C++

day4 SH-B260104 集训-讲解

T1 费用平摊

题目大意

已知长度为n的整数序列p[]，求有多少个子段和恰为m的倍数？

请同学阅读[数据规模和约定]
识别部分得分点

对于10%数据, m=1。

对于20%数据, m<=2。

对于30%数据, n<=100, 0<=p[i]<=100

对于50%数据, n<=1000。

对于100%数据, n<=200000, m<=1000000, 保证p[i]绝对值均不超过10000。

手算

输入	输出？
5 1 2 3 5 1 5	15
6 2 2 4 1 8 3 9	9

暴力

```
10 void solveBF(){
11     ll ans=0;
12     for(ll i=1;i<=n;++i){
13         ll sum=0;
14         for(ll j=i;j<=n;++j){
15             sum=(sum+p[j])%m;
16 
17         }
18     }
19     cout<<ans<<endl;
20 }
```

请写出时间复杂度
$O(n^2)$

思考如何加速

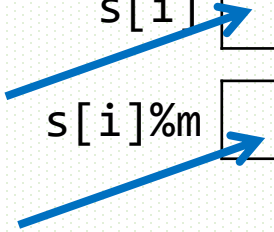
连续和/子段和 → 前缀和做差

子段和为m倍数 → 2个前缀和
模m同余

n=5
m=2

s[]是p[]的前缀和数组

	i=0	i=1	i=2	i=3	i=4	i=5
p[i]	-	2	4	1	8	3
s[i]	0	2	6	7	15	18
s[i]%m	0	0	0	1	1	0



对s[i]按照除以m的余数进行统计

cnt[x]表示有几个s[i]除以m的余数 恰为x

cnt[0]为4, cnt[1]为2

答案 = 4*(4-1)/2 + 2*(2-1)/2 = 7

前缀和+同余系计数

```
24 void solve(){
25     for(ll i=1;i<=n;++i){
26         s[i]=s[i-1]+p[i];
27         
28         cnt[s[i]]++;
29     }
30     
31     ll ans=0;
32     for(ll r=0;r<m;++r){
33         if(cnt[r]<=1)continue;
34         ans+=
35     }
36     cout<<ans<<endl;
37 }
```

可能有负数参与取模

s[0]为0参与计数

能否删除第33行?

请写出时间复杂度
O(n+m)

前缀和+同余系计数

对于10%数据, m=1。
对于20%数据, m<=2。
对于30%数据, n<=100, 0<=p[i]<=100
对于50%数据, n<=1000。
对于100%数据, n<=200000, m<=1000000, 保证p[i]绝对值均不超过10000。

```
const ll N=200009;
const ll M=1000009;

ll s[N];
ll cnt[M];
```

请设置N和M的大小

大小数据分离
确保部分分

```
38 int main(){
39     freopen("share.in","r",stdin);
40     freopen("share.out","w",stdout);
41     input();
42     if(n<=2000)
43         solveBF();
44     else
45         solve();
46     return 0;
47 }
```

T2 学术圈2

请同学写出题目大意
已知什么求什么

给出k个长度为n的人名序列
每个序列中人名按贡献量降序排列
贡献相同则按字典序排
更资深者不会比资历浅者贡献更多
求n人的资历关系矩阵

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】

1≤N,K≤100

输入样例
1 3 dean elsie mildred elsie mildred dean

输出样例
B11 0B? 0?B

1、单独一份论文 elsie-mildred-dean 并不能提供足够信息判断 Elsie 比 Mildred 资历更深或更浅，因为有可能两人贡献相等，只是按字典序排了先后。

2、elsie mildred dean

我们可以推断出 Dean 一定比这两名研究员资历更深，因为如果Dean跟前两人贡献相等，按字典序他不应该排在最后一位

3资历排序为 Elsie<Mildred<Dean 和 Mildred<Elsie<Dean 均是可能的

算法思路

扫描每个“贡献不增”序列

若发现i号比i-1号字典序小

i只可能比i-1贡献严格小、更资深

更新pos：

记录最后一个明确比前序资深的位置

此时i号明确比1到pos号资深：

通过遍历，标记矩阵

准备工作

```
3   int n,k,a[109][109];
4   map<string,int> name;
5   string s,x[109];
```

输入,初始化

```
7       cin>>k>>n;
8       for(int i=1;i<=n;i++)
9           for(int j=1;j<=n;j++)
10              a[i][j]=2;
11      for(int i=1;i<=n;i++)
12          cin>>s,name[s]=i;
```

```
13      while(k--){
14          int pos=1;
15          for(int i=1;i<=n;i++){
16              cin>>x[i];
17              if(x[i]<x[i-1]) pos=i;
18              for(int j=1;j<pos;j++)
19                  a[name[x[i]]][name[x[j]]]=1,
20                  a[name[x[j]]][name[x[i]]]=0;
21          }
22      }
```

若i号比i-1号字典序小
说明i只可能比i-1贡献小,更资深
此时更新pos:记录最后一个明确
比前序资深的位置

贡献序列是不增的
那么i相比pos及前序点
也一定贡献小,更资深
通过map更新资历矩阵

输出

```
23      for(int i=1;i<=n;i++){
24          for(int j=1;j<=n;j++){
25              if(i==j)
26                  printf("B");
27              else if(a[i][j]!=2)
28                  printf("%d",a[i][j]);
29              else printf("?");
30          }
31          puts("");
32      }
```

T3 文武双缺

请同学写出题目大意
已知什么求什么

n个元素的集合，每个元素两个属性(a[i],b[i])

划分为两个部分A和B,

{A中a[i]最大值+B中b[i]最大值} 最小化。

请同学设计部分分策略

测试点编号	特殊性质
1	所有x[i]等于100，所有y[i]等于200
2~5	n<=20
6~10	n<=100000

问题分析

x[i]（唱歌难听程度） y[i]（长跑缓慢程度）

n 个学生分配到合唱团或长跑队，合唱团的难听程度最高分记为 X，长跑队的缓慢程度最高分记为Y

目标： X+Y最小化

假设分组后，合唱团最大值为 A 长跑队最大值为B

对于人员的分配要满足什么条件?

所有大于**A**的学生必须在长跑队　　所有大于**B**的学生必须在合唱团

如果有一个学生同时满足 大于A 且 大于B？怎么办?　**无法分配**

问题总结：选择两个非负整数，使得不存在学生满足 xi>A且yi>B，并最小化A+B。

算法分析

集合的最优二划分(partition)

