r-eps)
{
    double ret = acos((a.x*b.x+a.y*b.y)/a.sqr()/b.sqr())*r*r/2.0;
    if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
        ret = -ret;
    return ret;
}
point p1, p2;
intersection_line_circle(o, r, a, b, p1, p2);
double cm = r/(distancex(o, a)-r);
point m = point((o.x+cm*a.x)/(1+cm), (o.y+cm*a.y)/(1+cm));
double cn = r/(distancex(o, b)-r);
point n = point((o.x+cn*b.x)/(1+cn), (o.y+cn*b.y)/(1+cn));
double ret1 = acos((m.x*n.x+m.y*n.y)/m.sqr()/n.sqr())*r*r;
double ret2 = acos((p1.x*p2.x+p1.y*p2.y)/p1.sqr()/p2.sqr())*r*r-fabs(xmult(p1,
↪ p2, o));
double ret = (ret1-ret2)/2.0;
if (xmult(a, b, o)<eps && sign>0.0 || xmult(a, b, o)>eps && sign<0.0)
    ret = -ret;
return ret;
}
double Inter(double x,double y,double R,int n,point *area){
    area[n] = area[0];
    point temp = point(x, y);
    double sum = 0;
    for (int i=0; i<n-1; i++)
        sum += Direct_Triangle_Circle_Area(area[i], area[i+1], temp, R);

    sum += Direct_Triangle_Circle_Area(area[n-1], area[0], temp, R);
    return fabs(sum);
}
double Cross(point A,point B)

```

```

{
    return A.x*B.y - A.y*B.x;
}
int N,M;
double PolygonArea (point * p,int n)
{
    double area = 0;
    for(int i = 1; i < n - 1; ++i)
    {
        area += Cross(p[i]-p[0],p[i+1]-p[0]);
    }
    return fabs(area/2);
}

int dcmp(double x)
{
    if(fabs(x)<eps)
        return 0;
    else
        return x < 0?-1:1;
}

double S ;
double xi,yi,P,Q;
bool check(double R){
    //      cout<<xi<<" "<<yi<<" "<<P<<" "<<Q<<endl;
    //      printf("r = %lf Intersect = %lf\n",R,Inter(xi,yi,R,N,p) );
    //      printf("%lf\n", (1-P/Q)*S);
    return dcmp(Inter(xi,yi,R,N,p) - (1-P/Q)*S) > 0;
}

int main()
{
    cin>>N;
    for(int i=0;i< N;i++)
    {
        scanf("%lf%lf",&p[i].x,&p[i].y);
    }

    S= PolygonArea(p,N);
    //cout<<S<<endl;
    cin>>M;
    for(int i = 0;i < M; ++i){

        scanf("%lf %lf %lf %lf",&xi,&yi,&P,&Q);

        double l = 0,r = 1e6;
        for(int j = 0;j < 100; ++j){
            double mid = l+(r-l)/2;
            if(check(mid))
                r = mid;
            else
                l = mid;
            // printf("%lf %lf\n",l,r);
        }
        printf("%.8lf\n",r);
    }
}

```

```

    return 0;
}

```

2.1.7 求圆与直线的交点.cpp

```

int getLineCircleIntersection(Point A, Point B, Point C, double r, double& t1, double&
↪ t2, vector<Point> &sol){
    // 初始方程:  $(A.x + t(B.x - A.x) - C.x)^2 + (A.y + t(B.y - A.y) - C.y)^2 = r^2$ 
    // 整理得:  $(at + b)^2 + (ct + d)^2 = r^2$ 
    double a = B.x - A.x;
    double b = A.x - C.x;
    double c = B.y - A.y;
    double d = A.y - C.y;
    // 展开得:  $(a^2 + c^2)t^2 + 2(ab + cd)t + b^2 + d^2 - r^2 = 0$ , 即  $et^2 + ft + g = 0$ 
    double e = a * a + c * c;
    double f = 2 * (a * b + c * d);
    double g = b * b + d * d - r * r;
    double delta = f * f - 4 * e * g; // 判别式
    if(dcmp(delta) < 0) return 0; // 相离
    if(dcmp(delta) == 0){ // 相切
        t1 = t2 = -f / (2 * e);
        sol.push_back(A+(B-A)*t1);
        return 1;
    }
    t1 = (-f - sqrt(delta)) / (2 * e);
    t2 = (-f + sqrt(delta)) / (2 * e);
    sol.push_back(A+(B-A)*t1);
    sol.push_back(A+(B-A)*t2);
    return 2;
}

```

2.2 3D

2.2.1 三维几何的基本操作.cpp

```

#include <bits/stdc++.h>

using namespace std;
struct Point3
{
    double x,y,z;
    Point3(double x = 0, double y = 0, double z = 0):x(x),y(y),z(z) {}
};
typedef Point3 Vector3;

Vector3 operator +(Vector3 v1, Vector3 v2)
{
    return Vector3(v1.x+v2.x, v1.y+v2.y, v1.z+v2.z);
}
Vector3 operator -(Vector3 v1, Vector3 v2)
{
    return Vector3(v1.x-v2.x, v1.y-v2.y, v1.z-v2.z);
}
Vector3 operator *(Vector3 v, double c)
{
    return Vector3(v.x*c, v.y*c, v.z*c);
}

```

```

}
Vector3 operator /(Vector3 v,double c)
{
    return Vector3(v.x/c,v.y/c,v.z/c);
}
double Dot(Vector3 A,Vector3 B)
{
    return A.x*B.x+A.y*B.y+A.z*B.z;
}
double Length(Vector3 A)
{
    return sqrt(Dot(A,A));
}
double Angle(Vector3 A,Vector3 B)
{
    return acos(Dot(A,B)/(2*Length(A)*Length(B)));
}
double DistanceToplane(const Point3 &p,const Point3 &p0,const Vector3& n)
{
    return fabs(Dot(p-p0,n))/Length(n);
}
Point3 GetPlaneProjection(const Point3&p,const Point3&p0,const Vector3&n)
{
    return p-n*Dot(p-p0,n);
}
//直线 p1-p2 到平面 p0-n 的交点。假定交点唯一存在
Point3 LinePlaneIntetsection(Point3 p1,Point3 p2,Point3 p0,Vector3 n)
{
    Vector3 v= p2 - p1;
    //    /*if(dcmp(Dot(v,n))==0)
    //    {
    //        if(dcmp(Dot(p1-p0,n))==0)
    //            直线在平面上
    //        else
    //            直线与平面平行
    //    }
    //    */
    double t = Dot(n,p0-p1)/Dot(n,p2-p1);
    return p1 + v*t;
}

```

2.2.2 三维几何的模版.cpp

```

#include <bits/stdc++.h>
const double eps = 1e-6;
using namespace std;

struct Point3
{
    double x,y,z;
    Point3(double x = 0,double y = 0,double z = 0):x(x),y(y),z(z) {}
};
typedef Point3 Vector3;
int dcmp(double d)
{
    if(fabs(d)< eps)

```

```

        return 0;
    else
        return d < 0?-1:1;
}
Vector3 operator +(Vector3 v1,Vector3 v2)
{
    return Vector3(v1.x+v2.x,v1.y+v2.y,v1.z+v2.z);
}
Vector3 operator -(Vector3 v1,Vector3 v2)
{
    return Vector3(v1.x-v2.x,v1.y-v2.y,v1.z-v2.z);
}
Vector3 operator *(Vector3 v,double c)
{
    return Vector3(v.x*c,v.y*c,v.z*c);
}
Vector3 operator /(Vector3 v,double c)
{
    return Vector3(v.x/c,v.y/c,v.z/c);
}
bool operator ==(Point3 A,Point3 B)
{
    return !dcmp(A.x-B.x)&&!dcmp(A.y-B.y)&&!dcmp(A.z-B.z);
}
double Dot(Vector3 A,Vector3 B)
{
    return A.x*B.x+A.y*B.y+A.z*B.z;
}
double Length(Vector3 A)
{
    return sqrt(Dot(A,A));
}
double Angle(Vector3 A,Vector3 B)//求两向量的夹角
{
    return acos(Dot(A,B)/(2*Length(A)*Length(B)));
}
double DistanceToplanة(const Point3 &p,const Point3 &p0,const Vector3& n)//
{
    return fabs(Dot(p-p0,n))/Length(n);
}
Point3 GetPlaneProjection(const Point3&p,const Point3&p0,const Vector3&n)
{
    return p-n*Dot(p-p0,n);
}
//直线 p1-p2 到平面 p0-n 的交点。假定交点唯一存在
Point3 LinePlaneIntetsection(Point3 p1,Point3 p2,Point3 p0,Vector3 n)
{
    Vector3 v= p2 - p1;
    //    /*if(dcmp(Dot(v,n))==0)
    //    {
    //        if(dcmp(Dot(p1-p0,n))==0)
    //            直线在平面上
    //        else
    //            直线与平面平行
    //    }
    //    */

```



```

    double t = Dot(n,p0-p1)/Dot(n,p2-p1);
    return p1 + v*t;
}
Point3 LinePlaneIntersection(Point3 p1,Point3 p2,double A,double B,double C,double D)
{
    Vector3 v = p2-p1;
    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));
    return p1 + v*t;
}
Vector3 Cross(Vector3 A,Vector3 B)
{
    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);
}
double Area2(Point3 A,Point3 B,Point3 C)
{
    return Length(Cross(B-A,C-A));
}
////已知平面的三点，求出点法式
//Vector3 Solven(Point3 A,Point3 B,Point3 C)
//{
//    return Cross(B-A,C-A);
//}
//判断一个点是否在三角形内，可以用面积法
bool PointInTri(Point3 P,Point3 A,Point3 B,Point3 C)
{
    double area1 = Area2(P,A,B);
    double area2 = Area2(P,A,C);
    double area3 = Area2(P,B,C);
    double area4 = Area2(A,B,C);
    return dcmp(area1+area2+area3-area4)==0;
}
//判断线段是否与三角形相交
bool TriSegIntersection(Point3 P0,Point3 P1,Point3 P2,Point3 A,Point3 B,Point3 &P)
{
    Vector3 n = Cross(P1-P0,P2-P0);

    if(dcmp(Dot(n,B-A))==0)
        return false;

    double t = Dot(n,P0-A)/Dot(n,B-A);
    if(dcmp(t) < 0 || dcmp(t-1) > 0)
        return false;
    P = A + (B-A) * t;
    return PointInTri(P,P0,P1,P2);
}
double DistanceToLine(Point3 P,Point3 A,Point3 B)
{
    return Length(Cross(A-P,B-P))/Length(A-B);
}
double DistanceToSegment(Point3 P,Point3 A,Point3 B)
{
    if(A==B) return Length(P-A);
    Vector3 v1 = B - A, v2 = P - A,v3 = P-B;
    if(dcmp(Dot(v1,v2)) == 0) return Length(v2);
    if(dcmp(Dot(v1,v3)) > 0) return Length(v3);
    return Length(Cross(v1,v2))/Length(v1);
}

```

```

}
double Volume6(Point3 A,Point3 B,Point3 C,Point3 D)
{
    return Dot(D-A,Cross(B-A,C-A));
}
//
int main(void)
{
    Point3 A(0,0,0),B(0,100,0),C(100,0,0),D(25,25,0);
    cout<<PointInTri(D,A,B,C)<<endl;
    return 0;
}

```

2.2.3 三维凸包.cpp

```

struct Face{
    int v[3];
    Vector3 normal(Vector *P)
    {
        return Cross(P[v[1]]-P[v[0]],P[v[2]]-P[v[0]]);
    }
    int cansee(Point *P,int i)const
    {
        return Dot(P[i]-P[v[0]],normal(P)) > 0?1 : 0;
    }
};
vector <Face> CH3D(Point3* P,int n)
{
    vector <Face> cur;
    cur.push_back((Face){0,1,2});
    cur.push_back((Face){2,1,0});
    for(int i = 3;i < n; ++i)
    {
        vector<Face> next;
        //计算每条边“左面”的可见性
        for(int j= 0;j < cur.size(); ++j)
        {
            Face &f = cur[j];
            int res = f.cansee(P,i);
            if(!res) next.push_back(f);
            for(int k = 0;k < 3; ++k)
                vis[f.v[k]][f.v[(k+1)%3]] = res;
        }
        for(int j = 0;j < cur.size(); ++j)
        {
            for(int k = 0;k < 3; ++k)
            {
                int a = cur[j].v[k],b = cur[j].v[(k+1)%3];
                if(vis[a][b] != vis[b][a]&&vis[a][b])//(a,b) 是分界线, 左边对 P[i] 可见
                    next.push_back((Face){a,b,i});
            }
        }
        cnr = next;
    }
    return cur;
}

```

```

}
double rand01() {return rand() / (double) RAND_MAX;} //0-1 的随机数
double randeps() {return (rand01()-0.5) * eps;}
Point3 add_noise(Point3 p)
{
    return Point3(p.x + randeps(),p.y+randeps(),p.z+randeps());
}

//.....
struct Face{
    int v[3];
    Vector3 normal(Vector *P)
    {
        return Cross(P[v[1]]-P[v[0]],P[v[2]]-P[v[0]]);
    }
    int cansee(Point *P,int i)const
    {
        return Dot(P[i]-P[v[0]],normal(P)) > 0?1 : 0;
    }
};
vector <Face> CH3D(Point3* P,int n)
{
    vector <Face> cur;
    cur.push_back((Face){0,1,2});
    cur.push_back((Face){2,1,0});
    for(int i = 3;i < n; ++i)
    {
        vector<Face> next;
        //计算每条边“左面”的可见性
        for(int j= 0;j < cur.size(); ++j)
        {
            Face &f = cur[j];
            int res = f.cansee(P,i);
            if(!res) next.push_back(f);
            for(int k = 0;k < 3; ++k)
                vis[f.v[k]][f.v[(k+1)%3]] = res;
        }
        for(int j = 0;j < cur.size(); ++j)
        {
            for(int k = 0;k < 3; ++k)
            {
                int a = cur[j].v[k],b = cur[j].v[(k+1)%3];
                if(vis[a][b] != vis[b][a]&&vis[a][b])//(a,b) 是分界线, 左边对 P[i] 可见
                    next.push_back((Face){a,b,i});
            }
        }
        cnr = next;
    }
    return cur;
}
double rand01() {return rand() / (double) RAND_MAX;} //0-1 的随机数
double randeps() {return (rand01()-0.5) * eps;}
Point3 add_noise(Point3 p)
{
    return Point3(p.x + randeps(),p.y+randeps(),p.z+randeps());
}

```

2.2.4 维度转换为三维坐标.cpp

```
// 经纬度转换为球坐标
double torad(double deg)
{
    return deg/180*acos(-1);
}
void get_coordinate(double R,double lat,double lng,double &x,double &y,double &z)
{
    lat = torad(lat);
    lng = torad(lng);
    x = R*cos(lat)*cos(lng);
    y = R*cos(lat)*sin(lng);
    z = R*sin(lat);
}
```

3 动态规划

3.1 1 单调队列.cpp

```
//https://ac.nowcoder.com/acm/contest/223/C
//C 区间区间
//$$ v_{l,r} = max(a_i-a_j) (l <= i,j <= r)$$
//$$ \sum_{i=1}^n \sum_{j=1}^n v_{i,j}$$
const int maxn = 1e5+100;
int a[maxn];
int s[maxn]; // 单调栈
// 第一遍求在这个区间里面最大
int pre[maxn];
int nxt[maxn];
int main(void)
{
    int T,n;
    cin>>T;
    while(T--){
        scanf("%d",&n);
        for(int i = 1;i <= n; ++i){
            scanf("%d",&a[i]);
        }
        int t = 0;
        for(int i = 1;i <= n; ++i){
            pre[i] = nxt[i] = 0;
            while(t > 0&&a[i] > a[s[t]]) nxt[s[t]] = i,t--;
            pre[i] = s[t];
            s[++t] = i;
            // cout<<pre[i]<<" ";
        }
        while(t > 0)
            nxt[s[t]] = n+1,t--;
        LL ans = 0;
        for(int i = 1;i <= n; ++i){
            ans += 1ll*a[i]*(nxt[i]-i)*(i-pre[i]);
        }
        t = 0;
        for(int i = 1;i <= n; ++i){
            pre[i] = nxt[i] = 0;
        }
    }
}
```

```

        while(t > 0&&a[i] < a[s[t]]) nxt[s[t]] = i,t--;
        pre[i] = s[t];
        s[++t] = i;
    }
    while(t > 0)
        nxt[s[t]] = n+1,t--;
    for(int i = 1;i <= n; ++i){
        ans -= 1ll*a[i]*(nxt[i]-i)*(i-pre[i]);
    }
    printf("%lld\n",ans);
}

return 0;
}

```

3.2 1 最长上升子序列.cpp

//最长上升子序列 *The longest increasing sequence*

```

template <class It>
int n_lisLength(It begin,It end)
{
    typedef typename iterator_traits<It>::value_type T;
    T inf = 1<<30;
    vector<T> best(end-begin,inf);
    for(It i = begin; i != end; ++i)
        *lower_bound(best.begin(),best.end(),*i) = *i;
    return lower_bound(best.begin(),best.end(),inf) - best.begin();
}

```

3.3 string dp

3.3.1 trie+dp.cpp

/*

Margot 有一个 长度为字符串 *aa*, 给定 *nn* 个子串,
 每一个子串一个价值 *wi*, 从原串中取出一个子串后,
 原串的左右结合组合成一个新的串,
 并且得到改子串的价值 *wi*。问能取到的最大价值

*/

// SWERC 2017 D candy

#include<bits/stdc++.h>

using namespace std;

const int maxn = 55;

const int maxm = 11000;// 200 个串 200*50 tire 树节点

inline void up(int &a,int b){

a<b?(a=b):0;

}

// tire 树

const int maxnode = 4e5+100;

const int sigma_size = 26;

struct Trie

```

{
    int ch[maxnode][sigma_size];
    int val[maxnode];
    int sz;
    Trie()
    {
        sz = 1;
        memset(ch[0],0,sizeof(ch[0]));
        memset(val,-1,sizeof(val));
    }
    int idx(char c)
    {
        return c-'a';
    }
    void insert(char *s,int v)
    {
        int u = 0, n = strlen(s);
        for(int i = 0; i < n; ++i)
        {
            int c = idx(s[i]);
            if(!ch[u][c])
            {
                memset(ch[sz],0,sizeof(ch[sz]));
                //val[sz] = 0;
                ch[u][c] = sz++;
            }
            u = ch[u][c];
        }
        up(val[u], v);
    }
};

Trie tr;

int dp[maxn],f[maxn][maxn],g[maxn][maxm];
char ar[maxn];
char br[maxn];
int main(void){

    scanf("%s",ar+1);
    int n = strlen(ar+1);
    for(int i = 1;i <= n; ++i)
        ar[i] -= 'a';
    int C;
    scanf("%d",&C);
    while(C--){
        int u;
        scanf("%s %d",br,&u);
        int nn = strlen(br);
        tr.insert(br,u);
        reverse(br,br+nn);
        tr.insert(br,u);
    }

    // 初始化
    // for(int i = 1;i < tr.sz; ++i)

```

```

// cout<<tr.val[i]<<" ";
// cout<<endl;
for(int i = 0;i <= n+1; ++i)
    for(int j = 0;j <= n+1; ++j)
        f[i][j] = -1;
for(int i = n; i; --i){
    for(int j = i - 1;j <= n; ++j)
        for(int k = 0;k < tr.sz; ++k)
            g[j][k] = -1;
    // cout<<tr.sz<<endl;
    g[i-1][0] = 0;
    for(int j = i-1;j <= n; ++j){
        for(int k = 0;k < tr.sz; ++k){
            if(~g[j][k]){// 我为人人递推
                for(int x = j+1;x <= n; ++x)
                    if(~f[j+1][x])
                        up(g[x][k],g[j][k]+f[j+1][x]);
                int y = tr.ch[k][(int)ar[j+1]];
                // cout<<y<<endl;
                if(y != 0){
                    up(g[j+1][y],g[j][k]);
                    if(~tr.val[y]){
                        // cout<<tr.val[y]<<endl;
                        up(g[j+1][0],g[j][k]+tr.val[y]);
                    }
                }
            }
            if(k == 0)
                up(f[i][j],g[j][k]);
        }
    }
}
}
}

```

```

// cout<<f[1][n]<<endl;
for(int i = 1;i <= n; ++i){
    dp[i] = dp[i-1];
    for(int j = 1;j <= i; ++j)
        if(~f[j][i])
            up(dp[i],dp[j-1]+f[j][i]);
}
cout<<dp[n]<<endl;

```

```

return 0;
}

```

3.4 zhuangyadp

3.4.1 1 多米诺骨牌覆盖.cpp

```

/* 状态压缩 dp+ 矩阵快速幂, 用 1*2 的小方块填满 N*M 的矩形 */
//1033 骨牌覆盖 V2

```

```

#include<bits/stdc++.h>

using namespace std;
typedef long long LL;
const int maxn = 13;
const int mod = 1e9+7;
int n,m;
LL f[12][1<<11];
bool in_s[1<<11];

struct Matrix{
    #define maxn 100
    int n,m;
    Matrix(int nn = 1,int mm = 1):n(nn),m(mm){ memset(a,0,sizeof(a));};
    long long a[maxn][maxn];
};

void print(const Matrix &a)
{
    for(int i = 1;i <= a.n; ++i,cout<<endl)
        for(int j = 1;j <= a.m; ++j)
            cout<<a.a[i][j]<<" ";
}

Matrix operator*(Matrix a,Matrix b)
{
    assert(a.m == b.n);
    Matrix c(a.n,b.m);
    for(int i = 1;i <= a.n; ++i)
    {
        for(int j = 1;j <= b.m; ++j)
        {
            for(int k = 1;k <= a.m; ++k)
            {
                c.a[i][j] += a.a[i][k] * b.a[k][j];
                c.a[i][j] %= mod;
            }
        }
    }
    // print(c);
    return c;
}

Matrix B;
void solve(int m){
    for(int i = 0;i < (1<<m); ++i){
        bool cnt = 0,has_odd = 0;
        for(int j = 0;j < m; ++j){
            if(i >>j &1) has_odd |= cnt,cnt = 0;
            else cnt ^= 1;
            in_s[i] = has_odd | cnt?0:1;
        }
    }

    // f[0][0] = 1;
    // for(int i = 1;i <= n; ++i){
        for(int j = 0;j < (1<<m); ++j){
            // f[i][j] = 0;

```



```

        for(int k = 0;k < (1<<m); ++k){
            if((j&k) == 0&& in_s[j|k])
                B.a[j+1][k+1] = 1;
            // f[i][j] += f[i-1][k];
        }
    }
    // print(B);
    // cout<<f[n][0]<<endl;
}

```

```

LL M,N;
int main(void){
    scanf("%lld%lld",&M,&N);
    B.n = B.m = 1<<N;
    solve(N);
    Matrix ans(1,1<<N);

    ans.a[1][1] = 1;
    // print(ans);
    // cout<<endl;
    // print(B);
    while(M > 0){
        if(M & 1)
            ans = ans*B;
        B = B*B;
        // cout<<endl;
        // print(B);
        M >>= 1;
    }
    cout<<ans.a[1][1]<<endl;

    return 0;
}

```

/* 加强版
 1*1 和 2*1 的小方块
 SWERC2017 C - Macarons
 搜索求状态 */

// 矩阵快速幂
 // 注意修改 maxn 的值, 要不然容易 T

```

const int maxn = 260;
int n;
struct Matrix{
    int n,m;
    Matrix(int nn = 1,int mm = 1):n(nn),m(mm){ memset(a,0,sizeof(a));};
    int a[maxn][maxn];
};
void print(const Matrix &a)
{
    for(int i = 1;i <= a.n; ++i,cout<<endl)
        for(int j = 1;j <= a.m; ++j)

```

```

        cout<<a.a[i][j]<<" ";
    }
Matrix operator*(Matrix a,Matrix b)
{
    Matrix c(a.n,b.m);
    for(int i = 1;i <= a.n; ++i)
    {
        for(int j = 1;j <= b.m; ++j)
        {
            for(int k = 1;k <= a.m; ++k)
            {
                c.a[i][j] = (1ll*c.a[i][j]+1ll*a.a[i][k] * b.a[k][j])%mod;
            }
        }
    }
    // print(c);
    return c;
}
// 状态压缩

```

```

LL MM[maxn][maxn];
LL N,M;
// 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
void dfs(int a,int now,int nxt){
    // cout<<a<<endl;
    int tmpnow = now,tmpnxt = nxt;
    int one[10],two[10];
    memset(one,0,sizeof(one));
    memset(two,0,sizeof(two));
    int cnt = 0;
    while(tmpnow > 0){
        one[cnt++] = tmpnow&1;
        tmpnow >>= 1;
    }
    bool flag = true;
    for(int i = 0;i < N; ++i){
        if(!one[i]){
            flag = false;
            break;
        }
    }
    if((now & NN) == NN){
        MM[a][nxt]++;
        return ;
    }
    cnt = 0;
    while(tmpnxt > 0){
        two[cnt++] = tmpnxt&1;
        tmpnxt >>= 1;
    }
    for(int i = 0;i < N; ++i){
        if(!one[i]){
            dfs(a,now|(1<<i),nxt);
            dfs(a,now|(1<<i),nxt|(1<<i));
            if(i + 1 < N&& !one[i+1]){

```

```

        dfs(a,now|(1<<i)|(1<<(i+1)),nxt);
    }
    break;
}
}

}

int NN;
Matrix ans(NN,NN);
Matrix B(NN,NN);
void solve(){
    B.n = B.m = ans.n = ans.m = NN;
    for(int i = 1;i <= NN; ++i){
        for(int j = 1;j <= NN; ++j)
        {
            B.a[i][j] = MM[i-1][j-1];
        }
    }

    for(int i = 1;i <= NN; ++i) ans.a[i][i] = 1;
    while(M > 0){
        if(M & 1)
            ans = ans*B;
        B = B*B;
        M >>= 1;
    }
    cout<<ans.a[1][1]<<endl;
}
int main(void)
{
    scanf("%lld%lld",&N,&M);
    // cout<<N<<" "<<M<<endl;
    NN = 1<<N;
    // cout<<N<<" "<<NN<<endl;
    for(int i = 0;i < NN; ++i){
        dfs(i,i,0);
    }
    solve();
    return 0;
}

```

3.5 树上的分治

3.5.1 1 树的重心.cpp

```

// Size[u] 代表以节点 u 为根的子树节点个数
// dp[u] 代表去除 u 节点后最大子树的节点个数
const int maxn = 2e4+100;
vector<int> G[maxn];
int dp[maxn];
int Size[maxn];
int n;
int ans;
void dfs(int u,int fa){
    dp[u] = Size[u] = 0;
    for(int i = 0;i < G[u].size(); ++i){

```

```

        if(fa==G[u][i])continue;
        dfs(G[u][i],u);
        // sum += tmp;
        Size[u] += Size[G[u][i]];
        dp[u] = max(dp[u],Size[G[u][i]]);
    }
    Size[u]++;
    dp[u] = max(n-Size[u],dp[u]);
    if(dp[u] < dp[ans]) ans = u;
}
int main(void)
{
    int T;
    cin>>T;
    while(T--){
        scanf("%d",&n);
        for(int i = 1;i <= n; ++i) G[i].clear();
        for(int i = 1;i <= n-1; ++i){
            int u,v;
            scanf("%d%d",&u,&v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        ans = 0;
        dp[0] = INF;
        dfs(1,-1);
        printf("%d %d\n",ans,dp[ans]);
    }
    return 0;
}

```

4 图论

4.1 DFS

4.1.1 1. 无向图的割点和桥.cpp

SPF POJ - 1523

// 如果有割点，那么割点与子节点边就是割边

```

int dfs(int u,int fa){
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for(int i = 0;i < G[u].size(); ++i){
        int v = G[u][i];
        if(!pre[v]){
            child++;
            int lowv = dfs(v,u);
            lowu = min(lowu,lowv);
            if(lowv >= pre[u]){
                iscut[u]++;
            }
        }
        else if(pre[v] < pre[u] && v != fa){
            lowu = min(lowu,pre[v]);
        }
    }
}

```

