

长寿基因



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3分钟

请同学校对题目大意
不能遗漏核心要点

在 n 个字符串中统计TAT出现次数（可重叠）最多的字符串

$n \leq 100$ ，样本名字长度不超过20，
样本序列长度不超过200

简单模拟

- 查找子串并统计出现次数
- 出现次数打擂台求出最大值
- 最大值的名字保存到一个`map<int,vector>`
- 或者直接连接到字符串中，注意两个名字之间有空格


70分，错在哪儿？

```
5  freopen("longlife.in","r",stdin);
6  freopen("longlife.out","w",stdout);
7  ll n,i;
8  cin>>n;
9  string ans="",name="",seq="";
10 ll maxx=-1;
11 while(n--){
12     cin>>name>>seq;
13     ll cnt=0;
14     for(i=0;i<[redacted];i++)
15         if(seq[i]=='T'&&seq[i+1]=='A'&&seq[i+2]=='T')
16             cnt++;
17     if(cnt>maxx)[redacted]
18     else [redacted]
19 }
20 cout<<ans<<endl<<maxx<<endl;
21 return 0;
```

序列长度为1时出错

```
11 白 while(n--){
12      cin>>name>>seq;
13      ll cnt=0;
14      for(i=0;i<(ll)seq.size()-2;i++)//size返回无符号数,小心下溢
15          if(seq[i]=='T'&&seq[i+1]=='A'&&seq[i+2]=='T')
16              cnt++;
```

```
11 白 while(n--){
12      cin>>name>>seq;
13      ll cnt=0;
14      ll len=seq.size();
15      for(i=0;i<len-2;i++)
16          if(seq[i]=='T'&&seq[i+1]=='A'&&seq[i+2]=='T')
17              cnt++,i++;
```



找到以后下
标可以加2

可怜的蜗牛



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3分钟

请同学核对题目大意 不能遗漏核心要点

n 个传送装置，第 i 个装置位于 a_i 处，如果启用，则路程增加 a_i 。蜗牛速度为 v 。 q 个询问，第 i 个询问要让蜗牛到达时间大于 t_i ，求启动装置的最小数目。

$1 \leq n, q \leq 2 \times 10^5$, $1 \leq v \leq L \leq 10^9$, $1 \leq a_i \leq L$,

$1 \leq t_i \leq 10^9$ 。

数据保证 a_i 两两不同。

手算样例

3 6 3

3 5 1

4

1

3

4

5

输出？

手算样例

$n=3, l=6, v=3$

$a[]: 3 \ 5 \ 1$

$q=4$

$t[]: 1 \ 3 \ 4 \ 5$

0
1
2
-1

从中找到解题思路

贪心

如果要启动最多*i*个装置以完成任务，应该选哪些装置？

应该启动 a_i 最大的那些装置。

a_i 按照从大到小排序，预计算前缀和数组sum

$$\text{sum}[0] = 1$$

$$\text{sum}[i+1] = \text{sum}[i] + a[i], \quad 0 \leq i < n$$

利用二分搜索库函数找到k，使得

$$\text{sum}[k] > v * t_i, \text{ 而 } \text{sum}[k-1] \leq v * t_i$$

```
3  #define N 200009
4  #define ll long long
5  ll n,l,v,q,t,a[N],sum[N];
6  int main(){
7      freopen("snail.in","r",stdin);
8      freopen("snail.out","w",stdout);
9      cin>>n>>l>>v;
10     for(ll i=0;i<n;i++)
11         cin>>a[i];
12     sum[0]=
13     sort(a,a+n);
14     reverse(a,a+n);
15     for(ll i=0;i<n;i++)
16         sum[i+1]=sum[i]+a[i];
```

```
17 cin>>q;
18 for(ll i=0;i<q;i++){
19     cin>>t;
20     ll s=t*v;
21     ll ans=upper_bound( , )-sum;
22     if( )
23         cout<<"-1"<<endl;
24     else
25         cout<<ans<<endl;
26 }
27 return 0;
```

月球脱险



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3分钟

请同学核对题目大意
不能遗漏核心要点

起点终点相距 L ，其间有 N 个氧气站，第 i 个氧气站距离起点 D_i ，氧气价格为 P_i 。起点氧气价格为 P ，氧气背包容量为 C ，起始氧气量为 0 。每升氧气行走距离为 U ，计算到达终点的最小花费？（保留两位小数）无法到达返回 -1 。