每个元素有两个属性
数据可视化

A	二维平面点
B	鱼骨图

剔除无效信息

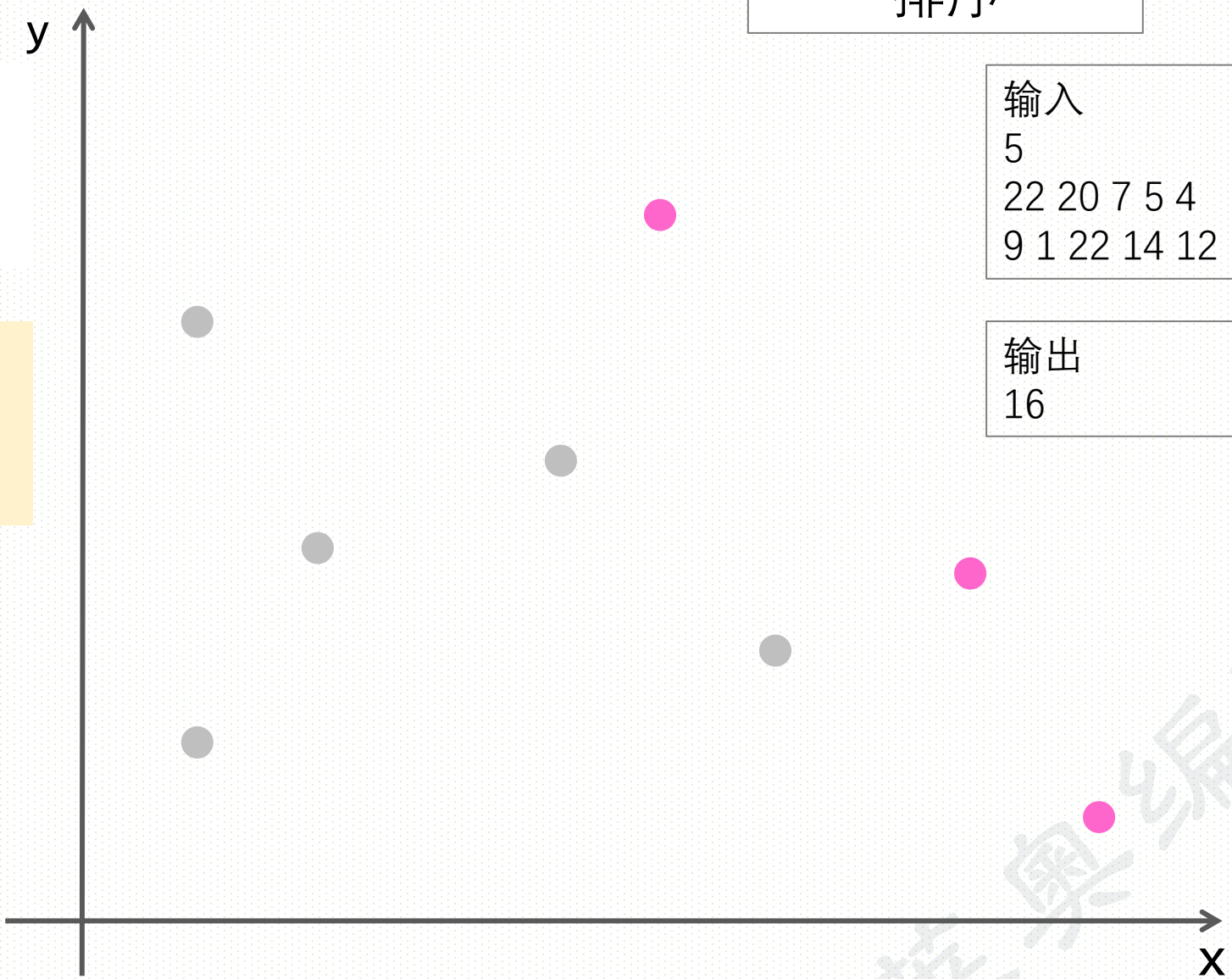
排序

什么是无效信息?

分配到哪一方，都不影响最终的结果。

有效的数据要严格满足:

排序后x 递增，y 严格递减

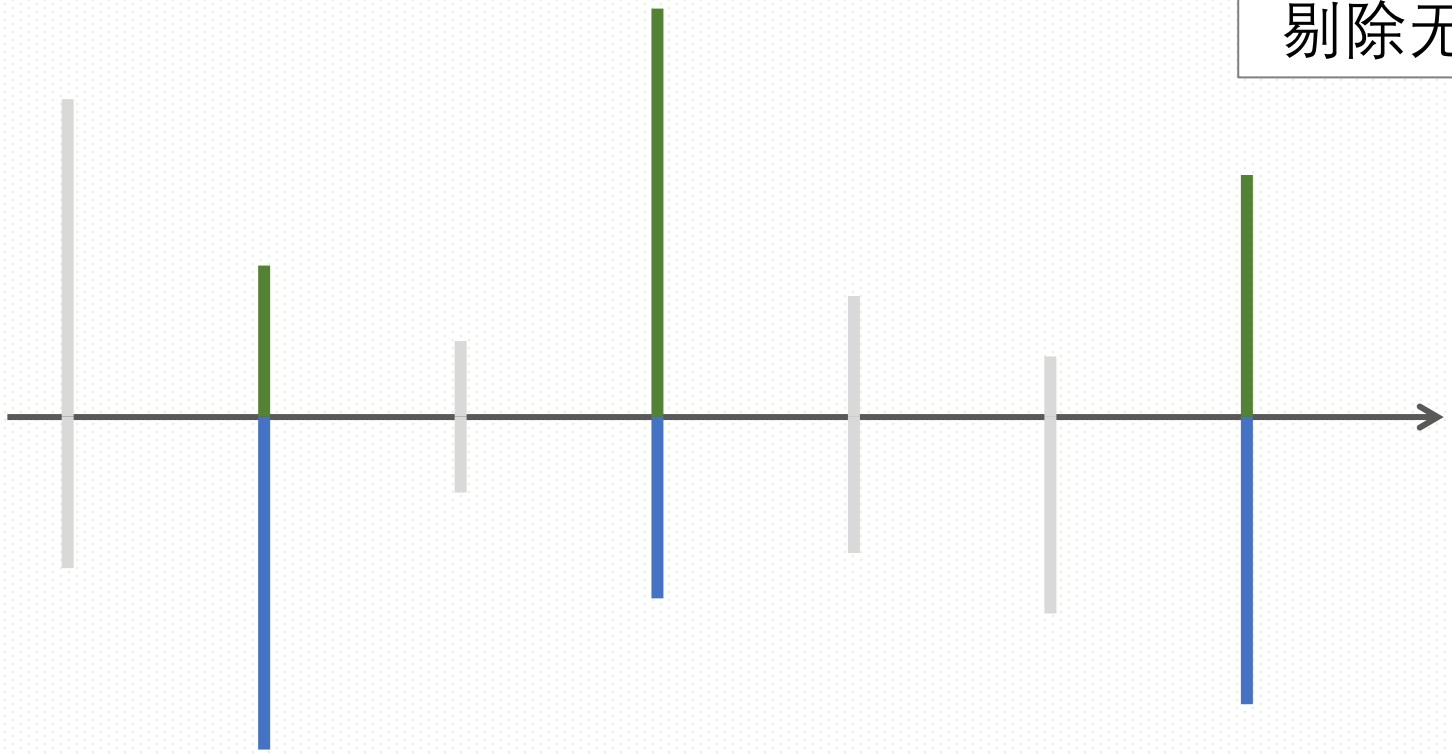


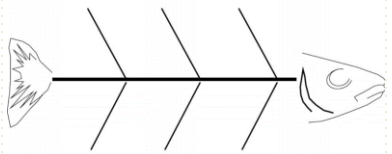
输入
5
22 20 7 5 4
9 1 22 14 12

输出
16

优点：可以表达序列编号
缺点：要求属性数值非负
缺点：较难发现双关键字排序

剔除无效信息

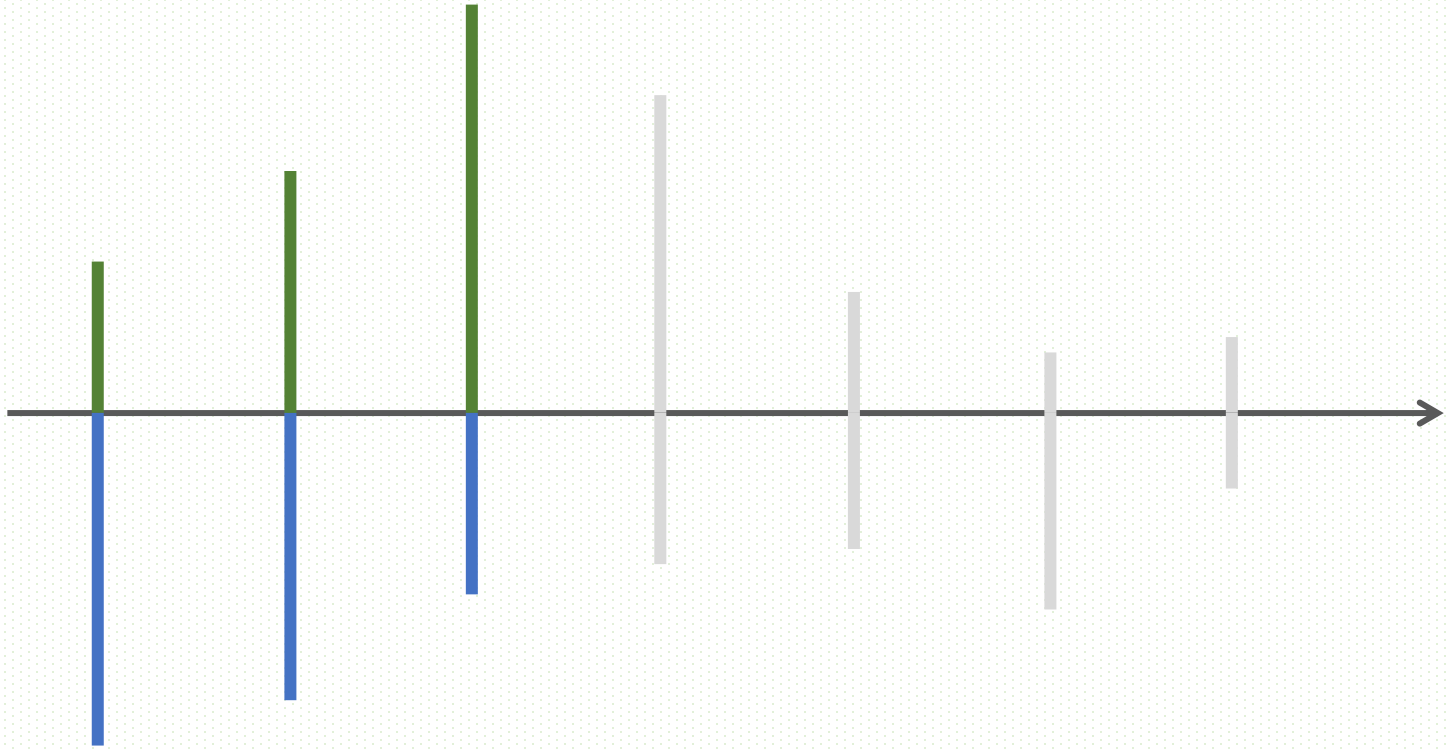




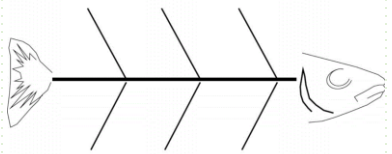
剔除无效信息
排序

有效的数据要严格满足:

排序后x 递增，y 严格递减



思考：如何处理筛选需要的数据?