```

    if(fa < 0&&child == 1) iscut[u] = 0;
    else if(fa < 0&&child >= 2) iscut[u] = child-1;
    return low[u] = lowu;
}

```

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

```

#include<iostream>
#include<cstdio>
#include<cctype>
#include<cstring>
#include<algorithm>
#include<vector>
#include<stack>
#include<map>
#include<queue>
#include<cmath>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define IOS ios::sync_with_stdio(false)
#define DEBUG cout<<endl<<"DEBUG"<<endl;
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
const int maxn = 1000+100;
// const int maxm = 1e6+100
int pre[maxn];
int dfs_clock = 0;
vector<int> G[maxn];
int iscut[maxn];
int low[maxn];

void init(){
    dfs_clock = 1;
    rep(i,1,maxn) G[i].clear();
    me(iscut);
    me(low);
    me(pre);
}

int dfs(int u,int fa){

```

```

int lowu = pre[u] = ++dfs_clock;
int child = 0;
for(int i = 0; i < G[u].size(); ++i){
    int v = G[u][i];
    if(!pre[v]){
        child++;
        int lowv = dfs(v,u);
        lowu = min(lowu,lowv);
        if(lowv >= pre[u]){
            iscut[u]++;
        }
    }
    else if(pre[v] < pre[u] && v != fa){
        lowu = min(lowu,pre[v]);
    }
}
if(fa < 0 && child == 1) iscut[u] = 0;
else if(fa < 0 && child >= 2) iscut[u] = child-1;
return low[u] = lowu;
}
// #define Debug
int main(void)
{
    #ifdef Debug
    freopen("input.txt","r",stdin);
    freopen("output.txt","w+",stdout);
    #endif
    int kase = 0;
    while(1){
        init();
        int u,v;
        int t = 0;
        while(scanf("%d",&u)==1 && u != 0){
            t++;
            scanf("%d",&v);
            G[u].Pb(v);
            G[v].Pb(u);
        }
        if(t==0) break;
        // rep(i,1,maxn) if(!G[i].empty()){

        //   dfs(i,-1);
        //   break;
        // }
        dfs(1,-1);
        int num = 0;
        rep(i,1,1001) if(iscut[i]) num++;

        printf("Network #%d\n",++kase);
        if(num > 0)
        {
            rep(i,1,1001) if(iscut[i]){
                printf("  SPF node %d leaves %d subnets\n",i,iscut[i]+1);
            }
        }
        else

```

```

        printf("  No SPF nodes\n");
    if(kase) puts("");
}

return 0;
}

```

4.1.2 2. 无向图的双连通分量.cpp

// 无向图的点联通分量

```

const int maxn= 1000+10;
int pre[maxn],iscut[maxn],bccno[maxn],dfs_clock,bcc_cnt;
vector<int> G[maxn],bcc[maxn];

stack<Edge> S;
int dfs(int u,int fa){
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for(int i = 0;i < G[u].size(); ++i){
        int v = G[u][i];
        Edge e = (Edge) {u,v};
        if(!pre[v]){
            S.push(e);
            child++;
            int lowv = dfs(v,u);
            lowu = min(lowu,lowv);
            if(lowv >= pre[u]){
                iscut[u] = true;
                bcc_cnt++;
                bcc[bcc_cnt].clear();
                for(;;){
                    Edge x = S.top(); S.pop();
                    if(bccno[x.u] != bcc_cnt) {bcc[bcc_cnt].push_back(x.u); bccno[x.u] =
                        ↪ bcc_cnt;}
                    if(bccno[x.v] != bcc_cnt) {bcc[bcc_cnt].push_back(x.v); bccno[x.v] =
                        ↪ bcc_cnt;}
                    if(x.u == u&&v == x.v) break;
                }
            }
        }
        else if(pre[v] < pre[u]&&v != fa){
            S.push(e);lowu = min(pre[v],lowu);
        }
    }
    if(fa < 0&& child == 1) iscut[u] = 0;
    return lowu;
}

void find_bcc(int n){
    memset(pre,0,sizeof(pre));
    memset(iscut,0,sizeof(iscut));
    memset(bccno,0,sizeof(bccno));
    dfs_clock = bcc_cnt = 0;
    for(int i = 0;i < n; ++i) if(!pre[i]) dfs(i,-1);
}

```

```
}
```

//无向图的边-双联通分量

// 第一边 *dfs* 求出所有的割边, 然后第二边 *dfs* 求出所有边一双连通分量 (不经过割边)

4.1.3 3 有向图的强联通分量.cpp

// *tarjan* 算法

```
const int maxn = 2e4+100;
```

```
vector<int> G[maxn];
```

```
int pre[maxn],lowlink[maxn],sccno[maxn],dfs_clock,scc_cnt;
```

```
stack<int> S;
```

```
void dfs(int u){
```

```
    pre[u] = lowlink[u] = ++dfs_clock;
```

```
    S.push(u);
```

```
    for(int i = 0;i < G[u].size(); ++i){
```

```
        int v = G[u][i];
```

```
        if(!pre[v]){
```

```
            dfs(v);
```

```
            lowlink[u] = min(lowlink[u],lowlink[v]);
```

```
    }
```

```
    else if(!sccno[v]){
```

```
        lowlink[u] = min(lowlink[u],pre[v]);
```

```
    }
```

```
}
```

```
if(lowlink[u] == pre[u]){
```

```
    scc_cnt++;
```

```
    for(;;){
```

```
        int x = S.top(); S.pop();
```

```
        sccno[x] = scc_cnt;
```

```
        if(x == u) break;
```

```
    }
```

```
}
```

```
}
```

```
void find_scc(int n){
```

```
    dfs_clock= scc_cnt = 0;
```

```
    me(sccno),me(pre);
```

```
    rep(i,0,n) if(!pre[i]) dfs(i);
```

```
}
```

// *kosaraju*

```
const int maxn = 2e4+100;
```

```
vector<int> G[maxn],G2[maxn];
```

```
vector<int> S;
```

```
int vis[maxn],sccno[maxn],scc_cnt;
```

```
void dfs1(int u){
```

```
    if(vis[u]) return ;
```

```
    vis[u] = 1;
```

```
    for(int i = 0;i < G[u].size(); ++i) dfs1(G[u][i]);
```

```
    S.push_back(u);
```

```
}
```



```

void dfs2(int u){
    if(sccno[u]) return ;
    sccno[u] = scc_cnt;
    for(int i = 0; i < G2[u].size(); ++i) dfs2(G2[u][i]);
}
void find_scc(int n){
    scc_cnt = 0;
    S.clear();
    memset(sccno,0,sizeof(sccno));
    memset(vis,0,sizeof(vis));
    for(int i = 0; i < n; ++i) dfs1(i);
    for(int i = n-1; i >= 0; --i){
        if(!sccno[S[i]]) {
            scc_cnt++;
            dfs2(S[i]);
        }
    }
}

```

4.1.4 4 2-sat 问题.cpp

*// $O(n*m)$ 复杂度不确定*

```

const int maxn = 2000 + 10;

struct TwoSAT {
    int n;
    vector<int> G[maxn*2];
    bool mark[maxn*2];
    int S[maxn*2], c;

    bool dfs(int x) {
        if (mark[x^1]) return false;
        if (mark[x]) return true;
        mark[x] = true;
        S[c++] = x;
        for (int i = 0; i < G[x].size(); i++)
            if (!dfs(G[x][i])) return false;
        return true;
    }

    void init(int n) {
        this->n = n;
        for (int i = 0; i < n*2; i++) G[i].clear();
        memset(mark, 0, sizeof(mark));
    }

    // x = xval or y = yval
    void add_clause(int x, int xval, int y, int yval) {
        x = x * 2 + xval;
        y = y * 2 + yval;
        G[x].push_back(y^1); // G[0].Pb(1)
        G[y].push_back(x^1); // G[1].Pb(0);
    }

    bool solve() {

```

```

    for(int i = 0; i < n*2; i += 2)
        if(!mark[i] && !mark[i+1]) {
            c = 0;
            if(!dfs(i)) {
                while(c > 0) mark[S[--c]] = false;
                if(!dfs(i+1)) return false;
            }
        }
    return true;
}
};

```

4.2 LCA

4.2.1 1 DFS+RMQ.cpp

```

#include<cstdio>
#include<cstring>
#include<vector>
#include<cmath>
#include<iostream>
using namespace std;

const int maxn = 40000+100;
const int maxlogv = 17;
struct Edge{
    int to,weight;
    Edge(int t,int w):to(t),weight(w){};
};
vector<Edge> G[maxn];

int id[maxn],dis[maxn];
int vs[maxn*2],depth[maxn*2];
int dp[maxn*2][maxlogv];
void dfs(int node,int fa,int d,int &k){
    id[node] = k;
    vs[k] = node;
    depth[k++] = d;
    // dis[node] = distance;
    for(int i = 0;i < G[node].size(); ++i){
        Edge &t = G[node][i];
        if(t.to == fa) continue;
        dis[t.to] = dis[node]+t.weight;
        dfs(t.to,node,d+1,k);
    }
    vs[k] = node;
    depth[k++] = d;
}

void init_rmq(int n){

    for(int i = 0;i < n ; ++i) dp[i][0] = i;
    for(int j = 1;(1<<j) <= n; ++j){
        for(int i = 0;i + (1<<j)-1 < n; ++i){
            if(depth[dp[i][j-1]] < depth[dp[i+(1<<(j-1))][j-1]])
                dp[i][j] = dp[i][j-1];
        }
    }
}

```

```

        else
            dp[i][j] = dp[i+(1<<(j-1))][j-1];
    }
}

int query(int l,int r){
    int k = 0;
    while((1<<(k+1)) <= r-l+1) k++;
    if(depth[dp[l][k]] < depth[dp[r-(1<<k)+1][k]])
        return dp[l][k];
    else
        return dp[r-(1<<k)+1][k];
}

int lca(int u,int v){
    return vs[query(min(id[u],id[v]),max(id[u],id[v]))];
}

void init(int n){
    int k = 0;
    dfs(0,-1,0,k);
    init_rmq(2*n-1);
}

int main(void){
    int n,m,q;
    while(~scanf("%d%d",&n,&m)){
        for(int i = 0;i < n; ++i) G[i].clear();
        int u,v,w;
        for(int i = 0;i < m; ++i){
            scanf("%d%d%d",&u,&v,&w);
            u--,v--;
            G[u].push_back(Edge(v,w));
            G[v].push_back(Edge(u,w));
        }
        init(n);
        scanf("%d",&q);
        while(q--){
            int u,v;
            scanf("%d %d",&u,&v);
            u--,v--;
            int f = lca(u,v);
            printf("%d\n",dis[u]+dis[v]-2*dis[f]);
        }
    }
    return 0;
}

```

4.2.2 2 倍增算法.cpp

```

// POJ1330
// LCA 的倍增算法

#include<vector>
#include<cstdio>
#include<cstring>
using namespace std;

```

```

const int maxn = 1e4+100;
const int maxlogv = 14;
vector<int> G[maxn];
int root;

int parent[maxlogv][maxn];
int depth[maxn];

void dfs(int v,int p,int d){
    parent[0][v] = p;
    depth[v] = d;
    for(int i = 0;i < G[v].size(); ++i){
        if(G[v][i] != p){
            dfs(G[v][i],v,d+1);
        }
    }
}

void init(int V){
    dfs(root,-1,0);
    for(int k = 0;k+1 < maxlogv; ++k){
        for(int v = 0; v < V; ++v){
            if(parent[k][v] < 0) parent[k+1][v] = -1;
            else parent[k+1][v] = parent[k][parent[k][v]];
        }
    }
}

int lca(int u,int v){
    if(depth[u] > depth[v]) swap(u,v);
    for(int k = 0;k < maxlogv; ++k){
        if(((depth[v] - depth[u]) >> k) & 1){
            v = parent[k][v];
        }
    }
    if(u == v) return u;
    for(int k = maxlogv-1; k >= 0; --k){
        if(parent[k][u] != parent[k][v]){
            u = parent[k][u];
            v = parent[k][v];
        }
    }
    return parent[0][u];
}

bool OUT[maxn];
int main(void)
{
    int T;
    scanf("%d",&T);
    while(T--){
        int n;
        for(int i = 0;i < n; ++i) G[i].clear();
        memset(OUT,0,sizeof(OUT));
        scanf("%d",&n);

```

```

        for(int i = 1; i < n; ++i) {
            int u, v;
            scanf("%d %d", &u, &v);
            u--, v--;
            G[u].push_back(v);
        }
        OUT[v] = 1;
    }
    for(int i = 0; i < n; ++i) if(!OUT[i]){
        root = i;
        break;
    }
    init(n);
    int u, v;
    scanf("%d %d", &u, &v);
    u--, v--;
    printf("%d\n", lca(u, v) + 1);
}

return 0;
}

```

4.3 Maxflow

4.3.1 1 Dinic.cpp

```

// dinic
#include <cstdio> // C 语言 io
#include <cstring> // 以下是 c 语言常用头文件
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <cctype>
#include <cstring>
#include <cmath>
#include <iostream> // c++ IO
#include <sstream>
#include <string>
#include <list> // c++ 常用容器
#include <vector>
#include <set>
#include <map>
#include <queue>
#include <stack>
#include <algorithm> // c++ 泛型的一些函数
#include <functional> // 用来提供一些模版
#define fo0(i, n) for(int i = 0; i < n; ++i)
#define fo1(i, n) for(int i = 1; i <= n; ++i)
#define mem(ar, num) memset(ar, num, sizeof(ar))
#define me(ar) memset(ar, 0, sizeof(ar))
#define lowbit(x) (x & (-x))
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF = 0x7FFFFFFFFFFFFFFF;

```

```

const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL      mod = 1e9 + 7;
const int LEN = 20000+1000;
const int maxn = 1e8;
struct Edge{
    int from,to,cap,flow;
    Edge(int u,int v,int w,int f): from(u),to(v),cap(w),flow(f){}
};
struct Dinic{
    int n,m,s,t;
    vector<Edge> edges;
    vector<int> G[LEN];
    int a[LEN];
    int vis[LEN];
    int d[LEN];
    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
    void init(int n)
    {
        this->n = n;
        for(int i = 0;i < n; ++i)
            G[i].clear();
        edges.clear();
    }
    void Add(int u,int v,int w)
    {
        edges.push_back(Edge(u,v,w,0));
        edges.push_back(Edge(v,u,0,0));
        m = edges.size();
        G[u].push_back(m-2);
        G[v].push_back(m-1);
    }
    bool Bfs(void) //分层
    {
        me(d);
        me(vis);
        d[s] = 0;
        vis[s] = 1;

        queue<int> Q;
        Q.push(s);
        while(!Q.empty())
        {
            int q = Q.front();Q.pop();

            for(size_t i = 0;i < G[q].size();++i)
            {
                Edge &tmp = edges[G[q][i]];
                if(!vis[tmp.to]&&tmp.cap>tmp.flow)
                {
                    vis[tmp.to] = 1;
                    d[tmp.to] = d[q] + 1;
                    Q.push(tmp.to);
                }
            }
        }
    }
};

```

```

    }
}
return vis[t];
}
int Dfs(int node,int a)
{
    if(node == t || a == 0)
        return a;
    int flow = 0,f;
    for(int &i = cur[node];i < G[node].size();++i)
    {
        Edge &tmp = edges[G[node][i]];
        if(d[tmp.to]==d[node]+1&&(f=Dfs(tmp.to,min(a,tmp.cap-tmp.flow)))>0)
        {
            flow += f;
            tmp.flow += f;
            edges[G[node][i]^1].flow -= f;
            a -= f;
            if(a==0)
                break;
        }
    }
    return flow;
}
int MaxFlow(int s,int t)
{
    this->s = s;
    this->t = t;
    int flow = 0;
    while(Bfs())
    {
        me(cur);
        flow += Dfs(s,maxn);
    }
    return flow;
}

};
Dinic dinic;
int main()
{
    int N,M,S,T;
    while(cin>>N>>M)
    {
        S = 1, T = N;
        dinic.init(N);
        int u,v,w;
        for(int i = 0;i < M;++i)
        {
            scanf("%d %d %d",&u,&v,&w);
            dinic.Add(u,v,w);
        }
        int ans = 0;

```

```

        ans = dinic.MaxFlow(S,T);
        printf("%d\n",ans);

    }

```

```

        return 0;
    }

```

4.3.2 2 ISAP.cpp

// 点的下标从零开始, 注意初始化

```

#include<cstdio>
#include<cstring>
#include<queue>
#include<vector>
#include<algorithm>
using namespace std;

const int maxn = 10000 + 10;
const int INF = 1000000000;

struct Edge {
    int from, to, cap, flow;
};

bool operator < (const Edge& a, const Edge& b) {
    return a.from < b.from || (a.from == b.from && a.to < b.to);
}

struct ISAP {
    int n, m, s, t;
    vector<Edge> edges;
    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
    bool vis[maxn]; // BFS 使用
    int d[maxn]; // 从起点到 i 的距离
    int cur[maxn]; // 当前弧指针
    int p[maxn]; // 可增广路上的上一条弧
    int num[maxn]; // 距离标号计数

    void AddEdge(int from, int to, int cap) {
        edges.push_back((Edge){from, to, cap, 0});
        edges.push_back((Edge){to, from, 0, 0});
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool BFS() {
        memset(vis, 0, sizeof(vis));
        queue<int> Q;
        Q.push(t);
        vis[t] = 1;
        d[t] = 0;
        while(!Q.empty()) {

```



```

    int x = Q.front(); Q.pop();
    for(int i = 0; i < G[x].size(); i++) {
        Edge& e = edges[G[x][i]^1];
        if(!vis[e.from] && e.cap > e.flow) {
            vis[e.from] = 1;
            d[e.from] = d[x] + 1;
            Q.push(e.from);
        }
    }
}

return vis[s];
}

void init(int n) {
    this->n = n;
    for(int i = 0; i < n; i++) G[i].clear();
    edges.clear();
}

int Augment() {
    int x = t, a = INF;
    while(x != s) {
        Edge& e = edges[p[x]];
        a = min(a, e.cap - e.flow);
        x = edges[p[x]].from;
    }
    x = t;
    while(x != s) {
        edges[p[x]].flow += a;
        edges[p[x]^1].flow -= a;
        x = edges[p[x]].from;
    }
    return a;
}

int Maxflow(int s, int t) {
    this->s = s; this->t = t;
    int flow = 0;
    BFS();
    memset(num, 0, sizeof(num));
    for(int i = 0; i < n; i++) num[d[i]]++;
    int x = s;
    memset(cur, 0, sizeof(cur));
    while(d[s] < n) {
        if(x == t) {
            flow += Augment();

            x = s;
        }
        int ok = 0;
        for(int i = cur[x]; i < G[x].size(); i++) {
            Edge& e = edges[G[x][i]];
            if(e.cap > e.flow && d[x] == d[e.to] + 1) { // Advance
                ok = 1;
                p[e.to] = G[x][i];
            }
        }
        if(!ok) x = t;
        cur[x]++;
    }
    return flow;
}

```

```

        cur[x] = i; // 注意
        x = e.to;
        break;
    }
}
if(!ok) { // Retreat
    int m = n-1; // 初值注意
    for(int i = 0; i < G[x].size(); i++) {
        Edge& e = edges[G[x][i]];
        if(e.cap > e.flow) m = min(m, d[e.to]);
    }
    if(--num[d[x]] == 0) break;
    num[d[x] = m+1]++;
    cur[x] = 0; // 注意
    if(x != s) x = edges[p[x]].from;
}
}
return flow;
}
};

```

ISAP g;

```

int main() {

    int N,M;
    int S,T;
    scanf("%d %d",&N,&M);
    scanf("%d %d",&S,&T);
    int u,v,w;
    g.init(N);
    while(M--){
        scanf("%d %d %d",&u,&v,&w);
        u--,v--;
        g.AddEdge(u,v,w);
    }
    printf("%d",g.Maxflow(S-1,T-1));

    return 0;
}

```

4.3.3 3 MCMF.cpp

// 最小费用最大流，下标从 1 开始

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)

```

```

using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 1e8;
const LL INFF = 0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;}
LL gcd(LL a, LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Edge{
    int from,to,cap,flow,cost;
};
const int maxn = 5000+100;
struct MCMF{
    int n,m,s,t;
    vector<Edge> edges;
    vector<int> G[maxn];
    int inq[maxn];
    int d[maxn];
    int p[maxn];
    int a[maxn];
    void init(int n){
        this->n = n;
        for(int i = 0;i < n; ++i) G[i].clear();
        edges.clear();
    }
    void AddEdge(int from,int to,int cap,int cost){
        edges.push_back((Edge){from,to,cap,0,cost});
        edges.push_back((Edge){to,from,0,0,-cost});
        int m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }
    bool BellmanFord(int s,int t,int &flow,int &cost){
        for(int i = 0;i < n; ++i) d[i] = INF;
        memset(inq,0,sizeof(inq));
        d[s] = 0,inq[s] = 1;p[s] = 0,a[s] = INF;

        queue<int> Q;
        Q.push(s);
        while(!Q.empty()){

            int u = Q.front(); Q.pop();
            inq[u] = 0;
            for(int i = 0;i < G[u].size(); ++i){
                Edge& e = edges[G[u][i]];
                if(e.cap > e.flow && d[e.to] > d[u]+e.cost){
                    d[e.to] = d[u]+e.cost;
                    p[e.to] = G[u][i];
                    a[e.to] = min(a[u],e.cap-e.flow);
                }
            }
            if(!inq[t]) Q.push(t);
        }
        flow = a[t];
        cost += d[t]*flow;
        while(t != s){
            int u = t;
            int i = p[u];
            Edge& e = edges[G[u][i]];
            e.flow += flow;
            int v = G[u][i]-1;
            Edge& e2 = edges[v];
            e2.flow -= flow;
            t = v;
        }
        return true;
    }
};

```

```

        if(!inq[e.to]) {
            Q.push(e.to); inq[e.to] = 1;
        }
    }
}

if(d[t] == INF) return false;

flow += a[t];
cost += d[t]*a[t];
int u = t;
while(u != s){
    edges[p[u]].flow += a[t];
    edges[p[u]^1].flow -= a[t];
    u = edges[p[u]].from;
}
return true;
}

int Mincost(int s,int t,int &flow,int &cost){
    flow = 0,cost = 0;

    while(BellmanFord(s,t,flow,cost));
    return cost;
}

};
MCMF mcmf;
int main(void)
{
    int n,m,s,t;
    scanf("%d %d %d %d",&n,&m,&s,&t);
    int u,v,w,c;
    mcmf.init(n+1);
    while(m--){
        scanf("%d %d %d %d",&u,&v,&w,&c);
        mcmf.AddEdge(u,v,w,c);
    }
    int flow,cost;
    flow = 0,cost = 0;
    mcmf.Mincost(s,t,flow,cost);
    printf("%d %d\n",flow,cost);

    return 0;
}

```

4.4 二分图

4.4.1 1 匈牙利算法.cpp

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back

```

```

#define FI first
#define SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF = 0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
const int maxn = 1000+10;
vector<int> G[maxn];
int match[maxn];
bool used[maxn];
int N,M;
bool dfs(int v){
    used[v] = true;
    for(int i = 0;i < G[v].size(); ++i){
        if(used[u]) continue; used[u] = true;
        int u = G[v][i],w = match[u];
        if(w < 0||!used[w]&&dfs(w)){
            match[v] = u;
            match[u] = v;
            return true;
        }
    }
    return false;
}
int main(void)
{
    scanf("%d %d",&N,&M);

    while(M--){
        int u,v;
        scanf("%d %d",&u,&v);
        G[u].Pb(v);
        G[v].Pb(u);
    }
    int ans = 0;
    memset(match,-1,sizeof(match));
    for(int i = 1;i <= N; ++i){
        if(match[i] < 0){
            memset(used,0,sizeof(used));
            if(dfs(i)){
                ans++;
            }
        }
    }
}

```

```

    cout<<ans<<endl;
    return 0;
}

```

4.4.2 2 KM.cpp

```

const int maxn = 500+5;
struct KM{
    int n;
    vector<int> G[maxn];
    int W[maxn][maxn];
    int Lx[maxn];
    int Ly[maxn];
    int Left[maxn];
    bool S[maxn],T[maxn];
    void init(int n){
        this->n = n;
        for(int i = 1;i <= n; ++i) G[i].clear();
        memset(W,0,sizeof(W));
    }
    void AddEdge(int u,int v,int w){
        G[u].push_back(v);
        W[u][v] = w;
    }
    bool match(int u){
        S[u] = true;
        for(int i = 0;i < G[u].size(); ++i){
            int v = G[u][i];
            if(Lx[u]+Ly[v] == W[u][v]&&!T[v]){
                T[v] = true;
                if(Left[v] == -1||match(Left[v])){
                    Left[v] = u;
                    return true;
                }
            }
        }
        return false;
    }
    void update(){
        int a = INF;
        for(int u = 0;u < n; ++u)
            if(S[u])
                for(int i = 0;i < G[u].size(); ++i){
                    int v = G[u][i];
                    if(!T[v])
                        a = min(a,Lx[u]+Ly[v]-W[u][v]);
                }
        for(int i = 0;i < n; ++i){
            if(S[i]) Lx[i] -= a;
            if(T[i]) Ly[i] += a;
        }
    }
    void solve(){
        for(int i = 0;i < n; ++i){
            Lx[i] = *max_element(W[i],W[i]+n);
            Left[i] = -1;
        }
    }
}

```

```

        Ly[i] = 0;
    }
    for(int u = 0; u < n; ++u){
        for(;;){
            for(int i = 0; i < n; ++i) S[i] = T[i] = 0;
            if(match(u)) break;
            else update();
        }
    }
};

```

4.4.3 3 一般图最大匹配.cpp

```

#include<cstdio>
#include<algorithm>
#include<cmath>
#include<cstring>
#include<vector>
#define SF scanf
#define PF printf
#define MAXN 510
using namespace std;
int mk[MAXN], fa[MAXN], nxt[MAXN], q[MAXN], vis[MAXN], match[MAXN];
int fr, bk, t, n, m;
vector<int> a[MAXN];
int find(int x){
    if(fa[x]==x)
        return x;
    fa[x]=find(fa[x]);
    return fa[x];
}
int LCA(int x, int y){
    t++;
    while(1){
        if(x){
            x=find(x);
            if(vis[x]==t)
                return x;
            vis[x]=t;
            if(match[x])
                x=nxt[match[x]];
            else
                x=0;
        }
        swap(x, y);
    }
}
void Union(int x, int y){
    if(find(x)!=find(y))
        fa[fa[x]]=fa[y];
}
void gr(int a, int p){
    while(a!=p){
        int b=match[a];
        int c=nxt[b];
    }
}

```

```

        if(find(c)!=p)
            nxt[c]=b;
        if(mk[b]==2){
            q[++bk]=b;
            mk[b]=1;
        }
        Union(a,b);
        Union(b,c);
        a=c;
    }
}

void aug(int S){
    for(int i=1;i<=n;i++){
        mk[i]=nxt[i]=0;
        fa[i]=i;
    }
    mk[S]=1;
    fr=bk=0;
    q[fr]=S;
    while(fr<=bk){
        int x=q[fr++];
        for(int i=0;i<a[x].size();i++){
            int y=a[x][i];
            if(match[x]==y)
                continue;
            else if(find(x)==find(y))
                continue;
            else if(mk[y]==2)
                continue;
            else if(mk[y]==1){
                int r=LCA(x,y);
                if(find(x)!=r)
                    nxt[x]=y;
                if(find(y)!=r)
                    nxt[y]=x;
                gr(x,r);
                gr(y,r);
            }
            else if(!match[y]){
                nxt[y]=x;
                for(int u=y;u;){
                    int v=nxt[u];
                    int mv=match[v];
                    match[u]=v;
                    match[v]=u;
                    u=mv;
                }
                return;
            }
            else{
                nxt[y]=x;
                mk[y]=2;
                q[++bk]=match[y];
                mk[match[y]]=1;
            }
        }
    }
}

```



