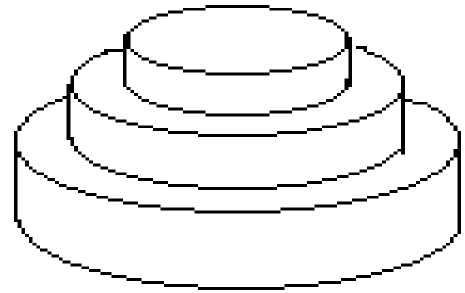
搜索的优化

主讲人：朱全民

生日蛋糕(Cake)(n0I99-3)



条件1：$V = n\pi$

$H = m$ 层

形状：每层都是一个圆柱体。

条件2：

设从下往上数第i(1<=i<=m)层蛋糕是半径为R_i, 高度为H_i的圆柱。

当i<m时，要求$R_i>R_{i+1}$且$H_i>H_{i+1}$。

条件3：

表面积Q最小，令$Q= S\pi$

问题：

给出的n和m，

找出蛋糕的制作方案（适当的R_i和H_i的值），使S最小。

(除Q外，以上所有数据皆为正整数)

输入

n (n<=10000)，

m (m<=20)

输出

S（若无解则S=0）。

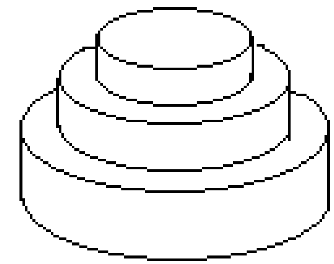
圆柱公式

$V=\pi R^2H$

$S_{侧}=2\pi RH$

$S_{底}=\pi R^2$

解析法？



(1) $$V = \Pi \sum_{i=1}^{m} R_i * R_i * H_i)$$

(2) $$R_i > R_j \quad and \quad 1 \le i < j \le m$$

(3) $$H_i > H_j \quad and \quad 1 \le i < j \le m$$

其中，$S = \Pi(R_1 * R_1 + 2\sum_{i=1}^{m} R_i * H_i)$

$Q_{\min} = \min\{R_1 * R_1 + 2\sum_{i=1}^{m} R_i * H_i\}$ 满足1,2,3

转变思路，搜索？

- **数据库**

用(i , R_i , H_i , V_i , S_i)表示第i层蛋糕的一个状态。其中R_i ,H_i分别为第i层蛋糕的半径和高，V_i , S_i分别表示做完第i层蛋糕后剩下的蛋糕体积和当前蛋糕的表面积。可见，

初始状态：$(1,R_1,H_1,n-R_1*R_1*H_1,R_1*R_1+2*R_1*H_1)$

目标状态：$(m,R_m,H_m,0,S_m)$

于是，我们的目标是找到一条从初始状态到任意目标状态的路径，并且S_m最小.

- **扩展的规则**

$(i , R_i , H_i , V_i , S_i) \rightarrow (i+1,R_{i+1},H_{i+1},V_{i+1},S_{i+1})$

满足:

（1） $R_i > R_{i+1}$

（2） $H_i > H_{i+1}$

（3） $V_{i+1} = V_i - R_{i+1} * R_{i+1} * H_{i+1}$

（4） $S_{i+1} = S_i + 2 * R_{i+1} * H_{i+1}$

基本算法

- 确定第一层蛋糕的大小
- 根据上一层蛋糕的大小确定下一层蛋糕该怎么做
- 看是否符合条件

  1）是否做到了m层

  2）是否最终体积为0

  3）是否当前面积最小
- 若上述条件成立，则保留当前最优值，否则继续做下一层蛋糕，若重做蛋糕

- Search (i, R_i , H_i , S_i , V_i)
  {对每层蛋糕进行搜索}
  if (i=m) and (V_i =0) then 更新最优值
  else
  for $R_{i+1} \leftarrow R_i$ -1 downto i
  for $H_{i+1} \leftarrow H_i$ -1 downto i [
  $S_{i+1} \leftarrow S_i + 2 * R_{i+1} * H_{i+1}$
  $V_{i+1} \leftarrow V_i - R_{i+1} * R_{i+1} * H_{i+1}$
  Search (i+1, R_{i+1}, H_{i+1}, S_{i+1},V_{i+1})
  ]

