A - 幻想门

先把所有的 A 变为 BB 然后再从前往后扫字符串，如果检查到两个连续的 B 就变成 A，这样我们能保证能变出来的 A 排在最前面，字典序最小。

具体实现的时候我们可以创建一个新的字符串 a，在从左往右扫描 s_i 的时候遇到 A 就加入两个 B，其他就正常加入扫描到的 s_i。

然后在输出的时候同样从左往右扫的时候遇到两个连续的 B 就输出为 A。

std：

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define ll long long
using namespace std;
const int N=4e5+10;
int n,m,a[N];
char s[N];
signed main()
{
    scanf("%d",&n);
    scanf("%s",s+1);
    for(int i=1;i<=n;i++)
        if(s[i]=='A')m+=2,a[m]=a[m-1]=1;
        else if(s[i]=='B')m++,a[m]=1;
        else m++,a[m]=0;
    int i=1;
    while(i<=m){
        if(a[i]&&a[i+1])putchar('A'),i+=2;
        else if(a[i])putchar('B'),i++;
        else putchar('C'),i++;
    }
    putchar('\n');
    return 0;
}
```

搬自：https://atcoder.jp/contests/arc136/tasks/arc136_a

B - 结论题

我们考虑 a_1，记 $S=\sum_{i=2}^{n}a_i\times i$，那么题目要求的就是 $S+a_1$ 是一个奇数，而 a_1 是 0 或者 1。

也就是如果 S 是奇数时只有 $a_1=0$ 合法，S 是偶数时只有 $a_1=1$ 合法。

所以无论 S 等于多少，只有一个 a_1 的取值是合法的。

所以 $a_i(i\geq 2)$ 是可以随意取的，而根据结果调整 a_1 的取值即可。

所以答案就是后面每个 a_i 取值范围的乘积也就是 $3\times 4\times\ldots\times n\times(n+1)$。

因为乘起来的时候会超出 int 的范围所以记得开 long long。

std：

```cpp
#include<cstdio>
#include<cstring>
#include<algorithm>
#define file(x)
#define ll long long
using namespace std;
const ll P=998244353;
ll n,ans;
signed main()
{
    scanf("%lld",&n);ans=1;
    for(ll i=3;i<=n+1;i++)
        ans=ans*i%P;
    printf("%lld\n",ans);
    return 0;
}
```

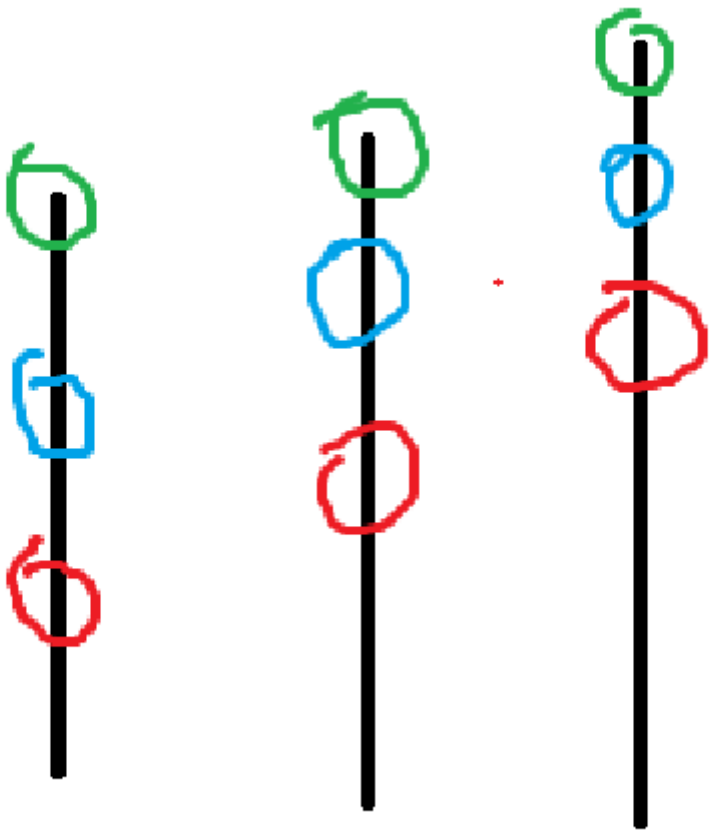
C - Color Your Night

我们考虑二分答案，我们对于区间答案 $[l,r]$ 取一个中点 mid 。

考虑答案能否超过 mid ，我们考虑能不能进行 mid 次装点就行了。

如果确定了要进行 mid 次，那么我们肯定是取红灯最低的 mid 盏灯和绿灯最高的 mid 盏灯。

先把这些灯取出来，我们最终把所有的装点结果排出来，我们最终肯定会弄成下图这种形式的最优的



也就是高的和高的一起装点，低的和低的一起装点。

所以我们相当于确定了红色的和绿色的用来装点的灯的高度了。

至于蓝色，我们可以定义一个 $z=1$，每次去判断 z 能不能塞进最左边没被装点的灯之间，如果能就塞进去，不能就让 $z=z+1$ 往下一个灯去找。

时间复杂度：$O(n\log n)$

std：

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N=1e5+10;
int n,a[N],b[N],c[N];
bool check(int x){
    int z=1;
    for(int i=1;i<=x;i++){
        int l=a[i],r=c[n-x+i];
        while(b[z]<=l)z++;
        if(b[z]>=r)return 0;
        z++;
    }
    return 1;
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);sort(a+1,a+1+n);
    for(int i=1;i<=n;i++)scanf("%d",&b[i]);sort(b+1,b+1+n);
    for(int i=1;i<=n;i++)scanf("%d",&c[i]);sort(c+1,c+1+n);
    int l=1,r=n;b[n+1]=1e9+7;
    while(l<=r){
          int mid=(l+r)>>1;
          if(check(mid))l=mid+1;
          else r=mid-1;
    }
    printf("%d\n",r);
}
```