```

    }
}
int main(){
    SF("%d%d",&n,&m);
    int u,v;
    for(int i=1;i<=m;i++){
        SF("%d%d",&u,&v);
        a[u].push_back(v);
        a[v].push_back(u);
    }
    for(int i=1;i<=n;i++)
        if(!match[i])
            aug(i);
    int sum=0;
    for(int i=1;i<=n;i++)
        if(match[i])
            sum++;
    PF("%d\n",sum/2);
    for(int i=1;i<=n;i++)
        PF("%d ",match[i]);
}

```

4.5 最小生成树

4.5.1 1 Krustal 卡鲁斯卡尔算法.cpp

```

/*
复杂度  $E \cdot \log(E)$ , 适用于稀疏图
https://vjudge.net/problem/HDU-1863
*/

#include<bits/stdc++.h>

using namespace std;

const int maxn = 100+100;
struct Edge//边
{
    int from,to,cost;
    bool operator< ( const Edge & a)
    {
        return cost < a.cost;
    }
};
Edge edge[maxn];
int F[maxn];
int Find(int x)//并查集算法
{
    return x == F[x] ? x:F[x] = Find(F[x]);
}
int main(void)
{
    int N,M;
    while(cin>>N>>M&&N)// N 代表的是道路数量, M 代表村庄的数量
    {

```

```

    for(int i = 0; i <= M; ++i)
        F[i] = i;
    for(int i = 0; i < N; ++i)
    {
        Edge &t = edge[i];
        scanf("%d %d %d",&t.from,&t.to,&t.cost);
    }
    sort(edge,edge+N); // 对边进行排序
    int sum = 0;
    int num = M;
    for(int i = 0; i < N ; ++i) // 一个个将边加进去
    {
        Edge t = edge[i];
        if(Find(t.from) == Find(t.to))
            continue;
        F[Find(t.from)] = F[Find(t.to)];
        sum += t.cost;
        num--;
    }
    if(num == 1)
        cout<<sum<<endl;
    else
        cout<<"?"<<endl;
}

return 0;
}

```

4.5.2 2 prim 算法.cpp

```

/*
prim 算法是进行加点，使用于稠密图，可以选择用堆或者不用
不用堆  $O(V^2)$ ;
用堆  $O(E * \log(V))$ ;
https://vjudge.net/problem/HDU-1863
*/

typedef pair<int,int> P;
const int LEN = 2e6+100;
int Away[LEN]; //记录从当前已选结点到 j 节点的路径的最小值
bool vis[LEN];
int N,M; //N 道路数目, M 村庄个数
vector<vector<P>> vec(LEN);
int main()
{
    cin>>M>>N;

    int from,to,weight;
    while(N--)
    {
        scanf("%d %d %d",&from,&to,&weight);
        vec[from].push_back(P(weight,to));
        vec[to].push_back(P(weight,from));
    } // 添加边
}

```

```

    for(int i = 2; i <= M; ++i)
        Away[i] = INF; //初始化 Away 数组
    Away[1] = 0;
    int Left = M;
    int All_cost = 0;
    priority_queue<P,vector<P>,greater<P> > q; // 小顶堆
    q.push(P(0,1));
    while(!q.empty() && Left>0)
    {
        P tmp = q.top();q.pop();
        int To = tmp.second;
        if(vis[To])
            continue;
        vis[To] = 1;
        Left--;
        All_cost += tmp.first;
        for(int i = 0; i < vec[To].size(); ++i) // 更新 Away 数组
        {
            P &t = vec[To][i];
            if(!vis[t.second] && Away[t.second] > t.first)
            {
                Away[t.second] = t.first;
                q.push(t);
            }
        }
    }

    cout<<All_cost<<endl;

    return 0;
}

```

4.5.3 3 最小限制生成树.cpp

```

// 限制某一点的度数不能超过 K
#include<cstring>
#include<map>
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<set>
using namespace std;
#define me(ar) memset(ar,0,sizeof(ar))
const int INF = 1e8;
//.....
const int LEN = 30;
int K;
int n,m;
struct Edge
{
    int x,y;
    int weight;
}

```

```

    bool operator <(const Edge &a) const
    {
        return weight < a.weight;
    }
} edge[LEN*LEN+10]; //邻接表存边, Kruskal 算法要用
int dis[LEN][LEN]; //邻接矩阵
int sign[LEN][LEN]; //记录那些边已经在生成树里面了
int vis[LEN]; //记录是否相连
int F[LEN]; //并查集所用
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
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
int Find(int x) //并查集所用 Find 函数
{
    return x == F[x]?x:F[x] = Find(F[x]);
}
void Dfs(int x) //Dfs 动态规划记忆化搜索
{
    // vis[x] = 1;
    for(int i = 1; i <= n; ++i)
    {
        if(sign[i][x]&!vis[i]) //如果有边相连并且下一个节点没有被访问
        {
            if(x==0)
                Best[i] = -INF; //与 park 直接相连的边不能删除

            else
                Best[i] = max(Best[x], dis[x][i]); //状态转移方程
            Father[i] = x;
            vis[i] = 1;
            Dfs(i);
        }
    }
}
void init(){
    for(int i = 0; i < LEN; ++i)
        F[i] = i;
    me(sign); //初始化标记数组
    me(vis);
    //初始化邻接矩阵
    for(int i = 0; i < LEN; ++i)
        for(int j = 0; j < LEN; ++j)
            dis[i][j] = INF;
}
int main(void)
{
    while(cin>>m)
    {
        //初始化并查集数组
        init();
        n = 0; //用来记录共有多少个节点
        // set<string> se;
        map<string, int> ma; //将地点编号
        ma["Park"] = 0; //将 park 加入节点
        string s1, s2;
        int a, b;
        int weight = 0;
    }
}

```

```

for(int i = 0; i < m; ++i)
{
    cin>>s1>>s2>>weight;
    if(s1 == "Park" || ma[s1] != 0)
        a = ma[s1]; //如果节点已编号，则直接使用
    else
        a = ma[s1] = ++n; //如果没有编号，编号
    if(s2 == "Park" || ma[s2] != 0)
        b = ma[s2];
    else
        b = ma[s2] = ++n;
    dis[a][b] = dis[b][a] = weight;
    edge[i].x = a;
    edge[i].y = b;
    edge[i].weight = weight;
}
//求最小生成树
int ans = 0; //kruskal 算法求最小生成树
sort(edge, edge+m);
for(int i = 0; i < m; ++i)
{
    int x = edge[i].x;
    int y = edge[i].y;
    weight = edge[i].weight;
    if(x==0 || y==0) //去掉 park 这个点
        continue;
    int xx = Find(x);
    int yy = Find(y);
    if(xx!=yy)
    {
        F[xx] = F[yy];
        ans += weight;
        sign[x][y] = sign[y][x] = 1;
    }
}

cin>>K; //最小 k 度生成树
int Min[LEN]; //用来记录每一个最小生成树到 park 点的最小路径
for(int i = 0; i < LEN; ++i)
    Min[i] = INF; //初始化
int index[LEN]; //用来记录最小路径的点
for(int i = 1; i <= n; ++i)
{
    if(dis[i][0] < Min[Find(i)])
    {
        Min[Find(i)] = dis[i][0];
        index[Find(i)] = i;
    }
}

//// cout<<se.size()<<endl;
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
for(int i = 1; i <= n; ++i)
{
    if(Min[i] != INF)
    {

```

```

        ans += Min[i];
        sign[index[i]][0] = sign[0][index[i]] = 1; //将这个最小路径的点与 park
        ↪ 相连
        m++;
    }
}
int MMin = ans;
for(int i = m + 1; i <= K; ++i) //从 m+1 到 K 求最小 i 度生成树
{
    me(vis);
    vis[0] = 1;
    Dfs(0);
    int select = -1; //select 用来记录选择哪个与 park 点相连是最小的
    int sum = INF;
    for(int i = 1; i <= n; ++i)
    {
        if(!sign[0][i] && dis[0][i] != INF)
        {
            if(dis[i][0] - Best[i] < sum)
            {
                select = i;
                sum = dis[i][0] - Best[i];
            }
        }
    }
    if(select == -1) //如果找不到，就跳出循环
        break;
    ans += sum;
    sign[select][0] = sign[0][select] = 1;
    MMin = min(MMin, ans);
    for(int i = select; i != 0; i = Father[i])
    {
        if(dis[Father[i]][i] == Best[select])
        {
            sign[i][Father[i]] = sign[Father[i]][i] = 0;
            break;
        }
    }
}
cout << ans << endl;

}
printf("Total miles driven: %d\n", MMin);
// cout << MMin << endl;
}
return 0;
}

```

4.5.4 4 次小生成树.cpp

```

#include<iostream>
#include<cstdio>
#include<cstring>
#include<string>
#include<algorithm>
#include<cmath>
#include<vector>

```

```

#include<queue>
#define ll long long
using namespace std;

int getint()
{
    int i=0,f=1;char c;
    for(c=getchar();(c<'0' || c>'9')&&c!='-';c=getchar());
    if(c=='-')f=-1,c=getchar();
    for(;c>='0'&&c<='9';c=getchar())i=(i<<3)+(i<<1)+c-'0';
    return i*f;
}

const int N=100005,M=300005;
struct node
{
    int x,y,w;
    inline friend bool operator < (const node &a,const node &b)
    {
        return a.w<b.w;
    }
}bian[M];
int n,m;
int id[N],fa[N][20],mx1[N][20],mx2[N][20],dep[N];
int tot,first[N],nxt[N<<1],to[N<<1],w[N<<1];
ll totlen,ans;
bool chs[M];

void add(int x,int y,int z)
{
    nxt[++tot]=first[x],first[x]=tot,to[tot]=y,w[tot]=z;
}

int find(int x)
{
    return id[x]==x?x:id[x]=find(id[x]);
}

void kruskal()
{
    for(int i=1;i<=n;i++)id[i]=i;
    sort(bian+1,bian+m+1);
    int cnt=0;
    for(int i=1;i<=m;i++)
    {
        int x=find(bian[i].x),y=find(bian[i].y);
        if(x!=y)
        {
            cnt++;
            totlen+=bian[i].w;
            chs[i]=true;
            add(bian[i].x,bian[i].y,bian[i].w);
            add(bian[i].y,bian[i].x,bian[i].w);
            id[y]=x;
            if(cnt==n-1)break;
        }
    }
}

```

```

    }
}

void dfs(int u)
{
    for(int i=1;i<20;i++)fa[u][i]=fa[fa[u][i-1]][i-1];
    for(int i=1;i<20;i++)mx1[u][i]=max(mx1[u][i-1],mx1[fa[u][i-1]][i-1]);
    for(int i=1;i<20;i++)
    {
        mx2[u][i]=max(mx2[u][i-1],mx2[fa[u][i-1]][i-1]);
        if(mx1[u][i-1]<mx1[fa[u][i-1]][i-1]&&mx2[u][i]<mx1[u][i-1])
            mx2[u][i]=mx1[u][i-1];
        if(mx1[u][i-1]>mx1[fa[u][i-1]][i-1]&&mx1[fa[u][i-1]][i-1]>mx2[u][i])
            mx2[u][i]=mx1[fa[u][i-1]][i-1];
    }
    for(int e=first[u];e;e=nxt[e])
    {
        int v=to[e];
        if(v==fa[u][0])continue;
        fa[v][0]=u;mx1[v][0]=w[e];
        dep[v]=dep[u]+1;
        dfs(v);
    }
}

int Find(int x,int y,int len)
{
    int Mx1=0,Mx2=0;
    if(dep[x]<dep[y])swap(x,y);
    int delta=dep[x]-dep[y];
    for(int i=19;i>=0;i--)
        if(delta&(1<<i))
        {
            if(Mx1>mx1[x][i]&&mx1[x][i]>Mx2)Mx2=mx1[x][i];
            if(Mx1<mx1[x][i])Mx2=max(Mx1,mx2[x][i]),Mx1=mx1[x][i];
            x=fa[x][i];
        }
    if(x==y)return Mx1==len?Mx2:Mx1;
    for(int i=19;i>=0;i--)
        if(fa[x][i]!=fa[y][i])
        {
            if(Mx1>mx1[x][i]&&mx1[x][i]>Mx2)Mx2=mx1[x][i];
            if(Mx1<mx1[x][i])Mx2=max(Mx1,mx2[x][i]),Mx1=mx1[x][i];
            x=fa[x][i];
            if(Mx1>mx1[y][i]&&mx1[y][i]>Mx2)Mx2=mx1[y][i];
            if(Mx1<mx1[y][i])Mx2=max(Mx1,mx2[y][i]),Mx1=mx1[y][i];
            y=fa[y][i];
        }
    if(Mx1>mx1[x][0]&&mx1[x][0]>Mx2)Mx2=mx1[x][0];
    if(Mx1<mx1[x][0])Mx2=max(Mx1,mx2[x][0]),Mx1=mx1[x][0];
    x=fa[x][0];
    if(Mx1>mx1[y][0]&&mx1[y][0]>Mx2)Mx2=mx1[y][0];
    if(Mx1<mx1[y][0])Mx2=max(Mx1,mx2[y][0]),Mx1=mx1[y][0];
    y=fa[y][0];
    return Mx1==len?Mx2:Mx1;
}

```



```

void solve(int e)
{
    int x=bian[e].x,y=bian[e].y,len=bian[e].w;
    int tmp=Find(x,y,len);
    ans=min(ans,totlen-tmp+len);
}

int main()
{
    //freopen("lx.in","r",stdin);
    n=getint(),m=getint();
    for(int i=1;i<=m;i++)
    {
        bian[i].x=getint();
        bian[i].y=getint();
        bian[i].w=getint();
    }
    kruskal();
    dfs(1);
    ans=1e18;
    for(int i=1;i<=m;i++)
        if(!chs[i])solve(i);
    printf("%lld",ans);
}

```

4.6 最短路

4.6.1 1 Dijkstra.cpp

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
//typedef unsigned long long ULL;
//const int prime = 999983;
//const int INF = 0x7FFFFFFF;
//const LL INFF = 0x7FFFFFFFFFFFFFFF;
//const double pi = acos(-1.0);
//const double inf = 1e18;
//const double eps = 1e-6;
//const LL mod = 1e9 + 7;
//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;}
//LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
//int dr[2][4] = {1,-1,0,0,0,0,-1,1};
//typedef pair<int,int> P;
struct Dijkstra{
    #define maxn 1234
    #define INF 123456789

```

```

int n,m;
int s,t;

int dis[maxn],M[maxn][maxn];
bool vis[maxn];
void init(){
    scanf("%d %d %d %d",&n,&m,&s,&t);
    int u,v,c;
    for(int i = 1;i <= n; ++i)
        for(int j = 1;j <= n; ++j)
            if(i != j)
                M[i][j] = INF;
    for(int i = 0;i < m; ++i){
        scanf("%d %d %d",&u,&v,&c);
        M[u][v] = M[v][u] = min(M[u][v],c);
    }
}

void solve(){
    memset(vis,0,sizeof(vis));
    fill(dis+1,dis+n+1,INF);
    dis[s] = 0;
    for(int i = 1;i <= n; ++i){
        int x,Min = INF;
        for(int j = 1;j <= n; ++j){
            if(!vis[j]&&dis[j] <= Min)
                Min = dis[x=j];
        }
        vis[x] = 1;

        for(int j = 1;j <= n; ++j){
            if(!vis[j]&&dis[j] > dis[x]+M[x][j])
                dis[j] = dis[x]+M[x][j];
        }
    }

    printf("%d\n",dis[t]);
}

};
Dijkstra Dij;
int main(void)
{
    Dij.init();
    Dij.solve();

    return 0;
}
// 加了堆优化的 dij

```

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define For(i,a,b) for(int i = a; i < b; ++i)

```

```

#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;

int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Edge{
    int u,v,d;
    Edge(int uu,int vv,int dd):u(uu),v(vv),d(dd){
    }
};
struct Dijkstra{
    #define maxn 123456
    #define INF 123456789
    int N,M,S,T;

    typedef pair<int,int> P;
    vector<Edge> edges;
    vector<int> G[maxn];
    bool done[maxn];
    int d[maxn];
    int p[maxn];
    void init(){
        for(int i = 1;i <= N; ++i) G[i].clear();
        edges.clear();
        scanf("%d %d %d %d",&N,&M,&S,&T);
        //      cout<<N<<M<<S<<T<<endl;
        int u,v,w;
        for(int i = 1;i <= M; ++i){
            scanf("%d %d %d",&u,&v,&w);
            AddEdge(u,v,w);
            AddEdge(v,u,w);
        }

        void AddEdge(int u,int v,int d){
            edges.push_back(Edge(u,v,d));
            int m = edges.size();
            G[u].push_back(m-1);
        }

        void solve(){
            priority_queue<P,vector<P>,greater<P>> Q;
            for(int i = 1;i <= N; ++i) d[i] = INF;
            d[S] = 0;
            memset(done,0,sizeof(done));
            Q.push(P(0,S));
            while(!Q.empty()){
                P x = Q.top(); Q.pop();
                int u = x.second;
                if(done[u]) continue;
                done[u] = true;
                for(int i = 0;i < G[u].size(); ++i){
                    Edge &e = edges[G[u][i]];
                    if(!done[e.v]&&d[e.v] > d[u]+e.d){

```

```

        d[e.v] = d[u]+e.d;
        p[e.v] = G[u][i];
        Q.push(P(d[e.v],e.v));
    }
}

    }

    printf("%d\n",d[T]);
}

};
Dijkstra Dij;
int main(void)
{
    Dij.init();
    Dij.solve();

    return 0;
}

```

4.6.2 2 Bellman-ford.cpp

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;
}
LL gcd(LL a,LL b) {
    return b?gcd(b,a%b):a;
}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Edge{

```

```

    int from,to,dist;
    Edge(int u,int v,int d):from(u),to(v),dist(d){
    }
};

struct Bellman_ford {
    #define maxn 1234567
    bool inq[maxn]; // 用来记录入队次数
    int cnt[maxn], d[maxn], p[maxn];
    // 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
    int n,m;
    int s,t;
    vector<Edge> edges;
    vector<int> G[maxn];
    void AddEdge(int from,int to,int dist){
        edges.push_back(Edge(from,to,dist));
        edges.push_back(Edge(to,from,dist));
        int m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }
    void init(){

        scanf("%d %d %d %d",&n,&m,&s,&t);
        int u,v,c;
        for(int i = 0; i < m; ++i){
            scanf("%d %d %d",&u,&v,&c);
            AddEdge(u,v,c);
        }
        /// cout<<"test"<<endl;
    }
    bool bellman_ford() {
        queue<int> Q;
        memset(inq,0,sizeof(inq));
        memset(cnt,0,sizeof(cnt));
        for(int i = 1; i <= n; ++i)
            d[i] = INF;
        d[s] = 0;
        inq[s] = true;
        Q.push(s);

        while(!Q.empty()) {
            int u = Q.front();
            Q.pop();
            inq[u] = false;
            for(int i = 0; i < G[u].size(); ++i) {
                Edge &e = edges[G[u][i]];
                if(d[u] < INF&& d[e.to] > d[u]+e.dist) {
                    d[e.to] = d[u]+e.dist;
                    p[e.to] = G[u][i];
                    if(!inq[e.to]) {
                        Q.push(e.to);
                        inq[e.to] = true;
                        if(++cnt[e.to] > n)
                            return false;
                    }
                }
            }
        }
    }
};

```

```

    }
}
printf("%d\n",d[t]);

}

};
Bellman_ford bell;
int main(void) {
    bell.init();
    bell.bellman_ford();

    return 0;
}

```

4.6.3 3 floyed.cpp

```

// https://hihocoder.com/problemset/problem/1089?sid=1348128
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF = 0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Floyd{
    // 复杂度  $O(n^3)$ 
    #define maxn 300
    int d[maxn][maxn];
    int n,m;
    void init(void){
        scanf("%d %d",&n,&m);
        for(int i = 1;i <= n ;++i)
            for(int j = 1;j <= n; ++j)
                if(i != j)
                    d[i][j] = INF;
        int u,v,c;
        for(int i = 0;i < m; ++i){
            scanf("%d %d %d",&u,&v,&c);
            d[u][v] = d[v][u] = min(d[v][u],c);
        }
    }
}

```

```

    }
    void floyd(void){
        for(int k = 1; k <= n; ++k)
            for(int i = 1; i <= n; ++i)
                for(int j = 1; j <= n; ++j)
                    if(d[i][k] < INF && d[j][k] < INF)
                        d[i][j] = min(d[i][j], d[i][k] + d[j][k]);
    }
    void print(void){
        for(int i = 1; i <= n; ++i){
            for(int j = 1; j <= n; ++j)
                printf("%d%c", d[i][j], " \n"[j==n]);
        }
    }
};

Floyd floyd;
int main(void)
{
    floyd.init();
    floyd.floyd();
    floyd.print();

    return 0;
}

```

4.6.4 堆优化的有限队列.cpp

```

#include <cstdio>
#include <iostream>
#include <algorithm>
#include <ext/pb_ds/priority_queue.hpp>
#define N 1000010
#define M 10000010
#define inf 1000000000000000ll

using namespace std;
using namespace __gnu_pbds;

typedef long long ll;
typedef pair<ll, int> pairs;
typedef __gnu_pbds::priority_queue<pairs, greater<pairs>, pairing_heap_tag> heap;

heap Q;
heap::point_iterator p[N];
int n, m, t, cnt;
ll rxa, rxc, rya, ryc, rp;
int G[N], vis[N];
ll dis[N];
struct edge{
    int t, nx;
    ll w;
}E[M];

inline void InsertT(int x, int y, ll w){

```

```

    E[++cnt].t=y;E[cnt].nx=G[x];E[cnt].w=w;G[x]=cnt;
}

inline void dijkstra(){
    for(int i=1;i<=n;i++) dis[i]=inf;
    dis[1]=0; vis[1]=0; p[1]=Q.push(pairs(0,1));
    while(!Q.empty()){
        int x=Q.top().second; Q.pop(); vis[x]=0;
        for(int i=G[x];i;i=E[i].nx)
            if(dis[E[i].t]>dis[x]+E[i].w){
                dis[E[i].t]=dis[x]+E[i].w;
                if(vis[E[i].t]) Q.modify(p[E[i].t],pairs(dis[E[i].t],E[i].t));
                else p[E[i].t]=Q.push(pairs(dis[E[i].t],E[i].t)),vis[E[i].t]=1;
            }
    }
}

int main(){
    freopen("1.in","r",stdin);
    freopen("1.out","w",stdout);
    scanf("%d%d%d%d%d%d%d", &n, &m, &t, &rx, &rx, &ry, &ry, &rp);
    ll x=0,y=0,z=0,a,b;
    for(int i=1;i<=t;i++){
        x=(x*rx+rx)%rp;
        y=(y*ry+ry)%rp;
        a=min(x%n+1,y%n+1);
        b=max(y%n+1,y%n+1);
        Insert(a,b,1e8-100*a);
    }
    for(int i=1;i<=m-t;i++){
        scanf("%lld%lld%lld",&x,&y,&a);
        Insert(x,y,a);
    }
    dijkstra();
    printf("%lld\n",dis[n]);
}

```

5 数学

5.1 3 FWT 模板.cpp

// 异或

```

void FWT(int *a,int N,int opt){
    const int inv2 = qpow(2,mod-2);
    // 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
    for(int i = 1; i < N; i <= 1){
        for(int p = i<<1, j = 0; j < N; j += p){
            for(int k = 0; k < i; ++k){
                int X = a[j+k], Y = a[i+j+k];
                a[j+k] = (X+Y)%mod;
                a[i+j+k] = (X-mod+Y)%mod;
                if(opt == -1) a[j+k] = 1ll*a[j+k]*inv2%mod, a[i+j+k] =
                    1ll*a[i+j+k]*inv2%mod;
            }
        }
    }
}

```



```

    }
}
}

```

或

```

if(opt == 1) F[i+j+k] = (F[i+j+k]+F[j+k]) %mod;
else          F[i+j+k] = (F[i+j+k+mod-F[j+k]]) %mod;

```

和

```

if(opt == 1) F[j+k] = (F[j+k]+F[i+j+k]) %mod;
else          F[j+k] = (F[j+k] +mod-F[i+j+k])%mod;

```

5.2 4 单纯形法.cpp

```

// UVa10498 Happiness!
// Rujia Liu
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cassert>
using namespace std;

// 改进单纯性法的实现
// 参考：http://en.wikipedia.org/wiki/Simplex\_algorithm
// 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
// 第 i 个约束为  $a[i][0]*x[0] + a[i][1]*x[1] + \dots \leq a[i][n]$ 
// 目标为  $\max(a[m][0]*x[0] + a[m][1]*x[1] + \dots + a[m][n-1]*x[n-1] - a[m][n])$ 
// 注意：变量均有非负约束  $x[i] \geq 0$ 
const int maxm = 500; // 约束数目上限
const int maxn = 500; // 变量数目上限
const double INF = 1e100;
const double eps = 1e-10;

struct Simplex {
    int n; // 变量个数
    int m; // 约束个数
    double a[maxm][maxn]; // 输入矩阵
    int B[maxm], N[maxn]; // 算法辅助变量

    void pivot(int r, int c) {
        swap(N[c], B[r]);
        a[r][c] = 1 / a[r][c];
        for(int j = 0; j <= n; j++) if(j != c) a[r][j] *= a[r][c];
        for(int i = 0; i <= m; i++) if(i != r) {
            for(int j = 0; j <= n; j++) if(j != c) a[i][j] -= a[i][c] * a[r][j];
            a[i][c] = -a[i][c] * a[r][c];
        }
    }

    bool feasible() {
        for(;;) {
            int r, c;
            double p = INF;
            for(int i = 0; i < m; i++) if(a[i][n] < p) p = a[r = i][n];
            if(p > -eps) return true;
            p = 0;

```

```

    for(int i = 0; i < n; i++) if(a[r][i] < p) p = a[r][c = i];
    if(p > -eps) return false;
    p = a[r][n] / a[r][c];
    for(int i = r+1; i < m; i++) if(a[i][c] > eps) {
        double v = a[i][n] / a[i][c];
        if(v < p) { r = i; p = v; }
    }
    pivot(r, c);
}
}

// 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
int simplex(int n, int m, double x[maxn], double& ret) {
    this->n = n;
    this->m = m;
    for(int i = 0; i < n; i++) N[i] = i;
    for(int i = 0; i < m; i++) B[i] = n+i;
    if(!feasible()) return 0;
    for(;;) {
        int r, c;
        double p = 0;
        for(int i = 0; i < n; i++) if(a[m][i] > p) p = a[m][c = i];
        if(p < eps) {
            for(int i = 0; i < n; i++) if(N[i] < n) x[N[i]] = 0;
            for(int i = 0; i < m; i++) if(B[i] < n) x[B[i]] = a[i][n];
            ret = -a[m][n];
            return 1;
        }
        p = INF;
        for(int i = 0; i < m; i++) if(a[i][c] > eps) {
            double v = a[i][n] / a[i][c];
            if(v < p) { r = i; p = v; }
        }
        if(p == INF) return -1;
        pivot(r, c);
    }
}
};

////////// 题目相关
#include<cmath>
Simplex solver;

int main() {
    int n, m;
    while(scanf("%d%d", &n, &m) == 2) {
        for(int i = 0; i < n; i++) scanf("%lf", &solver.a[m][i]); // 目标函数
        solver.a[m][n] = 0; // 目标函数常数项
        for(int i = 0; i < m; i++)
            for(int j = 0; j < n+1; j++)
                scanf("%lf", &solver.a[i][j]);
        double ans, x[maxn];
        assert(solver.simplex(n, m, x, ans) == 1);
        ans *= m;
        printf("Nasa can spend %d taka.\n", (int)floor(ans + 1 - eps));
    }
}

