

胜利队形



请同学简述题意 突出核心要点

已知每个英雄的性别和身高
求排列：

男英雄全部排在左边，从高到低
女英雄全部排在右边，从低到高

男的身高放一个数组man[]

女的身高放另一个数组woman[]

对man[]数组从小到大排序

反向输出man[]数组
完成从高到低

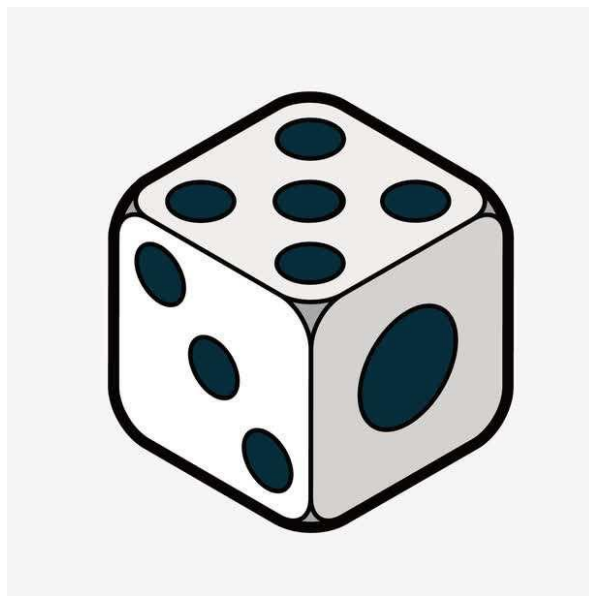
对woman[]数组从小到大排序

反向输出woman[]数组
完成从低到高

```
9  cin>>n;
10 for(int i=0;i<n;i++)cin>>gender[i];
11 int m=0,w=0;
12 for(int i=0;i<n;i++)
13     if(gender[i]=="man")
14         cin>>man[m++];
15     else
16         cin>>woman[w++];
17 sort(man,man+m);
18 sort(woman,woman+w);
19 for( ) cout<<man[i]<<" ";
20 for(int i=0;i<=w-1;i++)cout<<woman[i]<<" ";
```

m和w代表什么

非传递的骰子



请同学写出题目大意
已知什么求什么

定义X击败Y:X比Y更可能赢
给出两个四面骰子a,b的各面数字
求是否存在某个骰子c
使得它们彼此击败一个且被一个击败

若a击败b,那么:b击败c且c击败a
若b击败a,那么:a击败c且c击败b

请同学阅读[数据规模和约定]
识别部分得分点

【数据规模与约定】

所有4面骰子面上的数字必须是 1 到 10 的整数。
每个测试用例包含 T ($1 \leq T \leq 10$) 个独立子测试用例。

输入样例

1

4 5 6 7 2 4 5 10

输出样例

yes

A: 4 5 6 7

B: 2 4 5 10

C: 1 4 8 9

输入样例

1

4 5 6 7 2 4 5 10

输出样例

yes

A: 4 5 6 7

B: 2 4 5 10

C: 1 4 8 9

在16种情况中,A>B共9次^{超过一半}
(4>2,5>2,5>4,6>2,6>4,6>5,7>2,7>4,7>5)

B>C共8次^{因还有平局,显然超过B<C的数量}
(2>1,4>1,5>1,5>4,10>1,10>4,10>8,10>9)

C>A共8次^{也存在平局,因此超过A>C的数量}
(8>4,8>5,8>6,8>7,9>4,9>5,9>6,9>7)

形成A>B,B>C,C>A的关系,输出yes

模拟

`win()`函数判断两个骰子谁更有赢面

若a,b赢面相同,输出no
若b比a赢面大,交换a和b
至此已保证a比b赢面大
只要枚举到某个c
使得b赢面大过c且c赢面大过a,即yes

枚举c:
四重for?
DFS?


```
28 cin>>T;
29 for(int i=1;i<=T;i++){
30     for(int i=1;i<=LEN;i++) cin>>a[i];
31     for(int i=1;i<=LEN;i++) cin>>b[i];
32     if(!win(a,b)&&!win(b,a)){
33         cout<<"no"<<endl;
34         continue;
35     }
36     if(!win(a,b))
37         for(int i=1;i<=LEN;i++)
38             swap(a[i],b[i]);
39     cout<<(dfs(1,1)?"yes":"no")<<endl;
40 }
```