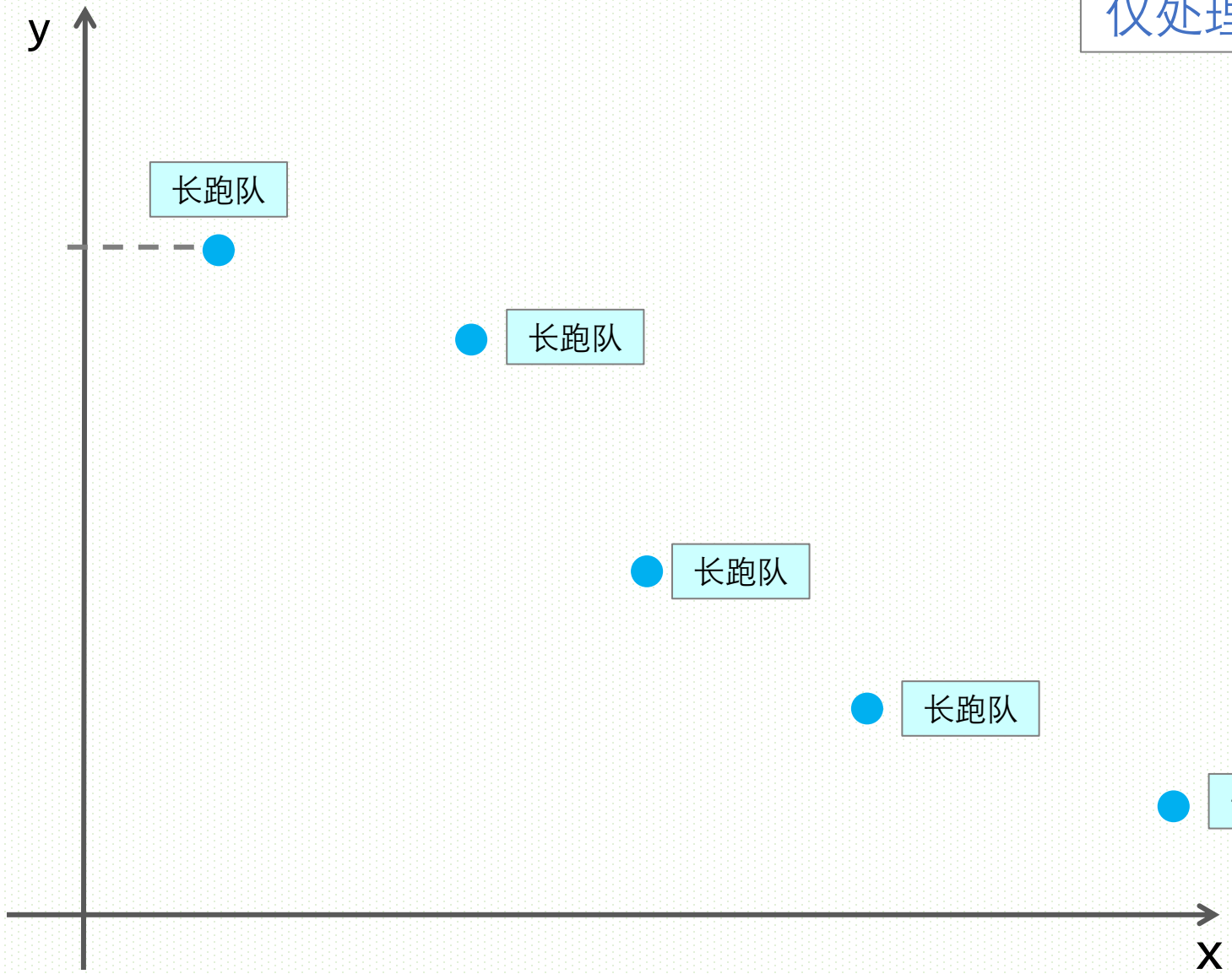
文武双缺模型

	分成两组
原始决策	每组人选是谁

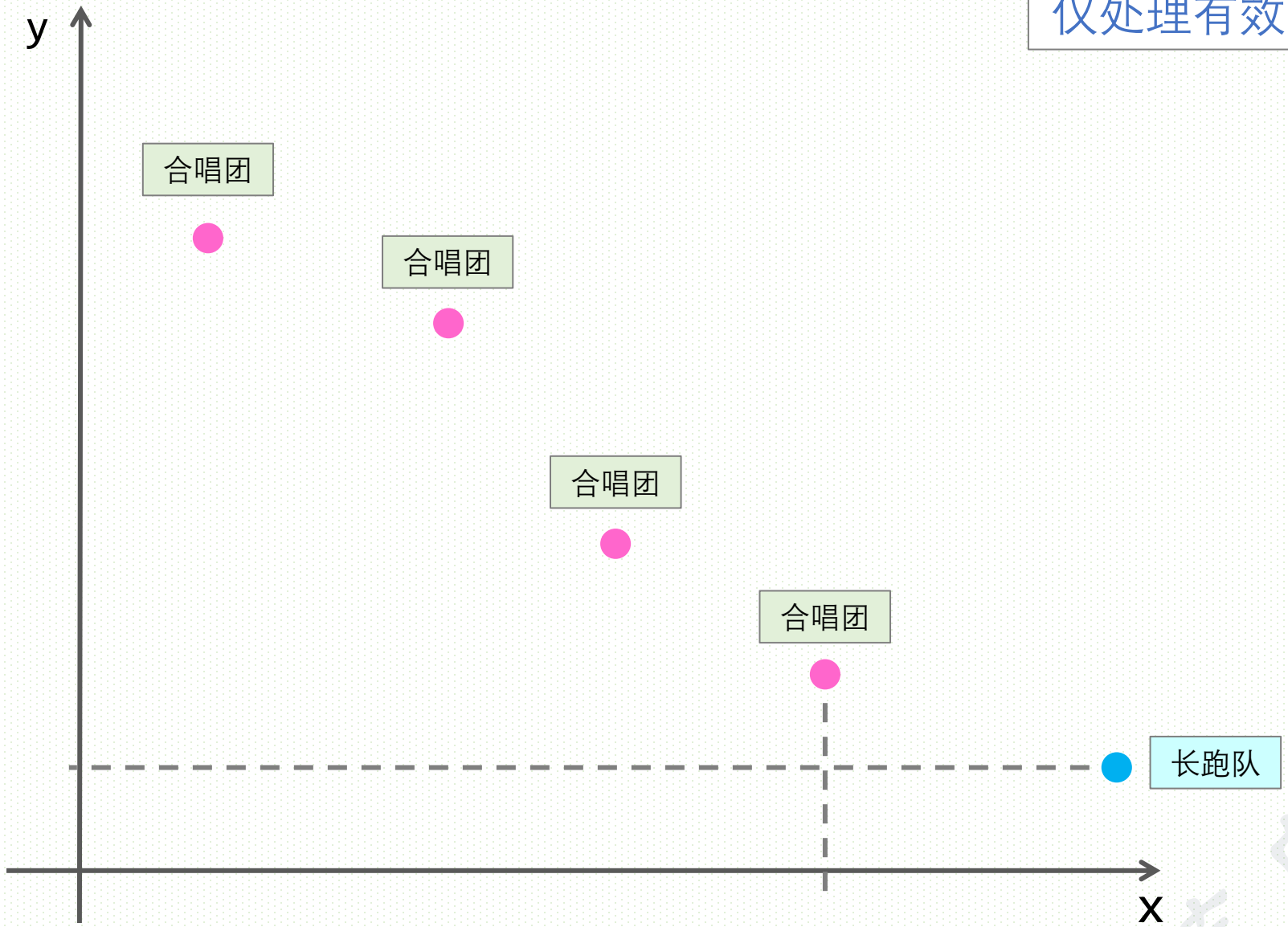
	最优解形态的简化
简化决策	剔除无效元素 有效元素排序后 枚举分割点前后分两组

枚举分割点

仅处理有效数据

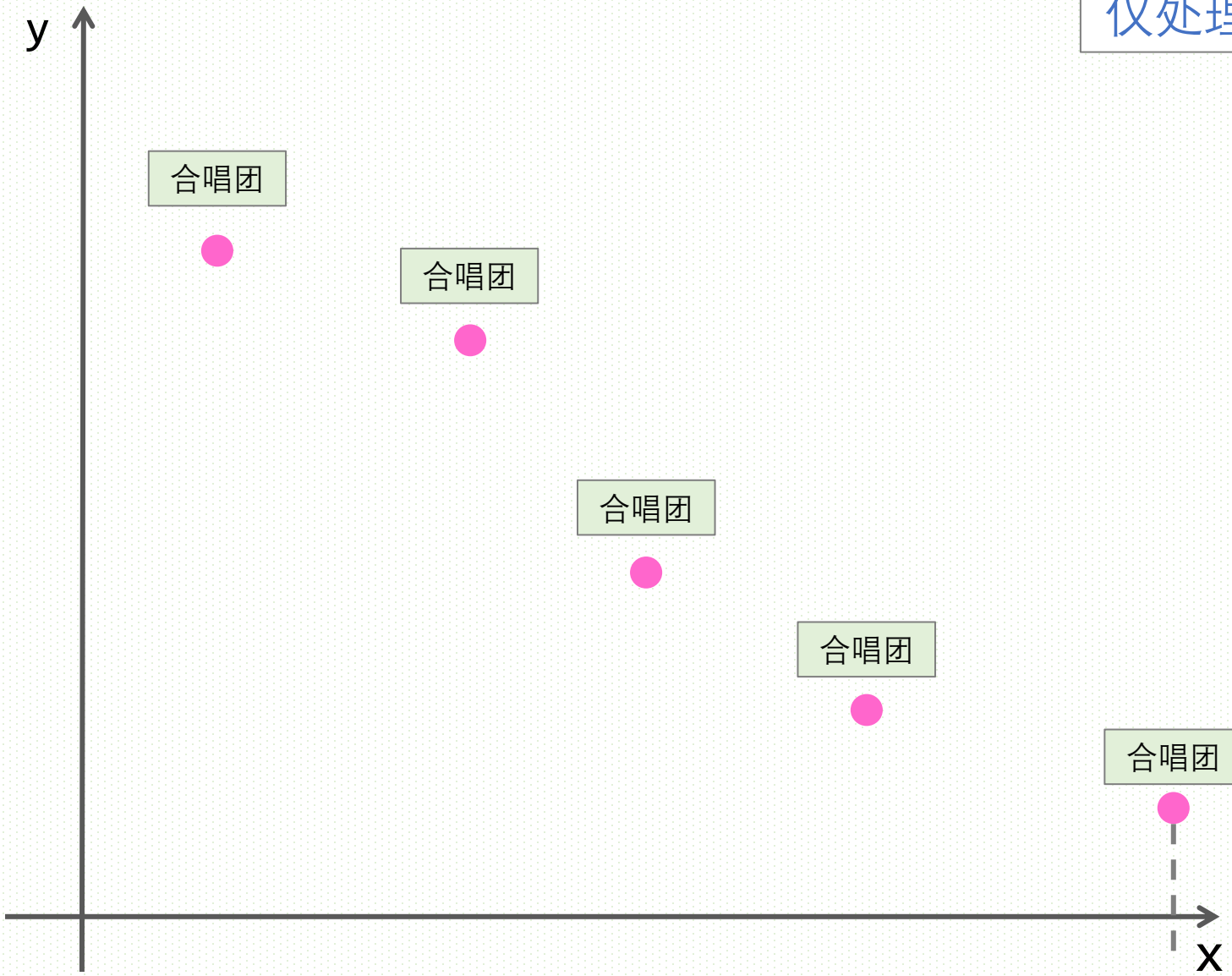


枚举分割点
仅处理有效数据



枚举分割点

仅处理有效数据



文武双缺模型

	最优解形态的简化
简化决策	剔除无效元素 有效元素排序后 枚举分割点前后分两组

n人共有几种可能的分组方案?

n+1

```
4  struct Point{int x,y;} p[N];
5  bool cmp(const Point&a,const Point&b){
6      return a.x<b.x||a.x==b.x&&a.y<b.y;
7  }
```

x升序，x相同时按y升序排序

```
26      cin>>n;
27      for(int i=1;i<=n;++i)cin>>p[i].x;
28      for(int i=1;i<=n;++i)cin>>p[i].y;
29      divide();
```

```
 9 void divide(){
10     sort(p+1,p+1+n,cmp);
11     int j=1;
12     for(int i=2;i<=n;++i){
13         
14         
15         
16     }
17     int m=j;
18     int ans=
19     for(int i=1;i<=m-1;++i)
20         ans=min(ans,
21     cout<<ans<<endl;
22 }
```

