

程序设计基础

Fundamental of Programming

清华大学软件学院

刘玉身

liuyushen@tsinghua.edu.cn

课前思考题

右表中每个黑色框中均有4个彩色数字。

如何编写程序，找出满足如下条件的黑框？

该框中最大的数字，在右表所有黑色框各自包含的最大数字中，是最小的。

0	3	3	2	2	1	3	2	2	3
1	3	2	2	3	3	2	2	1	3
3	2	4	3	1	2	2	3	3	0
2	2	3	3	0	4	3	1	2	2
2	3	1	2	4	1	3	2	2	3
1	3	2	2	3	1	2	2	1	3
3	2	2	3	3	4	2	1	3	2
4	2	1	3	2	2	1	3	4	0
2	3	3	2	2	3	3	2	4	3
3	3	2	2	3	1	2	2	3	3

Lecture 11: 宽度优先搜索

——骑士聚会问题

宽度优先搜索算法 (Breadth-First-Search, BFS)

骑士聚会问题

在 $T \times T$ 的棋盘上有 n 名骑士（马），问：各自跳多少步，才能在某处聚在一起？允许骑士不移动，即呆在原处，此时步数为0。

要求给出的“聚会方案”满足：

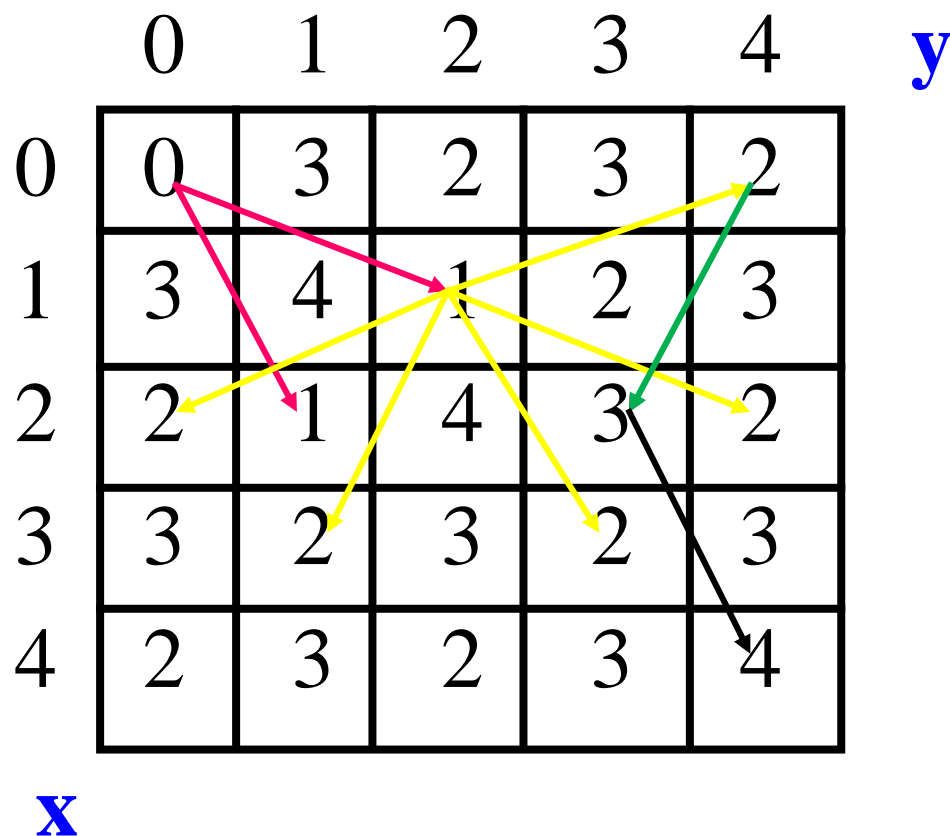
- （1）聚会时间越早越好，即最晚到达某处的骑士所用的时间最短（在所有方案中）
- （2）若聚会时间一样，选择总步数最少的方案

解题思路：

1. 从小到大，从简到繁。不妨以 $5*5$ 的棋盘为例来讨论。
2. 从个别到一般。假定有4个骑士，初始位置分别在 $(0,0)$, $(1,4)$, $(1,2)$, $(3,4)$ 。