```

```

    return 0;
}

```

5.3 5. 线性基.cpp

```

#include<bits/stdc++.h>
#define reg register
using namespace std;
typedef long long LL;
const int MN=60;
LL a[61],tmp[61];
bool flag;
void ins(LL x){
    for(reg int i=MN;~i;i--){
        if(x&(1LL<<i))
            if(!a[i]){a[i]=x;return;}
            else x^=a[i];
    }
    flag=true;
}
bool check(LL x){
    for(reg int i=MN;~i;i--){
        if(x&(1LL<<i))
            if(!a[i])return false;
            else x^=a[i];
    }
    return true;
}
LL qmax(LL res=0){
    for(reg int i=MN;~i;i--){
        res=max(res,res^a[i]);
    }
    return res;
}
LL qmin(){
    if(flag)return 0;
    for(reg int i=0;i<=MN;i++){
        if(a[i])return a[i];
    }
}
LL query(LL k){
    reg LL res=0;reg int cnt=0;
    k-=flag;if(!k)return 0;
    for(reg int i=0;i<=MN;i++){
        for(int j=i-1;~j;j--){
            if(a[i]&(1LL<<j))a[i]^=a[j];
            if(a[i])tmp[cnt++]=a[i];
        }
        if(k>=(1LL<<cnt))return -1;
        for(reg int i=0;i<cnt;i++){
            if(k&(1LL<<i))res^=tmp[i];
        }
    }
    return res;
}
int main(){
    int n;LL x;scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%lld",&x),ins(x);
    printf("%lld\n",qmax());
    return 0;
}

```

5.4 BM.cpp

//O(n²) n 是传入的数
//输入的 n 是第几个数

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

```

        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        rep(n,0,SZ(s)) {
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
};
int gao(VI a,ll n){
    VI c=BM(a);
    c.erase(c.begin());
    rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
    return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
}
};
int main()
{
    int t;
    scanf("%d",&t);
    while(t--){
        scanf("%lld",&n);
        vector<int>v
            ↪ {2,3,4,5,7,9,12,15,19,24,31,40,52,67,86,110,141,181,233,300,386,496,637};
        // n = v.size();
        //
        ↪ v.push_back({2,3,4,5,7,9,12,15,19,24,31,40,52,67,86,110,141,181,233,300,386,496,637});
        ↪ //至少 8 项, 越多越好。
        printf("%lld\n",linear_seq:gao(v,n-1)%mod);
    }
}

```

5.5 Combinatorial mathematics

5.5.1 康托展开.cpp

```

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
    int ans=0,sum=0;
    for(int i=1;i<n;i++){

```

```

        for(int j=i+1;j<=n;j++)
            if(a[j]<a[i])
                sum++;
        ans+=sum*factorial[n-i]; //累积
        sum=0; //计数器归零
    }
    return ans+1;
}

```

```

static const int FAC[] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880}; // 阶乘

```

//康托展开逆运算

```

void decantor(int x, int n)
{
    vector<int> v; // 存放当前可选数
    vector<int> a; // 所求排列组合
    for(int i=1;i<=n;i++)
        v.push_back(i);
    for(int i=n;i>=1;i--)
    {
        int r = x % FAC[i-1];
        int t = x / FAC[i-1];
        x = r;
        sort(v.begin(),v.end()); // 从小到大排序
        a.push_back(v[t]); // 剩余数里第 t+1 个数为当前位
        v.erase(v.begin()+t); // 移除选做当前位的数
    }
}

```

5.6 FFT

5.6.1 FFT.cpp

```

const double PI = acos(-1.0);
struct Complex
{
    double r,i;
    Complex(double _r = 0,double _i = 0){
        r = _r; i = _i;
    }
    Complex operator +(const Complex &b) {
        return Complex(r+b.r,i+b.i);
    }
    Complex operator -(const Complex &b) {
        return Complex(r-b.r,i-b.i);
    }
    Complex operator *(const Complex &b){
        return Complex(r*b.r-i*b.i,r*b.i+i*b.r);
    }
};

```

```

void FFT(Complex y[],int n ,int on)
{
    for(int i = 0, j = 0; i < n; i++) {
        if(j > i) swap(y[i], y[j]);
    }
}

```

```

    int k = n;
    while(j & (k >>= 1)) j &= ~k;
        j |= k;
}
for(int h = 2; h <= n; h <<= 1){
    Complex wn(cos(-on*2*PI/h), sin(-on*2*PI/h));
    for(int j = 0; j < n; j += h){
        Complex w(1,0);
        for(int k = j; k < j+h/2; k++){
            Complex u = y[k];
            Complex t = w*y[k+h/2];
            y[k] = u+t;
            y[k+h/2] = u-t;
            w = w*wn;
        }
    }
}
if(on == -1)
    for(int i = 0; i < n; i++)
        y[i].r /= n;
}

```

5.6.2 kuangbin.cpp

```

#include <stdio.h>
#include <iostream>
#include <string.h>
#include <algorithm>
#include <math.h>
using namespace std;

const double PI = acos(-1.0);
struct complex
{
    double r,i;
    complex(double _r = 0, double _i = 0)
    {
        r = _r; i = _i;
    }
    complex operator +(const complex &b)
    {
        return complex(r+b.r, i+b.i);
    }
    complex operator -(const complex &b)
    {
        return complex(r-b.r, i-b.i);
    }
    complex operator *(const complex &b)
    {
        return complex(r*b.r-i*b.i, r*b.i+i*b.r);
    }
};

void change(complex y[], int len)
{
    int i, j, k;
    for(i = 1, j = len/2; i < len-1; i++)

```

```

{
    if(i < j)swap(y[i],y[j]);
    k = len/2;
    while( j >= k)
    {
        j -= k;
        k /= 2;
    }
    if(j < k)j += k;
}
}
void fft(complex y[],int len,int on)
{
    change(y,len);
    for(int h = 2;h <= len;h <= 1)
    {
        complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
        for(int j = 0;j < len;j += h)
        {
            complex w(1,0);
            for(int k = j;k < j+h/2;k++)
            {
                complex u = y[k];
                complex t = w*y[k+h/2];
                y[k] = u+t;
                y[k+h/2] = u-t;
                w = w*wn;
            }
        }
    }
    if(on == -1)
        for(int i = 0;i < len;i++)
            y[i].r /= len;
}

```

```

const int MAXN = 400040;
complex x1[MAXN];
int a[MAXN/4];
long long num[MAXN];//100000*100000 会超 int
long long sum[MAXN];

```

```

int main()
{
    int T;
    int n;
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        memset(num,0,sizeof(num));
        for(int i = 0;i < n;i++)
        {
            scanf("%d",&a[i]);
            num[a[i]]++;
        }
        sort(a,a+n);
    }
}

```



```

    int len1 = a[n-1]+1;
    int len = 1;
    while( len < 2*len1 )len <= 1;
    for(int i = 0;i < len1;i++)
        x1[i] = complex(num[i],0);
    for(int i = len1;i < len;i++)
        x1[i] = complex(0,0);
    fft(x1,len,1);
    for(int i = 0;i < len;i++)
        x1[i] = x1[i]*x1[i];
    fft(x1,len,-1);
    for(int i = 0;i < len;i++)
        num[i] = (long long)(x1[i].r+0.5);
    len = 2*a[n-1];
    //减掉取两个相同的组合
    for(int i = 0;i < n;i++)
        num[a[i]+a[i]]--;
    //选择的无序, 除以 2
    for(int i = 1;i <= len;i++)
    {
        num[i]/=2;
    }
    sum[0] = 0;
    for(int i = 1;i <= len;i++)
        sum[i] = sum[i-1]+num[i];
    long long cnt = 0;
    for(int i = 0;i < n;i++)
    {
        cnt += sum[len]-sum[a[i]];
        //减掉一个取大, 一个取小的
        cnt -= (long long)(n-1-i)*i;
        //减掉一个取本身, 另外一个取其它
        cnt -= (n-1);
        //减掉大于它的取两个的组合
        cnt -= (long long)(n-1-i)*(n-i-2)/2;
    }
    //总数
    long long tot = (long long)n*(n-1)*(n-2)/6;
    printf("%.7lf\n",(double)cnt/tot);
}
return 0;
}

```

5.6.3 lrj.cpp

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF = 0x7FFFFFFFFFFFFFFF;
//const double pi = acos(-1.0);

```

```

const double inf = 1e18;
const double eps = 1e-6;
const LL      mod = 1e9 + 7;
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
// UVa12298 Super Poker II
// Rujia Liu

const long double PI = acos(0.0) * 2.0;

typedef complex<double> CD;

// Cooley-Tukey 的 FFT 算法，迭代实现。inverse = false 时计算逆 FFT
inline void FFT(vector<CD> &a, bool inverse) {
    int n = a.size();
    // 原地快速 bit reversal
    for(int i = 0, j = 0; i < n; i++) {
        if(j > i) swap(a[i], a[j]);
        int k = n;
        while(j & (k >>= 1)) j &= ~k;
        j |= k;
    }

    double pi = inverse ? -PI : PI;
    for(int step = 1; step < n; step <= 1) {
        // 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”
        double alpha = pi / step;
        // 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
        // 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
        //   X[k]
        // 蝴蝶操作参考：http://en.wikipedia.org/wiki/Butterfly\_diagram
        for(int k = 0; k < step; k++) {
            // 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。
            // 有更快更精确的递推方法，为了清晰起见这里略去
            CD omegak = exp(CD(0, alpha*k));
            for(int Ek = k; Ek < n; Ek += step < 1) { // Ek 是某次 DFT 合并中 E[k] 在原始序
                //   列中的下标
                int Ok = Ek + step; // Ok 是该 DFT 合并中 O[k] 在原始序列中的下标
                CD t = omegak * a[Ok]; // 蝴蝶操作：x1 * omega~k
                a[Ok] = a[Ek] - t; // 蝴蝶操作：y1 = x0 - t
                a[Ek] += t; // 蝴蝶操作：y0 = x0 + t
            }
        }
    }

    if(inverse)
        for(int i = 0; i < n; i++) a[i] /= n;
}

// 用 FFT 实现的快速多项式乘法
inline vector<double> operator * (const vector<double>& v1, const vector<double>& v2)
    ↪ {
    int s1 = v1.size(), s2 = v2.size(), S = 2;
    while(S < s1 + s2) S <= 1;
    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
    for(int i = 0; i < s1; i++) a[i] = v1[i];

```

```

FFT(a, false);
for(int i = 0; i < s2; i++) b[i] = v2[i];
FFT(b, false);
for(int i = 0; i < S; i++) a[i] *= b[i];
FFT(a, true);
vector<double> res(s1 + s2 - 1);
for(int i = 0; i < s1 + s2 - 1; i++) res[i] = a[i].real(); // 虚部均为 0
return res;
}

```

5.7 Lagrange-poly

5.7.1 template.cpp

// 适用范围, 求 n 次多项式第 x 项的值

```

namespace polysum {
    #define rep(i,a,n) for (int i=a;i<n;i++)
    #define per(i,a,n) for (int i=n-1;i>=a;i--)
    const int D=1e6+10;
    ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
    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;}
    //.....
    // 已知  $a_i$  的  $d$  次多项式, 求第  $n$  项
    ll calcn(int d,ll *a,ll n) { //  $a[0] \dots a[d] \quad a[n]$ 
        if (n<=d) return a[n];
        p1[0]=p2[0]=1;
        rep(i,0,d+1) {
            ll t=(n-i+mod)%mod;
            p1[i+1]=p1[i]*t%mod;
        }
        rep(i,0,d+1) {
            ll t=(n-d+i+mod)%mod;
            p2[i+1]=p2[i]*t%mod;
        }
        ll ans=0;
        rep(i,0,d+1) {
            ll t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
            if ((d-i)&1) ans=(ans-t+mod)%mod;
            else ans=(ans+t)%mod;
        }
        return ans;
    }
    // 初始化, 初始化的时候记得将  $D$  的值
    void init(int M) {
        f[0]=f[1]=g[0]=g[1]=1;
        rep(i,2,M+5) f[i]=f[i-1]*i%mod;
        g[M+4]=powmod(f[M+4],mod-2);
        per(i,1,M+4) g[i]=g[i+1]*(i+1)%mod;
    }
    // 已知  $a_i$ , 并且知道  $a_i$  是  $m$  次多项式
    ll polysum(ll m,ll *a,ll n) { //  $a[0] \dots a[m] \quad \sum_{i=0}^n a[i]$ 
        ll b[D];

```

```

    ll b[D];
    for(int i=0;i<=m;i++) b[i]=a[i];
    b[m+1]=calcn(m,b,m+1);
    rep(i,1,m+2) b[i]=(b[i-1]+b[i])%mod;
    return calcn(m+1,b,n); // m 次多项式的和是 m+1 次多项式
}

ll qpolysum(ll R,ll n,ll *a,ll m) {
    // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
    if (R==1) return polysum(n,a,m);
    a[m+1]=calcn(m,a,m+1);
    ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;
    h[0][0]=0;h[0][1]=1;
    rep(i,1,m+2) {
        h[i][0]=(h[i-1][0]+a[i-1])*r%mod;
        h[i][1]=h[i-1][1]*r%mod;
    }
    rep(i,0,m+2) {
        ll t=g[i]*g[m+1-i]%mod;
        if (i&1) p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+mod)%mod;
        else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
    }
    c=powmod(p4,mod-2)*(mod-p3)%mod;
    rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]*c)%mod;
    rep(i,0,m+2) C[i]=h[i][0];
    ans=(calcn(m,C,n)*powmod(R,n)-c)%mod;
    if (ans<0) ans+=mod;
    return ans;
}
} // polysum::init();

```

5.8 三分.cpp

```

//1142 : 三分·三分求极值
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define For(i,a,b) for(int i = a; i < b; ++i)
#define IOS ios::sync_with_stdio(false)
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-9;
const LL mod = 1e9 + 7;
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;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};

```

```

typedef pair<int,int> P;
double a,b,c,X,Y;
double f(double xx){
    return a*xx*xx+b*xx+c;
}
double d(double x){
    double t = a*x*x+b*x+c;
    return sqrt((X-x)*(X-x)+(t-Y)*(t-Y));
}
int main(void)
{
    cin>>a>>b>>c>>X>>Y;

    double l,r,lm,rm;
    l = -200.0,r = 200.0;
    while(r - l >= eps){
        lm = (r+l)/2;
        rm = (r+lm)/2;
        if(d(rm)<d(lm))
            l = lm;
        else
            r = rm;
    }

    printf("%.3lf\n",d(l));

    return 0;
}

```

5.9 博弈

5.9.1 2. 威佐夫博弈.cpp

```

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

```

5.9.2 3 Nim 积.cpp

/* 在一个二维平面中，有 n 个灯亮着并告诉你坐标，
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，
然后我们把四个角落的灯状态反转，不能操作为败

```
*/  
#include<set>  
#include<map>  
#include<stack>  
#include<cmath>  
#include<queue>  
#include<vector>  
#include<cstdio>  
#include<cstring>  
#include<iostream>  
#include<algorithm>  
typedef long long ll;  
const int maxn = 1e6 + 10;  
const int seed = 131;  
const ll MOD = 1e9 + 7;  
const int INF = 0x3f3f3f3f;  
using namespace std;  
int m[2][2] = {0, 0, 0, 1};  
int Nim_Mul_Power(int x, int y){  
    if(x < 2) return m[x][y];  
    int a = 0;  
    for(; ; a++){  
        if(x >= (1 << (1 << a)) && x < (1 << (1 << (a + 1))))  
            break;  
    }  
    int m = 1 << (1 << a);  
    int p = x / m, s = y / m, t = y % m;  
    int d1 = Nim_Mul_Power(p, s);  
    int d2 = Nim_Mul_Power(p, t);  
    return (m * (d1 ^ d2)) ^ Nim_Mul_Power(m / 2, d1);  
}  
int Nim_Mul(int x, int y){  
    if(x < y) return Nim_Mul(y, x);  
    if(x < 2) return m[x][y];  
    int a = 0;  
    for(; ; a++){  
        if(x >= (1 << (1 << a)) && x < (1 << (1 << (a + 1))))  
            break;  
    }  
    int m = 1 << (1 << a);  
    int p = x / m, q = x % m, s = y / m, t = y % m;  
    int c1 = Nim_Mul(p, s), c2 = Nim_Mul(p, t) ^ Nim_Mul(q, s), c3 = Nim_Mul(q, t);  
    return (m * (c1 ^ c2)) ^ c3 ^ Nim_Mul_Power(m / 2, c1);  
}  
int main(){  
    int T;  
    scanf("%d", &T);  
    int ans;  
    while(T--){  
        ans = 0;  
        int n, x, y;
```

```

scanf("%d", &n);
while(n--){
    scanf("%d%d", &x, &y);
    ans ^= Nim_Mul(x, y);
}
if(ans)
    printf("Have a try, lxhgww.\n");
else
    printf("Don't waste your time.\n");
}
return 0;
}

```

5.9.3 4 K 倍动态减法.cpp

/*
 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，
 则下一个人可以取 1 到 $k*m$ 个，取完最后一个为胜，
 问先手是否会胜，如果会胜输出第一次取几个。

```

**/
const int maxn = 2e6+100;
int a[maxn], b[maxn];
int main(void)
{
    int T;
    cin >> T;
    for(int kase = 1; kase <= T; ++kase){
        int n, k;
        cin >> n >> k;
        a[0] = 1, b[0] = 1;
        int i = 0, j = 0;
        while(a[i] < n){
            i++;
            a[i] = b[i-1] + 1;
            if(a[j+1] * k < a[i]) j++;
            if(a[j] * k < a[i]) b[i] = b[j] + a[i];
            else b[i] = a[i];
        }
        printf("Case %d: ", kase);
        if(a[i] == n) {
            puts("lose");
            continue;
        }
        // i--;
        while(i >= 0){
            if(n - a[i] > 0)
                n -= a[i];
            if(n == a[i]) break;
            i--;
        }
        printf("%d\n", n);
    }

    return 0;
}

```

5.9.4 5 海盗分金问题.cpp

```
/*
A Puzzle for Pirates HDU - 1538
*/

int solve(int n,int m,int q){
    if(n <= 2*m+2){
        if(q == n){
            return m-(n-1)/2;
        }
        else{
            if(q % 2== n%2) return 1;
            else return 0;
        }
    }
    else{
        if(q <= 2*m+2) return 0;
        if(n == q)
        {
            LL t = 2*m+2;
            while(t < n)
                t = 2*(t-m);
            if(t == n) return 0;
            else return -1;
        }
        else{
            LL t = 2*m+2;
            while(t < q)
                t = 2*(t-m);
            if(t <= n) return 0;
            else return -1;
        }
    }
}

int main(void)
{
    int T;
    cin>>T;
    while(T--){
        LL n,m,q;
        cin>>n>>m>>q;
        LL ans = solve(n,m,q);

        if(ans == -1) puts("Thrown");
        else printf("%lld\n",ans);
    }

    return 0;
}
```

5.9.5 6 Green Hackbush.cpp

```
// N 个点, M 条边
```



```

#include<bits/stdc++.h>
using namespace std;
#define min(x,y) ((x)<(y))? (x):(y)

int Cases,N,M;
vector< list<int> > G,G2;
vector<int> GV;
vector<int> visited,from,time_disc,time_up;
int DFStime;

void DFS_Visit(int v){
    int edges_to_parent=0;
    visited[v]=1; time_disc[v]=time_up[v]=++DFStime;
    for (list<int>::iterator start=G[v].begin();start!=G[v].end();start++) {
        if (!visited[*start]) { from[*start]=v; DFS_Visit(*start);
            ↪ time_up[v]=min(time_up[v],time_up[*start]); }
        else {
            if ((*start)!=from[v]) { time_up[v]=min(time_up[v],time_disc[*start]); }
            else {
                if (edges_to_parent) { time_up[v]=min(time_up[v],time_disc[*start]); }
                edges_to_parent++;
            }
        }
    }
}

void FindBridges(void){
    time_disc.clear(); time_up.clear(); visited.clear(); from.clear();
    visited.resize(N+3,0); time_disc.resize(N+3,0); time_up.resize(N+3,0);
    ↪ from.resize(N+3,0);
    from[1]=1; DFStime=0;
    DFS_Visit(1);
}

int IsBridge(int v_lo, int v_high) {
    if (v_high!=from[v_lo]) return 0;
    return ( time_disc[v_lo]==time_up[v_lo] );
}

void ContractGraph(void){
    vector<int> color(N+3,0);
    int colors=1;
    color[1]=1;

    list<int> Q;
    Q.clear(); Q.push_back(1);
    while (!Q.empty()) {
        int where=Q.front(); Q.pop_front();
        for (list<int>::iterator it=G[where].begin(); it!=G[where].end(); it++) if
            ↪ (!color[*it]) {
                if (IsBridge(*it,where)) color[*it]=++colors; else color[*it]=color[where];
                visited[*it]=1; Q.push_back(*it);
            }
    }

    G2.clear(); G2.resize(N+3);

```

```

    for (int i=1;i<=N;i++)
        for (list<int>::iterator it=G[i].begin(); it!=G[i].end(); it++)
            G2[color[i]].push_back(color[*it]);
}

int GrundyValue(int v){
    int loops=0,gv=0;

    if (GV[v]!=-1) return GV[v]; GV[v]=1000000000;

    for (list<int>::iterator start=G2[v].begin(); start!=G2[v].end(); start++) {
        if ((*start)==v) loops++; else if (GV[*start]!=1000000000)
            ↪ gv^=(1+GrundyValue(*start));
    }
    loops/=2; if (loops%2) gv^=1;
    return GV[v]=gv;
}

int main(void){
    int v1,v2;
    // freopen("input.txt","r",stdin);
    // freopen("out.txt","w+",stdout);
    cin >> Cases;
    while (Cases--){
        // read graph dimensions
        cin >> N >> M;
        // read the graph
        G.clear(); G.resize(N+3);
        for (int i=0;i<M;i++){ cin >> v1 >> v2; G[v1].push_back(v2); G[v2].push_back(v1);
            ↪ }
        // collapse all circuits in the graph
        FindBridges();
        ContractGraph();
        // compute the SG value
        GV.clear(); for (int i=0;i<=N;i++) GV.push_back(-1);
        int result=GrundyValue(1);
        if (result) cout << "Alice\n"; else cout << "Bob\n"; // cout << result << "\n";

        //cout << result << "\n";
    }
    return 0;
}

typedef pair<int,int> P;
vector<P> edges;
// 边连通分量
const int maxn = 1000+100;
// const int maxm = 1e6+100
int pre[maxn];
int dfs_clock = 0;
vector<int> G[maxn];
vector<int> G2[maxn];
bool Is[maxn];
int low[maxn];

```

```

void init(){
    dfs_clock = 1;
    rep(i,1,maxn) G[i].clear(),G2[i].clear();
    me(low);
    me(pre);
    me(Is);
}

int dfs1(int u,int fa){
    int lowu = pre[u] = ++dfs_clock;
    int child = 0;
    for(int i = 0;i < (int)G[u].size(); ++i){
        int v = edges[G[u][i]].second;
        if(!pre[v]){
            child++;
            int lowv = dfs1(v,u);
            lowu = min(lowu,lowv);
            if(lowv >= pre[u]){
                // iscut[u]++;
                Is[G[u][i]] = 1;
            }
        }
        else if(pre[v] < pre[u] && v != fa){
            lowu = min(lowu,pre[v]);
        }
    }

    return low[u] = lowu;
}

// #define Debug

int belong[maxn];
int num[maxn];

void dfs(int u,int be){
    belong[u] = be;
    for(int i = 0;i < (int)G[u].size(); ++i){
        if(Is[G[u][i]])
            continue;
        int v = edges[G[u][i]].second;
        if(!belong[v])
            dfs(v,be);
    }
}

int SG(int u,int fa){
    int t = 0;
    for(int i = 0;i < (int)G2[u].size(); ++i){
        int v = G2[u][i];
        if(v==fa) continue;
        t ^= (SG(v,u)+1);
    }
    if(num[u]&1) t ^= 1;
    return t;
}

int main(void)
{
    int n,m,k;

```

```

while(cin>>n){
    int sum = 0;
    while(n--){
        init();
        edges.clear();
        me(belong);
        me(num);
        scanf("%d%d",&m,&k);
        rep(i,0,k){
            int u,v;
            scanf("%d%d",&u,&v);
            edges.push_back(P(u,v));
            edges.push_back(P(v,u));
            G[u].push_back(edges.size()-2);
            G[v].push_back(edges.size()-1);
        }
        dfs1(1,-1);

        int tot = 0;
        rep(i,1,m+1)
            if(!belong[i])
                dfs(i,++tot);
        // dfs(m+1,)
        for(int i = 0;i < (int)edges.size(); i += 2){
            int x = belong[edges[i].first];
            int y = belong[edges[i].second];
            if(x != y)
                G2[x].Pb(y),G2[y].Pb(x);
            else
                num[x]++;
        }

        // cout<<SG(1,-1)<<endl;
        sum ^= SG(1,-1);
    }
    if(sum)
        puts("Sally");
    else
        puts("Harry");
}
return 0;
}

```

5.9.6 7 反 nim 博弈.cpp

/*

先手必胜当且仅当：

- (1) 所有堆的石子数都为 1 且游戏的 SG 值为 0；
- (2) 有些堆的石子数大于 1 且游戏的 SG 值不为 0。

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，则

↪ 先手必胜当且仅当：

- (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；
- (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。

Every-SG 游戏规定，对于还没有结束的单一游戏，游戏者必须对该游戏进行一步决策；

Every-SG 游戏的其他规则与普通 SG 游戏相同

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。

*/

5.9.7 8 超自然数.cpp

```
//[POJ-2931]
// 超自然数求解不平等博弈问题
char ar[100];
bool b[100];
LL surreal(int n){
    LL k = 1;
    k <= 52;
    for(int i = 0; i < n; ++i){
        scanf("%s", ar);
        if(ar[0] == 'W')
            b[i] = 1;
        else
            b[i] = 0;
    }
    LL x = 0, i = 0;
    while(i < n && b[i] == b[0]){
        if(b[i]) x += k;
        else x -= k;
        i++;
    }
    k >>= 1;
    while(i < n){
        if(b[i])
            x += k;
        else
            x -= k;
        i++;
        k >>= 1;
    }
    return x;
}
int main(void)
{
    int T;
    cin >> T;
    while(T--){
        int n;
        char br[100];
        scanf("%s %d: ", br, &n);

        LL ans1 = 0, ans2 = 0;
        int a[3];
        rep(i, 0, 3) scanf("%d", &a[i]);
        rep(i, 0, 3) ans1 += surreal(a[i]);
        rep(i, 0, 3) scanf("%d", &a[i]);
        rep(i, 0, 3) ans2 += surreal(a[i]);
        // cout << ans1 << " " << ans2 << endl;
        printf("%s %d: ", br, n);
        if(ans1 >= ans2)
            puts("Yes");
        else
```

```

        puts("No");
    }

    return 0;
}

```

5.10 数论

5.10.1 1 加法.cpp

```

string add(string a,string b)
{
    string c;
    int len1=a.length();
    int len2=b.length();
    int len=max(len1,len2);
    for(int i=len1;i<len;i++)
        a="0"+a;
    for(int i=len2;i<len;i++)
        b="0"+b;
    int ok=0;
    for(int i=len-1;i>=0;i--)
    {
        char temp=a[i]+b[i]-'0'+ok;
        if(temp>'9')
        {
            ok=1;
            temp-=10;
        }
        else ok=0;
        c=temp+c;
    }
    if(ok) c="1"+c;
    return c;
}

```

5.10.2 1 逆元.cpp

```

// 欧几里得扩展
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
    if(b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    long long m = ex_gcd(b,a%b,y,x);
    y -= a/b * x;
    return m;
}

int main()
{
    long long a,b,x,y;
    cin>>a>>b; //求 a 关于 b 的逆元
    if(ex_gcd(a,b,x,y)==1)
        cout<<(x%b+b)%b<<endl;
}

```