此顺序中，x是严格递增的

单调栈

存储有效数据

后续处理要保证对应的y严格递减

初始化为2种情况的最小值

全进入长跑队或全进入合唱队

T4 涂鸦王子

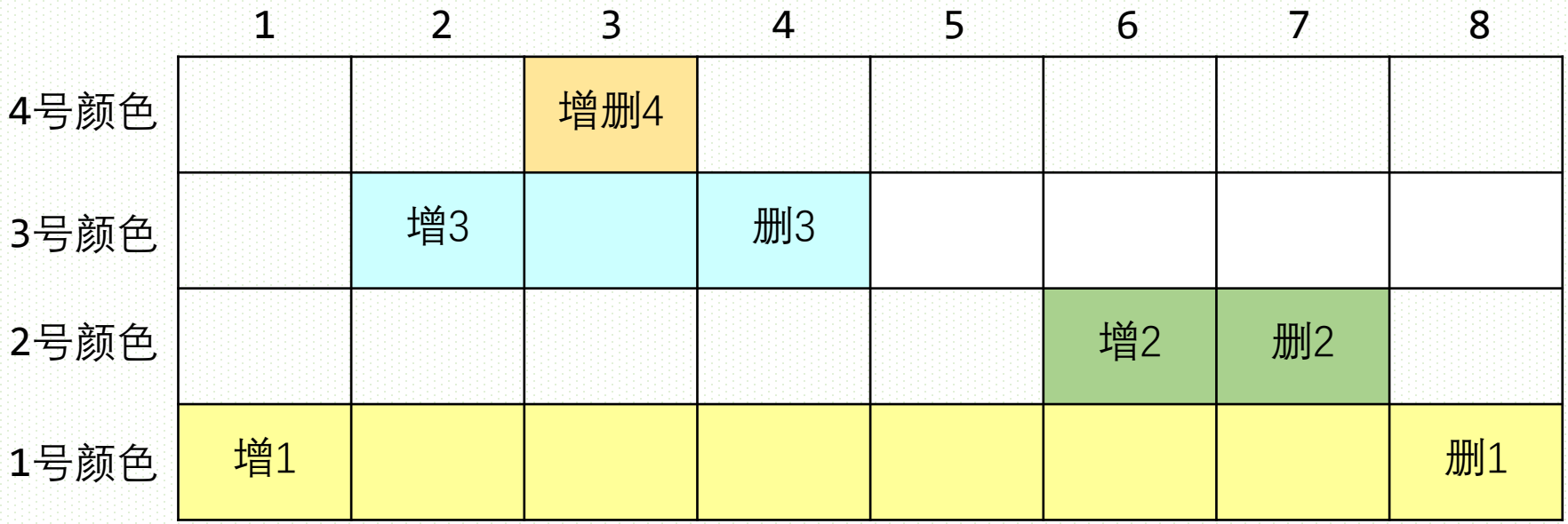
暴力

```
10 void solveBF(){
11     for(ll clr=1;clr<=m;++clr){
12         ll l=(clr*p+q)%n+1;
13         ll r=(clr*q+p)%n+1;
14         if(l>r) [          ];
15         for(ll id=l;id<=r;++id)
16             ans[id]=clr;
17     }
18     for(ll i=1;i<=n;++i)
19         cout<<ans[i]<<endl;
20 }
```