- Problem-Cake
  {枚举所有的初始状态 ----- 第一层蛋糕的大小}
  for $R_1 \leftarrow$m to sqrt (n) do /*假设 H_1=1，只做一层蛋糕 */
  for $H_1 \leftarrow$n div (R_1*R_1) downto m do
  [
  $S_1=2 * R_1 * H_1 + R_1 * R_1$
  $V_1=n - R_1 * R_1 * H_1$
  Search (1, R_1, H_1, S_1, V_1)
  ]

优化？？

（1）因为知道余下的蛋糕体积，因此可以估算一下余下侧面积，
这样我们可以就加入如下剪枝条件：

if 当前的表面积 + **余下的侧面积** > 当前最优值 then exit

设已经做了i层蛋糕，则还需做m-i层，
S_i'：为第i层蛋糕的侧面积，
FS_i：余下的侧面积，怎么求FS_i？
因为：

$$2V_i = 2R_{i+1} * R_{i+1} * H_{i+1} + ... + 2R_m * R_m * H_m$$
$$= R_{i+1} * S_{i+!}' + ... + R_m * S_m'$$
$$\leqslant R_{i+1} * (S_{i+1}' + ... + S_m')$$
$$= R_{i+1} * FS_i$$

所以：

$$FS_i \geqslant 2V_i / R_{i+1}$$

因此剪枝条件为：

if $S_{i-1} + 2 * V_{i-1} / R_i >$ 当前最优值 then exit

优化？？

(2)如果剩下的蛋糕材料太少，不能保证做到m层,那么没有必要继续往下做了，设，

最m层半径和高都为1，$R_m=H_m=1$

第m-1层半径和高都为2，$R_{m-1}=H_{m-1}=2$

…………

第 i +1层半径和高都为i, $R_i = H_i = m - i$

这样， 余下的m-i层的最小体积为

$$MIN_i = \sum_{k=1}^{m-i} k^3$$

因此,剪枝条件为,

if $V_i < MIN_i$ then exit

优化？？

(3)如果剩下的蛋糕材料太多，以最大的方式做完m层, 仍有材料剩余,那么没有必要继续往下做了，设，

第i+1层半径和高分别为，$R_{i+1} = R_i - 1$， $H_{i+1} = H_i - 1$

第i+2层半径和高分别为，$R_{i+2} = R_i - 2$， $H_{i+2} = H_i - 2$

…………

第 m层半径和高分别为，$R_{i+m} = R_i - m$，$H_{i+m} = H_i - m$

这样， 余下的m-i层的最大体积为

$$MAX_{i,R,H} = \sum_{j=i}^{M} (R_j - j)^2 * (H_j - j)$$

因此,剪枝条件为,

if V_i ＞ $MAX_{i,R,H}$ then exit

初始化

- 计算MIN_i

```
for i ← 1 to n do [
    S ← S + i * i * i;
    MIN_{m-i} ← S
]
```

- 计算$MAX_{i,R,H}$

```
for R ← 1 to sqrt(n) do
    for H ← 1 to n div (R*R) do [
        S ← 0;
        for i ← m downto 1 do [
            S ← S +(R-i)*(R-i)*(H-i);
            MAX_{i,R,H} ← S
        ]
    ]
```

- Search (i, R_i , H_i , S_i , V_i)

  {对每层蛋糕进行搜索}

  if $S_i + 2 * V_i / R_i >$***当前最优值 then exit; {剪枝1}***

  if $V_i < MIN_i$ ***then exit; {剪枝2}***

  if $V_i > MAX_i$ ***then exit; {剪枝3}***

  if i<m then

  for $R_{i+1} \leftarrow R_i$ downto i

  for $H_{i+1} \leftarrow \min(V_i \text{ div } (R_{i+1}*R_{i+1}), H_i)$ downto i [

  $S_{i+1} \leftarrow S_i + 2 * R_{i+1} * H_{i+1}$

  $V_{i+1} \leftarrow V_i - R_{i+1} * R_{i+1} * H_{i+1}$

  Search (i+1, R_{i+1}, H_{i+1}, S_{i+1}, V_{i+1})

  ]

  Else if V_i =0 then 更新最优值

- Problem-Cake

  1. 计算*MIN*$_i$和*MAX*$_{i\ R,H}$;
  2. for $R_1 \leftarrow m$ to sqrt (n) do /*假设 H_1=1，只做一层蛋糕 */
  3. for $H_1 \leftarrow n \text{ div } (R_1*R_1)$ downto m do [
  4. $S_1=2 * R_1 * H_1 + R_1 * R_1$
  5. $V_1=n - R_1 * R_1 * H_1$
  6. Search (1, R_1, H_1, S_1, V_1)
  7.]