```

    else
        cout<<"None"<<endl;
    return 0;
}
// 费马小定理求逆元
qpow(a,p-2,p);
// 逆元打表

int inv[10000];
int p;
cin>>p;
inv[1] = 1;
for(int i = 2;i < p; ++i)
{
    inv[i] = (p - p/i*inv[p%i]%p)%p;
}
for(int i = 1;i < p; ++i)
    cout<<inv[i]<<" ";
cout<<endl;
for(int i = 1;i < p; ++i)
    cout<<i * inv[i] % p<<" ";

// 快速阶乘逆元

const int maxn = 1e5+10;
long long fac[maxn],invfac[maxn];
void init(int n){
    fac[0] = 1;
    for(int i = 1;i <= n; ++i) fac[i] = fac[i-1]*i%mod;
    invfac[n] = qpow(fac[n],mod-2);
    for(int i = n-1;i >= 0; --i) invfac[i] = invfac[i+1]*(i+1)%mod;
}

```

5.10.3 2 减法.cpp

```

string sub(string a,string b)
{
    string c;
    bool ok=0;
    int len1=a.length();
    int len2=b.length();
    int len=max(len1,len2);
    for(int i=len1;i<len;i++)
        a="0"+a;
    for(int i=len2;i<len;i++)
        b="0"+b;
    if(a<b)
    {
        string temp=a;
        a=b;
        b=temp;
        ok=1;
    }
    for(int i=len-1;i>=0;i--)
    {
        if(a[i]<b[i])

```

```

    {
        a[i-1]-=1;
        a[i]+=10;
    }
    char temp=a[i]-b[i]+'0';
    c=temp+c;
}
int pos=0;
while(c[pos]!='0' && pos<len) pos++;
if(pos==len) return "0";
if(ok) return "-" + c.substr(pos);
return c.substr(pos);
}

```

5.10.4 3 乘法.cpp

```

string mul(string a,int b)
{
    string c;
    char s;
    int len=a.length();
    int ok=0;
    for(int i=len-1;i>=0;i--)
    {
        int temp=(a[i]-'0')*b+ok;
        ok=temp/10;
        s=temp%10+'0';
        c=s+c;
    }
    while(ok)
    {
        s=ok%10+'0';
        c=s+c;
        ok/=10;
    }
    return c;
}

```

5.10.5 4 除法.cpp

```

string div(string a,int b)
{
    string c;
    int len=a.length();
    int ans=0;
    char s;
    for(int i=0;i<len;i++)
    {
        ans=ans*10+a[i]-'0';
        s=ans/b+'0';
        ans%=b;
        c+=s;
    }
    int pos=0;
    while(pos<len && c[pos]!='0') pos++;
    if(pos==len) return "0";
}

```



```

    return c.substr(pos);
}

```

5.10.6 5. 蒙哥马利快速模.cpp

```

#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
↪ for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
// head

typedef unsigned long long u64;
typedef __int128_t i128;
typedef __uint128_t u128;
int _,k;
u64 A0,A1,M0,M1,C,M;

struct Mod64 {
    Mod64():n_(0) {}
    Mod64(u64 n):n_(init(n)) {}
    static u64 init(u64 w) { return reduce(u128(w) * r2); }
    static void set_mod(u64 m) {
        mod=m; assert(mod&1);
        inv=m; rep(i,0,5) inv*=2-inv*m;
        r2=-u128(m)%m;
    }
    static u64 reduce(u128 x) {
        u64 y=u64(x>>64)-u64((u128(u64(x)*inv)*mod)>>64);
        return ll(y)<0?y+mod:y;
    }
    Mod64& operator += (Mod64 rhs) { n_+=rhs.n_-mod; if (ll(n_)<0) n_+=mod; return
↪ *this; }
    Mod64 operator + (Mod64 rhs) const { return Mod64(*this)+=rhs; }
    Mod64& operator -= (Mod64 rhs) { n_-=rhs.n_; if (ll(n_)<0) n_+=mod; return *this; }
    Mod64 operator - (Mod64 rhs) const { return Mod64(*this)-=rhs; }
    Mod64& operator *= (Mod64 rhs) { n_ = reduce(u128(n_)*rhs.n_); return *this; }
    Mod64 operator * (Mod64 rhs) const { return Mod64(*this)*=rhs; }
    u64 get() const { return reduce(n_); }
    static u64 mod,inv,r2;
    u64 n_;
};
u64 Mod64::mod,Mod64::inv,Mod64::r2;

```

```

u64 pmod(u64 a,u64 b,u64 p) {
    u64 d=(u64)floor(a*(long double)b/p+0.5);
    ll ret=a*b-d*p;
    if (ret<0) ret+=p;
    return ret;
}

void bruteforce() {
    u64 ans=1;
    for (int i=0;i<=k;i++) {
        ans=pmod(ans,A0,M);
        u64 A2=pmod(M0,A1,M)+pmod(M1,A0,M)+C;
        while (A2>=M) A2-=M;
        A0=A1; A1=A2;
    }
    printf("%llu\n",ans);
}

int main() {
    for (scanf("%d",&_);_>0;_--) {
        scanf("%llu%llu%llu%llu%llu%llu%llu",&A0,&A1,&M0,&M1,&C,&M,&k);
        Mod64::set_mod(M);
        Mod64 a0(A0),a1(A1),m0(M0),m1(M1),c(C),ans(1),a2(0);
        for (int i=0;i<=k;i++) {
            ans=ans*a0;
            a2=m0*a1+m1*a0+c;
            a0=a1; a1=a2;
        }
        printf("%llu\n",ans.get());
    }
}

```

5.10.7 Euler.cpp

欧拉函数打表

$O(n \log(n))$

```

~ ~ ~

```

```

const int maxn = 1e6+100;
int phi[maxn],Prime[maxn];

void init2(int n){
    for(int i = 1;i <= n; ++i) phi[i] = i;
    for(int i = 2;i <= n; ++i){
        if(i == phi[i]){
            for(int j = i; j <= n; j += i) phi[j] = phi[j]/i*(i-1);
        }
    }
}

```

```

~ ~ ~

```

线性筛 $O(n)$

```

~ ~ ~

```

```

const int maxn = 1e6+100;
bool check[maxn];
int phi[maxn],Prime[maxn];

```

```

void init(int MAXN){
    int N = maxn-1;
    memset(check,false,sizeof(check));
    phi[1] = 1;
    int tot = 0;
    for(int i = 2;i <= N; ++i){
        if(!check[i]){
            Prime[tot++] = i;
            phi[i] = i-1;
        }
        for(int j = 0;j < tot; ++j){
            if(i*Prime[j] > N) break;
            check[i*Prime[j]] = true;
            if(i%Prime[j] == 0){
                phi[i*Prime[j]] = phi[i]*Prime[j];
                break;
            }
            else{
                phi[i*Prime[j]] = phi[i]*(Prime[j]-1);
            }
        }
    }
}
}

```

5.10.8 lucas , 组合数.cpp

```

LL qpow(LL a,LL b,LL m){
    LL ans = 1;
    a %= m;
    while(b > 0){
        if(b&1)
            ans = ans*a%m;
        a = a*a%m;
        b >>= 1;
    }
    return ans;
}
LL C(LL n,LL m,LL p){
    if(m > n) return 0;
    LL tmp1 = 1,tmp2 = 1;
    m = min(n-m,m);
    for(LL i = 1;i <= m; ++i){
        tmp1 = tmp1*(n-m+i)%p;
        tmp2 = tmp2*i%p;
    }
    return tmp1*qpow(tmp2,p-2,p)%p;
}
LL lucas(LL n, LL m, LL p){
    if(m == 0)
        return 1;
    return lucas(n/p,m/p,p)*C(n%p,m%p,p)%p;
}

```

5.10.9 miller-rabin-Pollard-rho.cpp

// 可以对一个 2^{63} 的素数进行判断。

可以分解比较大的数的因子。

```
#include<stdio.h>
#include<string.h>
#include<iostream>
#include<math.h>
#include<stdlib.h>
#include<time.h>
using namespace std;

typedef long long LL;
#define maxn 10000

LL factor[maxn];
int tot;
const int S=20;
LL muti_mod(LL a,LL b,LL c){    //返回 (a*b) mod c, a,b,c<2^63
    a%=c;
    b%=c;
    LL ret=0;
    while (b){
        if (b&1){
            ret+=a;
            if (ret>=c) ret-=c;
        }
        a<<=1;
        if (a>=c) a-=c;
        b>>=1;
    }
    return ret;
}

LL pow_mod(LL x,LL n,LL mod){    //返回  $x^n \bmod c$  , 非递归版
    if (n==1) return x%mod;
    int bit[90],k=0;
    while (n){
        bit[k++]=n&1;
        n>>=1;
    }
    LL ret=1;
    for (k=k-1;k>=0;k--){
        ret=muti_mod(ret,ret,mod);
        if (bit[k]==1) ret=muti_mod(ret,x,mod);
    }
    return ret;
}

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
    LL ret=pow_mod(a,x,n),last=ret;
    for (int i=1;i<=t;i++){
        ret=muti_mod(ret,ret,n);
```

```

        if (ret==1 && last!=1 && last!=n-1) return 1;
        last=ret;
    }
    if (ret!=1) return 1;
    return 0;
}

bool Miller_Rabin(LL n){
    LL x=n-1,t=0;
    while ((x&1)==0) x>>=1,t++;
    bool flag=1;
    if (t>=1 && (x&1)==1){
        for (int k=0;k<S;k++){
            LL a=rand()%(n-1)+1;
            if (check(a,n,x,t)) {flag=1;break;}
            flag=0;
        }
    }
    if (!flag || n==2) return 0;
    return 1;
}

LL gcd(LL a,LL b){
    if (a==0) return 1;
    if (a<0) return gcd(-a,b);
    while (b){
        LL t=a%b; a=b; b=t;
    }
    return a;
}

LL Pollard_rho(LL x,LL c){
    LL i=1,x0=rand()%x,y=x0,k=2;
    while (1){
        i++;
        x0=(muti_mod(x0,x0,x)+c)%x;
        LL d=gcd(y-x0,x);
        if (d!=1 && d!=x){
            return d;
        }
        if (y==x0) return x;
        if (i==k){
            y=x0;
            k+=k;
        }
    }
}

void findfac(LL n){
    //递归进行质因数分解 N
    if (!Miller_Rabin(n)){
        factor[tot++] = n;
        return;
    }
    LL p=n;
    while (p>=n) p=Pollard_rho(p,rand() % (n-1) +1);
    findfac(p);
}

```

```

        findfac(n/p);
    }

int main()
{
    // srand(time(NULL)); //POJ 上 G++ 要去掉这句话
    int T;
    scanf("%d",&T);
    long long n;
    while(T--)
    {
        scanf("%I64d",&n);
        if (!Miller_Rabin(n)) {printf("Prime\n"); continue; }
        tot = 0;
        findfac(n);
        long long ans=factor[0];
        for(int i=1;i<tot;i++)
            if(factor[i]<ans)ans=factor[i];
        printf("%I64d\n",ans);
    }
    return 0;
}

```

5.10.10 分段求和.cpp

```

int main(void)
{
    std::ios::sync_with_stdio(false);
    int T;
    cin>>T;
    int Kase = 0;
    while(T--)
    {
        LL n;
        cin>>n;
        int m = (int)sqrt(n);
        LL ans = 0;
        for(LL i = 1;i < m; ++i)
        {
            ans += n/i;
            ans += (LL)i*(n/i - n/(i+1));
        }
        ans += n/m;
        ans += m*(n/m-m);
        printf("Case %d: %lld\n",++Kase,ans);
    }
}

```

5.10.11 大数.cpp

```

#include<iostream>
#include<string>
#include<iomanip>
#include<algorithm>
using namespace std;

#define MAXN 9999

```

```

#define MAXSIZE 10
#define DLEN 4

class BigNum
{
private:
    int a[500];    //可以控制大数的位数
    int len;       //大数长度
public:
    BigNum(){ len = 1;memset(a,0,sizeof(a)); }    //构造函数
    BigNum(const int);    //将一个 int 类型的变量转化为大数
    BigNum(const char*);    //将一个字符串类型的变量转化为大数
    BigNum(const BigNum &);    //拷贝构造函数
    BigNum &operator=(const BigNum &);    //重载赋值运算符，大数之间进行赋值运算

    friend istream& operator>>(istream&, BigNum&);    //重载输入运算符
    friend ostream& operator<<(ostream&, BigNum&);    //重载输出运算符

    BigNum operator+(const BigNum &) const;    //重载加法运算符，两个大数之间的相加运
    ↪ 算
    BigNum operator-(const BigNum &) const;    //重载减法运算符，两个大数之间的相减运
    ↪ 算
    BigNum operator*(const BigNum &) const;    //重载乘法运算符，两个大数之间的相乘运
    ↪ 算
    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

    BigNum operator^(const int &) const;    //大数的 n 次方运算
    int operator%(const int &) const;    //大数对一个 int 类型的变量进行取模运
    ↪ 算
    bool operator>(const BigNum & T) const;    //大数和另一个大数的大小比较
    bool operator>(const int & t) const;    //大数和一个 int 类型的变量的大小比
    ↪ 较

    void print();    //输出大数
};

BigNum::BigNum(const int b)    //将一个 int 类型的变量转化为大数
{
    int c,d = b;
    len = 0;
    memset(a,0,sizeof(a));
    while(d > MAXN)
    {
        c = d - (d / (MAXN + 1)) * (MAXN + 1);
        d = d / (MAXN + 1);
        a[len++] = c;
    }
    a[len++] = d;
}

BigNum::BigNum(const char*s)    //将一个字符串类型的变量转化为大数
{
    int t,k,index,l,i;
    memset(a,0,sizeof(a));
    l=strlen(s);
    len=l/DLEN;
    if(l%DLEN)

```

```

        len++;
    index=0;
    for(i=l-1;i>=0;i-=DLEN)
    {
        t=0;
        k=i-DLEN+1;
        if(k<0)
            k=0;
        for(int j=k;j<=i;j++)
            t=t*10+s[j]-'0';
        a[index++]=t;
    }
}

BigNum::BigNum(const BigNum & T) : len(T.len)    //拷贝构造函数
{
    int i;
    memset(a,0,sizeof(a));
    for(i = 0 ; i < len ; i++)
        a[i] = T.a[i];
}

BigNum & BigNum::operator=(const BigNum & n)    //重载赋值运算符，大数之间进行赋值运算
{
    int i;
    len = n.len;
    memset(a,0,sizeof(a));
    for(i = 0 ; i < len ; i++)
        a[i] = n.a[i];
    return *this;
}

istream& operator>>(istream & in, BigNum & b)    //重载输入运算符
{
    char ch[MAXSIZE*4];
    int i = -1;
    in>>ch;
    int l=strlen(ch);
    int count=0,sum=0;
    for(i=l-1;i>=0;)
    {
        sum = 0;
        int t=1;
        for(int j=0;j<4&& i>=0;j++,i--,t*=10)
        {
            sum+=(ch[i]-'0')*t;
        }
        b.a[count]=sum;
        count++;
    }
    b.len =count++;
    return in;
}

ostream& operator<<(ostream& out, BigNum& b)    //重载输出运算符
{
    int i;
    cout << b.a[b.len - 1];
    for(i = b.len - 2 ; i >= 0 ; i--)

```



```

        {
            cout.width(DLEN);
            cout.fill('0');
            cout << b.a[i];
        }
        return out;
    }
}

BigNum BigNum::operator+(const BigNum & T) const    //两个大数之间的相加运算
{
    BigNum t(*this);
    int i, big;    //位数
    big = T.len > len ? T.len : len;
    for(i = 0 ; i < big ; i++)
    {
        t.a[i] += T.a[i];
        if(t.a[i] > MAXN)
        {
            t.a[i + 1]++;
            t.a[i] -= MAXN + 1;
        }
    }
    if(t.a[big] != 0)
        t.len = big + 1;
    else
        t.len = big;
    return t;
}

BigNum BigNum::operator-(const BigNum & T) const    //两个大数之间的相减运算
{
    int i, j, big;
    bool flag;
    BigNum t1, t2;
    if(*this > T)
    {
        t1 = *this;
        t2 = T;
        flag = 0;
    }
    else
    {
        t1 = T;
        t2 = *this;
        flag = 1;
    }
    big = t1.len;
    for(i = 0 ; i < big ; i++)
    {
        if(t1.a[i] < t2.a[i])
        {
            j = i + 1;
            while(t1.a[j] == 0)
                j++;
            t1.a[j--]--;
            while(j > i)
                t1.a[j--] += MAXN;
        }
    }
    if(flag)
        t1 = -t1;
    return t1 + t2;
}

```

```

        t1.a[i] += MAXN + 1 - t2.a[i];
    }
    else
        t1.a[i] -= t2.a[i];
}
t1.len = big;
while(t1.a[t1.len - 1] == 0 && t1.len > 1)
{
    t1.len--;
    big--;
}
if(flag)
    t1.a[big-1]=0-t1.a[big-1];
return t1;
}

```

```

BigNum BigNum::operator*(const BigNum & T) const    //两个大数之间的相乘运算
{
    BigNum ret;
    int i,j,up;
    int temp,temp1;
    for(i = 0 ; i < len ; i++)
    {
        up = 0;
        for(j = 0 ; j < T.len ; j++)
        {
            temp = a[i] * T.a[j] + ret.a[i + j] + up;
            if(temp > MAXN)
            {
                temp1 = temp - temp / (MAXN + 1) * (MAXN + 1);
                up = temp / (MAXN + 1);
                ret.a[i + j] = temp1;
            }
            else
            {
                up = 0;
                ret.a[i + j] = temp;
            }
        }
        if(up != 0)
            ret.a[i + j] = up;
    }
    ret.len = i + j;
    while(ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}

BigNum BigNum::operator/(const int & b) const    //大数对一个整数进行相除运算
{

```

```

    BigNum ret;
    int i,down = 0;
    for(i = len - 1 ; i >= 0 ; i--)
    {
        ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
        down = a[i] + down * (MAXN + 1) - ret.a[i] * b;
    }
}

```

```

        ret.len = len;
        while(ret.a[ret.len - 1] == 0 && ret.len > 1)
            ret.len--;
        return ret;
    }
    int BigNum::operator %(const int & b) const    //大数对一个 int 类型的变量进行取模运算
    {
        int i,d=0;
        for (i = len-1; i>=0; i--)
        {
            d = ((d * (MAXN+1))% b + a[i])% b;
        }
        return d;
    }
    BigNum BigNum::operator^(const int & n) const    //大数的 n 次方运算
    {
        BigNum t,ret(1);
        int i;
        if(n<0)
            exit(-1);
        if(n==0)
            return 1;
        if(n==1)
            return *this;
        int m=n;
        while(m>1)
        {
            t=*this;
            for( i=1;i<=1<=m;i<=1)
            {
                t=t*t;
            }
            m-=i;
            ret=ret*t;
            if(m==1)
                ret=ret*(t);
        }
        return ret;
    }
    bool BigNum::operator>(const BigNum & T) const    //大数和另一个大数的大小比较
    {
        int ln;
        if(len > T.len)
            return true;
        else if(len == T.len)
        {
            ln = len - 1;
            while(a[ln] == T.a[ln] && ln >= 0)
                ln--;
            if(ln >= 0 && a[ln] > T.a[ln])
                return true;
            else
                return false;
        }
        else

```

```

        return false;
    }
    bool BigNum::operator >(const int & t) const    //大数和一个 int 类型的变量的大小比较
    {
        BigNum b(t);
        return *this>b;
    }

    void BigNum::print()    //输出大数
    {
        int i;
        cout << a[len - 1];
        for(i = len - 2 ; i >= 0 ; i--)
        {
            cout.width(DLEN);
            cout.fill('0');
            cout << a[i];
        }
        cout << endl;
    }
    int main(void)
    {
        int i,n;
        BigNum x[101];    //定义大数的对象数组
        x[0]=1;
        for(i=1;i<101;i++)
            x[i]=x[i-1]*(4*i-2)/(i+1);
        while(scanf("%d",&n)==1 && n!=-1)
        {
            x[n].print();
        }
    }
}

```

5.10.12 快速数论变换.cpp

```

const int mod = 998244353;
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;}
const int g = 3;    //原根
LL quick_mod(LL a,LL b)
{
    LL ans=1;
    for(;b;b/=2)
    {
        if(b&1)
            ans=ans*a%mod;
        a=a*a%mod;
    }
    return ans;
}
int rev(int x,int r)    //蝴蝶操作
{
    int ans=0;
    for(int i=0; i<r; i++)
    {
        if(x&(1<<i))
        {

```

```

        ans+=1<<(r-i-1);
    }
}
return ans;
}
void NTT(int n, LL A[],int on) // 长度为 N (2 的次数)
{
    int r=0;
    for(;; r++)
    {
        if((1<<r)==n)
            break;
    }
    for(int i=0; i<n; i++)
    {
        int tmp=rev(i,r);
        if(i<tmp)
            swap(A[i],A[tmp]);
    }
    for(int s=1; s<=r; s++)
    {
        int m=1<<s;
        LL wn=quick_mod(g,(mod-1)/m);
        for(int k=0; k<n; k+=m)
        {
            LL w=1;
            for(int j=0; j<m/2; j++)
            {
                LL t,u;
                t=w*(A[k+j+m/2]%mod)%mod;
                u=A[k+j]%mod;
                A[k+j]=(u+t)%mod;
                A[k+j+m/2]=(u-t)%mod+mod)%mod;
                w=w*wn%mod;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        LL inv=quick_mod(n,mod-2);
        for(int i=0;i<n;i++)
            A[i]=A[i]%mod*inv%mod;
    }
}
}

```

5.10.13 欧拉函数打表.cpp

求任意一个数的欧拉函数值

```

```cpp
long long Euler(long long num)
{
 long long temp=num;

```

```

for(long long i=2;i*i<=num;i++)
if(num%i==0)
{
 while(num%i==0)
 num=num/i;
 temp=temp/i*(i-1);
}
if(num!=1)
temp=temp/num*(num-1);
return temp;
}

```

```

```

```

欧拉函数打表

$O(n\log(n))$

```

```cpp

```

```

const int maxn = 1e6+100;
int phi[maxn],Prime[maxn];

void init2(int n){
 for(int i = 1;i <= n; ++i) phi[i] = i;
 for(int i = 2;i <= n; ++i){
 if(i == phi[i]){
 for(int j = i; j <= n; j += i) phi[j] = phi[j]/i*(i-1);
 }
 }
}

```

```

```

```

线性筛 $O(n)$

```

```cpp

```

```

const int maxn = 1e6+100;
bool check[maxn];
int phi[maxn],Prime[maxn];
void init(int MAXN){
 int N = maxn-1;
 memset(check,false,sizeof(check));
 phi[1] = 1;
 int tot = 0;
 for(int i = 2;i <= N; ++i){
 if(!check[i]){
 Prime[tot++] = i;
 phi[i] = i-1;
 }
 for(int j = 0;j < tot; ++j){
 if(i*Prime[j] > N) break;
 check[i*Prime[j]] = true;
 if(i%Prime[j] == 0){
 phi[i*Prime[j]] = phi[i]*Prime[j];
 break;
 }
 else{
 phi[i*Prime[j]] = phi[i]*(Prime[j]-1);
 }
 }
 }
}

```

```

 }
}

}

```

#### 5.10.14 欧拉筛和埃氏筛.cpp

```

void Era_s(void){
 check[1] = 1;
 tot = 1;
 for(int i = 2; i < maxn; ++i){
 if(!check[i]){
 Prime[tot++] = i;
 for(int j = i+i; j < maxn; ++j) check[j] = 1;
 }
 }
}

void Euler_s(void){
 check[1] = 1;
 tot = 1;
 int n = 1e6;
 for(int i = 2; i <= n; ++i){
 if(!check[i]) Prime[tot++] = i;
 for(int j = 1; j < tot; ++j){
 if(i*Prime[j] > n) break;
 check[i*Prime[j]] = 1;
 if(i % Prime[j] == 0) break;
 }
 }
}

```

#### 5.10.15 素性检测.cpp

```

#include<bits/stdc++.h>

using namespace std;
//typedef long long LL;
const int LEN = 1e6+1;
bool vis[LEN];
//int prime[LEN];
int Prime[LEN];
int cnt = 1;
typedef unsigned long long LL;

LL modular_multi(LL x, LL y, LL mo) {
 LL t;
 x%=mo;
 for(t=0; y; x=(x<<1)%mo, y>>=1)
 if (y&1)
 t=(t+x)%mo;
 return t;
}

LL modular_exp(LL num, LL t, LL mo) {
 LL ret=1, temp=num%mo;

```

```

 for(;t;t>=1,temp=modular_multi(temp,temp,mo))
 if (t&1)
 ret=modular_multi(ret,temp,mo);
 return ret;
}

bool miller_rabin(LL n) {
 if (n==2||n==7||n==61)
 return true;
 if (n==1||(n&1)==0)
 return false;
 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
 LL a,x,y,u=n-1;
 while((u&1)==0)
 t++,u>=1;
 for(int i=0;i<3;i++) {
 a=num[i];
 x=modular_exp(a,u,n);
 for(int j=0;j<t;j++) {
 y=modular_multi(x,x,n);
 if (y==1&&x!=1&&x!=n-1)
 return false;
 //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。
 //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
 x=y;
 }
 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
 return false;
 }
 return true;
}

void init(void)
{
 int n = LEN - 1;
 for(int i = 2; i <= n; ++i)
 {
 if(!vis[i])
 {
 Prime[cnt++] = i;
 for(LL j = (LL)i * i; j <= n; j += i)
 vis[j] = 1;
 }
 }
}

bool isPrime(LL n)
{
 if(n < 1e6)
 {
 for(LL i = 1; i < cnt&&Prime[i] < n; ++i)
 {
 if(n % Prime[i] == 0)

```



```

 return false;
 }
 return true;
}
else
 return miller_rabin(n);
}

int main(void)
{
 init();

 int T;
 cin>>T;
 while(T-->0)
 {
 LL n;
 cin>>n;
 if(isPrime(n))
 cout<<"Yes"<<endl;
 else
 cout<<"No"<<endl;
 }

 return 0;
}

```

#### 5.10.16 素数筛.cpp

```

Eratosthenes 筛法 (埃拉托斯特尼筛法)
const int maxn = 1e6+10;
bool check[maxn];
int Prime[maxn];
int tot = 1;
void Eratosthenes(void){
 const int n = maxn - 1;
 memset(check,0,sizeof(check));
 for(int i = 2;i < n; ++i){
 if(!check[i]){
 Prime[tot++] = i;
 for(int j = i+i;j < n;j += i) check[j] = 1;
 }
 }
}

欧拉筛

const int maxn = 1e6+10;
bool check[maxn];
int Prime[maxn];
int tot = 1;
void Euler_shai(void){
 int n = maxn-1;
 memset(check,0,sizeof(check));
 for(int i = 2;i <= n; ++i){

```

```

 if(!check[i]){
 Prime[tot++] = i;
 }
 for(int j = 1;j < tot; ++j){
 if(i*Prime[j] > n) break;
 check[i*Prime[j]] = 1 ;
 if(i % Prime[j]==0) break;
 }
 }
}

```

...