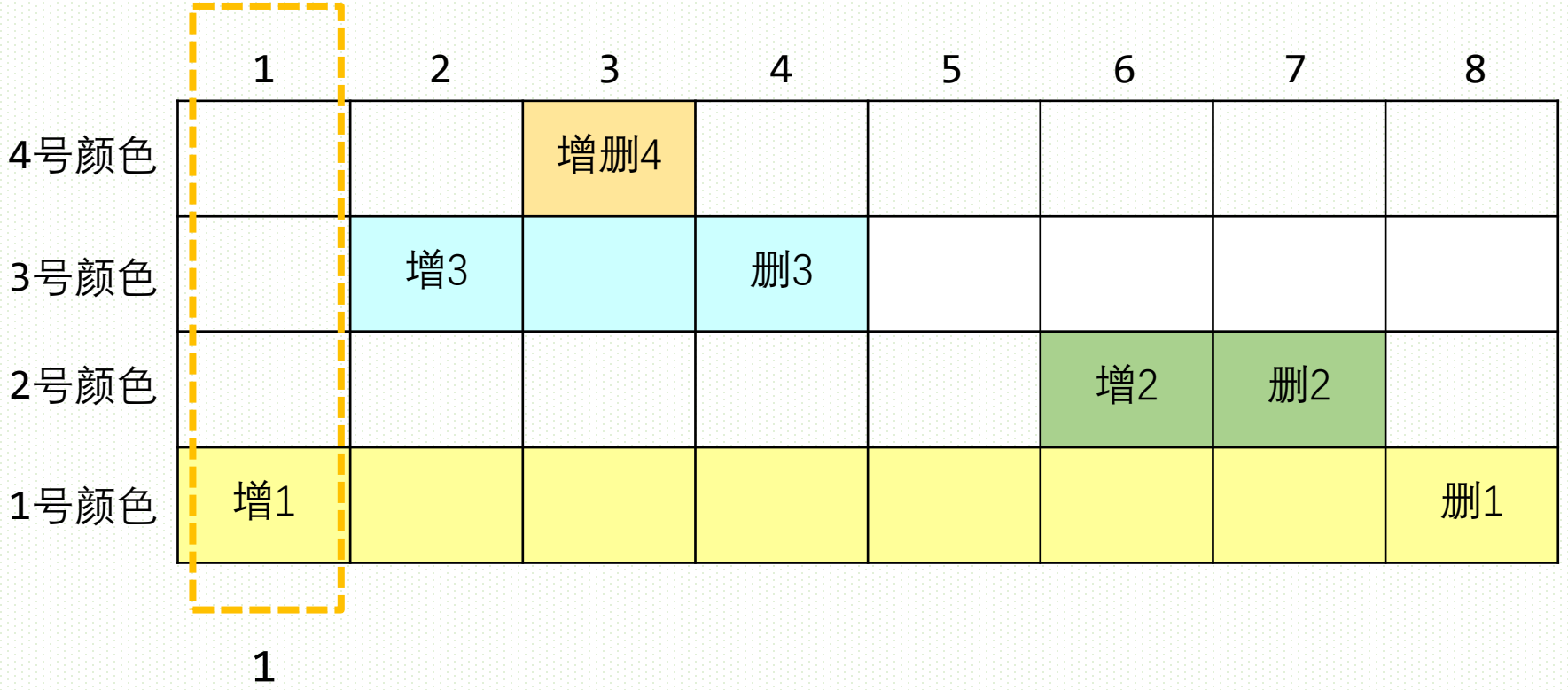
更快解法	扫描算法

从左到右依次对序列元素进行分析

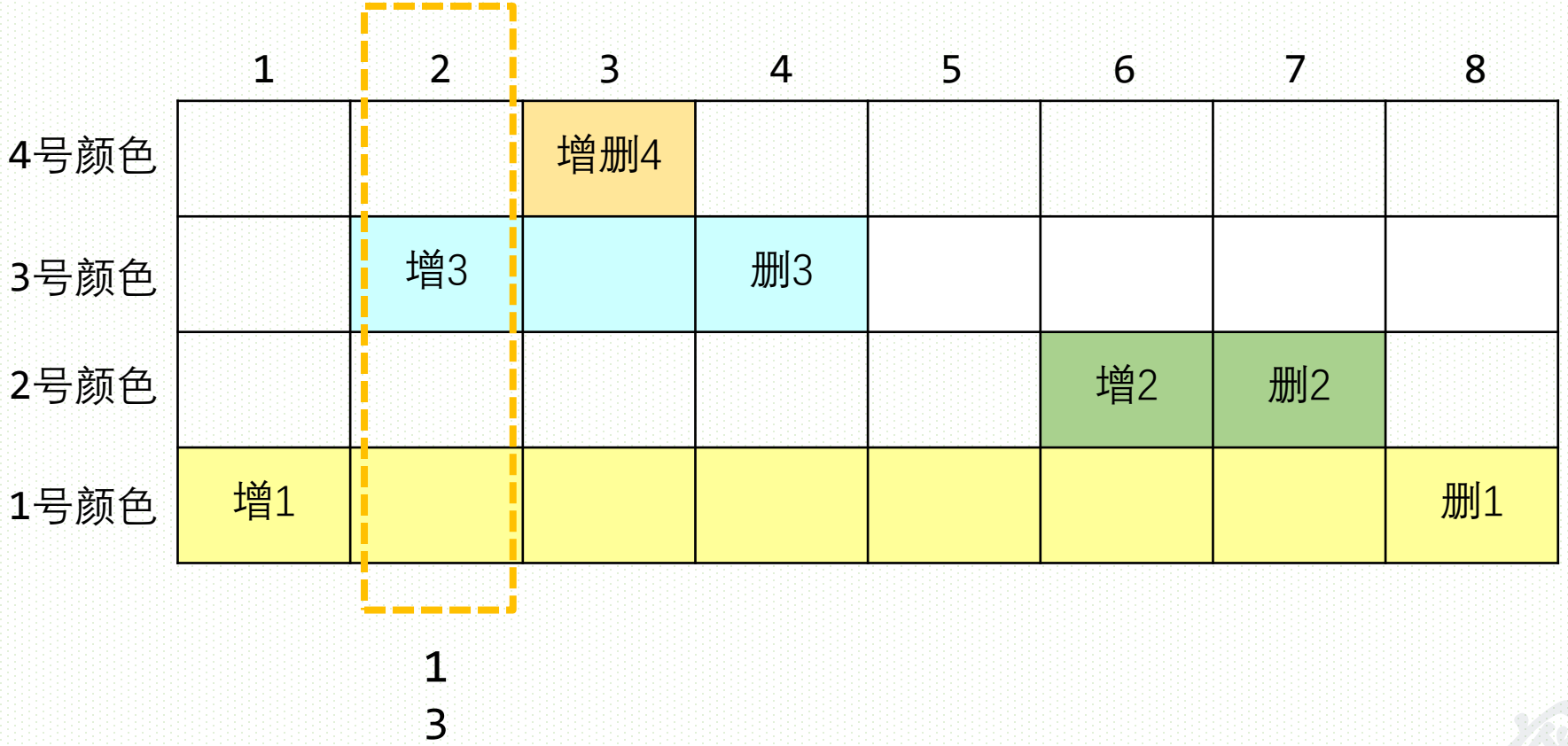
8格涂色4次
[1,8]涂1号颜色
[6,7]涂2号颜色
[2,4]涂3号颜色
[3,3]涂4号颜色



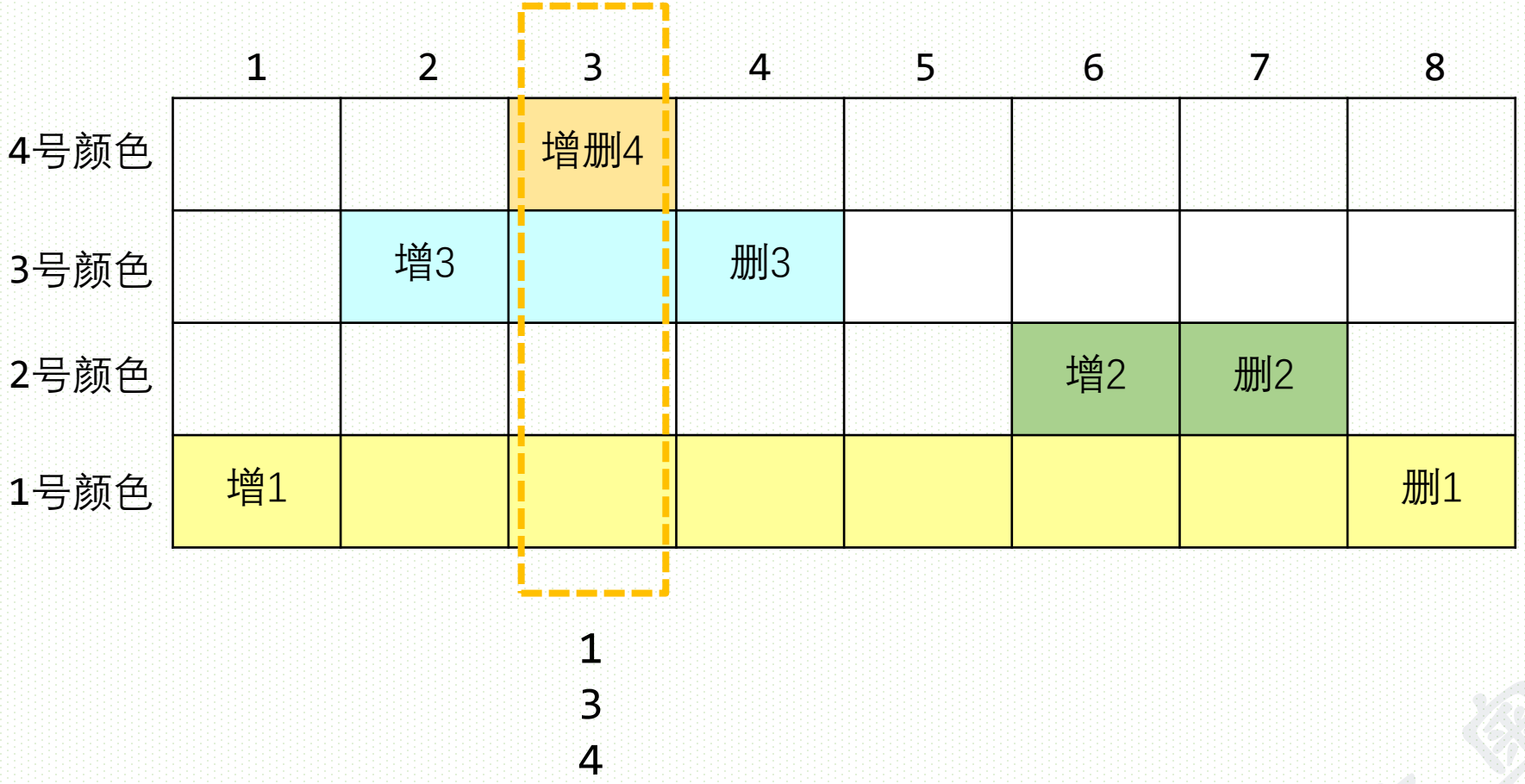
8格涂色4次
[1,8]涂1号颜色
[6,7]涂2号颜色
[2,4]涂3号颜色
[3,3]涂4号颜色