显然，关键的问题是：如何得到各个骑士到达各个位置的时间（步数）呢？

子任务： 在一个5*5的棋盘上，求处在(0,0)位置上的象棋马跳到任何一个位置所需步数

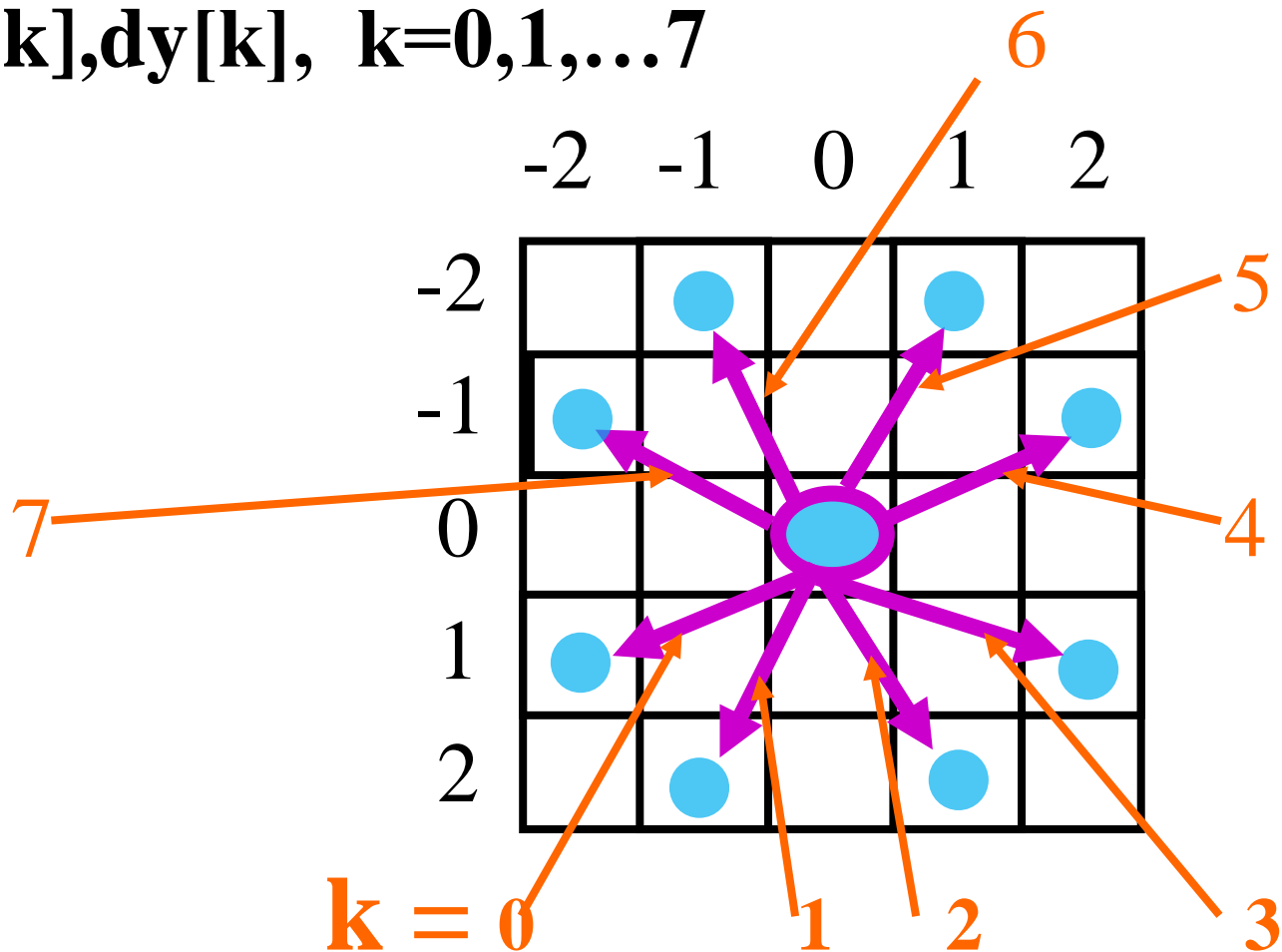


经初始化后的棋盘（5x5格子）

	0	1	2	3	4
0	-1	-1	-1	-1	-1
1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1
3	-1	-1	-1	-1	-1
4	-1	-1	-1	-1	-1

格子中的数为-1的格子，是尚未有马跳入过的地方。

根据马的跳步规则研究8个方向的跳步
增量 $dx[k], dy[k]$, $k=0,1,\dots,7$



马一共有8种跳法，通过编码“数字化”，使之“可计算”

k	0	1	2	3	4	5	6	7
dx	1	2	2	1	-1	-2	-2	-1
dy	-2	-1	1	2	2	1	-1	-2

dx----马跳一步在x方向上的增量

dy----马跳一步在y方向上的增量

k----- 方向号

从(x, y)马跳一步到(x1, y1)