小节

- 最优化剪枝

  剪枝1： if $S_{i-1} + 2 * V_{i-1} / Ri >$当前最优值 then exit

- 可行性剪枝

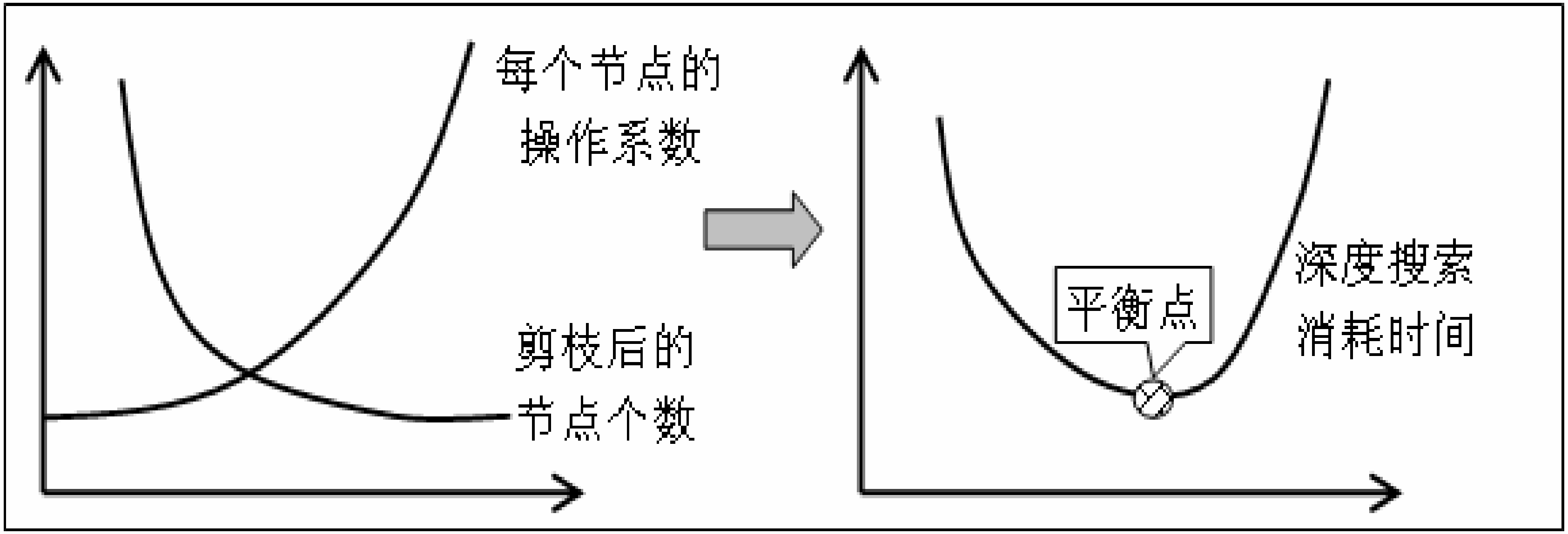
  剪枝2：if $V_i < MIN_i$ then exit

  剪枝3：if $V_i > MAX_{i,R,H}$ then exit

- 剪枝原则

  正确、高效

深度搜索消耗时间 ≈ 每个节点操作系数 × 节点个数



优化

1）减少节点个数——这就是**剪枝优化**；

2）减少每个节点的操作系数——即**程序操作量**。

彩票问题

已知：

彩票上的数字：1,2,…,M

彩民的选择：A1,A2,…,An，其中Ai属于1,2,…,M

每人只能买一张彩票，每人彩票选择都不同

抽出两个自然数X和Y。

如果1/A1+2/A2+…+1/An= X/Y，则中奖（获取纪念品）。

输入：

N，M，X，Y

输出：

所需准备的纪念品数量

1≤X, Y≤100，1≤N≤10，1≤M≤50。

输入数据保证输出结果不超过10^5。

分析：对于每个数，有选和不选两种可能性，显然可以建立如下模型：

$$x_1/1 + x_2/2 + x_3/3 + \ldots + x_m/m = X/Y$$

其中，$x_i=0$或者$1(1<=i<=m)$

$$x_1+x_2+x_3+\ldots+x_m=n$$

- 逐个搜索x_i
- $O(2^m)$

如何优化？？

$$x_1/1 + x_2/2 + x_3/3 + \ldots + x_m/m = X/Y$$

同时乘以m!*Y通分。

令$T_i=m!*Y/i\ (1<=i<=m)$, $T_0=m!*X$

则：

$$T_1x_1+T_2x_2+T_3x_3+\ldots+T_mx_m=T_0$$

这就变成了一个01背包问题。

每个包裹的体积是T_i，箱子体积T_0

从M个中选N个，填满箱子。

求方案数。