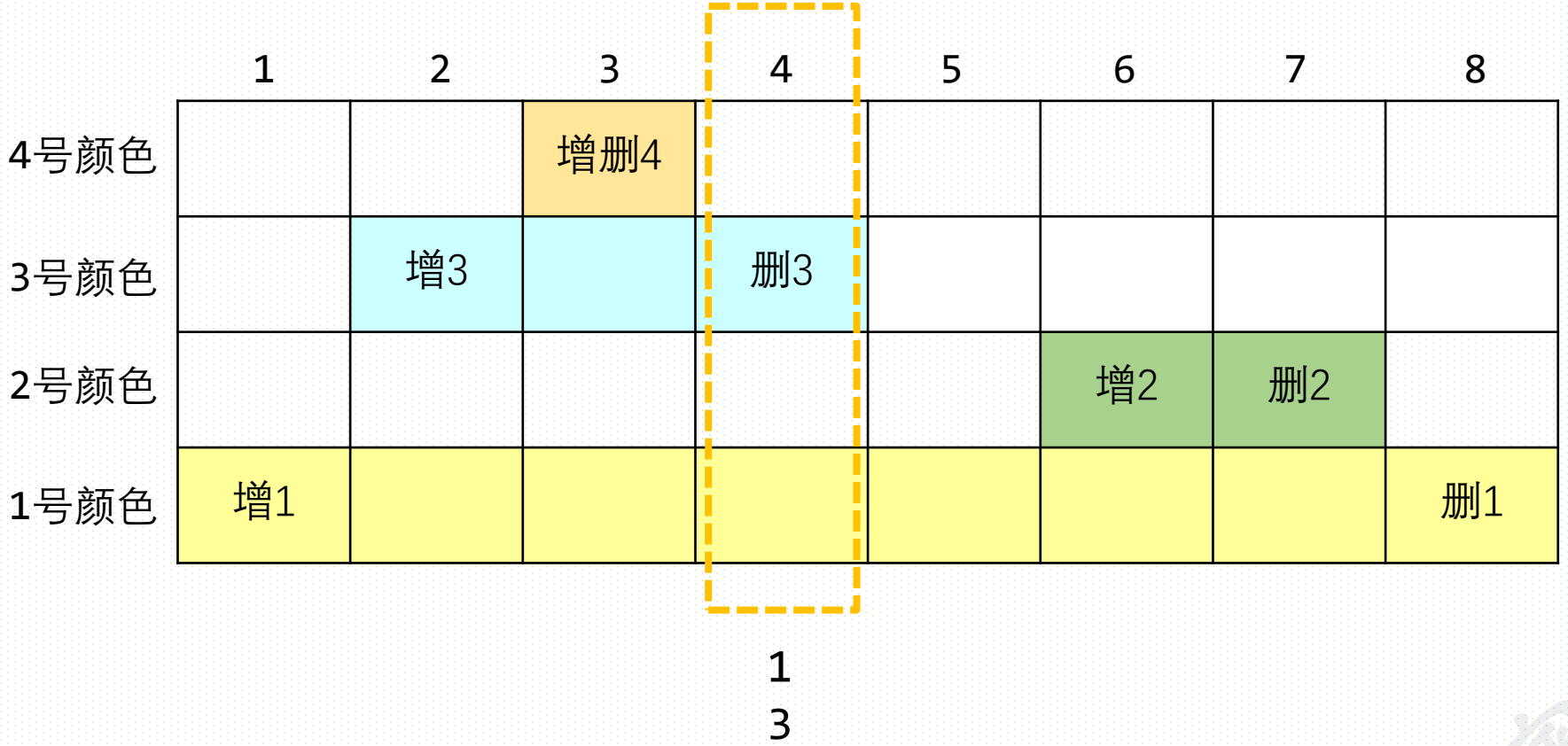
8格涂色4次
[1,8]涂1号颜色
[6,7]涂2号颜色
[2,4]涂3号颜色
[3,3]涂4号颜色



8格涂色4次
[1,8]涂1号颜色
[6,7]涂2号颜色
[2,4]涂3号颜色
[3,3]涂4号颜色



8格涂色4次
[1,8]涂1号颜色
[6,7]涂2号颜色
[2,4]涂3号颜色
[3,3]涂4号颜色



区间两端标记+扫描线

区间批量更新

只标记两端

扫描线：从左往右处理所有标记

实时维护扫描线上数据

需要支持最大值查询
请问用哪个数据容器

add增加
remove删除

```
20 vector<ll> ADD[N],RMV[N];
```

```
22     for(ll clr=1;clr<=m;++clr){
23         ll l=(clr*p+q)%n+1;
24         ll r=(clr*q+p)%n+1;
25         if(l>r) swap(l,r);
26         ADD[l].push_back(clr);
27         RMV[r].push_back(clr);
28     }
```

ADD[x]储存x号位置出现的区间左端点们对应的颜色

RMV[x]储存x号位置出现的区间右端点们对应的颜色

扫描线：从左往右处理所有标记

实时维护扫描线上数据

A 增加该位置上ADD信息

B 删除该位置上RMV信息

C 计算该位置答案

请为以上3种操作排序	A C B

```
30      multiset<ll> s;
31      for(int k=1;k<=n;++k){
32          for(int i=0;i<ADD[k].size();++i)
33              s.insert(ADD[k][i]);
34          if(s.empty())ans[k]=0;
35          else ans[k]=*s.rbegin();
36          for(int i=0;i<RMV[k].size();++i)
37              s.erase(                    );
38      }
```

multiset能否改成set?	可以

T5 魂器

现场挑战

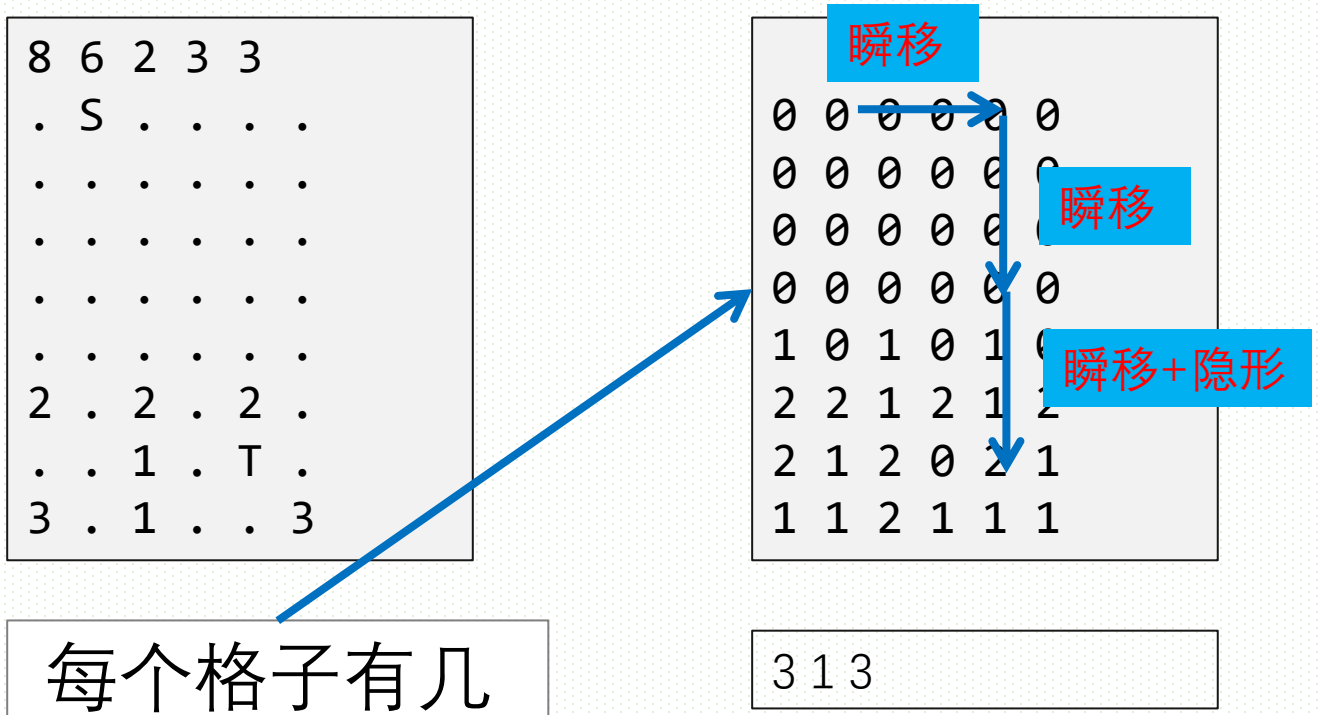
用纸和笔概括题目大意 已知什么求什么
写出算法步骤和复杂度 或者写出难点
限时5分钟