$$0 \leq N \leq 100$$

手算样例

300 10 20 2 2
100 3
200 1

输出？

手算样例

L C U P N

300 10 20 2 2
100 3
200 1

25.00

起点
价格2

1号站
价格3

2号站
价格1

终点

0

100

200

300

从中找到解题思路

模拟+贪心

- 采取何种贪心策略？

起点策略（粗略）

300 10 20 2 2
100 3
200 1

25.00

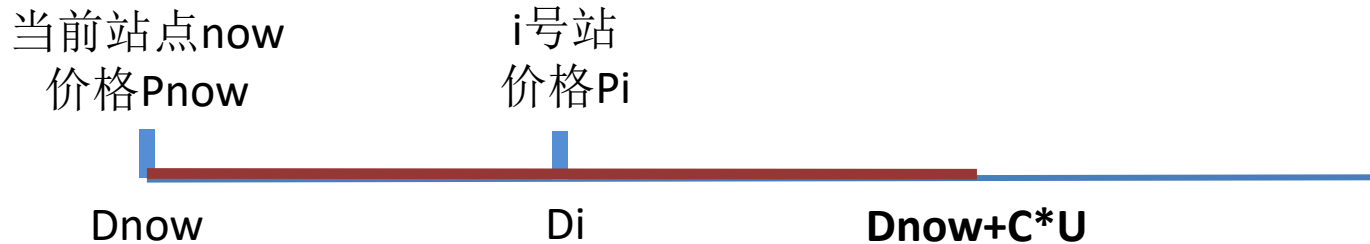


起点氧气量为0，必须加氧气。加满氧气价格 $C \cdot P$ ，能到的最远距离是 $C \cdot U$ 。

在能到达的范围内检查：

1. 是否到达终点？若是，则结束计算。
2. 有没有比当前价格低的站？若有，加能刚好到达那个站点的氧气。到达那个站。
3. 如果没有比当前价格低的站，找范围内价格最低的站。加氧气，到达那个站。

推广到一般情况并细化



假设在当前站点now加满氧气，已经花费ans，能到的最远距离是 $D_{now}+C*U$ 。

在能到达的范围内检查：

1. 是否到达终点？若是，则结束计算。

只要范围内有终点就结束计算吗？

不一定。有低价先到达低价站点。

2. 有没有比当前价格低的站？若有，加能刚好到达那个站点的氧气（原来假设在now加满，现在要扣除多余氧气）。到达那个站。

范围内若有多个低价站点呢？

先到达第一个低价站点。

3. 如果没有比当前价格低的站，找范围内价格最低的站。加氧气，到达那个站。

加能刚好到达那个站点的氧气吗？

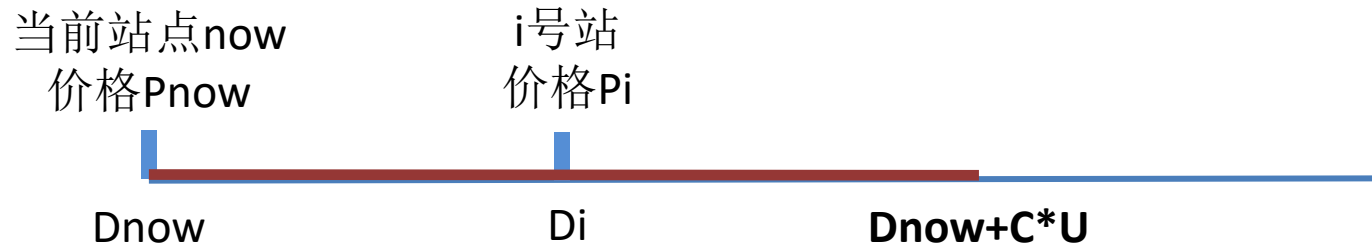
加满氧气更合算。

到达新站点重复上述过程

什么情况下输出-1？

范围内没有任何站点，包括终点。

模拟+贪心 完整策略



假设在当前站点now加满氧气，已经花费ans，能到的最远距离是 $D_{now}+C*U$ 。

在能到达的范围内由近及远依次检查：

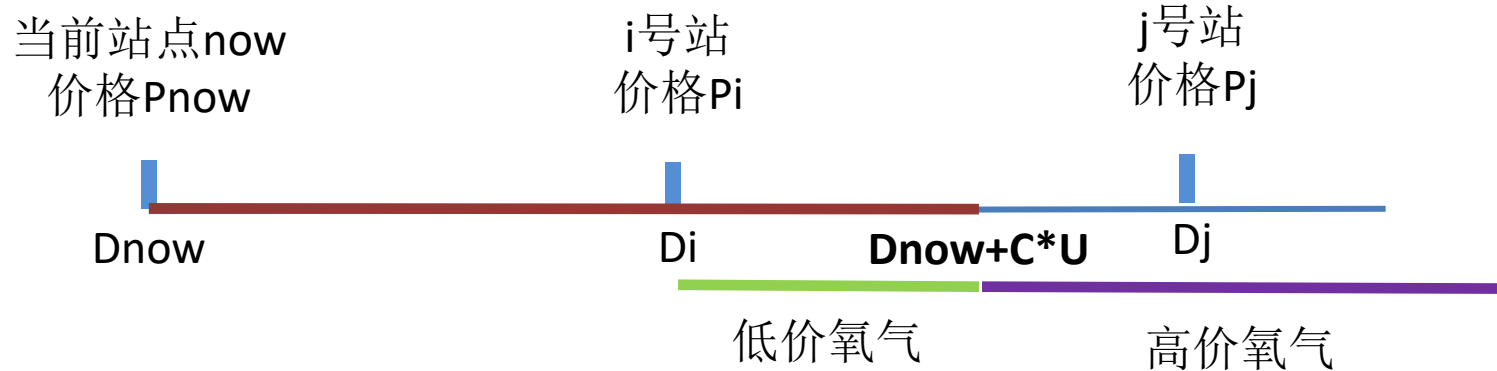
- 1.如果检查到终点，扣除多余氧气，结束计算。
- 2.如果检查到第一个低价站点，加能刚好到达那个站点的氧气（原来假设在now加满，现在要扣除多余氧气），到达那个站。
- 3.如果没有比当前价格低的站，找范围内价格最低的站。加满氧气，到达那个站。
- 4.如果范围内没有任何站点（低价、高价、终点都没有），输出-1。