### 5.10.17 逆元打表.cpp

```

int inv[10000];
int p;
cin>>p;
inv[1] = 1;
for(int i = 2;i < p; ++i)
{
 inv[i] = (p - p/i*inv[p%i]%p)%p;
}
for(int i = 1;i < p; ++i)
 cout<<inv[i]<<" ";
cout<<endl;
for(int i = 1;i < p; ++i)
 cout<<i * inv[i] % p<<" ";

```

### 5.11 矩阵快速幂.cpp

```

// 注意修改 maxn 的值，要不然容易 T
// 注意 maxn 值过大，栈可能会不够
const int maxn = 100;
int n;
struct Matrix{
 int n,m;
 Matrix(int nn = 1,int mm = 1):n(nn),m(mm){ memset(a,0,sizeof(a));};
 long long a[maxn][maxn];
};
// void print(const Matrix &a)
// {
// for(int i = 1;i <= a.n; ++i,cout<<endl)
// for(int j= 1;j <= a.m; ++j)
// cout<<a.a[i][j]<<" ";
// }
Matrix operator*(Matrix a,Matrix b)
{
 Matrix c(a.n,b.m);
 for(int i = 1;i <= a.n; ++i)
 {
 for(int j = 1;j <= b.m; ++j)
 {
 for(int k = 1;k <= a.m; ++k)
 {
 c.a[i][j] += a.a[i][k] * b.a[k][j];
 }
 }
 }
}

```

```

 c.a[i][j] %= mod;
 }
}

// print(c);
return c;
}

```

## 5.12 自适应辛普森积分.cpp

```

double F(double x)
{
 //Simpson 公式用到的函数
}
double simpson(double a, double b)//三点 Simpson 法, 这里要求 F 是一个全局函数
{
 double c = a + (b - a) / 2;
 return (F(a) + 4 * F(c) + F(b)) * (b - a) / 6;
}
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
{
 double c = a + (b - a) / 2;
 double L = simpson(a, c), R = simpson(c, b);
 if (fabs(L + R - A) <= 15 * eps) return L + R + (L + R - A) / 15.0;
 return asr(a, c, eps / 2, L) + asr(c, b, eps / 2, R);
}
double asr(double a, double b, double eps)//自适应 Simpson 公式 (主过程)
{
 return asr(a, b, eps, simpson(a, b));
}

```

## 6 数据结构

### 6.1 CDQ 分治

#### 6.1.1 CDQ 分治.cpp

```

// CDQ 解决 单点修改, 区间查询
/*

*/
const int maxn = 5e6+100;

struct node{
 int type,id;
 LL val;
 bool operator <(const node &a) const
 {
 if(a.id != id) return id < a.id;
 return type < a.type;
 }
};

node A[maxn],B[maxn];
LL ans[maxn];

```

```

void CDQ(int L,int R){
 // cout<<L<<" "<<R<<endl;
 if(L == R) return ;
 int M = (L+R)>>1;
 CDQ(L,M),CDQ(M+1,R);
 int t1 = L,t2 = M+1;
 LL sum = 0;
 for(int i = L;i <= R; ++i){
 if((t1 <= M && A[t1] < A[t2])||t2 > R){
 if(A[t1].type == 1) sum += A[t1].val;
 B[i] = A[t1++];
 }
 else{
 if(A[t2].type == 2) ans[A[t2].val] -= sum;
 else if(A[t2].type == 3) ans[A[t2].val] += sum;
 B[i] = A[t2++];
 }
 }

 for(int i = L;i <= R; ++i) A[i] = B[i];
}

int main(void)
{
 int n,q;
 cin>>n>>q;
 int tot = 0;
 for(int i = 1;i <= n; ++i){
 scanf("%lld",&A[i].val);
 A[i].type = 1;
 A[i].id = i;
 }
 tot = n;
 int sz = 0;
 rep(i,0,q){
 int type;
 scanf("%d",&type);
 if(type ==1){
 A[++tot].type = 1;
 scanf("%d%lld",&A[tot].id,&A[tot].val);
 }
 else{
 int l,r;
 scanf("%d%d",&l,&r);
 A[++tot].type = 2,A[tot].id = l-1,A[tot].val = ++sz;
 A[++tot].type = 3,A[tot].id = r, A[tot].val = sz;
 }
 }
 CDQ(1,tot);
 rep(i,1,sz+1){
 printf("%lld\n",ans[i]);
 }
}

```

```

 return 0;
}

```

### 6.1.2 CDQ 求动态逆序数.cpp

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define IOS ios::sync_with_stdio(false)
#define DEBUG cout<<endl<<"DEBUG"<<endl;
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF = 0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;

const int maxn = 2e5+100;
int n,m;
int a[maxn];
int del[maxn];
int id[maxn];
bool del2[maxn];
struct nd{
 int id,val;
};
bool operator <(const nd &a,const nd &b){
 return a.id < b.id;
}
bool operator >(const nd &a,const nd &b){
 return !(a < b);
}
nd A[maxn],B[maxn];
LL ans[maxn];
LL tree[maxn];
void Add(int x,int y){
 while(x <= n)
 {
 tree[x] += y;
 x += lowbit(x);
 }
}

```

```

LL Sum(int x){
 LL sum = 0;
 while(x > 0){
 sum += tree[x];
 x -= lowbit(x);
 }
 return sum;
}

void CDQ(int L,int R){
 // DEBUG;
 if(L == R) return ;
 int M = (L+R)>>1;
 CDQ(L,M),CDQ(M+1,R);
 int t1 = L,t2 = M+1;
 for(int i = L;i <= R; ++i){
 if((t1 <= M&&A[t1] < A[t2])||t2 > R){
 Add(A[t1].val,1);
 B[i] = A[t1++];
 }
 else{
 ans[id[A[t2].val]] += Sum(n)-Sum(A[t2].val);
 B[i] = A[t2++];
 }
 }

 for(int i = L;i <= M; ++i)
 Add(A[i].val,-1);
 t1 = M,t2 = R;
 for(int i = R;i >= L; --i){
 if((t1 >= L&&A[t1] > A[t2])||t2 <= M){
 Add(A[t1].val,1);
 t1--;
 // B[i] = A[t2++];?
 }
 else{
 ans[id[A[t2].val]] += Sum(A[t2].val);
 t2--;
 }
 }

 for(int i = L;i <= M; ++i)
 Add(A[i].val,-1);
 for(int i = L;i <= R; ++i)
 A[i] = B[i];
}

LL ans2[maxn];
int sign[maxn];
int main(void)
{
 // cout<<maxn*maxn/2<<endl;
 // freopen("input.txt","r",stdin);
 // freopen("output.txt","w",stdout);
 scanf("%d%d",&n,&m);
 // int s;
 for(int i = 1;i <= n; ++i){
 scanf("%d",&a[i]);
 id[a[i]] = i;
 }
}

```

```

}
for(int i = 1; i <= m; ++i){
 scanf("%d",&del[i]);
 del2[id[del[i]]] = 1;
}
// DEBUG;
int cnt = 0;
for(int i = 1; i <= n; ++i){
 if(!del2[i])
 {
 A[++cnt].id = i, A[cnt].val = a[i];
 sign[cnt] = a[i];
 }
}
for(int i = m; i >= 1; --i){
 // A[++cnt].op = 1, A[cnt].id = id[del[i]], A[cnt].val = a[i];
 A[++cnt].id = id[del[i]], A[cnt].val = del[i];
 sign[cnt] = del[i];
}
CDQ(1,n);
LL sum = 0;
for(int i = 1; i <= n; ++i){
 sum += ans[id[sign[i]]];
 ans2[i] = sum;
}
for(int i = n; i >= n-m+1; --i){
 printf("%lld\n", ans2[i]);
}

return 0;
}

```

### 6.1.3 陌上花开 CDQ 三位偏序.cpp

```

#include <cstdio>
#include <algorithm>
#include <iostream>
using namespace std;

const int N = 200005;
int w,q,c[500005];
struct nd {int op,x1,y1,x2,y2,z,id,ans;}a[N],b[N];
bool cmp(const nd &a, const nd &b) {return a.x1<b.x1 || (a.x1==b.x1&&a.op<b.op);}

int lowbit(int x) {return x & -x;}
void Add(int x, int y) {while(x <= w) c[x] += y, x += lowbit(x);}
int Sum(int x) {
 int r = 0;
 while(x) r += c[x], x -= lowbit(x);
 return r;
}
struct node{
 int x,y,z,id,num;
}Node[N],Node2[N];
bool operator<(const node &a,const node &b){
 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);
}

```

```

}
bool operator ==(const node &a,const node&b){
 return a.x == b.x && a.y == b.y&&a.z == b.z;
}
void CDQ(int l, int r) {
 if(l == r) return;

 // printf("%d %d\n",l,r);
 int m = (l+r) >> 1, cnt = 0;
 CDQ(l,m),CDQ(m+1,r);
 for(int i = l; i <= m; i++) if(a[i].op == 1) b[cnt++] = a[i];
 for(int i = m+1; i <= r; i++) if(a[i].op == 2) {
 b[cnt++] = a[i];
 b[cnt++] = a[i];
 b[cnt-2].x1--, b[cnt-1].x1=a[i].x2,
 b[cnt-1].op = 3;
 }
 sort(b, b+cnt, cmp);
 for(int i = 0; i < cnt; i++)
 if(b[i].op == 1) Add(b[i].y1, b[i].z);
 else if(b[i].op == 2) a[b[i].id].ans -= Sum(b[i].y2)-Sum(b[i].y1-1);
 else a[b[i].id].ans += Sum(b[i].y2)-Sum(b[i].y1-1);
 for(int i = 0; i < cnt; i++)
 if(b[i].op == 1) Add(b[i].y1, -b[i].z);
}
int ans[N];
int main() {
 // freopen("locust.in","r",stdin);
 // freopen("locust.out","w",stdout);
 scanf("%d%d",&q,&w);
 for(int i = 1;i <= q; ++i)
 scanf("%d%d%d",&Node2[i].x,&Node2[i].y,&Node2[i].z),Node2[i].id = i;
 // DEBUG;
 // cout<<"1"<<endl;
 int qq = q;
 sort(Node2+1,Node2+q+1);
 int cnt = 1;
 Node[cnt] = Node2[1];
 Node[cnt].num = 1;
 for(int i = 2;i <= q; ++i){
 if(Node2[i] == Node2[i-1])
 Node[cnt].num++;
 else
 Node[++cnt] = Node2[i],Node[cnt].num = 1;
 }
 q = cnt;

 for(int i = 1; i <= q; i++) {
 Node[i].id = i;
 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;
 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;

 a[2*i-1].id = a[2*i].id =Node[i].id;
 }
}

```



```

}
// puts("DEBUG");
CDQ(1, 2*q);

for(int i = 1; i <= q; i++) ans[a[i].ans+Node[i].num-1] += Node[i].num;
// cout<<endl;
// for(int i = 1;i <= q; ++i) cout<<a[i].ans<<endl;
// cout<<endl;
for(int i = 0; i < qq; ++i) printf("%d\n",ans[i]);
return 0;
}

```

## 6.2 fenkuai

### 6.2.1 区间修改区间查询.cpp

```

const int maxn = 100010;
LL a[maxn],add[maxn],sum[maxn];
int pos[maxn],R[maxn],L[maxn];
int n,m,t;
void change(int l,int r,LL d){
 int p = pos[l],q = pos[r];
 if(p == q){
 for(int i = l;i <= r; ++i) a[i] += d;
 sum[p] += (r-l+1)*d;
 }
 else{
 for(int i = p+1;i <= q-1; ++i) add[i] += d;
 for(int i = l;i <= R[p];++i)
 a[i] += d;
 sum[p] += (R[p]-l+1)*d;
 for(int i = L[q];i <= r; ++i)
 a[i] += d;
 sum[q] += (r-L[q]+1)*d;
 }
}
LL ask(int l,int r){
 LL ans = 0;
 int p = pos[l],q = pos[r];
 if(p == q){
 for(int i = l;i <= r; ++i)
 ans += a[i];
 ans += (r-l+1)*add[p];
 }
 else{
 for(int i = p+1;i <= q-1; ++i)
 ans += sum[i]+add[i]*(R[i]-L[i]+1);
 for(int i = l;i <= R[p]; ++i)
 ans += a[i];
 ans += add[p]*(R[p]-l+1);
 for(int i = L[q];i <= r; ++i)
 ans += a[i];
 ans += add[q]*(r-L[q]+1);
 }
 return ans;
}

```

```

int main(void){

 cin>>n>>m;
 for(int i = 1;i <= n; ++i) scanf("%lld",&a[i]);
 LL t = sqrt(n);
 for(int i = 1;i <= t; ++i){
 L[i] = (i-1)*sqrt(n)+1;
 R[i] = i*sqrt(n);
 }
 if(R[t] < n) t++,L[t] = R[t-1]+1,R[t] = n;
 // cout<<t<<endl;
 for(int i = 1;i <= t; ++i){
 for(int j = L[i];j <= R[i]; ++j){
 pos[j] = i;
 sum[i] += a[j];
 }
 }
 while(m--){
 char op[3];
 int l,r,x;
 scanf("%s%d%d",op,&l,&r);
 if(op[0] == 'C'){
 scanf("%d",&x);
 change(l,r,x);
 }
 else
 printf("%lld\n",ask(l,r));
 }
 return 0;
}

```

### 6.2.2 区间数的平方.cpp

```

const int maxn = 50000+10;
int n,m,k;
int pos[maxn];
int a[maxn];
int num[maxn];
LL Ans[maxn];
int L[maxn],R[maxn];
struct Query{
 int l,r,id;
};
Query q[maxn];
bool cmp1 (const Query &a,const Query &b){
 return a.l < b.l ||(a.l == b.l && a.r < b.r);
}
bool cmp2(const Query &a,const Query &b){
 return a.r < b.r;
}

void work(int x,LL &ans,int d){
 ans -= 1ll*num[x]*num[x];
 num[x] += d;
 ans += 1ll*num[x]*num[x];
}

```

```

int main(){
 cin>>n>>m>>k;
 rep(i,1,n+1) scanf("%d",&a[i]);
 rep(i,1,m+1){
 scanf("%d%d",&q[i].l,&q[i].r);
 q[i].id = i;
 }
 int t = sqrt(m);
 for(int i = 1;i <= t; ++i){
 L[i] = (i-1)*t;
 R[i] = i*t;
 }
 if(R[t] < m){
 L[t+1] = R[t]+1;
 R[t+1] = m;
 }
 sort(q+1,q+m+1,cmp1);
 for(int i = 1;i <= t; ++i){
 sort(q+L[i],q+R[i]+1,cmp2);
 LL ans = 0;
 me(num);
 int l = q[L[i]].l,r = q[L[i]].r;
 rep(i,l,r+1) work(a[i],ans,1);
 Ans[q[L[i]].id] = ans;
 for(int j = L[i]+1;j <= R[i]; ++j){
 // l = L[j].l,r = L[j].r;
 while(l < q[j].l) work(a[l++],ans,-1);
 while(l > q[j].l) work(a[--l],ans,1);
 while(r < q[j].r) work(a[++r],ans,1);
 while(r > q[j].r) work(a[r--],ans,-1);
 Ans[q[j].id] = ans;
 }
 }
 rep(i,1,m+1)
 printf("%lld\n",Ans[i]);
 return 0;
}

```

### 6.2.3 在线查询区间众数.cpp

```

const int N = 40006,T = 37;
int a[N],b[N],L[N],R[N],pos[N];
int c[T][T][N],f[T][T][2],now[2];
inline void work(int x,int y,int num){
 ++c[x][y][num];
 if(c[x][y][num] > now[0] || (c[x][y][num] == now[0] && num < now[1])){
 now[0] = c[x][y][num];
 now[1] = num;
 }
}
int ask(int l,int r){
 int p = pos[l],q = pos[r];
 int x = 0,y = 0;
 if(p+1 <= q-1){
 x = p+1;
 y = q-1;
 }
}

```

```

 }
 memcpy(now,f[x][y],sizeof(now));
 if(p == q){
 rep(i,l,r+1) work(x,y,a[i]);
 rep(i,l,r+1) --c[x][y][a[i]];
 }
 else{
 rep(i,l,R[p]+1) work(x,y,a[i]);
 rep(i,L[q],r+1) work(x,y,a[i]);
 rep(i,l,R[p]+1) --c[x][y][a[i]];
 rep(i,L[q],r+1) --c[x][y][a[i]];
 }
 return b[now[1]];
}

int main(void){
 // freopen("input.txt","r",stdin);

 // freopen("output1.txt","w+",stdout);
 int n,m;cin>>n>>m;
 rep(i,1,n+1) scanf("%d",&a[i]);
 memcpy(b,a,sizeof(a));
 sort(b+1,b+n+1);
 int tot = unique(b+1,b+n+1)-(b+1);
 rep(i,1,n+1) a[i] = lower_bound(b+1,b+tot+1,a[i])-b;
 int t = pow((double)n,(double)1/3);
 int len = t?n/t:n;
 for(int i = 1;i <= t; ++i){
 L[i] = (i-1)*len+1;
 R[i] = i*len;
 }
 if(R[t] < n){
 L[t+1] = R[t]+1;
 R[++t] = n;
 }
 rep(i,1,t+1)
 rep(j,L[i],R[i]+1)
 pos[j] = i;

 me(c),me(f);
 rep(i,1,t+1){
 rep(j,i,t+1){
 rep(k,L[i],R[j]+1)
 ++c[i][j][a[k]];
 rep(k,1,tot+1)
 if(c[i][j][k] > f[i][j][0]){
 f[i][j][0] = c[i][j][k];
 f[i][j][1] = k;
 }
 }
 }

 int x = 0;
 while(m--){
 int l,r;scanf("%d%d",&l,&r);
 l = (l+x-1)%n+1;
 r = (r+x-1)%n+1;
 if(l > r) swap(l,r);
 }
}

```

```

 printf("%d\n",x = ask(l,r));
 }

 return 0;
}

```

## 6.3 pbds

### 6.3.1 1 可合并优先队列.cpp

*// pbds zoj2334 合并 logn*

```

#include<bits/stdc++.h>
#include<ext/pb_ds/priority_queue.hpp>

using namespace std;
using namespace __gnu_pbds;
typedef pair<int,int> P;
typedef __gnu_pbds::priority_queue<int> Heap;

const int maxn = 1e5+10;
Heap heap[maxn];

int F[maxn];

int Find(int x){
 return x == F[x]?x:F[x] = Find(F[x]);
}

int main(void){
 int N,M;
 while(cin>>N){
 for(int i = 1;i <= N; ++i){
 int a;
 scanf("%d",&a);
 heap[i].clear();
 heap[i].push(a);
 F[i] = i;
 }
 cin>>M;
 int a,b;
 for(int i = 1;i <= M; ++i){
 scanf("%d%d",&a,&b);
 int fa = Find(a);
 int fb = Find(b);
 if(fa == fb){
 puts("-1");
 continue;
 }
 // cout<<fa<<" "<<fb<<endl;
 F[fb] = fa;
 int t;
 t = heap[fa].top(),heap[fa].pop(),t/=2,heap[fa].push(t);
 t = heap[fb].top(),heap[fb].pop(),t/=2,heap[fb].push(t);
 heap[fa].join(heap[fb]);
 }
 }
}

```

```

 printf("%d\n",heap[fa].top());
 }
}
return 0;
}

```

## 6.4 二叉搜索树

### 6.4.1 1 二叉树.cpp

// 通过中序遍历和后序遍历建立二叉树

//<https://vjudge.net/problem/UVA-548>

```

#include<bits/stdc++.h>

using namespace std;
const int maxn = 1e5+10;
const int INF = 1e8;
int in_order[maxn],post_order[maxn],l[maxn],r[maxn];
int n;
int read_order(int *a)
{
 string s;
 if(!getline(cin,s)) return false;
 stringstream ss(s);
 n = 0;
 int v;
 while(ss >> v)
 a[n++] = v;
 return n > 0;
}
int build_tree(int L1,int R1,int L2,int R2)
{
 if(L1 > R1)
 return 0;
 int root = post_order[R2];
 int p = L1;
 while(in_order[p] != root)
 p++;
 int cnt = p-L1;
 l[root] = build_tree(L1,p-1,L2,L2+cnt-1);
 r[root] = build_tree(p+1,R1,L2+cnt,R2-1);
 return root;
}
int best,bestsum;
void dfs(int a,int b)
{
 if(!l[a] && !r[a])
 {
 b += a;
 if(bestsum > b||(bestsum == b&&best > a))
 {
 best = a;
 bestsum = b;
 }
 }
}

```

```

 }
 if(l[a]) dfs(l[a],b+a);
 if(r[a]) dfs(r[a],b+a);
}

int main(void)
{
 while(read_order(in_order))
 {
 read_order(post_order);
 build_tree(0,n-1,0,n-1);
// cout<<0<<endl;
 bestsum = INF;
 dfs(post_order[n-1],0);
 cout<<best<<endl;
 }

 return 0;
}

```

#### 6.4.2 2 treap.cpp

// UVA LA 5031

/\*

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。

$D\ X$  删除  $ID$  为  $x$  的边

$Q\ X\ K$  计算与节点  $X$  连通的节点中权值第  $k$  大的数

$C\ X\ K$  把节点  $X$  的权值改为  $V$

\*/

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first
#define SE second
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define IOS ios::sync_with_stdio(false)
#define DEBUG cout<<endl<<"DEBUG"<<endl;
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF = 0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;}

```

```

LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Node{
 Node *ch[2]; // 左右子树
 int r; // 随机优先值
 int v; // 值
 int s; // 节点总数

 Node(int v):v(v){ch[0] = ch[1] = NULL; r = rand(); s = 1;}
 int cmp(int x) {
 if(x==v) return -1;
 return x < v?0:1;
 }

 void maintain(){
 s = 1;
 if(ch[0] != NULL) s += ch[0]->s;
 if(ch[1] != NULL) s += ch[1]->s;
 }
};

void rotate(Node * &o,int d){
 Node *k = o->ch[d^1]; o->ch[d^1] = k->ch[d]; k->ch[d] = o;
 o->maintain();k->maintain(); o = k;
}

void insert(Node * &o,int x){
 if(o==NULL) o = new Node(x);
 else{
 int d = (x < o->v?0:1);
 insert(o->ch[d],x);
 if(o->ch[d]->r > o->r) rotate(o,d^1);
 }
 o->maintain();
}

void remove(Node * &o,int x){
 int d = o->cmp(x);
 // int ret = 0;
 if(d == -1){
 Node *u = o;
 if(o->ch[0] != NULL && o->ch[1] != NULL){
 int d2 = (o->ch[0]->r > o->ch[1]->r?1:0);
 rotate(o,d2); remove(o->ch[d2],x);
 }
 else{
 if(o->ch[0] == NULL) o = o->ch[1];
 else o = o->ch[0];
 delete u;
 }
 } else
 remove(o->ch[d],x);
 if(o != NULL) o->maintain();
}

const int maxc = 5e5+10;

```



```

struct Command{
 char type;
 int x,p;

} commands[maxc];

const int maxn = 2e4+10;
const int maxm = 6e4+10;
int n,m,weight[maxn],from[maxm],to[maxm],removed[maxm];
// 并查集相关
int pa[maxn];
int findset(int x){ return pa[x] != x?pa[x] = findset(pa[x]) : x;}
// 名次数相关
Node *root[maxn]; // Treap;
int kth(Node *o,int k){
 if(o == NULL || k <= 0 || k > o->s) return 0;
 int s = (o->ch[1] == NULL?0:o->ch[1]->s);
 if(k == s+1) return o->v;
 else if(k <= s) return kth(o->ch[1],k);
 else return kth(o->ch[0],k-s-1);
}

void mergeto(Node* &src,Node * &dest){
 if(src->ch[0] != NULL) mergeto(src->ch[0],dest);
 if(src->ch[1] != NULL) mergeto(src->ch[1],dest);
 insert(dest,src->v);
 delete src;
 src = NULL;
}

void removetree(Node *&x){
 if(x->ch[0] != NULL) removetree(x->ch[0]);
 if(x->ch[1] != NULL) removetree(x->ch[1]);
 delete x;
 x = NULL;
}

void add_edge(int x){
 int u = findset(from[x]), v = findset(to[x]);
 if(u != v){
 if(root[u]->s < root[v]->s){ pa[u] = v; mergeto(root[u],root[v]);}
 else {pa[v] = u; mergeto(root[v],root[u]);}
 }
}

int query_cnt;
long long query_tot;
void query(int x,int k){
 query_cnt++;
 query_tot += kth(root[findset(x)],k);
}

void change_weight(int x,int v){
 int u = findset(x);
 remove(root[u],weight[x]);
 insert(root[u],v);
 weight[x] = v;
}

```

```

}

int main(void){
 int kase = 0;
 while(scanf("%d%d",&n,&m) == 2&& n){
 rep(i,1,n+1) scanf("%d",&weight[i]);
 rep(i,1,m+1) scanf("%d%d",&from[i],&to[i]);
 me(removed);
 int c = 0;
 for(;;){
 char type;
 int x,p = 0,v = 0;
 scanf(" %c",&type);
 if(type == 'E') break;
 scanf("%d",&x);
 if(type == 'D') removed[x] = 1;
 if(type == 'Q') scanf("%d",&p);
 if(type == 'C') {
 scanf("%d",&v);
 p = weight[x];
 weight[x] = v;
 }
 commands[c++] = (Command){type,x,p};
 }
 rep(i,1,n+1) {
 pa[i] = i; if(root[i] != NULL) removetree(root[i]);
 root[i] = new Node(weight[i]);
 }
 rep(i,1,m+1) if(!removed[i]) add_edge(i);
 // 反向操作
 query_tot = query_cnt = 0;
 per(i,0,c){
 if(commands[i].type == 'D') add_edge(commands[i].x);
 if(commands[i].type == 'Q')
 ⇨ query(commands[i].x,commands[i].p);
 if(commands[i].type == 'C')
 ⇨ change_weight(commands[i].x,commands[i].p);
 }
 printf("Case %d: %.6lf\n", ++kase, query_tot / (double)query_cnt);
 }
}

```

### 6.4.3 3 伸展树.cpp

```

/*
UVA 11922
序列反转 (a,b)

*/
#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
#define Pb push_back
#define FI first

```

```

#define SE second
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define IOS ios::sync_with_stdio(false)
#define DEBUG cout<<endl<<"DEBUG"<<endl;
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF = 0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 1e9 + 7;
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;}
LL gcd(LL a,LL b) {return b?gcd(b,a%b):a;}
int dr[2][4] = {1,-1,0,0,0,0,-1,1};
typedef pair<int,int> P;
struct Node{
 Node *ch[2];
 int s;
 int flip;
 int v;
 int cmp(int k) const {
 int d = k-ch[0]->s;
 if(d == 1) return -1;
 return d <= 0?0:1;
 }
 void maintain(){
 s = ch[0]->s+ch[1]->s+1;
 }
 void pushdown(){
 if(flip){
 flip = 0;
 swap(ch[0],ch[1]);
 ch[0]->flip = !ch[0]->flip;
 ch[1]->flip = !ch[1]->flip;
 }
 }
};
Node *null = new Node();
void rotate(Node *&o,int d){
 Node *k = o->ch[d^1];
 o->ch[d^1] = k->ch[d];
 k->ch[d] = o;
 o->maintain(); k->maintain(); o = k;
}

void splay(Node * &o,int k){
 // cout<<1<<endl;
 o->pushdown();
 int d = o->cmp(k);
 if(d == 1) k -= o->ch[0]->s + 1;
 // DEBUG;

```

```

 if(d != -1){
 Node *p = o->ch[d];
 p->pushdown();
 int d2 = p->cmp(k);
 int k2 = (d2==0?k:k-p->ch[0]->s-1);
 // cout<<k2<<endl;
 if(d2 != -1){
 splay(p->ch[d2],k2);
 if(d == d2) rotate(o,d^1);
 else rotate(o->ch[d],d);
 }
 rotate(o,d^1);
 }
}

Node * Merge(Node *left,Node*right){
 splay(left,left->s);
 left->ch[1] = right;
 left->maintain();
 return left;
}

void split(Node *o,int k,Node * &left,Node *&right){
 splay(o,k);
 left = o;
 right = o->ch[1];
 o->ch[1] = null;
 left->maintain();
}

const int maxn = 1e5+10;
struct SplaySequence{
 int n;
 Node seq[maxn];
 Node *root;
 Node *build(int sz){
 if(!sz) return null;
 Node *L = build(sz/2);
 Node *o = &seq[++n];
 o->v = n;
 o->ch[0] = L;
 o->ch[1] = build(sz-sz/2-1);
 o->flip = o->s = 0;
 o->maintain();
 return o;
 }
 void init(int sz){
 n = 0;
 null->s = 0;
 root = build(sz);
 }
};

vector<int> ans;
void print(Node *o){
 if(o!=null){
 o->pushdown();
 print(o->ch[0]);
 ans.push_back(o->v);
 }
}