$T_1x_1+T_2x_2+T_3x_3+\ldots+T_mx_m=T_0$

如何剪枝？

f[i, T]表示为了满足$T_1x_1+T_2x_2+\ldots+T_mx_m=T$，最少要让多少个$x_i$取1。

$f[i, T]=\min\{f[i-1,T], f[i-1,T-T_i]+1\}$

按照$x_m, x_{m-1}, x_{m-2}, \ldots, x_1$的顺序搜索。

假设x_p~x_m都已经取定，令$S=T_px_p+T_{p+1}x_{p+1}+\ldots+T_mx_m$，$L=x_p+x_{p+1}+\ldots+x_m$，如果$f[p-1, T-S]+L>N$，那么就可以回溯，不必继续搜索了。

T~O(m!)。太大了！

f数组开不下，时间上也不允许。

$$T_1x_1+T_2x_2+T_3x_3+\ldots+T_mx_m=T_0$$

$f[i, T]$表示为了满足$T_1x_1+T_2x_2+\ldots+T_mx_m=T$，至少要让多少个$x_i$取1。

$f[i, T]=\min\{f[i-1,T], f[i-1,T-T_i]+1\}$

动态规划的思想，空间矛盾太大。

抓住矛盾：解决空间问题！

T太大了，可不可以想办法把它变小呢？

同余

$$T_1x_1+T_2x_2+T_3x_3+\ldots+T_mx_m=T_0$$

f[i, T]表示为了满足$(T_1x_1+T_2x_2+\ldots+T_mX_m) \bmod P=T$，至少要让多少个$x_i$取1。

$f[i, T]=\min\{f[i-1,T], f[i-1,(T-T_i) \bmod P]+1\}$

$$0<=T<P$$

按照$x_m, x_{m-1}, x_{m-2}, \ldots, x_1$的顺序搜索。

假设x_p~x_m都已经取定，令$S=T_px_p+T_{p+1}x_{p+1}+\ldots+T_mx_m$，$L=x_p+x_{p+1}+\ldots+x_m$，如果$f[p-1, (T-S) \bmod P]+L>N$，那么就可以回溯，不必继续搜索了。