若x比y赢面大,win函数返回true

```
7 bool win(int *x,int *y){  
8     int cnt=0;  
9     for(int i=1;i<=LEN;i++){  
10         for(int j=1;j<=LEN;j++){  
11             if(x[i]>y[j]) cnt++;  
12               
13         }  
14     return   
15 }
```

dfs:枚举骰子c的每一面

```
16 bool dfs(int pos,int val){
17     if(pos>LEN){
18         if( ) return 1;
19         return 0;
20     }
21     for(int i= ;i<=R;i++){
22         c[pos]=i;
23         ;
24     }
25     return 0;
26 }
```



连环爆



请同学写出题目大意
已知什么求什么

给定 n 个数字 x_i ， r 的半径依次递增，
求从某个数字出发最多可以访问多少个数字

请同学阅读[数据规模和约定]
识别部分得分点

【数据规模与约定】

$0 \leq x_i \leq 10^9$

对于20%的数据， $n \leq 20$

对于40%的数据， $n \leq 50$

对于100%的数据， $n \leq 100$

算法：枚举+模拟

枚
举

题目描述的情况不确定性的根源是什么？

首先引爆哪一颗炸弹

枚举引爆的第一颗炸弹位置

模
拟

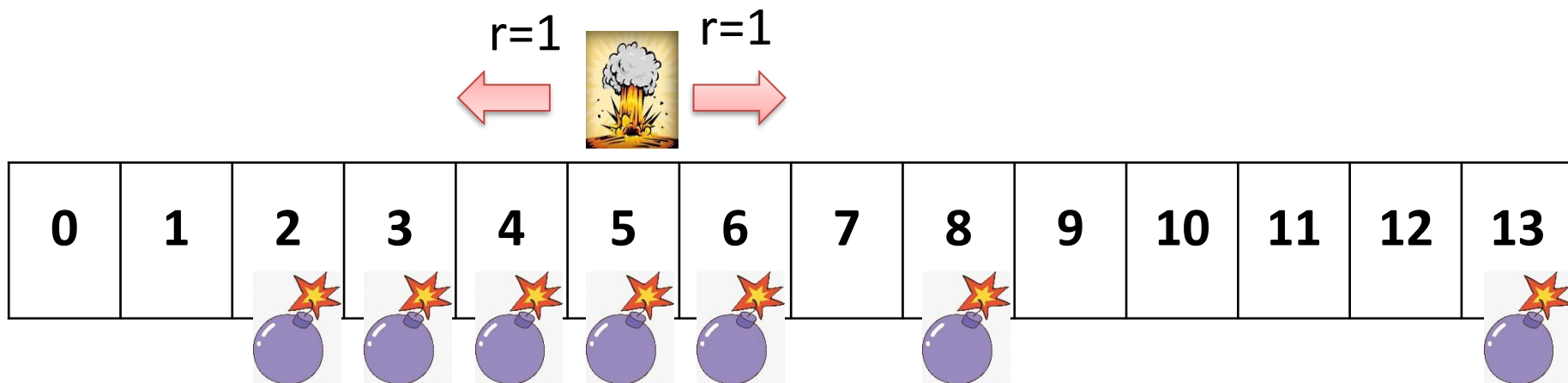
模拟该炸弹引发的连锁爆炸

如何记录当前炸弹半径
如何追踪连锁爆炸的炸弹

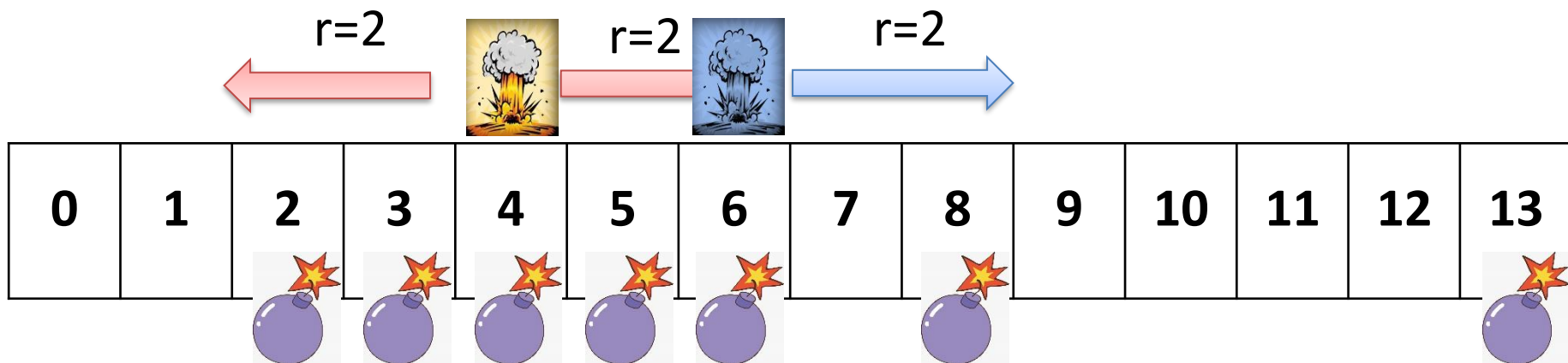
使用变量 r ，递增
分别模拟炸弹两侧的
爆炸情况

太戈编程
www.etiger.vip

算法：枚举+模拟

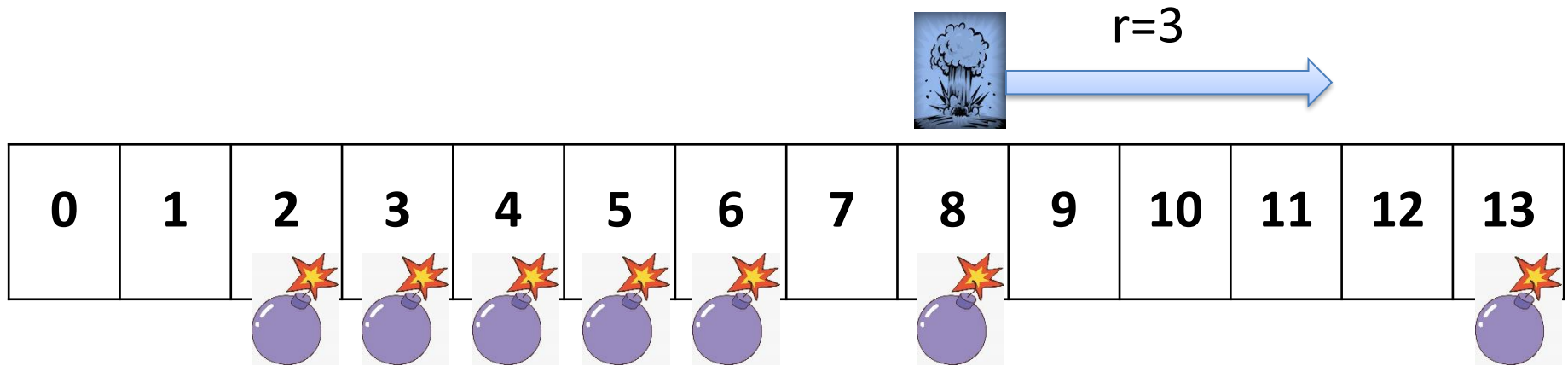


算法：枚举+模拟




炸弹范围每次递增1，只需要模拟一侧即可

算法：枚举+模拟



代码

枚举第一个引爆炸弹位置

```
29    
30  
31  
32  
33  
34  
for(int i=0;i<n;i++) {  
    ll left=reach(i,true);  
    ll right=reach(i,false);  
    ll len=right-left+1;  
    if(len>ans) ans=len;  
}
```