```

```

 print(o->ch[1]);
 }
}
void debug(Node *o){
 if(o!=null){
 o->pushdown();
 debug(o->ch[0]);
 printf("%d ",o->v-1);
 debug(o->ch[1]);
 }
}
SplaySequence ss;
int main(void)
{
 int n,m;
 scanf("%d%d",&n,&m);
 // cout<<n<<" "<<m<<endl;
 ss.init(n+1);

 while(m--){
 int a,b;
 scanf("%d %d",&a,&b);
 // cout<<a<<" "<<b<<endl;
 Node *left,*mid,*right,*o;
 split(ss.root,a,left,o);
 // DEBUG;
 split(o,b-a+1,mid,right);
 mid->flip ^= 1;
 ss.root = Merge(Merge(left,right),mid);
 }
 print(ss.root);
 for(int i = 1; i <ans.size(); i++)
 printf("%d\n",ans[i]-1);
 return 0;
}

```

## 6.5 基础数据结构

### 6.5.1 堆.cpp

// 堆的插入和删除操作

```

void Insert(int vv)
{
 int t = sz++;
 h[t] = vv;
 while(t > 1)
 {
 if(h[t] < h[t/2])
 {
 swap(h[t],h[t/2]);
 t /= 2;
 }
 else break;
 }
}

```

```

}
int Down(int i)
{
 int t;
 while(i * 2 <= n)
 {
 if(h[i] > h[2*i])
 t = 2*i;
 else
 t = i;
 if(i*2+1 <= n&&h[i*2+1] < h[t])
 t = i*2+1;
 if(i == t)
 break;
 swap(h[t],h[i]);
 i = t;
 }
}

```

## 6.6 字符串

### 6.6.1 1 Trie(前缀树).cpp

```

const int maxnode = 4e5+100;
const int sigma_size = 26;
struct Trie
{
 int ch[maxnode][sigma_size];
 int val[maxnode];
 int sz;
 Trie()
 {
 sz = 1;
 memset(ch[0],0,sizeof(ch[0]));
 }
 int idx(char c)
 {
 return c-'a';
 }
 void init(void)
 {
 memset(ch,0,sizeof(ch));
 memset(val,0,sizeof(val));
 }
 void insert(char *s,int v)
 {
 int u = 0, n = strlen(s);
 for(int i = 0; i < n; ++i)
 {
 int c = idx(s[i]);
 if(!ch[u][c])
 {
 memset(ch[sz],0,sizeof(ch[sz]));
 val[sz] = 0;
 ch[u][c] = sz++;
 }
 }
 }
}

```

```

 u = ch[u][c];
 }
 val[u] = v;
}
int query(char *s,int t)
{
 int sum = 0;
 int u = 0,n = strlen(s);
 for(int i = 0; i < n; ++i)
 {
 int c = idx(s[i]);
 if(ch[u][c])
 {
 if(val[ch[u][c]])
 sum = (sum+ans[i+t+1]) % mod;
 }
 else
 return sum;
 u = ch[u][c];
 }
 return sum;
}
};

```

### 6.6.2 2 KMP.cpp

```

#include <bits/stdc++.h>
#define mem(ar,num) memset(ar,num,sizeof(ar))
#define me(ar) memset(ar,0,sizeof(ar))
#define lowbit(x) (x&(-x))
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int prime = 999983;
const int INF = 0x7FFFFFFF;
const LL INFF =0x7FFFFFFFFFFFFFFF;
const double pi = acos(-1.0);
const double inf = 1e18;
const double eps = 1e-6;
const LL mod = 20071027 ;
int f[1100];
char ch[100];
void getFail(char *P,int *f)
{
 int m = strlen(P);
 f[0] = 0,f[1] = 0;
 for(int i = 1;i < m; ++i)
 {
 int j = f[i];
 while(j && P[i] != P[j]) j = f[j];
 f[i+1] = P[i] == P[j] ? j + 1: 0;
 }
}
}

```

```

void find(char * T,char * P,int* f)
{
 int n = strlen(T),m = strlen(P);
 getFail(P,f);
 int j = 0;
 for(int i = 0;i < n; ++i)
 {
 while(j&&P[j] != T[i]) j = f[j];
 if(P[j] == T[i]) j++;
 if(j == m) printf("%d\n",i-m+1);
 }
}

int main(void)
{
 cin>>ch;
 getFail(ch,f);
 printf("%d",f[strlen(ch)-1]);

 return 0;
}

```

### 6.6.3 3 AC 自动机.cpp

```

const int SIGMA_SIZE = 26;
const int MAXNODE = 11000;
const int MAXS = 150 + 10;

struct AhoCorasickAutomata {
 int ch[MAXNODE][SIGMA_SIZE];
 int f[MAXNODE]; // fail 函数
 int val[MAXNODE]; // 每个字符串的结尾结点都有一个非 0 的 val
 int last[MAXNODE]; // 输出链表的下一个结点
 int sz;

 void init() {
 sz = 1;
 memset(ch[0], 0, sizeof(ch[0]));
 }

 // 字符 c 的编号
 int idx(char c) {
 return c-'a';
 }

 // 插入字符串。v 必须非 0
 void insert(char *s, int v) {
 int u = 0, n = strlen(s);
 for(int i = 0; i < n; i++) {
 int c = idx(s[i]);
 if(!ch[u][c]) {
 memset(ch[sz], 0, sizeof(ch[sz]));
 val[sz] = 0;
 ch[u][c] = sz++;
 }
 }
 }
}

```



```

 u = ch[u][c];
}
val[u] = v;
}

// 递归打印以结点 j 结尾的所有字符串
void print(int j) {
 if(j) {
 print(last[j]);
 }
}

// 在 T 中找模板
int find(char* T) {
 int n = strlen(T);
 int j = 0; // 当前结点编号, 初始为根结点
 for(int i = 0; i < n; i++) { // 文本串当前指针
 int c = idx(T[i]);
 while(j && !ch[j][c]) j = f[j]; // 顺着细边走, 直到可以匹配
 j = ch[j][c];
 if(val[j]) print(j);
 else if(last[j]) print(last[j]); // 找到了!
 }
}

// 计算 fail 函数
void getFail() {
 queue<int> q;
 f[0] = 0;
 // 初始化队列
 for(int c = 0; c < SIGMA_SIZE; c++) {
 int u = ch[0][c];
 if(u) { f[u] = 0; q.push(u); last[u] = 0; }
 }
 // 按 BFS 顺序计算 fail
 while(!q.empty()) {
 int r = q.front(); q.pop();
 for(int c = 0; c < SIGMA_SIZE; c++) {
 int u = ch[r][c];
 if(!u) continue;
 q.push(u);
 int v = f[r];
 while(v && !ch[v][c]) v = f[v];
 f[u] = ch[v][c];
 last[u] = val[f[u]] ? f[u] : last[f[u]];
 }
 }
}

};

```

#### 6.6.4 4 KMP-KMP 变形.cpp

<https://www.nowcoder.com/acm/contest/119/E>

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

```

using namespace std;

const int N=200010;
int a[N],b[N];
int x[N],y[N],nxt[N];

void kmp_pre(int x[],int m,int nxt[])
{
 int i,j;
 j=nxt[0]=-1;
 i=0;
 while(i<m) {
 while(-1!=j && (x[i]!=x[j]&&x[j]!=-1))j=nxt[j];
 nxt[++i]=++j;
 }
}

int KMP_Count(int x[],int m,int y[],int n)
{
 // for (int i=0;i<n;i++) {
 // printf("%d ",y[i]);
 // }
 // puts("");
 // for (int i=0;i<m;i++) {
 // printf("%d ",x[i]);
 // }
 // puts("");
 int i,j;
 int ans=0;
 kmp_pre(x,m,nxt);
 i=j=0;
 while(i<n) {
 while(-1!=j && !(y[i]==x[j]||(x[j]==-1&&(y[i]==-1||j-y[i]<0)))) j=nxt[j];
 i++;
 j++;
 if(j>=m) {
 ans++;
 j=nxt[j];
 }
 }
 return ans;
}

int main()
{
 int n,m,k;
 scanf("%d%d",&n,&k);
 memset(x,-1,sizeof(x));
 memset(y,-1,sizeof(y));
 map<int,int> pre;
 for (int i=0;i<n;i++) {
 scanf("%d",&a[i]);
 auto pos=pre.find(a[i]);
 if (pos!=pre.end()) {
 y[i]=i-pos->second;
 }
 }
}

```

```

 pre[a[i]]=i;
 }
 scanf("%d",&m);
 pre.clear();
 for (int i=0;i<m;i++) {
 scanf("%d",&b[i]);
 auto pos=pre.find(b[i]);
 if (pos!=pre.end()) {
 x[i]=i-pos->second;
 }
 pre[b[i]]=i;
 }
 printf("%d\n",KMP_Count(x,m,y,n));
 return 0;
}

```

### 6.6.5 5 字符串 hash.cpp

```

// 字符串 hash, 查找在字符串中至少出现 k 次的最长字符串
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;

const int maxn = 40000+10;
const int x = 123;
int n,m,pos;

unsigned long long H[maxn],xp[maxn];

unsigned long long Hash[maxn];
int Rank[maxn];

int cmp(const int &a,const int &b){
 return Hash[a] < Hash[b] || (Hash[a] == Hash[b] &&a < b);
}

int possible(int L){
 int c = 0;
 pos = -1;
 for(int i = 0;i < n-L+1; ++i){
 Rank[i] = i;
 Hash[i] = H[i]-H[i+L]*xp[L];
 }
 sort(Rank,Rank+n-L+1,cmp);
 for(int i = 0;i < n-L+1; ++i){
 if(i == 0||Hash[Rank[i]] != Hash[Rank[i-1]]) c = 0;
 if(++c >= m) pos = max(pos,Rank[i]);
 }
 return pos >= 0;
}

char s[maxn];
int main(void)
{

```

```

while((scanf("%d",&m)) == 1&&&m){
 scanf("%s",s);
 n = strlen(s);
 H[n] = 0;
 for(int i = n-1;i >= 0; i--) H[i] = H[i+1]*x+(s[i]-'a');
 xp[0] = 1;
 for(int i = 1;i <= n; ++i) xp[i] = xp[i-1]*x;
 if(!possible(1)) printf("none\n");
else{
 int L = 1,R = n;
 while(R >= L){
 int M = (R+L)/2;
 if(possible(M)) L = M+1;
 else R = M-1;
 }
 possible(R);
 printf("%d %d\n",R,pos);
}
}

return 0;
}

```

### 6.6.6 6 后缀数组.cpp

```

const int maxn = 1e6 + 10;

struct SuffixArray {
 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)
 int sa[maxn]; // 后缀数组
 int rank[maxn]; // 名次数组. rank[0] 一定是 n-1, 即最后一个字符
 int height[maxn]; // height 数组
 int t[maxn], t2[maxn], c[maxn]; // 辅助数组
 int n; // 字符个数

 void clear() { n = 0; memset(sa, 0, sizeof(sa)); }

 // m 为最大字符值加 1。调用之前需设置好 s 和 n
 void build_sa(int m) {
 int i, *x = t, *y = t2;
 for(i = 0; i < m; i++) c[i] = 0;
 for(i = 0; i < n; i++) c[x[i] = s[i]]++;
 for(i = 1; i < m; i++) c[i] += c[i-1];
 for(i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
 for(int k = 1; k <= n; k <= 1) {
 int p = 0;
 for(i = n-k; i < n; i++) y[p++] = i;
 for(i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i]-k;
 for(i = 0; i < m; i++) c[i] = 0;
 for(i = 0; i < n; i++) c[x[y[i]]]++;
 for(i = 1; i < m; i++) c[i] += c[i-1];
 for(i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
 swap(x, y);
 p = 1; x[sa[0]] = 0;
 for(i = 1; i < n; i++)
 x[sa[i]] = y[sa[i-1]]==y[sa[i]] && y[sa[i-1]+k]==y[sa[i]+k] ? p-1 : p++;
 }
 }
}

```

```

 if(p >= n) break;
 m = p;
 }
}

void build_height() {
 int i, j, k = 0;
 for(i = 0; i < n; i++) rank[sa[i]] = i;
 for(i = 0; i < n; i++) {
 if(k) k--;
 int j = sa[rank[i]-1];
 while(s[i+k] == s[j+k]) k++;
 height[rank[i]] = k;
 }
}
};

```

## 6.7 并查集

### 6.7.1 加权并查集 + 区间合并.cpp

```

const int LEN = 234567;
int F[LEN];
int val[LEN];
int Find(int x){
 int k = F[x];
 if(x!=k){
 F[x] = Find(k);
 val[x] += val[k];
 }
 return F[x];
}
int main(void)
{
 int N,M;
 while(cin>>N>>M) {
 for(int i = 0;i <= N; ++i){
 F[i] = i;
 val[i] = 0;
 }
 int a,b,c;
 int Count = 0;
 while(M--){
 scanf("%d %d %d",&a,&b,&c);
 a--;
 int x1 = Find(a);
 int y1 = Find(b);
 if(x1==y1&&c+val[a]!=val[b])
 ++Count;
 else if(x1<y1) {
 F[y1] = x1;
 val[y1] = c+val[a]-val[b];
 }
 else if(x1>y1){
 F[x1] = y1;
 val[x1] = val[b]-val[a]-c;
 }
 }
 }
}

```

```

 }
}
cout<<Count<<endl;
}
return 0;
}

```

## 6.7.2 并查集.cpp

[//http://acm.hdu.edu.cn/showproblem.php?pid=1232](http://acm.hdu.edu.cn/showproblem.php?pid=1232)

```

#include <iostream>
#include <cstdio>
#include <set>
#include <cstring>
using namespace std;
const int LEN = 1000+5;
int N,M;
int ar[LEN];
int Find(int x)//并查集之 find 函数
{
 return x==ar[x]?x:ar[x]=Find(ar[x]);
}
int main()
{
 while(cin>>N&&N)
 {
 cin>>M;
 for(int i = 1;i <= N; ++i)
 ar[i] = i;
 while(M--)
 {
 int a,b;
 scanf("%d %d",&a,&b);
 if(Find(a)!=Find(b))//如果不在一个集合，合并
 {
 ar[Find(a)] = Find(b);
 }
 }
 int Count=0;
 for(int i = 1;i <= N; ++i)
 if(Find(ar[i]) == i)
 Count++;
 cout<<Count-1<<endl;
 }
 return 0;
}

```

## 6.8 树状数组

### 6.8.1 1 树状数组模板.cpp

```

void Add(int x,int p)//
{
 while(x<=N)

```

```

 {
 tree[x] += p;
 x += lowbit(x);
 }
 }
int Query(int x)
{
 int sum = 0;
 while(x)
 {
 sum += tree[x];
 x -= lowbit(x);
 }
 return sum;
}

```

## 6.8.2 2 区间出现两次的数的个数.cpp

```

//..... 离线树状数组
int n,m;
const int LEN = 2e5+100;
int tree[LEN]; //树状数组
int ans[LEN]; //答案数组
int ar[LEN];
int last[LEN]; //last[i] 上一个与 ar[i] 相等的元素的位置
map<int,int> ma; //存储每一个数对应的最后的位置
struct Q
{
 int l,r,ID;
};
Q q[LEN];
bool operator <(const Q &a,const Q &b)
{
 return a.r < b.r;
}
void modify(int x,int d)
{
 while(x <= n)
 {
 tree[x] += d;
 x += lowbit(x);
 }
}
int Query(int x)
{
 int sum = 0;
 while(x>0)
 {
 sum += tree[x];
 x -= lowbit(x);
 }
 return sum;
}

int main()
{

```

```

cin>>n>>m;

for(int i = 1; i <= n; ++i)
{
 scanf("%d",&ar[i]);
 last[i] = ma[ar[i]];
 ma[ar[i]] = i;
}
for(int i = 1; i <= m; ++i)
{
 scanf("%d %d",&q[i].l,&q[i].r);
 q[i].ID = i;
}
sort(q+1,q+m+1);
int index = 1;
/* 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*/
for(int i = 1; i <= n; ++i)
{
 if(last[i]!=0)
 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
 int p = last[last[i]];
 if(p != 0)
 {
 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 */
 int pp = last[p];
 if(pp != 0)//消除-2 的影响
 modify(pp,1);
 }
 // 分析后得知 C[i] 只有三种可能的值,0,-1,1,

 while(index <= m&&q[index].r == i)
 {
 ans[q[index].ID] = Query(i) - Query(q[index].l-1);/* 这个时候 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) 则不是 */
 index ++;
 }
}
for(int i = 1; i <= m; ++i)
 printf("%d\n",ans[i]);
return 0;
}

```

## 6.9 线段树

### 6.9.1 1. 区间更新区间查询.cpp

```

#include<bits/stdc++.h>
using namespace std;
#define lson (o << 1)
#define rson (o << 1|1)
const int maxn = 1e5+10;
const int INF = 1e9;

```



```

typedef long long LL;
struct Tree{
 LL min,max,sum,add;
};
Tree tree[maxn<<2];
LL a[maxn];
void pushup(int o,int l,int r){
 tree[o].min = min(tree[lson].min,tree[rson].max);
 tree[o].max = max(tree[lson].max,tree[rson].max);
 tree[o].sum = tree[lson].sum + tree[rson].sum;
}
void pushdown(int o,int l,int r){
 int m = (l+r)>>1;
 if(tree[o].add){
 tree[lson].add += tree[o].add;
 tree[lson].sum += (m-l+1)*tree[o].add;
 tree[lson].min += tree[o].add;
 tree[lson].max += tree[o].add;

 tree[rson].add += tree[o].add;
 tree[rson].sum += (r-m)*tree[o].add;
 tree[rson].min += tree[o].add;
 tree[rson].max += tree[o].add;
 tree[o].add = 0;
 }
}
void up(Tree & a,Tree b){
 a.min = min(a.min,b.min);
 a.max = max(a.max,b.max);
 a.sum += b.sum;
}
void build(int o,int l,int r){
 // cout<<l<<" "<<r<<endl;
 tree[o].add = 0;
 if(l == r)
 {
 tree[o].min = tree[o].max = tree[o].sum = a[l];
 // cout<<l <<" "<<a[l]<<endl;
 }
 else{
 int m = (l+r)>>1;
 build(lson,l,m);
 build(rson,m+1,r);
 pushup(o,l,r);
 }
}
void Update(int o,int l,int r,int L,int R,int v){
 if(L <= l && R >= r){
 tree[o].add += v;
 tree[o].sum += (r-l+1)*v;
 tree[o].max += v;
 tree[o].min += v;
 return ;
 }
 pushdown(o,l,r);
 int m = (l+r)/2;

```

```

 if(L <= m)
 Update(lson,l,m,L,R,v);
 if(R > m)
 Update(rson,m+1,r,L,R,v);
 pushup(o,l,r);
 }
 Tree Query(int o,int l,int r,int L,int R){

 if(L <= l && R >= r)
 {
 return tree[o];
 }
 Tree tmp;
 tmp.min = INF,tmp.max = -INF,tmp.sum = 0;
 pushdown(o,l,r);
 int m = (l+r)>>1;
 if(L <= m)
 up(tmp,Query(lson,l,m,L,R));
 if(R > m)
 up(tmp,Query(rson,m+1,r,L,R));
 // cout<<tmp.sum<<endl;
 return tmp;
 }
 int main(void){

 int N,Q;cin>>N>>Q;
 for(int i =1;i <= N; ++i)
 scanf("%lld",&a[i]);
 build(1,1,N);
 // cout<<Query(1,1,N,1,1).sum<<endl;
 while(Q--){
 LL c,x,y,v;
 scanf("%lld%lld%lld",&c,&x,&y);
 if(c == 1){
 scanf("%lld",&v);
 Update(1,1,N,x,y,v);
 }
 else{
 printf("%lld\n",Query(1,1,N,x,y).sum);
 }
 }

 return 0;
 }

```

### 6.9.2 2 主席树求第 k 大.cpp

```

// 主席树求第 k 大
// 先离散, 后可持续化建树
// poj 2104

```

```

#include <bits/stdc++.h>
#define me(ar) memset(ar,0,sizeof(ar))
#define rep(i,a,n) for (int i=a;i<n;i++)
using namespace std;

```

```

const int maxn = 1e5+10;
int sum[maxn<<5],L[maxn<<5],R[maxn<<5];
int rt[maxn];
int a[maxn],Hash[maxn];
int tot = 0;
int build(int l,int r){
 int rt = (++tot);
 sum[rt] = 0;
 if(l < r){
 int m = (l+r) >> 1;
 L[rt] = build(l,m);
 R[rt] = build(m+1,r);
 }
 return rt;
}

int update(int pre,int l,int r,int x){
 int rt = (++tot);
 L[rt] = L[pre],R[rt] = R[pre],sum[rt] = sum[pre]+1;
 if(l < r){
 int m = (l+r)>>1;
 if(x <= m)
 L[rt] = update(L[pre],l,m,x);
 else
 R[rt] = update(R[pre],m+1,r,x);
 }
 return rt;
}

int query(int u,int v,int l,int r,int k){
 if(l >= r) return r;
 int num = sum[L[v]]-sum[L[u]];
 int m = (l+r)>>1;
 if(num >= k)
 return query(L[u],L[v],l,m,k);
 return query(R[u],R[v],m+1,r,k-num);
}

int main(void)
{
 int T;
 scanf("%d",&T);
 while(T--){
 tot = 0;
 int n,m;
 scanf("%d%d",&n,&m);
 // map<int,int> ma;
 rep(i,1,n+1){scanf("%d",&a[i]);Hash[i] = a[i];}
 sort(Hash+1,Hash+1+n);
 int id = unique(Hash+1,Hash+1+n) - Hash-1;
 rt[0] = build(1,id);
 rep(i,1,n+1){
 int x = lower_bound(Hash+1,Hash+id+1,a[i]) - Hash;
 rt[i] = update(rt[i-1],1,id,x);
 }
 rep(i,0,m){
 int l,r,k;

```

```

 scanf("%d%d%d",&l,&r,&k);
 int ans = query(rt[l-1],rt[r],1,id,k);
 printf("%d\n",Hash[ans]);
 }
}

return 0;
}

```

### 6.9.3 2 树套树求动态第 k 大.cpp

```

/*
ZOJ
Dynamic Rankings ZOJ - 2112
动态第 k 大数
*/
//lowbit 自己写
#define lson l,m
#define rson m+1,r
const int N = 60006;
int a[N],Hash[N];
int T[N],L[N<<5],R[N<<5],sum[N<<5];
int S[N];
int n,m,tot;
struct node{
 int l,r,k;
 bool Q;
}op[10005];

int build(int l,int r){
 int rt = (++tot);
 sum[rt] = 0;
 if(l != r){
 int m = (l+r)>>1;
 L[rt] = build(lson);
 R[rt] = build(rson);
 }
 return rt;
}

int update(int pre,int l,int r,int x,int val){
 int rt = (++tot);
 L[rt] = L[pre],R[rt] = R[pre],sum[rt] = sum[pre]+val;
 if(l < r){
 int m = (l+r)>>1;
 if(x <= m)
 L[rt] = update(L[pre],lson,x,val);
 else
 R[rt] = update(R[pre],rson,x,val);
 }
 return rt;
}

int use[N];
void add(int x,int pos,int val){
 while(x <= n){
 S[x] = update(S[x],1,m,pos,val);
 }
}

```

```

 x += lowbit(x);
}
}
int Sum(int x){
 int ret = 0;
 while(x > 0){
 ret += sum[L[use[x]]];
 x -= lowbit(x);
 }
 return ret;
}

int query(int u,int v,int lr,int rr,int l,int r,int k){
 if(l >= r)
 return l;
 int m = (l+r)>>1;
 int tmp = Sum(v)-Sum(u)+sum[L[rr]]-sum[L[lr]];
 if(tmp >= k){
 for(int i = u;i;i -= lowbit(i))
 use[i] = L[use[i]];
 for(int i = v;i;i -= lowbit(i))
 use[i] = L[use[i]];
 return query(u,v,L[lr],L[rr],lson,k);
 }
 else{
 for(int i = u;i ;i -= lowbit(i))
 use[i] = R[use[i]];
 for(int i = v;i ;i -= lowbit(i))
 use[i] = R[use[i]];
 return query(u,v,R[lr],R[rr],rson,k-tmp);
 }
}

}

void modify(int x,int p,int d){
 while(x <= n){
 S[x] = update(S[x],1,m,p,d);
 x += lowbit(x);
 }
}

int main(){
 int t;
 scanf("%d",&t);
 while(t--){
 int q;
 scanf("%d%d",&n,&q);
 tot = 0;
 m = 0;
 for(int i = 1;i <= n; ++i)
 {
 scanf("%d",&a[i]);
 Hash[++m] = a[i];
 }
 for(int i = 0;i < q; ++i){
 char s[10];
 scanf("%s",s);

```

```

 if(s[0] == 'Q'){
 scanf("%d%d%d",&op[i].l,&op[i].r,&op[i].k);
 op[i].Q = 1;
 }
 else{
 scanf("%d%d",&op[i].l,&op[i].r);
 op[i].Q = 0;
 Hash[++m] = op[i].r;
 }
}
sort(Hash+1,Hash+1+m);
int mm = unique(Hash+1,Hash+1+m)-Hash-1;
m = mm;
T[0] = build(1,m);
for(int i = 1;i <= n; ++i)
 T[i] = update(T[i-1],1,m,lower_bound(Hash+1,Hash+1+m,a[i])-Hash,1);
// DEBUG;

for(int i = 1;i <= n; ++i)
 S[i] = T[0];
for(int i = 0;i < q; ++i){
 // DEBUG;
 if(op[i].Q){

 // cout<<op[i].l<<" "<<op[i].r<<" "<<endl;
 for(int j = op[i].l-1;j;j -= lowbit(j))
 use[j] = S[j];
 for(int j = op[i].r ;j;j -= lowbit(j))
 use[j] = S[j];
 // DEBUG;

 ↪ 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)]);

 }
 else{
 modify(op[i].l,lower_bound(Hash+1,Hash+1+m,a[op[i].l])-Hash,-1);
 modify(op[i].l,lower_bound(Hash+1,Hash+1+m,op[i].r)-Hash,1);
 a[op[i].l] = op[i].r;
 }

}
}
return 0;
}

/*
2
5 3
3 2 1 4 7
Q 1 4 3
C 2 6
Q 2 5 3
5 3
3 2 1 4 7
Q 1 4 3
C 2 6