请同学核对题目大意
不能遗漏核心要点

n行m列的地图，起点为S，终点为T，有若干卫兵可以观察与他曼哈顿距离小于 $a_{i,j}$的点。每秒8方向走1格， c1次机会使用隐形技能可以进入卫兵观察范围（不包括卫兵所在格）， c2次机会使用瞬移技能可以四方向移动d格。求最少时间，技能使用次数尽量少，隐形尽量少。

$2 \leqslant n,m \leqslant 350$，$1 \leqslant a_{i,j} \leqslant 350$，$0 \leqslant c1,c2 \leqslant 15$，$1 \leqslant d \leqslant 350$ 。

手算样例


8 6 2 3 3
. S . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
2 . 2 . 2 .
. . 1 . T .
3 . 1 . . 3
瞬移
瞬移
瞬移+隐形
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
1 0 1 0 1 0
2 2 1 2 1 2
2 1 2 0 2 1
1 1 2 1 1 1
每个格子有几名守卫能看到?
3 1 3
终点在卫兵观察范围内，需要隐形

记忆化搜索+差分

搜索:

- DFS
- BFS

剪枝:

1.每个格子只经过1次（记忆化vst数组有4个维度: *m**c1*c2 ）

2.当前答案>=目前最佳答案，停止搜索

记忆化搜索+差分

- 二维数组a记录每个格子**是否**有卫兵能看到
- 但是如果对每个卫兵都去标记他的观察范围，复杂度是$O(N^2M^2)$，会TE。
- 利用差分可以降低复杂度

记忆化搜索+差分

			$(i-a_{ij}+1,j)$+1	-1		
		+1			-1	
	+1					-1
$(i,j-a_{ij}+1)$+1			卫兵(i,j)			$(i,j+a_{ij}-1)$ -1
	+1					-1
		+1			-1	
			$(i+a_{ij}-1,j)$+1	-1		

差分数组: 横着看

(主要是去确定这个菱形的特征——行偏移得多，列摆动的幅度就小)

处理差分

```
11 void add(ll i,ll j,ll x){//i行j列卫兵能看到x距离
12     for(ll d=-x+1;d<=x-1;d++){//枚举卫兵观察范围的行偏移量
13         int nowi=i+d;//现在处理nowi行
14         if(                    )continue;
15         ll p=x-1-abs(d);//p为列偏移量，先减小后增大，代表菱形
16         int startj=j-p;//卫兵能看到的最左列
17         if(startj<1)a[nowi]    +=1;//覆盖该格的卫兵数+1
18         else a[nowi]           +=1;
19         int endj=      ;//卫兵能看到的最右列的右边一列
20         if(endj>m);
21         else a[nowi][endj]-=1;//覆盖该格的卫兵数-1
22     }
23 }
```

```
 4  #define N 359
 5  #define INF 1e9
 6  ll n,m,c1,c2,d;//如题
 7  ll sx,sy,tx,ty;//起点终点坐标
 8  string s;//读入的数据
 9  ll a[N][N],flag[N][N];
10  bool v[N][N][16][16];//剪枝一：已经搜过的节点不再搜索
11  void add(ll i,ll j,ll x){//i行j列卫兵能看到x距离
24  struct info{
25      ll x,y,u1,u2,t;
26      //坐标,隐身使用次数,瞬移使用次数,已经过了多长时间
27  };
28  ll ans=INF,ans1=INF,ans2=INF;
29  ll dx[8]={0,0,1,-1,1,1,-1,-1};
30  ll dy[8]={1,-1,0,0,1,-1,1,-1};
31  void bfs(){
88  int main(){
```

处理输入

```
91      cin>>n>>m>>c1>>c2>>d;
92      for(ll i=1;i<=n;i++){
93          for(ll j=1;j<=m;j++){
94              cin>>s;
95              if(s=="S")flag[i][j]=-2,sx=i,sy=j;
96              else if(s=="T")flag[i][j]=-1,tx=i,ty=j;
97              else if(s==".")
98              else{
99                  flag[i][j]=1;
100                 ll x=s[0]-'0';
101                 for(ll k=1;k<s.size();k++)x=x*10+s[k]-'0';
102                 add(i,j,x);//差分使复杂度降为n^3
103             }
104         }
105     }
```

```
106     for(ll i=1;i<=n;i++){
107         for(ll j=1;j<=m;j++){
108             a[i][j]+=a[i][j-1];//注意差分要加回去
109         }
110     }
111     bfs();
112     if(          )cout<<-1;//无解
113     else cout<<ans<<' '<<ans1<<' '<<ans2;//输出
114     return 0;
115 }
```

```
31  void bfs(){
32      queue<info> q;
33      q.push((info){sx,sy,0,0,0});
34      [          ]=1;//起点已经过
35      while(!q.empty()){
36          info now=q.front();
37          q.pop();
38          if([          ])continue;//剪枝二：已经不如目前最佳方案
39          if(now.x==tx&&now.y==ty){//已经到达终点
57          for(ll i=0;i<8;i++){//八方向移动一格
72          if([          ])continue;//无法使用瞬移
73          for(ll i=0;i<4;i++){//四方向瞬移
86      }
87  }
```

```
39      if(now.x==tx&&now.y==ty){//已经到达终点
40          if(now.t<ans){
41              ans=now.t;
42              ans1=now.u1;
43              ans2=now.u2;
44          }
45          else{
46              if(ans1+ans2>now.u1+now.u2){//魔法使用次数少
47
48
49              }
50              else if(ans1+ans2==now.u1+now.u2&&          ){ //魔法一样,隐身u1少
51
52
53              }
54          }
55          continue;
56      }
```

```
57  for(ll i=0;i<8;i++){//八方向移动一格
58      ll nx=now.x+dx[i],ny=now.y+dy[i];
59      if(nx<1||nx>n||ny<1||ny>m)continue;//越界
60      if(                    )continue;//有卫兵
61      if(a[nx][ny]<=0&&v[nx][ny][now.u1][now.u2]==0){
62      //不在卫兵的观察范围内
63          v[nx][ny][now.u1][now.u2]=1;//标记
64          q.push((info){nx,ny,now.u1,now.u2,          });
65      }
66      else if(now.u1+1<=c1&&v[nx][ny][now.u1+1][now.u2]==0){
67      //在卫兵的观察范围内,使用隐身
68          v[nx][ny][now.u1+1][now.u2]=1;//标记
69          q.push((info){nx,ny,                          });
70      }
71  }
```

```
72          if(now.u2+1>c2)continue;//无法使用瞬移
73          for(ll i=0;i<4;i++){//四方向瞬移
74              ll nx=now.x+          ,ny=now.y+          ;
75              if(nx<1||nx>n||ny<1||ny>m)continue;
76              if(flag[nx][ny]==1)continue;
77              if(a[nx][ny]<=0&&v[nx][ny][now.u1][now.u2+1]==0){
78                  v[nx][ny][now.u1][now.u2+1]=1;
79                  q.push((info){nx,ny,now.u1,now.u2+1,now.t+1});
80              }
81              else
82
83
84              }
85          }
```

记忆化搜索+差分

- 整体复杂度O(n*m*c1*c2*12)
- 为什么是常数12?
- 状态迁移的不同路径：8+4

SH-B260104 订正题号

T1　2718 费用平摊

T2　3154 学术圈2

T3　3072 文武双缺

T4　2124 涂鸦王子

T5　2096 魂器