剪枝效果有所削弱

但是空间复杂度降到了O(mP)，这里P可以任取。

- 取P为大质数。（比如<10000的最大质数）
- 剪枝效果相当的好。题目给定范围内的数据都在1s内解。
- 注意要使用高精度。

总结

1、空间换时间。

2、抓住主要矛盾，用同余法解决空间矛盾。

Amazing Robots: IOI2003

已知条件：

迷宫

i（i=1,2）

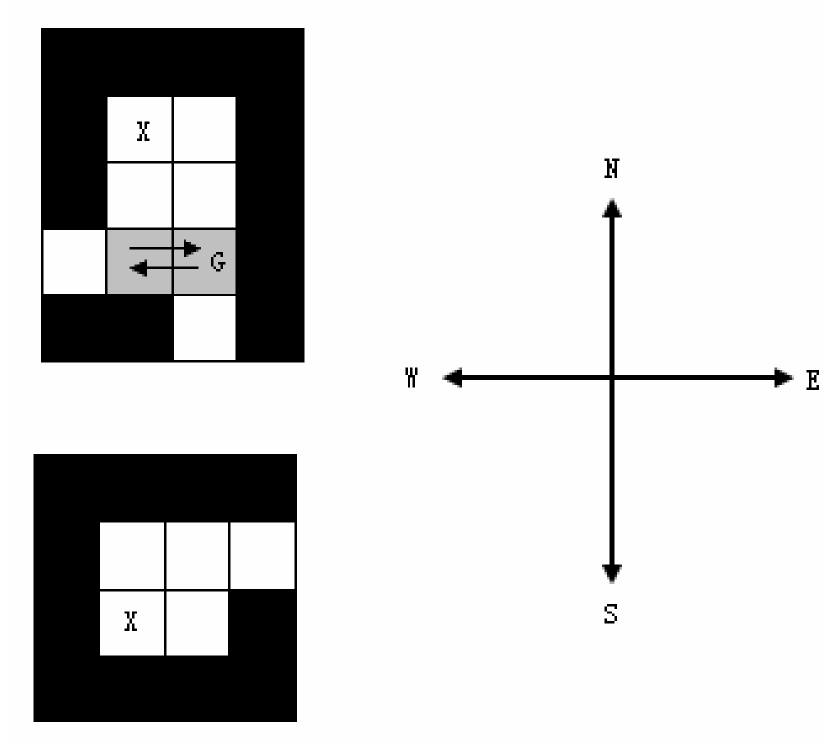
（每个不会大于20*20）

守卫

Gi（0<=Gi<=10）

（守卫循环移动进行执勤）

（守卫巡逻的方格数（2..4））

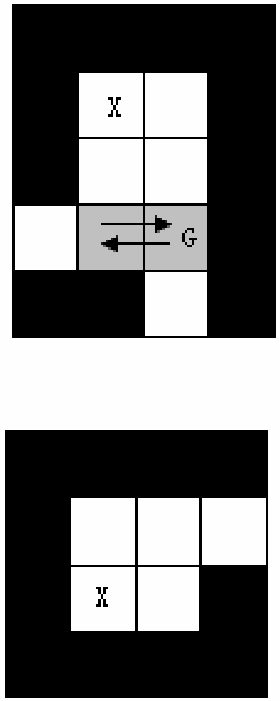


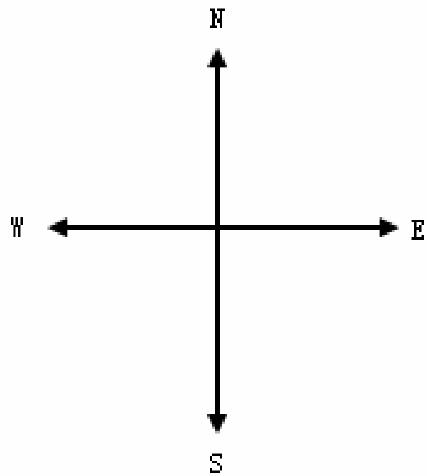
求：

两个机器人都离开迷宫所用的最少指令数目和离开制指令序列（10000 步以内）。

- 每一步可以发出的命令可以是N, E, S, W中的一种，有4种选择。
- 对每一步具体发出哪个命令，直接搜索。
- 假设最后结果是T。（也就是最少出宫时间）
- 时间复杂度是$O(4^T)$

这种方法时间复杂度太高，绝对不可行!!