```

Q 2 5 3  
\*/

### 6.9.4 3 树套树求动态逆序数.cpp

```
//数据范围 1-n 的全排列
#include<bits/stdc++.h>
#define inf 0x7fffffff
#define N 100005
#define M 5000005
using namespace std;
typedef long long ll;
ll ans;
int n,m,sz,a[100],b[100],val[N],pos[N],a1[N],a2[N];
int c[N*10],rt[N],ls[M],rs[M],sumv[M];
inline int lowbit(int x){return x&(-x);}
inline int ask(int x){
 int ans=0;
 for(int i=x;i;i-=lowbit(i))ans+=c[i];
 return ans;
}
void change(int &o,int l,int r,int q){
 if(!o)o=++sz;sumv[o]++;
 if(l==r)return;
 int mid=(l+r)>>1;
 if(q<=mid)change(ls[o],l,mid,q);
 else change(rs[o],mid+1,r,q);
}
int querysub(int x,int y,int v){
 int cnta=0,cntb=0;int ans=0;x--;
 for(int i=x;i;i-=lowbit(i))a[++cnta]=rt[i];
 for(int i=y;i;i-=lowbit(i))b[++cntb]=rt[i];
 int l=1,r=n;
 while(l!=r){
 int mid=(l+r)>>1;
 if(v<=mid){
 for(int i=1;i<=cnta;i++)ans-=sumv[rs[a[i]]];
 for(int i=1;i<=cntb;i++)ans+=sumv[rs[b[i]]];
 for(int i=1;i<=cnta;i++)a[i]=ls[a[i]];
 for(int i=1;i<=cntb;i++)b[i]=ls[b[i]];
 r=mid;
 }
 else{
 for(int i=1;i<=cnta;i++)a[i]=rs[a[i]];
 for(int i=1;i<=cntb;i++)b[i]=rs[b[i]];
 l=mid+1;
 }
 }
 return ans;
}
int querypre(int x,int y,int v){
 int cnta=0,cntb=0,ans=0;x--;
 for(int i=x;i;i-=lowbit(i))a[++cnta]=rt[i];
 for(int i=y;i;i-=lowbit(i))b[++cntb]=rt[i];
 int l=1,r=n;
 while(l!=r){
```

```

 int mid=(l+r)>>1;
 if(v>mid){
 for(int i=1;i<=cnta;i++)ans-=sumv[ls[a[i]]];
 for(int i=1;i<=cntb;i++)ans+=sumv[ls[b[i]]];
 for(int i=1;i<=cnta;i++)a[i]=rs[a[i]];
 for(int i=1;i<=cntb;i++)b[i]=rs[b[i]];
 l=mid+1;
 }
 else{
 for(int i=1;i<=cnta;i++)a[i]=ls[a[i]];
 for(int i=1;i<=cntb;i++)b[i]=ls[b[i]];
 r=mid;
 }
}
return ans;
}
inline int read(){
 int f=1,x=0;char ch;
 do{ch=getchar();if(ch=='-')f=-1;}while(ch<'0' || ch>'9');
 do{x=x*10+ch-'0';ch=getchar();}while(ch>='0'&&ch<='9');
 return f*x;
}
int main(){
 n=read();m=read();
 for(int i=1;i<=n;i++){
 val[i]=read();pos[val[i]]=i;
 a1[i]=ask(n)-ask(val[i]);
 ans+=a1[i];
 for(int j=val[i];j<=n;j+=lowbit(j))c[j]++;
 }
 memset(c,0,sizeof(c));
 for(int i=n;i;i--){
 a2[i]=ask(val[i]-1);
 for(int j=val[i];j<=n;j+=lowbit(j))c[j]++;
 }
 for(int i=1;i<=m;i++){
 printf("%lld\n",ans);
 int x=read();x=pos[x];
 ans-=(a1[x]+a2[x]-querysub(1,x-1,val[x])-querypre(x+1,n,val[x]));
 for(int j=x;j<=n;j+=lowbit(j))change(rt[j],1,n,val[x]);
 }
 return 0;
}

```

// 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。

```

#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<algorithm>
#include<cstring>
#include<queue>
#include<vector>
#define ll long long
const int maxn=100000+9999;
using namespace std;
int n,m,num[maxn],H[maxn],Q[maxn],cnt,root[maxn*50],t[maxn],pos[maxn];

```



```

int A[100],B[100];
ll ans;
int LO(int x){return x&-x;}
int qsum(int x){
 int tmp=0;
 for(int i=x;i;i-=LO(i))
 tmp+=t[i];
 return tmp;
}
int read(){
 int an=0,f=1;
 char ch=getchar();
 while(ch<'0' || ch>'9'){if(ch=='-')f=-1;ch=getchar();}
 while('0'<=ch&&ch<='9'){an=an*10+ch-'0';ch=getchar();}
 return an*f;
}
struct saber{
 int r,l,sum;
}T[maxn*50];
int askmore(int x,int y,int wi){
 int cnt1,cnt2,tmp=0;cnt1=cnt2=0;
 for(int i=x;i;i-=LO(i))cnt1++,A[cnt1]=root[i];
 for(int i=y;i;i-=LO(i))cnt2++,B[cnt2]=root[i];
 int l=1,r=n;
 while(l!=r){
 int mid=(l+r)>>1;
 if(wi<=mid){
 for(int i=1;i<=cnt1;i++)tmp-=T[T[A[i]].r].sum;
 for(int i=1;i<=cnt2;i++)tmp+=T[T[B[i]].r].sum;
 for(int i=1;i<=cnt1;i++)A[i]=T[A[i]].l;
 for(int i=1;i<=cnt2;i++)B[i]=T[B[i]].l;
 r=mid;
 }
 else {
 for(int i=1;i<=cnt1;i++)A[i]=T[A[i]].r;
 for(int i=1;i<=cnt2;i++)B[i]=T[B[i]].r;
 l=mid+1;
 }
 }
 return tmp;
}
int askless(int x,int y,int wi){
 int cnt1,cnt2,tmp=0;
 cnt1=cnt2=0;x--;
 for(int i=x;i;i-=LO(i))cnt1++,A[cnt1]=root[i];
 for(int i=y;i;i-=LO(i))cnt2++,B[cnt2]=root[i];
 int l=1,r=n;
 while(l!=r){
 int mid=(l+r)>>1;
 if(wi>mid){
 for(int i=1;i<=cnt1;i++)tmp-=T[T[A[i]].l].sum;
 for(int i=1;i<=cnt2;i++)tmp+=T[T[B[i]].l].sum;
 for(int i=1;i<=cnt1;i++)A[i]=T[A[i]].r;
 for(int i=1;i<=cnt2;i++)B[i]=T[B[i]].r;
 l=mid+1;
 }
 }
}

```

```

 else {
 for(int i=1;i<=cnt1;i++)A[i]=T[A[i]].l;
 for(int i=1;i<=cnt2;i++)B[i]=T[B[i]].l;
 r=mid;
 }
 }
 return tmp;
}

void add(int &y,int l,int r,int wi){
 if(!y)cnt++,y=cnt;
 T[y].sum++;
 if(l==r)return ;
 int mid=(l+r)>>1;
 if(wi<=mid)add(T[y].l,l,mid,wi);
 else add(T[y].r,mid+1,r,wi);
}

struct da{
 int wi,i;
}data[maxn];

bool cmp1(da x,da y){
 return x.wi<y.wi;
}

bool cmp2(da x,da y){
 return x.i<y.i;
}

void prepare(){
 n=read();m=read();
 for(int i=1;i<=n;i++){
 data[i].wi=read();
 data[i].i=i;
 }
 sort(data+1,data+1+n,cmp1);
 for(int i=1;i<=n;i++){
 data[i].wi=i;
 }
 sort(data+1,data+1+n,cmp2);
 for(int i=1;i<=n;i++)
 num[i]=data[i].wi;
}

int main(){
 prepare();
 for(int i=1;i<=n;i++){
 Q[i]=qsum(n)-qsum(num[i]);//Q 在 i 这个点前面比 i 大的数贡献
 ans+=Q[i];
 for(int j=num[i];j<=n;j+=LO(j)){
 t[j]++;
 }
 }
 memset(t,0,sizeof(t));
 for(int i=n;i;i--){
 H[i]=qsum(num[i]-1);
 for(int j=num[i];j<=n;j+=LO(j))
 t[j]++;
 }
 printf("%lld ",ans);
 while(m){m--;

```

```

 int x=read();
 ans+=(H[x]+Q[x]-askmore(0,x-1,num[x])-askless(x+1,n,num[x])));
 for(int j=x;j<=n;j+=L0(j))add(root[j],1,n,num[x]);
 printf("%lld ",ans);
}
return 0;
}

```

#### 6.9.5 4 李超树.cpp

// 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

// 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  
 // 插入点的时候  
 // 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  
 // 共有四种情况  
 // 插入直线的斜率大于节点存的斜率,  
 // 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  
 ↪  $l, mid$   
 // 如果插入直线的值小, 那么用插入直线更新  $mid+1, r$ ;  
 // 如果插入直线的斜率小于节点存的斜率  
 // 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  $mid+1, r$   
 // 如果插入直线的值小, 那么用插入直线更新  $l, mid+1$ ;

```

#include <bits/stdc++.h>
using namespace std;
const int N = 5e5+10;
int n,m,tree[N*4];
double a[N*2],b[N*2];
int cmp(int x,int y,int pos){
 return a[x] + (pos-1)*b[x] > a[y] + (pos-1)*b[y];
}
void update(int o,int l,int r,int x){
 if(l == r){
 if(cmp(x,tree[o],l))
 tree[o] = x;
 return ;
 }
 int mid = (l+r)/2;
 if(b[x] > b[tree[o]]){
 if(cmp(x,tree[o],mid)){
 update(o<<1,l,mid,tree[o]),tree[o] = x;
 }
 else
 update(o<<1|1,mid+1,r,x);
 }
 if(b[x] < b[tree[o]]){
 if(cmp(x,tree[o],mid)){
 update(o<<1|1,mid+1,r,tree[o]),tree[o] = x;
 }
 else
 update(o<<1,l,mid,x);
 }
}

```

```

 }

}

double cal(int k,int x){
 return a[k] + (x-1)*b[k];
}

double query(int o,int l,int r,int x){
 if(l==r) return cal(tree[o],x);
 int mid = (l+r)/2;
 double ans = cal(tree[o],x);
 if(x <= mid) ans = max(ans,query(o<<1,l,mid,x));
 else
 ans = max(ans,query(o<<1|1,mid+1,r,x));
 return ans;
}

int main(void)
{
 scanf("%d",&n);
 for(int i = 1;i <=n; ++i){
 char s[20];
 scanf("%s",s);
 if(s[0] == 'P'){
 m++;
 scanf("%lf%lf",&a[m],&b[m]);
 update(1,1,N,m);
 }
 else{
 int x;
 scanf("%d",&x);
 double t = query(1,1,N,x);
 int k = t;
 printf("%d\n",k/100);
 }
 }

 return 0;
}

```

### 6.9.6 5 线段树-区间最小乘积.cpp

// 单点更新，区间查询

```

#include <bits/stdc++.h>
#define me(ar) memset(ar,0,sizeof(ar))
using namespace std;
const int INF = 100000;
const int maxn = 1e6+10;
const int maxnode = 4*maxn;
int ql,qr;
int _p,_v;
struct T{
 int a,b,c,d;
 T(int aa = -INF,int bb = -INF,int cc = INF,int dd = INF):a(aa),b(bb),c(cc),d(dd){
 }
}

```

```

};
T up(T x,T y)
{
 int a[4] = {x.a,x.b,y.a,y.b};
 sort(a,a+4);
 x.a = a[3];
 x.b = a[2];
 int b[4] = {x.c,x.d,y.c,y.d};
 sort(b,b+4);
 x.c = b[0];
 x.d = b[1];
 return x;
}
T vv[maxnode];
T a[maxn];
void build(int o,int l,int r)
{
 int m = (r+l)>>1;
 if(l == r) vv[o] = a[l];
 else
 {
 build(o*2,l,m);
 build(o*2+1,m+1,r);
 vv[o] = up(vv[o*2],vv[o*2+1]);
 }
}
void update(int o,int l,int r)
{
 if(l == r) vv[o] = T(_v,-INF,_v,INF);
 else
 {
 int m = (r+l)>>1;
 if(_p <= m)
 update(o*2,l,m);
 else
 update(o*2+1,m+1,r);
 vv[o] = up(vv[o*2],vv[o*2+1]);
 }
}
T query(int o,int l,int r)
{
 if(l >= ql&&r <= qr)
 return vv[o];
 int m = l+(r-l)/2;
 T ans;
 if(ql <= m&&m < qr)
 ans = up(query(o*2,l,m),query(o*2+1,m+1,r));
 else if(ql <= m)
 ans = query(o*2,l,m);
 else if(m < qr)
 ans = query(o*2+1,m+1,r);
 return ans;
}

int main(void)

```

```

{
 int N,Q;
 while(scanf("%d",&N) != EOF&&N)
 {
 for(int i = 1;i <= N; ++i)
 {
 int aa;
 scanf("%d",&aa);
 a[i] = T(aa,-INF,aa,INF);
 }
 build(1,1,N);
 cin>>Q;
 while(Q-->0)
 {
 int op;
 scanf("%d",&op);
 if(op == 1)
 {
 scanf("%d %d",&p,&v);
 update(1,1,N);
 }
 else
 {
 scanf("%d %d",&l,&r);
 T ans = query(1,1,N);
 long long an = min(ans.a*ans.b,min(ans.a*ans.c,ans.c*ans.d));
 printf("%lld\n",an);
 }
 }
 }

 return 0;
}

```

### 6.9.7 6 区间加斐波那契数.cpp

*//CodeForces 446C DZY Loves Fibonacci Numbers*

```

#include <cstdio>

const int maxn=300000;
const long long mod=1e9+9;

struct fenw {
 long long tree[maxn+10];
 void add(int i, long long d) {
 for (;i<maxn+10;i|=(i+1)) tree[i]=tree[i]+d;
 }
 long long get(int i) {
 long long ans=0;
 for (;i>=0; i=(i&(i+1))-1) ans+=tree[i];
 return ans%mod;
 }
};

```

```

fenv t1, t2, t3;
long long fb[maxn+10], s[maxn+10];
int n, m, a, t, l, r;
char ss[20];

inline long long getfb(int i) {
 if (i>0) return fb[i];
 else if (i%2) return fb[-i];
 else return mod-fb[-i];
}

inline int geti() {
 char ch=getchar();
 while (ch<'0' || ch>'9') ch=getchar();
 int ans=0;
 while (ch>='0' && ch<='9') ans=(ans*10+ch-'0'), ch=getchar();
 return ans;
}

inline void puti(int i) {
 int j=0;
 while (i) ss[j]=(i%10)+'0', j++, i/=10;
 for (j--; j>=0; j--) putchar(ss[j]);
 putchar('\n');
}

int main() {
 fb[1]=fb[2]=1;
 for (int i=3; i<maxn+10; i++) fb[i]=(fb[i-1]+fb[i-2])%mod;
 n=geti(), m=geti();
 for (int i=1, sum=0; i<=n; i++) a=geti(), sum=(sum+a)%mod, s[i]=sum;
 for (int i=0; i<m; i++) {
 t=geti(), l=geti(), r=geti();
 if (t==1) {
 long long c=getfb(2-l), d=getfb(3-l);
 t1.add(l, c);
 t2.add(l, d);
 t3.add(l, -1);
 t1.add(r, -c);
 t2.add(r, -d);
 t3.add(r, fb[r-l+3]);
 } else {
 puti((int)
 ↪ ((t3.get(r)+t1.get(r)*fb[r]+t2.get(r)*fb[r+1]-t3.get(l-1)-t1.get(l-1)*fb[l-1]-
)
 }
 }
 return 0;
}

// #include <bits/stdc++.h>
#define eps 1e-6
#define LL long long
#define pii pair<int, int>
#define pb push_back
#define mp make_pair
// #pragma comment(linker, "/STACK:1024000000,1024000000")

```

```

using namespace std;

const int MAXN = 1500000;
const int MOD = 1e9+9;
LL bas = 276601605;
LL q1 = 691504013;
LL q2 = 308495997;
LL mul1[MAXN], mul2[MAXN];
int c[MAXN];
LL s[MAXN];

struct Node {
 LL a, b, sum;
} node[MAXN];
int n, k;

void init(int m) {
 mul1[0] = mul2[0] = 1;
 for (int i = 1; i <= m; i++) {
 mul1[i] = mul1[i-1] * q1 % MOD;
 mul2[i] = mul2[i-1] * q2 % MOD;
 }
}

void build(int o, int l, int r) {
 node[o].a = node[o].b = node[o].sum = 0;
 if (l == r) return;
 int m = (l+r) >> 1;
 build(o<<1, l, m);
 build((o<<1)+1, m+1, r);
}

void push_down(int o, int l, int r) {
 LL aa = node[o].a, bb = node[o].b;
 if (!aa && !bb) return;
 int lc = o << 1, rc = (o<<1)|1, mid = (l+r) >> 1;
 int len1 = mid-l+1, len2 = r - mid;

 node[lc].a = (node[lc].a+aa) % MOD;
 node[lc].b = (node[lc].b+bb) % MOD;
 node[lc].sum = (node[lc].sum+aa*(mul1[len1+2]-mul1[2])) % MOD;
 node[lc].sum = (node[lc].sum-bb*(mul2[len1+2]-mul2[2])) % MOD;

 node[rc].a = (node[rc].a+aa*mul1[len1]) % MOD;
 node[rc].b = (node[rc].b+bb*mul2[len1]) % MOD;
 node[rc].sum = (node[rc].sum + aa*mul1[len1]%MOD*(mul1[len2+2]-mul1[2])%MOD)
 ↪ % MOD;
 node[rc].sum = (node[rc].sum - bb*mul2[len1]%MOD*(mul2[len2+2]-mul2[2])%MOD)
 ↪ % MOD;

 node[o].a = node[o].b = 0;
}

void push_up(int o) {
 node[o].sum = (node[o<<1].sum+node[(o<<1)|1].sum) % MOD;
}

LL query(int o, int l, int r, int ql, int qr) {
 if (l == ql && r == qr)
 return node[o].sum;
}

```



```

push_down(o, l, r);
int mid = (l+r) >> 1;
if (qr <= mid)
 return query(o<<1, l, mid, ql, qr);
else if (ql > mid)
 return query((o<<1)|1, mid+1, r, ql, qr);
else
 return (query(o<<1, l, mid, ql, mid)+query((o<<1)|1, mid+1, r, mid+1,
 ↪ qr)) % MOD;
}

void update(int o, int l, int r, int ql, int qr, LL x, LL y) {
 if (l == ql && r == qr) {
 node[o].a = (node[o].a+x) % MOD;
 node[o].b = (node[o].b+y) % MOD;
 node[o].sum = (node[o].sum+x*(mul1[r-l+3]-mul1[2])) % MOD;
 node[o].sum = (node[o].sum-y*(mul2[r-l+3]-mul2[2])) % MOD;
 return;
 }
 push_down(o, l, r);
 int mid = (l+r) >> 1;
 if (qr <= mid)
 update(o<<1, l, mid, ql, qr, x, y);
 else if (ql > mid)
 update((o<<1)|1, mid+1, r, ql, qr, x, y);
 else {
 int len = mid - ql + 1;
 update(o<<1, l, mid, ql, mid, x, y);
 update((o<<1)|1, mid+1, r, mid+1, qr, x*mul1[len]%MOD,
 ↪ y*mul2[len]%MOD);
 }
 push_up(o);
}

int main()
{
 //freopen("input.txt", "r", stdin);
 scanf("%d%d", &n, &k);
 for (int i = 1; i <= n; i++) {
 scanf("%d", &c[i]);
 s[i] = s[i-1] + c[i];
 }
 init(301000);
 build(1, 1, n);
 for (int i = 1; i <= k; i++) {
 int op, l, r;
 scanf("%d%d%d", &op, &l, &r);
 if (op == 1)
 update(1, 1, n, l, r, 1, 1);
 else {
 LL ans = (bas*query(1, 1, n, l, r)%MOD+s[r]-s[l-1]) % MOD;
 if (ans < 0) ans += MOD;
 printf("%I64d\n", ans);
 }
 }
 return 0;
}

```

### 6.9.8 7 区间加 + 区间乘.cpp

//洛谷 P3373

```
const int maxn = 100000+10;
LL n,m,mod;
LL sumv[maxn<<2],addv[maxn<<2],mulv[maxn<<2];
LL a[maxn];
#define lc (o<<1)
#define rc (o<<1|1)
void maintain(int o,int l,int r){
 sumv[o] = sumv[lc]+sumv[rc];
 sumv[o] %= mod;
}
void pushdown(int o,int l,int r){
 int m = (l+r)>>1;
 if(mulv[o] != 1){
 sumv[lc] = sumv[lc]*mulv[o]%mod,sumv[rc] = sumv[rc]* mulv[o]%mod;
 addv[lc] = addv[lc] *mulv[o]%mod,addv[rc] = addv[rc] * mulv[o]%mod;
 mulv[lc] = (mulv[lc]*mulv[o])%mod,mulv[rc] = (mulv[rc]* mulv[o]%mod);
 mulv[o] = 1;
 }
 if(addv[o]){
 sumv[lc] = (sumv[lc]+addv[o]*(m-l+1))%mod;
 addv[lc] = (addv[lc]+addv[o])%mod;
 sumv[rc] = (sumv[rc]+addv[o]*(r-m))%mod;
 addv[rc] = (addv[rc]+addv[o])%mod;
 addv[o] = 0;
 }
}

void build(int o,int l,int r){
 if(l == r){
 sumv[o] = a[l];
 addv[o] = 0;
 mulv[o] = 1;
 return ;
 }
 int m = (l+r)>>1;
 build(lc,l,m);
 build(rc,m+1,r);
 // sumv[o] =
 addv[o] = 0,mulv[o] = 1;
 maintain(o,l,r);
}

int op;
void update(int o,int l,int r,int L,int R,LL v){
 if(L <= l &&R >= r){
 if(op == 2){
 sumv[o] = (sumv[o]+v*(r-l+1))%mod;
 addv[o] += v;
 }
 else{

```

```

 sumv[o] = (sumv[o]*v)%mod;
 addv[o] = (addv[o]*v)%mod;
 mulv[o] = (mulv[o]*v)%mod;
 }
}
else{
 int m = (l+r)>>1;
 pushdown(o,l,r);
 if(L <= m)
 update(lc,l,m,L,R,v);
 if(R > m)
 update(rc,m+1,r,L,R,v);
 maintain(o,l,r);
}
}
LL _sum;
void query(int o,int l,int r,int L,int R){
 if(L <= l && R >= r){
 _sum += sumv[o];
 _sum %= mod;
 return ;
 }
 pushdown(o,l,r);
 int m = (l+r)>>1;
 if(L <= m)
 query(lc,l,m,L,R);
 if(R > m)
 query(rc,m+1,r,L,R);
 // pushup()
}

```

```

int main(void){
 cin>>n>>m>>mod;
 for(int i = 1;i <= n; ++i)
 scanf("%lld",&a[i]);
 build(1,1,n);
 // _sum = 0;
 // query(1,1,n,1,n);
 // cout<<_sum<<endl;
 for(int i = 1;i <= m; ++i){
 int x,y,v;
 scanf("%d%d%d",&op,&x,&y);
 if(op == 1||op == 2){
 scanf("%d",&v);
 update(1,1,n,x,y,v);
 }
 else{
 _sum = 0;
 query(1,1,n,x,y);
 _sum %= mod;
 printf("%lld\n",_sum);
 }
 }
}

```

```

 return 0;
 }

```

## 7 模拟

### 7.1 1 日期.cpp

1 计算日期差

```

#include <stdio.h>
#include <stdlib.h>

bool isLeapYear(int year)
{
 return ((year%4==0 && year%100!=0) || year%400==0);
}
// 以公元 1 年 1 月 1 日为基准, 计算经过的日期
int getDays(int year, int month, int day)
{
 int m[] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
 if(isLeapYear(year))
 m[2]++;
 int result = 0;
 for(int i = 1; i < year; i++)
 {
 result += 365;
 if(isLeapYear(i))
 result ++;
 }
 for(int i = 1; i < month; i++)
 {
 result += m[i];
 }
 result += day;

 return result;
}
int dayDis (int year1, int month1, int day1,
 int year2, int month2, int day2)
{
 return abs(getDays(year2, month2, day2) - getDays(year1, month1, day1));
}

int main(void)
{
 printf("%d\n", dayDis(2012, 9, 1, 2018, 3, 25));

 return 0;
}
2 计算某一天星期几
int cal1(int y, int m, int d)
{
 if(m==1 || m==2)
 m+=12, y--;

```

```

 int w=(d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
 return ++w;
}
int cal2(int y,int m,int d)
{
 if(m==1||m==2)
 m+=12,y--;
 int c=y/100,ty=y%100;
 int w=ty+ty/4+c/4-2*c+26*(m+1)/10+d-1;
 return w%7==0?7:(w+7)%7;
}
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
#include<bits/stdc++.h>

using namespace std;

int year,month,day;
int a1,b1,c1,a2,b2,c2;

const int maxn = 1e4+100;
int a[maxn];
int c[maxn]; // 代表当前年所有的 9
// int mon[30] = {0,2,2,2,2}
int run(int y){
 return y%400 == 0 || (y%4==0&& y%100!=0);
}
int wanyue(int t,int y){
 if(t == 2) return 2+run(y);
 if(t == 9) return 3+30;
 return 3;
}
int wanyear(int t){
 int num = 0;
 int tt = t;
 while(tt > 0){
 if(tt % 10 == 9) num++;
 tt /= 10;
 }
 a[t] = num;
 int tmp = run(t);
 return num*(365+tmp)+65+tmp;
}
int mo[20] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
int Howmuchday(int y,int t){
 if(t==2){
 return run(y)+28;
 }
 return mo[t];
}
int subday(int a,int b){
 int sum = 0;
 for(int i = a;i <= b; ++i)
 if(i%10 == 9)
 sum++;
}

```

```

 return sum;
 }
 int numsubday(int a,int b){
 return b-a+1;
 }

 int numsubday(int y,int b1,int c1,int b2,int c2){
 int num = 0;
 if(b1 == b2)
 return numsubday(c1,c2);
 for(int i = b1+1;i < b2; ++i)
 num += mo[i]+(i==2&&run(y));
 num += numsubday(c1,Howmuchday(y,b1));
 num += numsubday(1,c2);
 return num;
 }
 int FF(int t){
 int num = 0;
 int tt = t;
 while(tt > 0){
 if(tt % 10 == 9) num++;
 tt /= 10;
 }
 return num;
 }
 int submonth(int y,int b1,int c1,int b2,int c2){
 if(b1 == b2)
 return subday(c1,c2)+(c2-c1+1)*FF(b1);
 int sum = 0;
 for(int i = b1+1;i < b2; ++i)
 sum += wanyue(i,y);

 sum += subday(c1,Howmuchday(y,b1))+FF(b1)*(Howmuchday(y,b1)-c1+1);
 // cout<<sum<<endl;
 sum += subday(1,c2)+FF(b2)*(c2);
 return sum;
 }

 int subyear(int a1,int b1,int c1,int a2,int b2,int c2){
 if(a1 == a2)
 return numsubday(a1,b1,c1,b2,c2)*a[a1] + submonth(a1,b1,c1,b2,c2);
 int ans = 0;
 ans += c[a2-1]-c[a1];
 ans += numsubday(a1,b1,c1,12,31)*a[a1];
 ans += numsubday(a2,1,1,b2,c2)*a[a2];
 return ans + submonth(a1,b1,c1,12,31)+submonth(a2,1,1,b2,c2);
 }

 int main(void){

 for(int i = 2000;i < maxn; ++i){
 c[i] = wanyear(i);
 c[i] += c[i-1];
 }
 int T;
 cin>>T;

```

```

 while(T--){
 scanf("%d%d%d %d%d%d",&a1,&b1,&c1,&a2,&b2,&c2);
 int ans = subyear(a1,b1,c1,a2,b2,c2);
 printf("%d\n",ans);
 }
 return 0;
}
// 同上
#include <stdio.h>
#include <string.h>

int sum[10005][15][35],pre[10005][15][35];
int mon[15] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

int leap(int x)
{
 if (x % 400 == 0) return 1;
 if (x % 100 == 0) return 0;
 if (x % 4 == 0) return 1;

 return 0;
}

int check(int y,int m,int d)
{
 int num = 0;

 while (y)
 {
 y % 10 == 9 ? ++num : num += 0;
 y /= 10;
 }

 while (m)
 {
 m % 10 == 9 ? ++num : num += 0;
 m /= 10;
 }

 while (d)
 {
 d % 10 == 9 ? ++num : num += 0;
 d /= 10;
 }

 return num;
}

void init(int y1,int m1,int d1,int y2,int m2,int d2)
{
 int tmp = 0;

 while (y1 != y2 || m1 != m2 || d1 != d2)
 {

```

```

mon[2] = leap(y1) + 28;

pre[y1][m1][d1] = tmp; //tmp 是到前一个日期显示的 9 的数量。

tmp += check(y1,m1,d1);

sum[y1][m1][d1] = tmp; //现在的日期显示的 9 的数量

if (++d1 > mon[m1])
{
 d1 = 1;

 if (++m1 > 12)
 {
 m1 = 1;
 mon[2] = 28 + leap(++y1);
 }
}
}

int main()
{
 int t;

 scanf("%d",&t);

 init(2000,1,1,10000,1,1);

 while (t--)
 {
 int y1,m1,d1,y2,m2,d2;

 scanf("%d%d%d%d%d%d",&y1,&m1,&d1,&y2,&m2,&d2);

 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。
 }

 return 0;
}

```