reach函数分别计算向两侧可以最多引爆多少颗炸弹
true表示向左侧引爆
false表示向右侧引爆

代码

```
6  ll reach(int start, bool goleft) {
7      ll last=start;
8      ll r=1;
9      ll dir=0;
10 while(last>0 && last<n-1) {
11     if(goleft) dir=-1;
12     else dir=1;
13     ll next=last;
14     while(next+dir>=0 && next+dir<n && abs(x[next+dir]-x[last])<=r)
15         next+=dir;
16     if(next==last) break;
17     last=next;
18     r++;
19 }
20 return last;
21 }
```

last 起爆的炸弹

r当前炸弹范围

dir 爆炸方向上下标的变化

下一个炸弹的位置next+dir

更新下一轮爆炸

拍照片



请同学写出题目大意
已知什么求什么

给出偶数个由‘G’和‘H’组成的序列
通过对偶数长的前缀序列进行反转
希望使尽可能多的‘G’处于偶数位
求最少反转次数

请同学阅读[数据规模和约定]
识别部分得分点

【数据规模与约定】

$2 \leq N \leq 2 \cdot 10^5$ ，N为偶数。

测试点 2-6 满足 $N \leq 1000$ 。

测试点 7-11 没有额外限制。

输入样例

8

HHHHGHGH

输出样例

1

将前八个点反转

→HGHGHHHH

输入样例

6

GHHGGH

输出样例

3

将前2个点反转

→HGHGH

再将前4个点反转

→GHGHGH

再将前6个点反转

→HGHGHG

要点分析

每次反转偶数长的前缀序列
会将该前缀中所有位置奇偶性对调

序列中两位一组
有HH, GG, HG, GH四种情况
HH, GG不用主动去翻转
若被牵连反转对结果也无影响
可以当成空气忽略

序列可简化成HG, GH排列
HG无需反转, 设为0
GH需反转, 设为1

构造成01串
串中纯0/纯1前缀部分
反转后可变纯1/纯0状态

分析样例

HGGH: [01]

若直接前4位反转,仍是01不变
应先反转前2位,使01变成11
再反转前4位,使11变成00

HGHGHGHGGH: [00001]

同理,可将所有0看成整体
先反转前8位,使00001变成11111
再反转前10位,使11111变成00000

分析样例

HG**GH**HG**HGHGHGH**HG: [0**1**00**111**0]

先将前2位反转: **0**1001110 -> **1**1001110

再将前4位反转: **11**001110 -> **00**001110

再将前8位反转: **0000**1110 -> **1111**1110

再将前14位反转: **1111111**0 -> **00000000**

步骤总结

每次将纯**0**的前缀反转成纯**1**
跟后继纯**1**子段合并成新的纯**1**前缀
再将纯**1**前缀反转成更长的纯**0**
直到整个序列变成全**0**

统计**0**段+**1**段总数(若末尾是**0**则忽略)

```
6 cin>>n>>s;
```

```
7 s=" "+s;
```

```
8 int type=-1;
```

← type记录上一个01小组是0还是1

```
9 for(int i=1;i<=n;i+=2){
```

```
10     if(s[i]=='G'&& s[i+1]=='G' ||
```

```
11     s[i]=='H'&& s[i+1]=='H')
```

← 若是GG或HH,则无视

```
12         continue;
```

```
13     if(s[i]=='G'&& s[i+1]=='H'){
```

```
14         if(type!=1) type=1,cnt++;
```

← 0段转1段

```
15     }
```

```
16     else if(s[i]=='H'&& s[i+1]=='G'){
```

← 1段转0段

```
17     }
```

```
18 }
```

```
19  
20  
21 cout<<cnt;
```

← 若末尾是0段,扣除