到达新站点重复上述过程

更多细节考虑

- 当宇航服内同时有两种价格的氧气，要不要记录每种的价格和数量？
- 如果不记录，扣除多余的高价氧气时，是否会把低价氧气也当成高价的扣除了，使得计算结果偏低？

两种价格氧气的情形



$P_{now} < P_i, P_i > P_j$, 在 j 号站点要扣除多余较贵的氧气。

此时，一定有 $D_j > D_{now} + C * U$ 。

否则就不会到达 i 号站，而是直接到达 j 号站。（无论 P_j 是否低于 P_{now} ）

因此，低价氧气已经耗尽，不存在被当成高价氧气被扣除的问题。

存在两种价格的氧气时，优先使用低价氧气。因为高价氧气后面多余的话可以扣除。

所以，不需要记录多种氧气的价格和数量

```

3  #define N 109
4  #define INF 1e9
5  #define Err 1e-6
6  int n;
7  double L,C,U,range,ans;//range=C*U
8  struct stop{
9      double d,p;//每个氧气站的距离及价格
10 }s[N];
11 bool cmp(const stop& a, const stop& b) { return a.d<b.d; }

```

```

33  cin>>L>>C>>U>> >>n;
34  >=L;
35  for(int i=1;i<=n;i++)
36      cin>>s[i].d>>s[i].p;
37  sort(s,s+n+2,cmp);//按照距离从小到大排序
38  range=C*U;//装满氧气能行走的最大距离
39  ans=C*s[0].p;//假定先装满氧气，计算当前成本
40  int k=0,t;//k为当前站点，从起点开始
41  do{
42      t=Solve(k),k=t;//迭代，寻找下一到达站点t
43  }
44  if(t==-1)
45      cout<<-1;// 不可能完成
46  else
47      cout<<fixed<<setprecision(2)<<ans;
48  return 0;

```

```

12 int Solve(int now){
13     double minP=INF;
14     int minI=N;
15     double rangei=s[now].d+range+Err;//计算可到达范围
16     int i;
17     for(i=now+1;s[i].d<=rangei;i++){//枚举范围中的每个点
18         if(i==n+1||s[i].p<=s[now].p){//如果已到终点, 或者发现低价
19             ans -= [redacted]; //扣除多余较贵氧气
20             ans += C*s[i].p; //以低价加满氧气 (终点氧气价格为0)
21             return i; //到达i号氧气站或者终点
22         }
23         if(s[i].p<minP) //在价格更高的站中找最便宜的
24             minP=s[i].p,minI=i;
25     }
26     if(minP==INF) return -1; //如果范围内无任何站点
27     ans += [redacted]; //补充消耗掉的氧气
28     return minI; //在所有高价站中找最便宜的, 到达它
29 }

```

条件也可以写成
i==now+1

更多细节

- 所有浮点数比较都要考虑误差Err吗？

如果比较的两个浮点数都是键盘读入或者直接用字面量赋值，可以不考虑Err。

```
11 bool cmp(const stop& a, const stop& b) { return a.d<b.d; }  
35     for(int i=1;i<=n;i++)  
36         cin>>s[i].d>>s[i].p;
```

```
26     if(minP==INF) return -1; //如果范围内无任何站点
```

如果参与比较的两个数之一经过了浮点运算，必须考虑Err。

```
15     double rangei=s[now].d+range+Err; //计算可到达范围  
16     int i;  
17     for(i=now+1;s[i].d<=rangei;i++){ //枚举范围中的每个点
```

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，
 c_1 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），
 c_2 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。

$2 \leq n, m \leq 350$, $1 \leq a_{i,j} \leq 350$, $0 \leq c_1, c_2 \leq 15$, $1 \leq d \leq 350$ 。

手算样例

8	6	2	3	3	
.	S
.
.
.
.
2	.	2	.	2	.
.	.	1	.	T	.
3	.	1	.	.	3

输出？

手算样例

8	6	2	3	3	
.	S
.
.
.
.
2	.	2	.	2	.
.	.	1	.	T	.
3	.	1	.	.	3

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？

手算样例

8	6	2	3	3
.	S	.	.	.
.
.
.
.
2	.	2	.	2
.	.	1	.	T
3	.	1	.	.

瞬移					
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

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`vis`数组有4个维度： $n * m * c1 * c2$ ）
2. 当前答案 \geq 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	
			$(i+a_{ij}-1,j)+1$	-1		

```
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(){

```

```
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         else if(ans1+ans2==now.u1+now.u2&&){//魔
50 ☐             
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