A.Dividing Candy 模拟 构造
B.Fireworks 三分 数学
C.Filth Rich Trees 树状数组 树 数学 思维
D.Make 10 贪心
E.Can you answer there queries I 线段树
F.Diluc and Kaeya 计算几何 数学
G.Quadruple 数学
H.Non-Integer Donuts 模拟
个人建议顺序 A->D->H->G->F->B

A. Dividing Candy

题意：给定 n 个a_i，每个位置代表2^{a_i}，问是否最终能正好合并（加）成两个 2 的乘幂

肯定是尽量合并成最大的数，看最后剩下的数是否正好剩下两个

```
#include <cstdio>

using namespace std;

int n, left;
int vis[500005];

int main(){
	scanf("%d",&n);
	for(int i = 1, a ; i <= n ; ++i){
		scanf("%d",&a);
		++vis[a];
	}
	for(int i = 0 ; i <= 200000 ; ++i){
		if(vis[i]){
			left += vis[i]&1;
			vis[i+1] += vis[i]>>1;
		}
	}
	if(left==2||(left==1&&n^1)) puts("Y");
	else puts("N");
}
```

B. Fireworks

记得这个是 2020ICPC 南京的铜牌题

设每轮制作k个烟花，则每轮的开销为$k*n+m$，最少一个成功烟花的的概率为$(1-(1-p)^k)$，根据几何分布得出此时期望$(1-(1-p)^k)^{-1}$轮做出一个成功烟花，所以总的期望花费为$\frac{k*n+m}{(1-(1-p)^k)}$

遍历k打表得知这是一个单凹的函数，最后三分求解即可

```
#include <cstdio>
#include <cmath>

using namespace std;
typedef long long ll;

int t, n, m;
long double p;

inline double fun(ll k){
	return (double)(k*n+m)/(double)(1.0-pow(1.0-p,k));
}

void solve(){
	scanf("%d %d %Lf",&n,&m,&p);
	p = p * 1e-4;
	// dabiao
	/*
	for(int i = 1 ; i <= 1e4 ; ++i){
		printf("%.10lf\n",(double)(i*n+m)/(double)(1.0-pow(1.0-p,i)));
	}
	*/
	ll l = 1, r = 0xffffffff;
	while(l<r){
		ll mid1 = l + (r - l) / 3, mid2 = r - (r - l) / 3;
		if(fun(mid1)<fun(mid2)) r = mid2 - 1;
		else l = mid1 + 1;
	}
	printf("%.10lf\n",fun(l));
}

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

C. Filthy Rich Trees

用对数存储可以解决数据过大的问题，即每个节点不存放原来的值，转换为存储 $\ln(a_i)$

1 操作 在对数上操作 $a*b = \ln a + \ln b$ $a/b = \ln a - \ln b$

2 操作 求倍数关系，从对数还原成小数 $a/b = e^{\ln(a)-\ln(b)}$

在树上更新值的时间复杂度为$O(n)$

所以可以用树状数组存储节点信息，用一次 DFS 在节点和树状数组间建立关系即可

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
#include <string.h>
#include <algorithm>
#include <cmath>
#define max(a,b) ((a) > (b) ? (a) : (b))
#define min(a,b) ((a) < (b) ? (a) : (b))
#define pb push_back
#define cl clear
#define MAXN 300005

using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> P;
const int INF = 1e9;

inline int read(){
	int x = 0, f = 1;
	char ch = getchar();
	while(ch<'0'||ch>'9'){
		if(ch=='-')
			f = -1;
		ch = getchar();
	}
	while(ch>='0'&&ch<='9'){
		x=(x<<1)+(x<<3)+(ch^48);
		ch = getchar();
	}
	return x*f;
```

```
}

int head[MAXN], tot;

struct Edge{
	int u, to, next;
}G[MAXN<<1];

inline void addEdge(int u, int v){
	G[++tot].u = u;
	G[tot].to = v;
	G[tot].next = head[u];
	head[u] = tot;
}

P seg[MAXN];
int n, q;
double tree[MAXN], valOfNode[MAXN];

int DFS(int x, int counter, int fat){
	seg[x].first = counter;
	for(int i = head[x], to ; i ; i = G[i].next){
		to = G[i].to;
		if(to^fat){
			counter = DFS(to,counter+1,x);
		}
	}
	return seg[x].second = counter;
}

inline int lowbit(int x){
	return x & -x;
}

void update(int x, double val){
	while(x<=n){
		tree[x] += val;
		x += lowbit(x);
	}
}

double sum(int x){
	double res = 0;
	while(x>=1){
```

```
        res += tree[x];
        x -= lowbit(x);
    }
    return res;
}

void solve(){
    n = read();
    for(int i = 1, u, v ; i < n ; ++i){
        u = read(); v = read();
        addEdge(u,v); addEdge(v,u);
    }
    DFS(1,1,0);
    q = read();
    for(int i = 1, t, x, y ; i <= q ; ++i){
        t = read(); x = read(); y = read();
        if(t&1){
            double val = log(y) - valOfNode[x];
            update(seg[x].first,val);
            valOfNode[x] = log(y);
        }else{
            double a = sum(seg[x].second) - sum(seg[x].first-1);
            double b = sum(seg[y].second) - sum(seg[y].first-1);
            double ans = exp(a-b);
            if(ans>=1e9) printf("%.10lf\n",1e9);
            else printf("%.10lf\n",ans);
        }
    }
}

int main(){
    int t = 1;
    // t = read();
    while(t--) solve();
    return 0;
}
```