搬自：https://atcoder.jp/contests/arc123/tasks/arc123_b

D - Melty Land Nightmare

首先我们要知道 Nim 游戏的结论：

> 有若干堆石子，每堆石子的数量都是有限的，合法的移动是“选择一堆石子并拿走若干颗（不能不拿）”，如果轮到某个人时所有的石子堆都已经被拿空了，则判负（因为他此刻没有任何合法的移动）。
>
> 结论：记每一堆石子的个数 a_i 的情况下，把所有 a_i 异或起来，如果为 0 则后手必胜，否则先手必胜

证明的话，我们可以看 OI Wiki 上的详细证明

证明

为什么异或值会和状态的胜负有关？下面给出了这个定理的证明过程。

为了证明该定理，只需要证明下面三个定理：

- 定理 1：没有后继状态的状态是必败状态。
- 定理 2：对于 $a_1 \oplus a_2 \oplus \ldots \oplus a_n \neq 0$ 的局面，一定存在某种移动使得 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$。
- 定理 3：对于 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$ 的局面，一定不存在某种移动使得 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$。

对于定理 1，没有后继状态的状态只有一个，即全 0 局面。此时 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$。

对于定理 2，不妨假设 $a_1 \oplus a_2 \oplus \ldots a_n = k \neq 0$。如果我们要将 a_i 改为 a_i'，则 $a_i' = a_i \oplus k$。

假设 k 的二进制最高位 1 为 d，即 $2^d \leq k < 2^{d+1}$。根据异或定义，一定有奇数个 a_i 的二进制第 d 位为 1。满足这个条件的 a_i 一定也满足 $a_i > a_i \oplus k$，因而这也是个合法的移动。

对于定理 3，如果我们要将 a_i 改为 a_i'，则根据异或运算律可以得出 $a_i = a_i'$，因而这不是个合法的移动。

在知道 Nim 游戏结论的前提下，我们来看这道题。先手和后手会先各自取空两个格子，后手显然不会取与先手同一行或者同一列的格子，否则先手直接取走该行/列剩下那个的全部就赢了。

所以先后手第一次操作后肯定存在两行和两列都被取空了一个格子，此时我们考虑如果存在某次操作后有人取空了这两行/列的另外一个格子，此时另一个人还是取走剩下那个就赢了。

我们记 $b_{i,j}$ 表示格子 (i,j) 最多能取走多少个石子，那么对于这两行和两列的格子上的石头，我们令 $b_{i,j} = a_{i,j} - 1$（也就是至少要留下一个不能取空）。

而对于剩下的一个格子，它取不取空都没有任何影响，也就是 $b_{i,j} = a_{i,j}$。

至于先手和后手的第一步怎么取，我们可以直接暴力枚举即可。

std

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define ll long long
using namespace std;
ll T,sum,ans,a[3][3];
int uni(int x,int y){
    int z=0;
    if(x==0||y==0)z++;
    else return z;
    if(x==1||y==1)z++;
    return z;
}
bool check(ll x,ll y){
    for(ll i=0;i<3;i++)
        for(ll j=0;j<3;j++){
            int c=uni(i,x),z=uni(j,y);
```

```
            if(((a[x][y]^a[i][j])==(sum^a[c][z]^(a[c][z]+1)))&&(i!=x)&&(j!=y))
                return 0;
        }
    return 1;
}
int main()
{
    scanf("%lld",&T);
    while(T--){
        sum=ans=0;
        for(ll i=0;i<3;i++)
            for(ll j=0;j<3;j++)
                scanf("%lld",&a[i][j]),a[i][j]--,sum=sum^a[i][j];
        for(ll i=0;i<3;i++)
            for(ll j=0;j<3;j++)
                ans+=check(i,j);
        printf("%lld\n",ans);
    }
    return 0;
}
```

E - 我是雨

我们先二分一个答案 mid，然后变成考虑把小于 mid 的视为黑色，大于 mid 的视为白色，那它的下方有两个或两个以上的颜色就是这个格子的颜色。

而在扩散一次之后最两边的我们可以直接不考虑了，因为它不再会影响到最终答案。

直接考虑最初的颜色序列，找到一个离中间最近的相邻的相同颜色，这个颜色就是答案，因为这两个数顶上连续到中间的都是这个颜色，因为只有另一个连续相同的颜色才能隔开这些颜色，但是如果找到了这两个，那么就有更优的答案案了，所以结论成立。

std：

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N=2e5+10;
int n,a[N];
int check(int x){
    for(int i=1;i<=n;i++){
        if(a[n-i+1]<x&&a[n-i]<x||a[n+i-1]<x&&a[n+i]<x)return 0;
        if(a[n-i+1]>=x&&a[n-i]>=x||a[n+i-1]>=x&&a[n+i]>=x)return 1;
    }
    return a[1]>=x;
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=2*n-1;i++)
        scanf("%d",&a[i]);
```

```
    int l=1,r=1e9;
    while(l<=r){
        int mid=(l+r)>>1;
        if(check(mid))l=mid+1;
        else r=mid-1;
    }
    printf("%d",r);
}
```