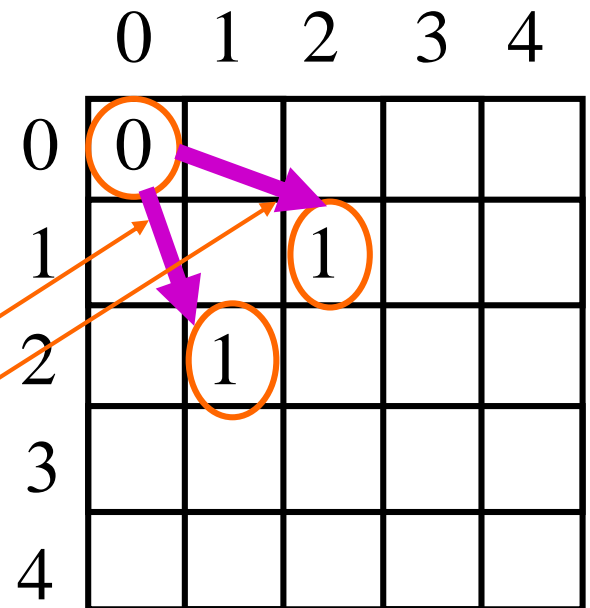
$$x1=x+dx[k];$$

$$y1=y+dy[k];$$

如马的初始位置在(0,0)则

$$x1=0+dx[k]$$

$$y1=0+dy[k], \quad k=0,1\dots7$$



k	0	1	2	3	4	5	6	7
x1	1	2	2	1	-1	-2	-2	-1
y1	-2	-1	1	2	2	1	-1	-2

8种跳法中的第2和第3种，分别到达图中标识为1的格子

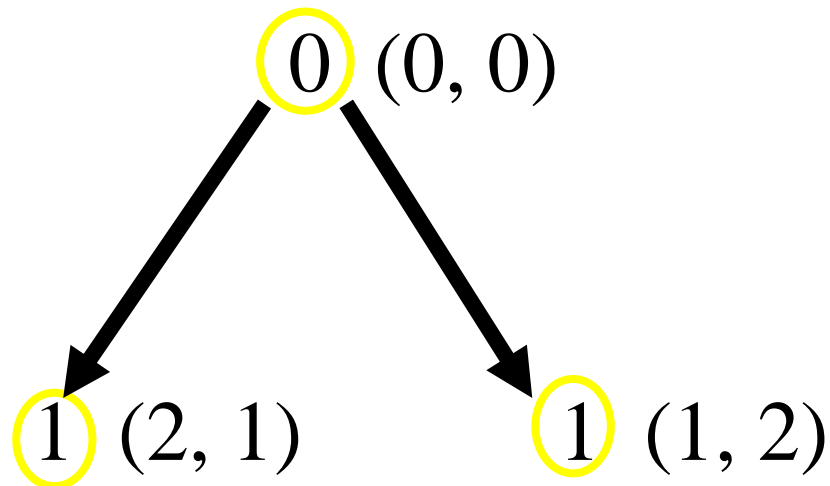
马从 $(0, 0)$ 跳一步，有两个可行位置

➤ 第2种跳法，跳至 $(2, 1)$ ；

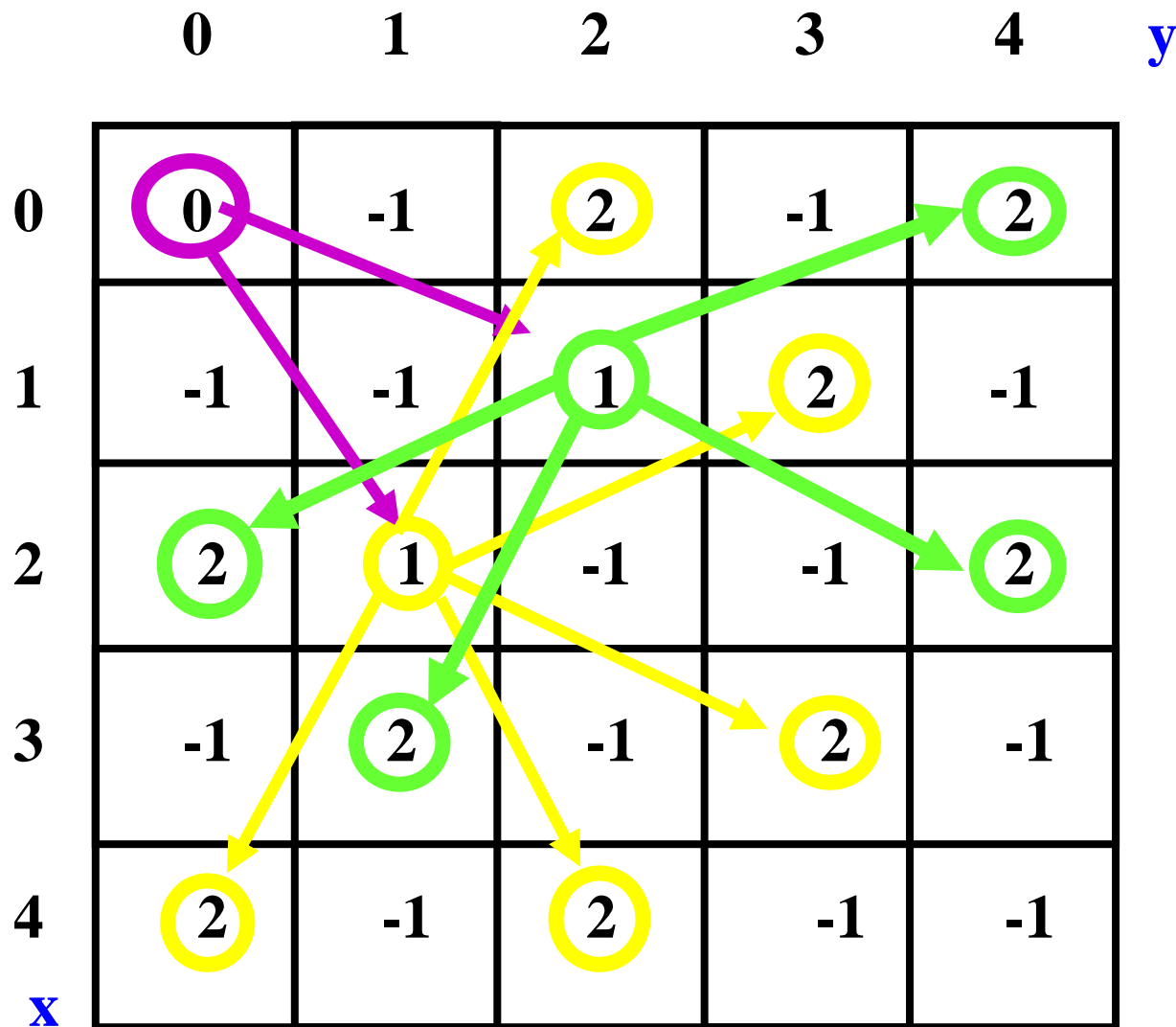
➤ 第3种跳法，跳至 $(1, 2)$ 。

称 $(0, 0)$ 为结点， $(2, 1)$ 和 $(1, 2)$ 为由 $(0, 0)$ 扩展出的结点。

示意图如下：



马由 $(0, 0)$ 跳一步可以到 $(2, 1)$ 和 $(1, 2)$ 。如果继续再跳一步，则可以到如下位置： $(4, 0)$ $(4, 2)$ $(3, 3)$ $(1, 3)$
 $(0, 2)$ $(2, 0)$ $(3, 1)$ $(2, 4)$ $(0, 4)$



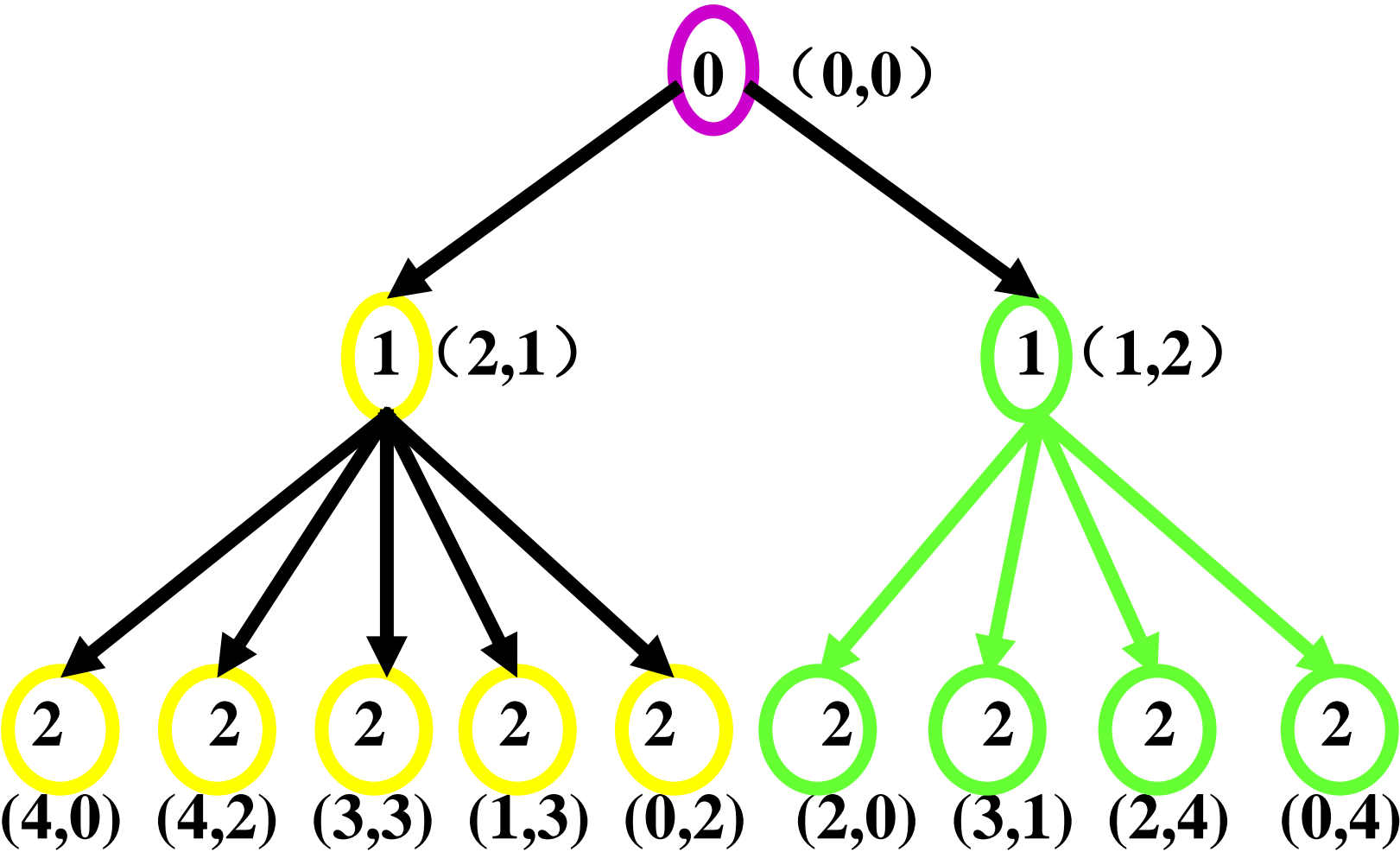
跳2步

马从 $(0, 0)$ 到 $(2, 1)$ 和 $(1, 2)$ ，再到
 $(4, 0)$ $(4, 2)$ $(3, 3)$ $(1, 3)$ $(0, 2)$ $(2, 0)$
 $(3, 1)$ $(2, 4)$ $(0, 4)$