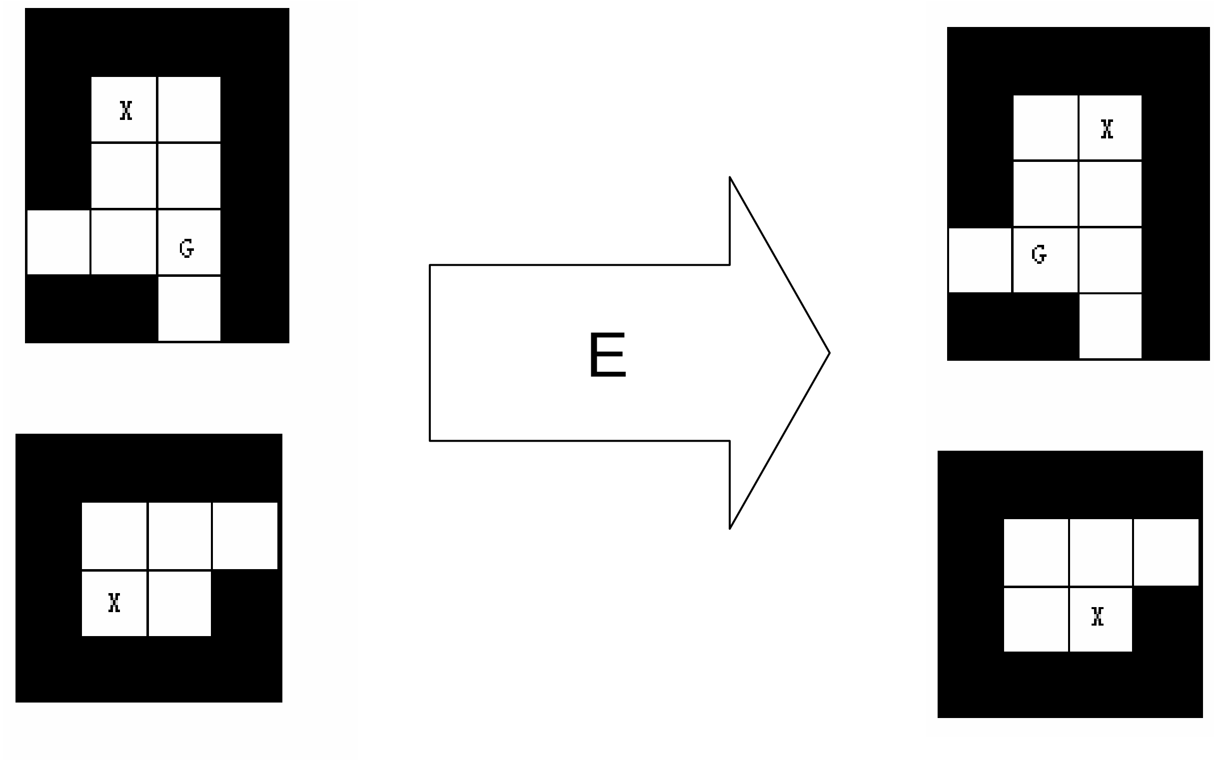




- 5*4和4*4的迷宫
- 第一个机器人的位置是(2,2)
- 第二个机器人的位置是(3,2)
- 当前时间是0。
- 状态((2,2),(3,2),0)

状态表示：

（第一个机器人位置，第二个机器认位置，时间）



((2,2),(3,2),0)

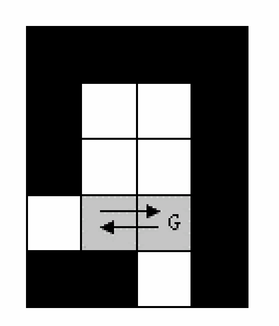
((2,3),(3,3),1)

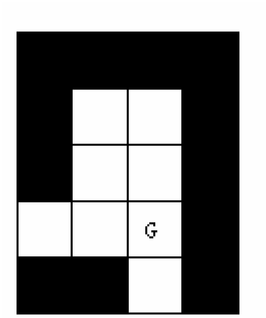
- 时间已知，则所有Guard的位置可知。
- Guard、Robot的位置均已知，所以状态可以转移

- 设出宫时间是T。
- 总共(n*n)*(n*n)*T种状态。
- 用BFS实现算法，HASH表判重。
- 总的复杂度是$O(Tn^4)$。

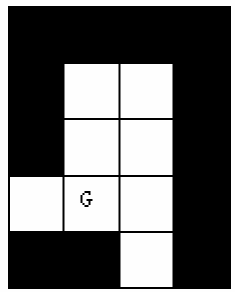
T<=10000, n<=20。

时间复杂度太大，不可承受！

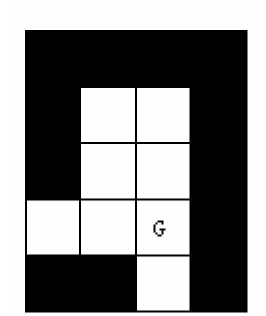




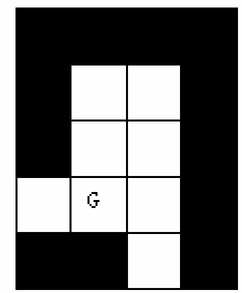
0时刻



1时刻



2时刻



3时刻

- 0时刻和2时刻是一样的
- 1时刻和3时刻是一样的。
- 稍加分析：此Guard循环以2为周期循环。

- 状态：**(position of Robot1, position of Robot2, Time)**
- Time<=10000，这是状态数过多的罪魁祸首！
- 状态转移，需要的信息是：Robot位置，Guard位置。
- Position of Robot1, 2是的作用就是记录Robot位置。
- Time的作用就是为了计算Guard的位置
- 题目说：Guard巡逻经过的格子数只可能是2, 3, 4。
- 也就是说机器人巡逻周期只能是2, 4, 6。
- [2, 4, 6]=12，所以第0时刻、12时刻、24时刻……Guard的状态完全相同。
- 12可以看作Guard的周期。Time只要记录当前是第几个周期。因为周期确定了，Guard的位置也完全确定了！

0<=Time<=11

- 状态数(n*n)*(n*n)*12=12n^4。
- 用BFS算法，标志数组判重。
- 时间复杂度O(12n^4)。

n<=20

完全可以承受

智破连环阵（N0I2003）

已知:

M个武器($1 \le M \le 100$)