D. Make 10

贪心，从用棍数少的线开始合成

```
#include <cstdio>
#include <algorithm>
```

```
using namespace std;
typedef long long ll;

ll n1, n2, n3;
ll ans;

void solve(){
	ans = 0;
	scanf("%lld %lld %lld",&n1,&n2,&n3);
	// 3 3 4
	if(n2>=2&&n3>=1){
		ll w = min(n2>>1,n3);
		n2 -= w<<1;
		n3 -= w;
		ans += w;
	}
	// 2 4 4
	if(n1>=1&&n3>=2){
		ll w = min(n1,n3>>1);
		n1 -= w;
		n3 -= w<<1;
		ans += w;
	}
	// 2 2 3 3
	if(n1>=2&&n2>=2){
		ll w = min(n1>>1,n2>>1);
		n1 -= w<<1;
		n2 -= w<<1;
		ans += w;
	}
	// 2 2 2 4
	if(n1>=3&&n3>=1){
		ll w = min(n1/3,n3);
		n1 -= w*3;
		n3 -= w;
		ans += w;
	}
	// 2 2 2 2 2
	if(n1>=5){
		ll w = n1 / 5;
		n1 -= w*5;
		ans += w;
	}
```

```
	printf("%lld\n",ans);
}

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

E. Can you answer there queres I

我的锅没给数据范围..本题的时间复杂度要做到$O(n \log n + q)$

所以要一种离线查询的数据结构，叫做猫树

build 时求出最大子区间和包含端点的最大子区间

具体实现看代码及注释

```
#include <iostream>
#include <cstdio>
#define re register
#define MAXN 200005

using namespace std;

inline int read(){
	int x = 0, f = 1;
	char ch = getchar();
	while(ch<'0'||ch>'9'){
		if(ch=='-')
			f = -1;
		ch = getchar();
	}
	while(ch>='0'&&ch<='9'){
		x=(x<<1)+(x<<3)+(ch^48);
		ch = getchar();
	}
	return x*f;
}

// p[a][b], 最大子区间和, 其中 a 是区间所在的树层数,b 代表[b,mid]或[mid+1,b](取决小于等于 mid 还是大于)区间的子区间最大和
// s[a][b], 包含端点的最大子区间和, 同上
// 2^16 > 50000
```

```
int N, data[MAXN], p[17][MAXN], s[17][MAXN], pos[MAXN];
int M, lg2[MAXN<<2];

inline void build(int node, int l, int r, int d){
	if(l==r){
		pos[l] = node;
		return ;
	}
	// pre, 从 mid 向左开始的子段和
	// sum, 从 mid 向左开始包含端点的子段和
	int mid = (l+r)>>1, left_node = node<<1, right_node = node<<1|1, pre, sum;
	p[d][mid] = s[d][mid] = pre = sum = data[mid];
	// 子段和小于 0 就舍弃
	pre = max(pre,0);
	// mid-1 向左开始遍历
	for(re int i = mid - 1 ; i >= l ; --i){
		pre += data[i];
		sum += data[i];
		p[d][i] = max(p[d][i+1],pre);
		s[d][i] = max(s[d][i+1],sum);
		pre = max(pre,0);
	}
	// pre, 从 mid+1 向右开始的子段和
	// sum, 从 mid+1 向右开始包含端点的区间和
	p[d][mid+1] = s[d][mid+1] = pre = sum = data[mid+1];
	pre = max(pre,0);
	// mid+1 向右开始遍历
	for(re int i = mid + 2 ; i <= r ; ++i){
		pre += data[i];
		sum += data[i];
		p[d][i] = max(p[d][i-1],pre);
		s[d][i] = max(s[d][i-1],sum);
		pre = max(pre,0);
	}
	build(left_node,l,mid,d+1);
	build(right_node,mid+1,r,d+1);
}