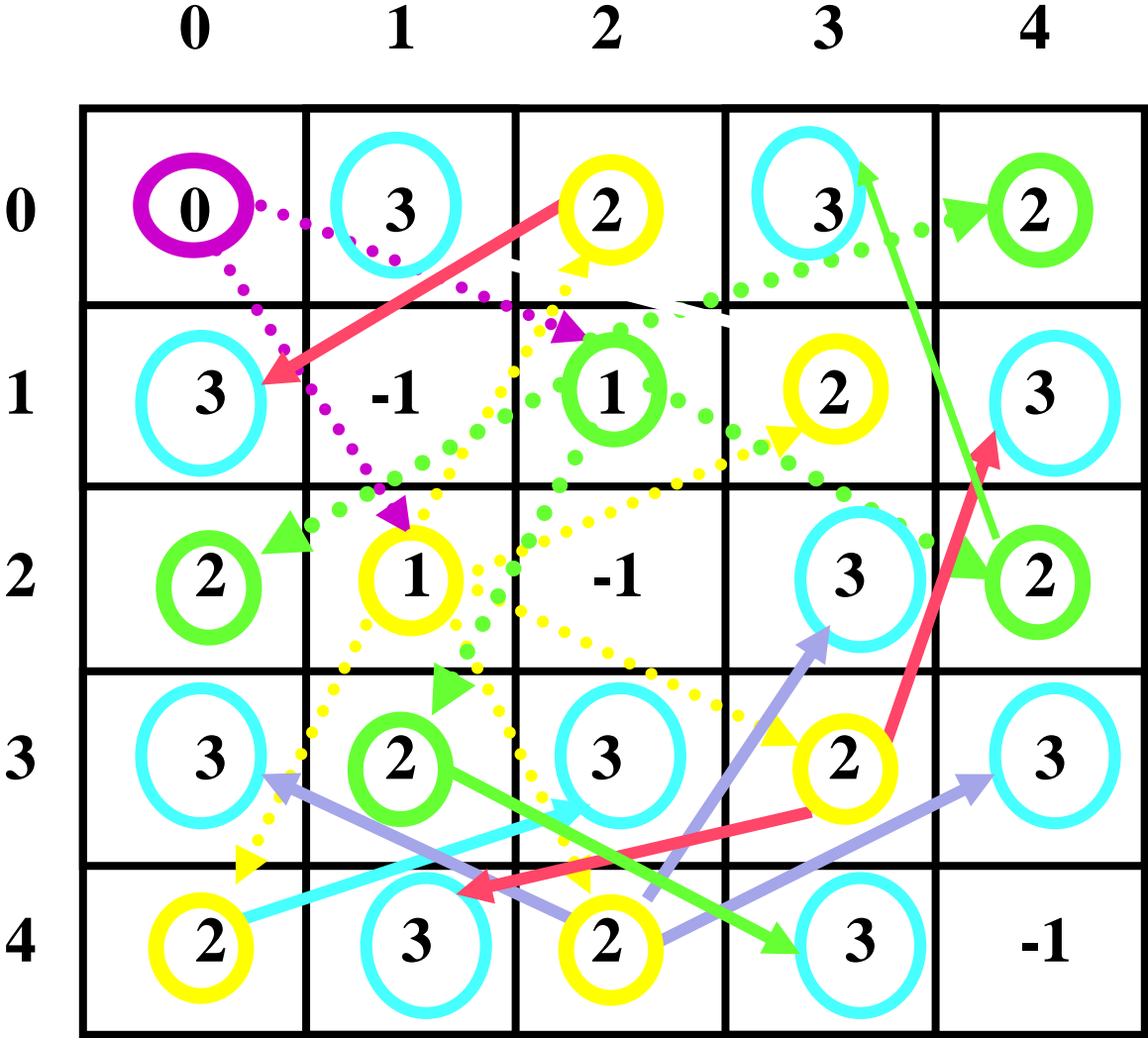
这个过程也可用如下的示意图来表示（见下页）。其中，

- 标有2的黄色的5个结点是由结点 $(2, 1)$ 扩展出来的；
- 标有2的绿色的4个结点是由结点 $(1, 2)$ 扩展出来的。

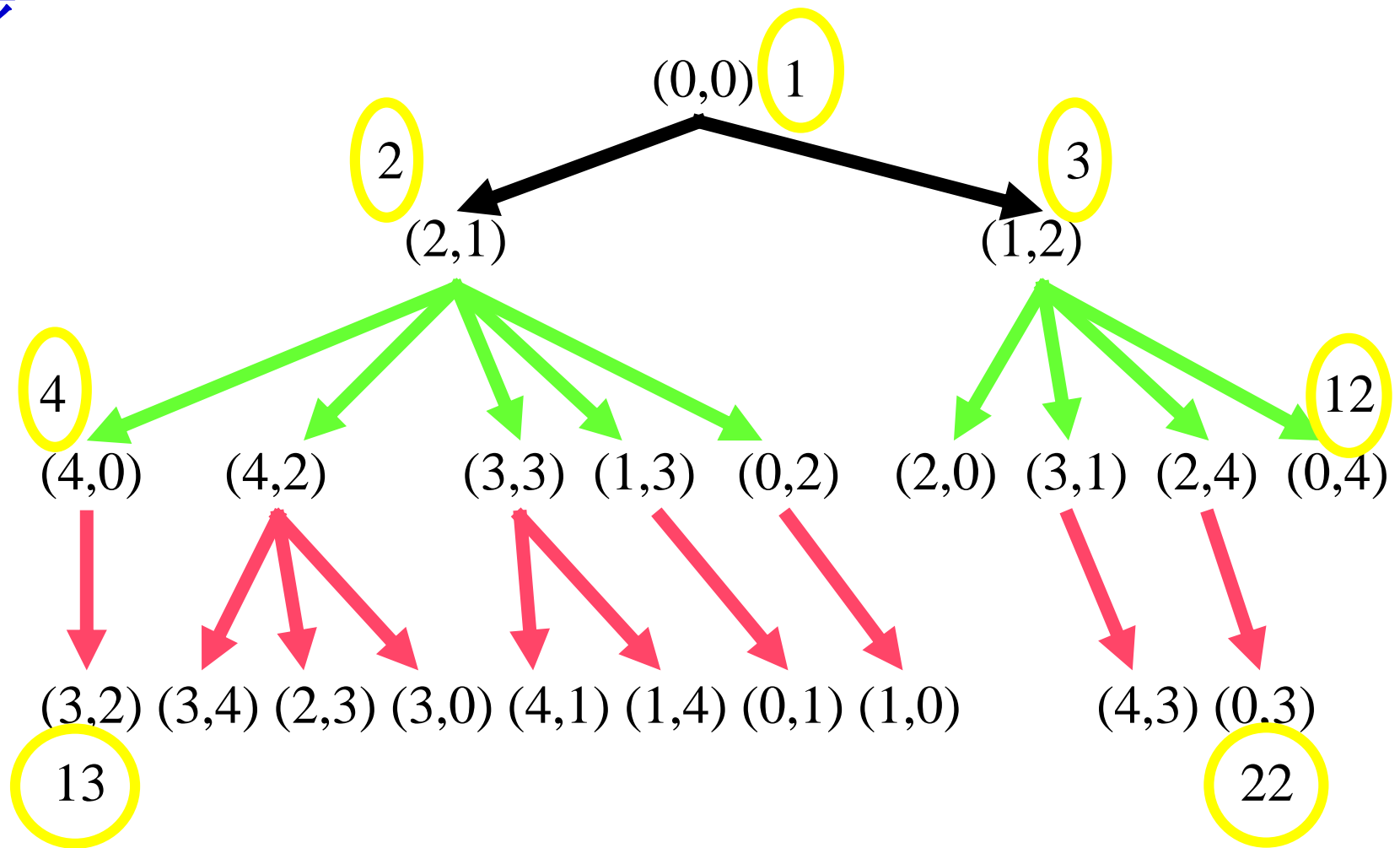