N棵炸弹($1 \le N \le 100$)

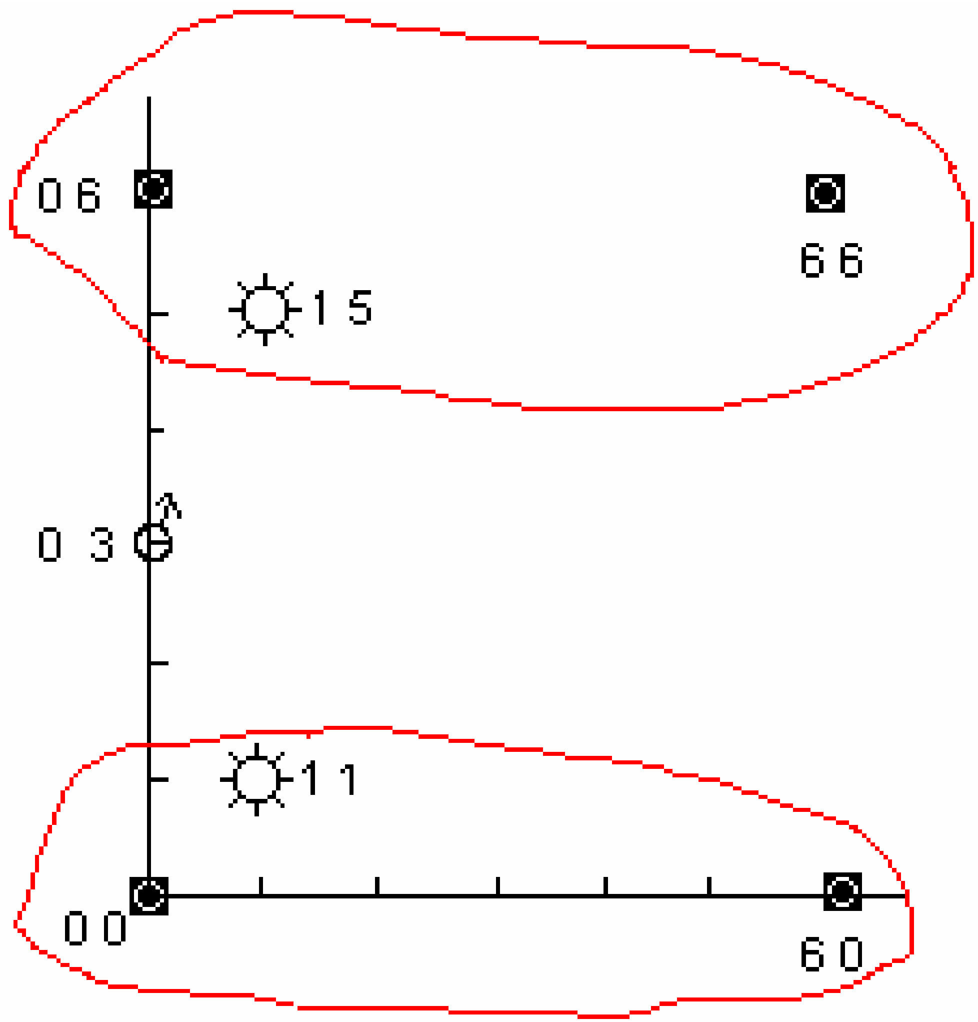
炸弹爆炸半径($1 \le k \le 1000$)

炸弹消灭半径范围以内的,处于攻击状态的武器

第i号武器被消灭后,第i+1号武器立即处于攻击状态

要求:用最少的炸弹消灭所有的武器

示例图



输入数据

4 3 6
0 6
6 6
6 0
0 0
1 5
0 3
1 1

输出数据

2
1 3

问题转化

- 设炸弹的攻击的半径为r
- 第i号炸弹的坐标为(u[i],v[i])
- 第j号武器的坐标为(x[j],y[j])