inline int query(int l, int r){
	if(l==r) return data[l];
	// 它们的 LCA 所在的层数
	int d = lg2[pos[l]] - lg2[pos[l] ^ pos[r]];
	// mid 左边的最大子区间和/mid+1 右边的最大子区间和/包含端点的最大子区间和
相加
```

```
    return max(max(p[d][l],p[d][r]),s[d][l]+s[d][r]);
}

int main(){
    N = read();
    int len = 2;
    while(len<N) len<<=1;
    for(re int i = 2, l = len<<1 ; i <= l  ; ++i)
        lg2[i] = lg2[i>>1] + 1;
    for(re int i = 1 ; i <= N ; ++i)
        data[i] = read();
    // 保证区间长度为 1 的节点在同一层
    build(1,1,len,1);
    for(re int i = read() ; i ; --i){
        int l = read(), r = read();
        printf("%d\n",query(l,r));
    }
    return 0;
}
```

F. Diluc and Kaeya

令(x,y)为(D 出现的次数,K 出现的次数)，从原点出发做一条线，线穿过的点就是答案

```
#include <cstdio>
#include <map>
#include <algorithm>

using namespace std;
typedef pair<int,int> P;

map<P,int> ans;
int n;
P xy[500005];
char s[500005];

void solve(){
    ans.clear();
    scanf("%d",&n);
    getchar();
    for(int i = 1 ; i <= n ; ++i){
        s[i] = getchar();
```

```
            int *x = &xy[i].first, *y = &xy[i].second;
            *x = *(x-2) + (s[i]=='D');
            *y = *(y-2) + (s[i]=='K');

            ++ans[make_pair(*x,*y)];

            int g = __gcd(*x,*y);
            int gx = *x / g, gy = *y / g;
            if(gx==*x&&gy==*y){
                printf("%d%c",ans[make_pair(gx,gy)]," \n"[i==n]);
            }else{
                printf("%d%c",++ans[make_pair(gx,gy)]," \n"[i==n]);
            }

        }
}

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

G. Quadruple

令$A=a+b, B=c+d$,$(2<=A,B<=2*N)$，那么$K=A-B$，枚举A得到B

再求A,B能有多少种组合即可$(n-|x-n-1|)$

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
#include <queue>
#include <cmath>
#include <algorithm>
#define re register
#define pb push_back
#define cl clear
#define MAXN 100005

using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> P;
```

```
const int INF = 1e9;

inline int read(){
	int x = 0, f = 1;
	char ch = getchar();
	while(ch<'0'||ch>'9'){
		if(ch=='-')
			f = -1;
		ch = getchar();
	}
	while(ch>='0'&&ch<='9'){
		x=(x<<1)+(x<<3)+(ch^48);
		ch = getchar();
	}
	return x*f;
}

int n, k;
ll ans;

ll f(int x){
	return n-abs(x-n-1);
}

void solve(){
	n = read();
	k = read();
	for(re int A = 2 ; A <= (n<<1) ; ++A){
		int B = A - k;
		if(B>=2&&B<=(n<<1)){
			ans = ans + f(A)*f(B);
		}
	}
	printf("%lld\n",ans);
}

int main(){
	int t = 1;
	//t = read();
	while(t--) solve();
	return 0;
}
```

H. Non-Integer Donuts

题意：给初始的资金和后面 N 天增加的资金，问有几天的资产不是整数？

阅读题

*X*并没有什么用

一个建议就是在整数下操作而不是在小数

```
#include <cstdio>
#include <iostream>

using namespace std;

int n, ans, x, y;

int main(){
    scanf("%d\n",&n);
    scanf("$%d.%d\n",&x,&y);
    for(int i = 1, dx, dy ; i <= n ; ++i){
        scanf("$%d.%d\n",&dx,&dy);
        y += dy;
        y %= 100;
        ans += !!y;
    }
    printf("%d\n",ans);
    return 0;
}
```