跳2步



跳3步

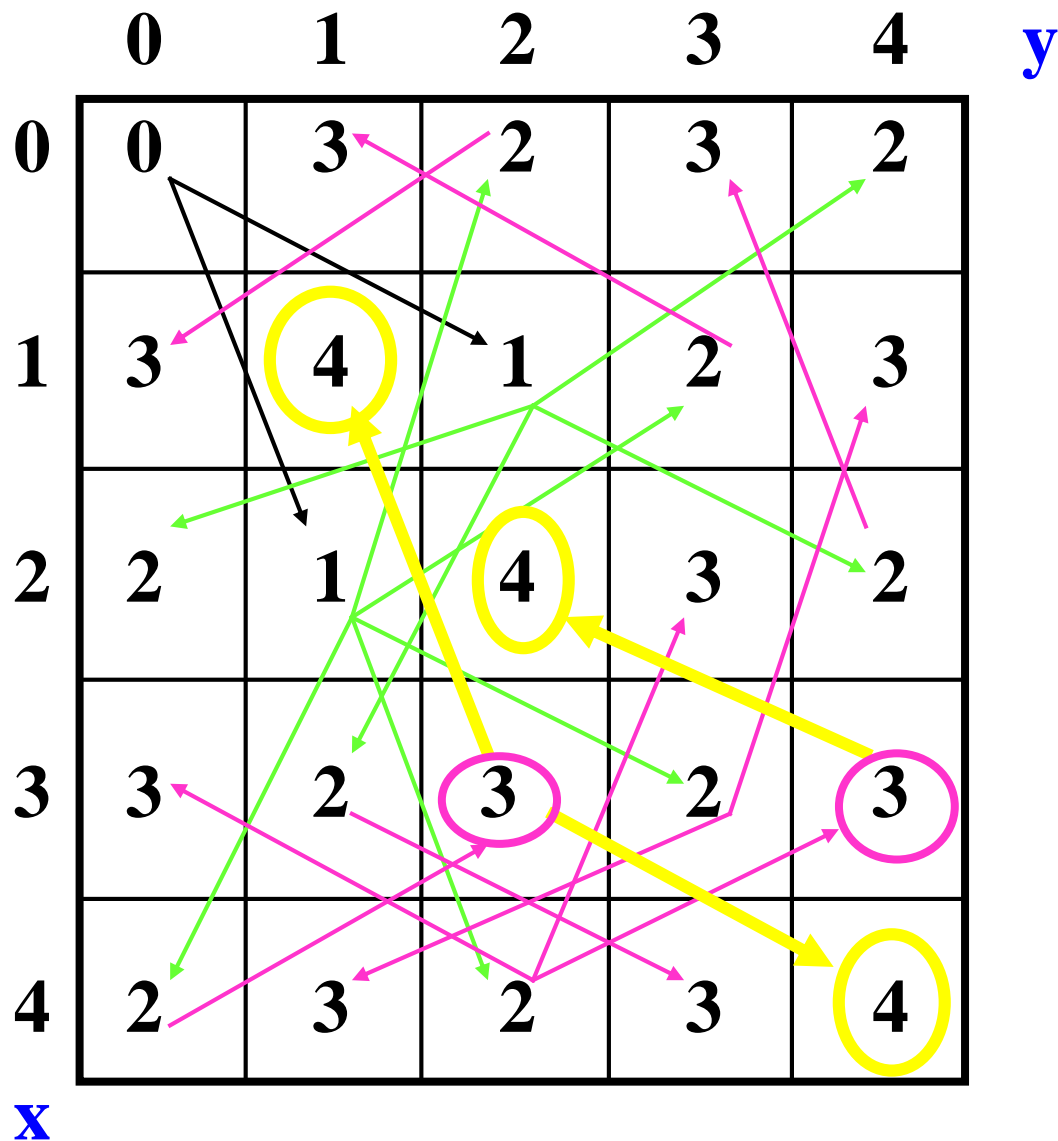


跳3步



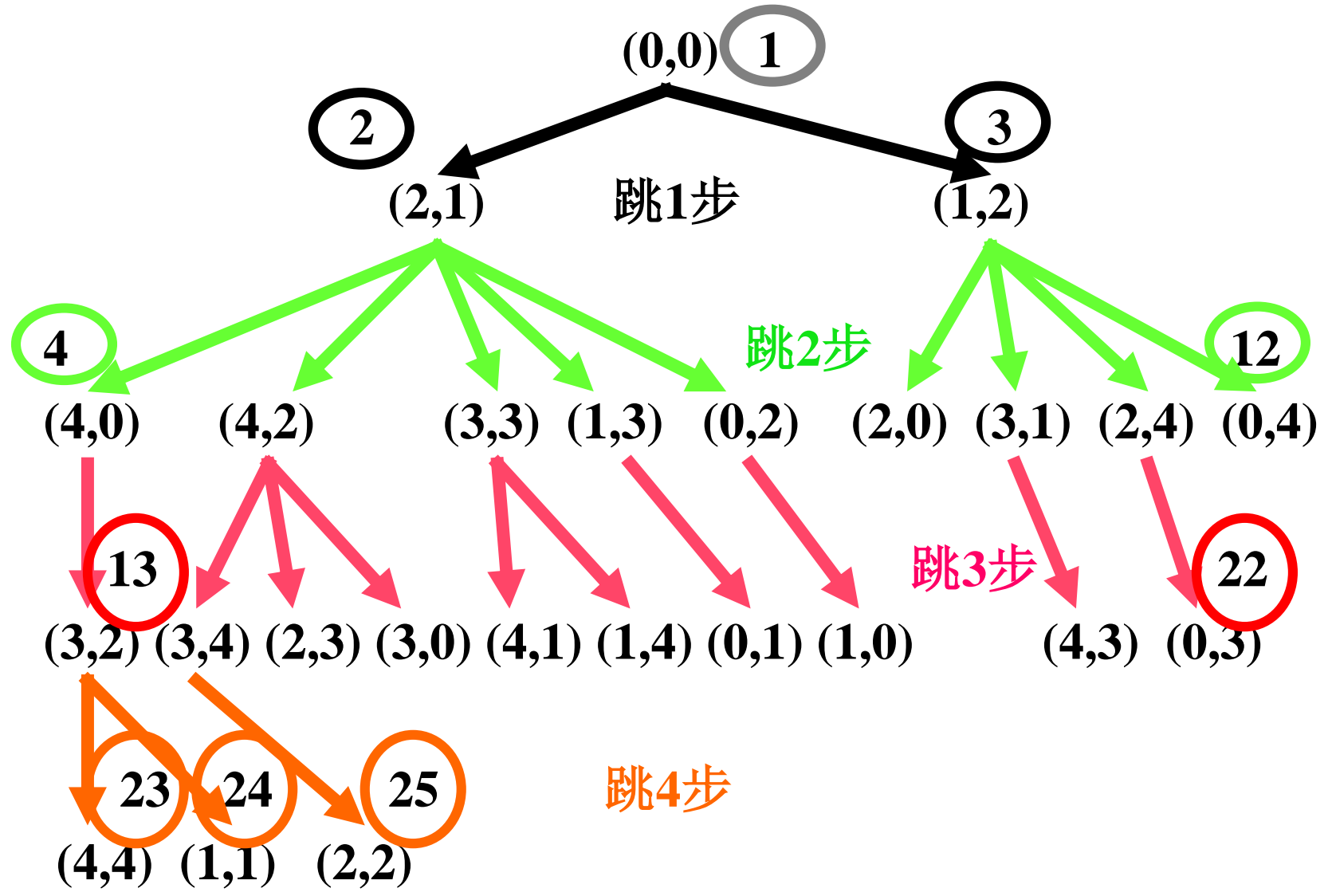
圆圈中的数字为该结点在队列中的序号
(其他结点为保持版面整洁省略了没有标出)