则炸弹i能炸到炸弹j的基本条件是

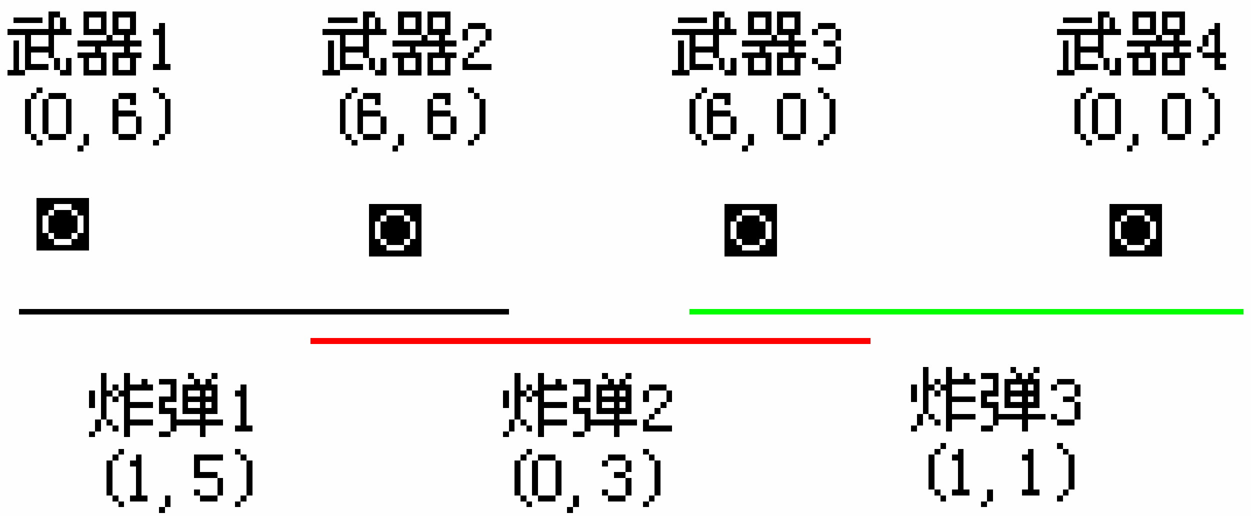
$$(u[i]-x[j])^2+(v[i]-y[j])^2<=r^2$$

我们可以枚举哪些炸弹是否爆炸,使得消灭所有得武器,时间复杂度为$M*2^n$,M,N可达到100,显然此方法时间复杂度巨大。

按武器搜索怎么样？

1）我们必须将武器按它的编号划分成若干段

2）每一段的武器，哪些炸弹能对它们进行销毁，如下图：



算法

- 将武器根据编号分为x段，[Si,Ti] (S1=1,Ti>=Si,Ti+1=Si+1)。
- 然后判断是否可以从A国的N颗炸弹中选出x颗，分别可以炸掉其中的一段。

如何分段：搜索算法

　　如何分段，理论复杂度为2^m，需要优化，留给学生思考？

如何选取炸弹：二分图的匹配

子串

- 给定一个由自然数串联而成的无限数列 1234567891011121314……（母串）
- 求任意一个长度不超过200的数列（子串）在其中最早出现的位置。

分析：

- 首先，由于母串可无限扩充，显然我们不可能把它全部生成出来。
- 如果边生成母串，边判断所生成的数串是否包含给定的子串，即使是使用字符串处理中高效的KMP算法，耗费的工作量也是巨大的。

1121314

- 先枚举第1位1
- 自然数1之后为2,3,……
- 母串中形如123……
- 与子串从第2位开始不符

- 枚举前2位11
- 自然数11之后为12,13……
- 母串中形如111213……
- 与子串从第3位开始不符

- 考虑第2位开始12
- 自然数12之前为11,末位与子串第1位相同，之后为13,14,……
- 母串中形如1314……
- 与子串匹配！

如何优化呢？

- 较好的方法是另辟蹊径：
- 刚才我们枚举的是母串的每一位，不妨换一个角度，从子串着手。
- 先来观察一些片断：

12345678910111213141516……

很自然得到算法：

- 枚举子串所包含第一个完整的数a的位数La。
- 假设a在母串的第k位出现（k<=La）
- 判断接下来由a生成的序列是否与子串吻合。
- 如果吻合，则最优解的判断：
  - (1) La<Ans_L
  - (2) La=Ans_L & k>Ans_k
- 4．跟据Ans_a及Ans_k计算出位数

- 总时间复杂度约为O(n^3)，与之前的不可估量相比有了本质性提高。
- 第4步可通过多种途径求得，这里不作介绍。
- 由于其中牵涉到许多高精度的计算及字符串处理，要求我们细致认真。

小结

- 充分挖掘题目特性是解决本题的关键。
- 得以使枚举这类看似低效率的方法得到较好的运用。
- 同时细致全面的考虑问题也是必不可少的。

搜索试题推荐

1）八数码、14数码问题

2）埃及分数问题

3）多个列车轨道的列车调度

4）贴邮票问题

等等