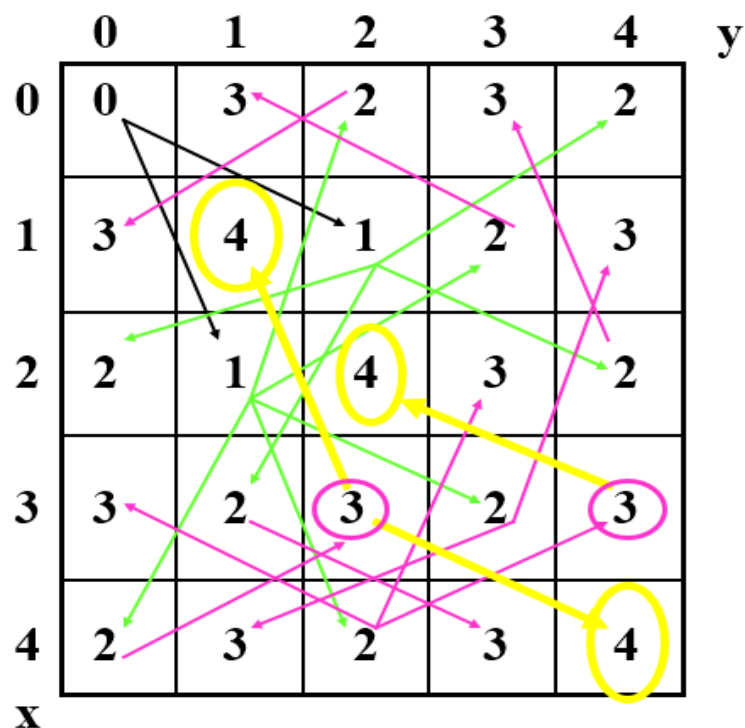
跳4步



至此，将所有格子均跳到过了

从1步到4步的另一种解法

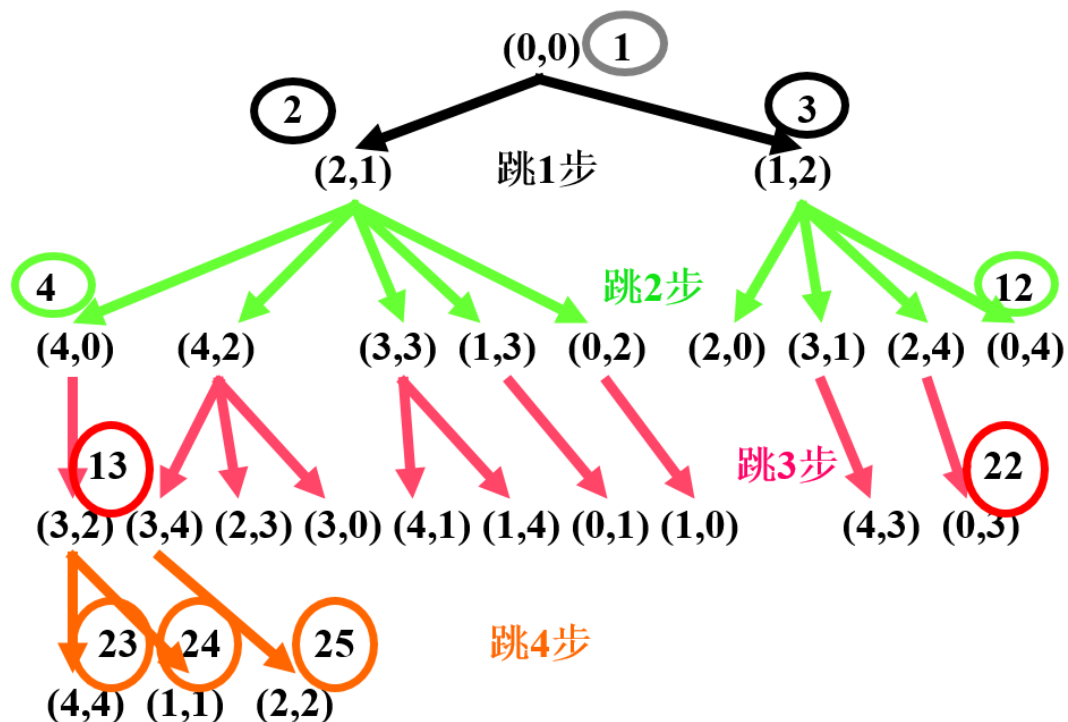




“小学水平”

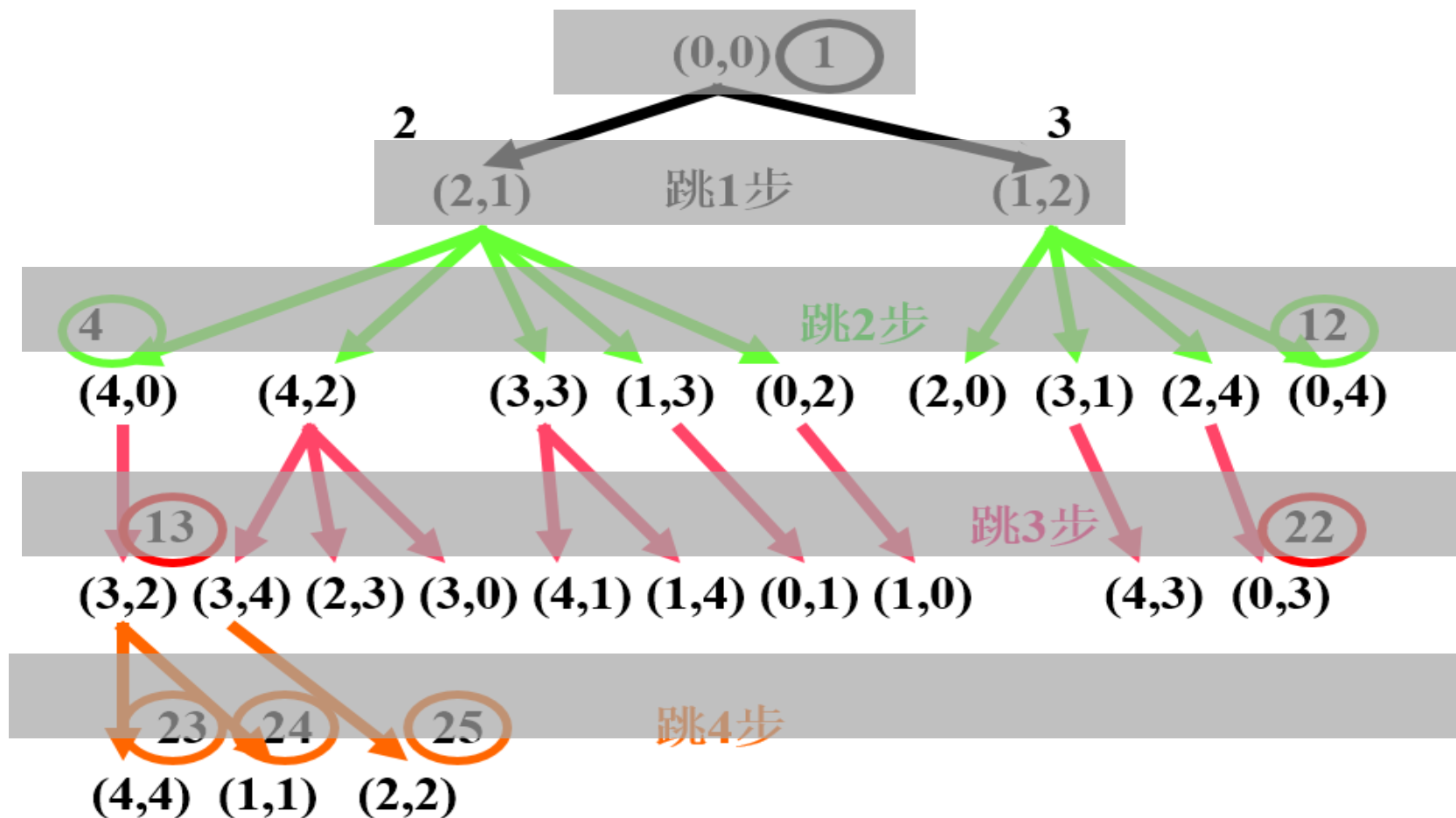
比较这两种“求解”方法

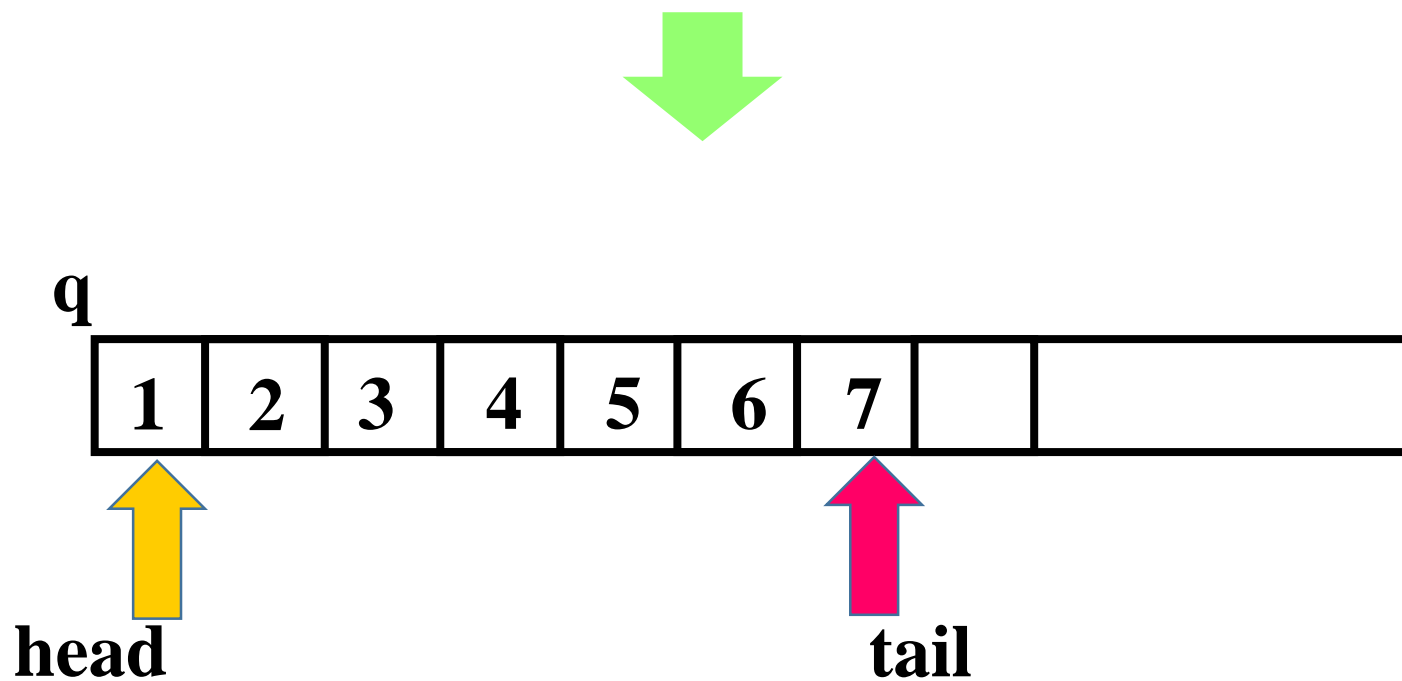
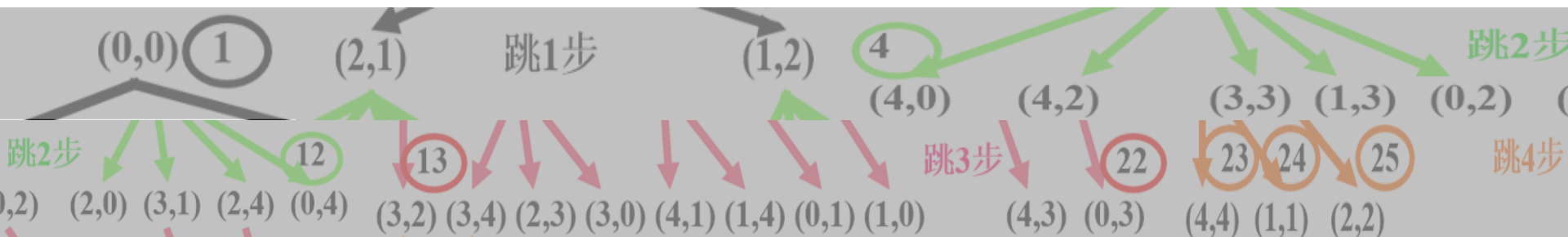
“中学水平”



“大学水平”

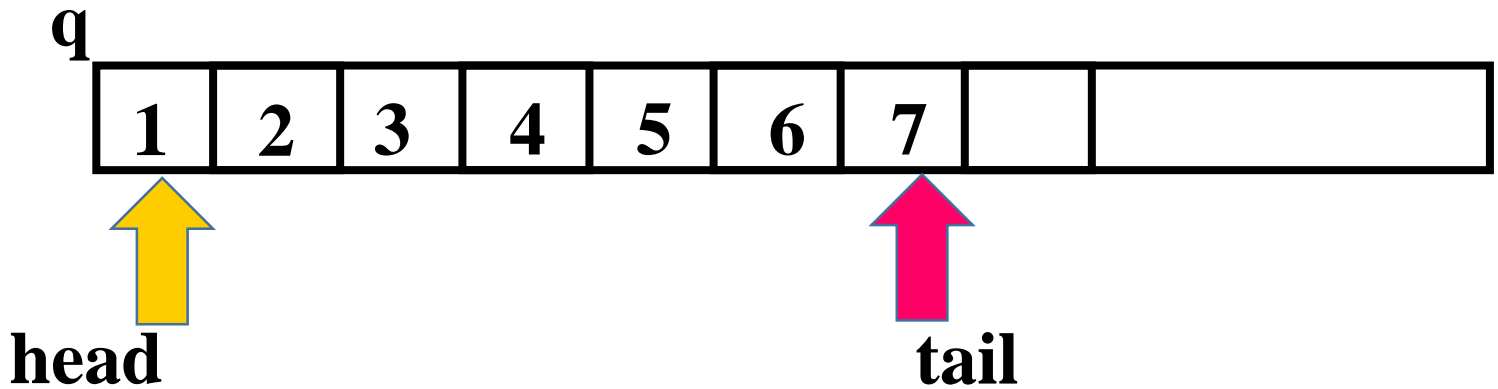
怎么让计算机来求解（“计算”）马的跳步扩展过程呢？



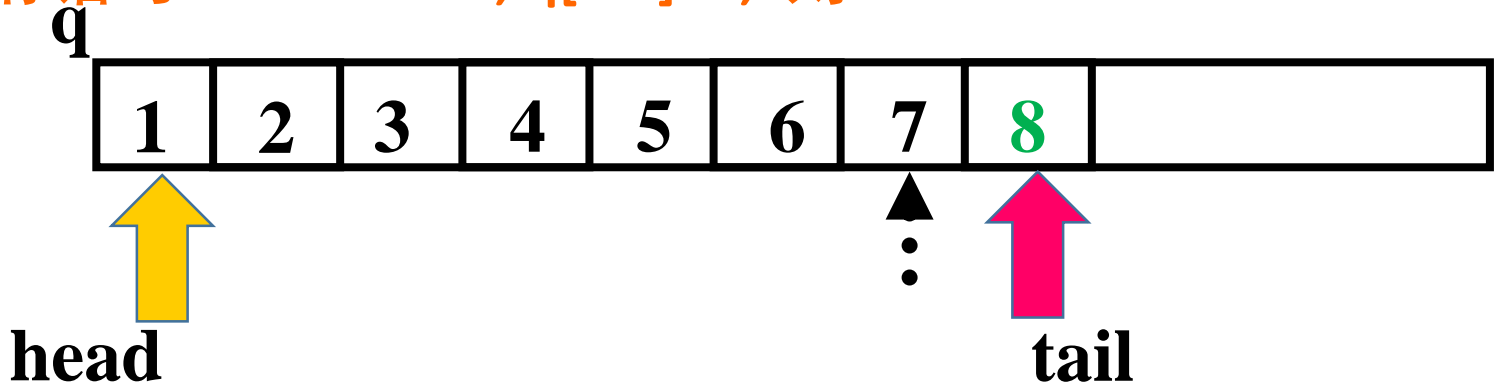


“大学水平”

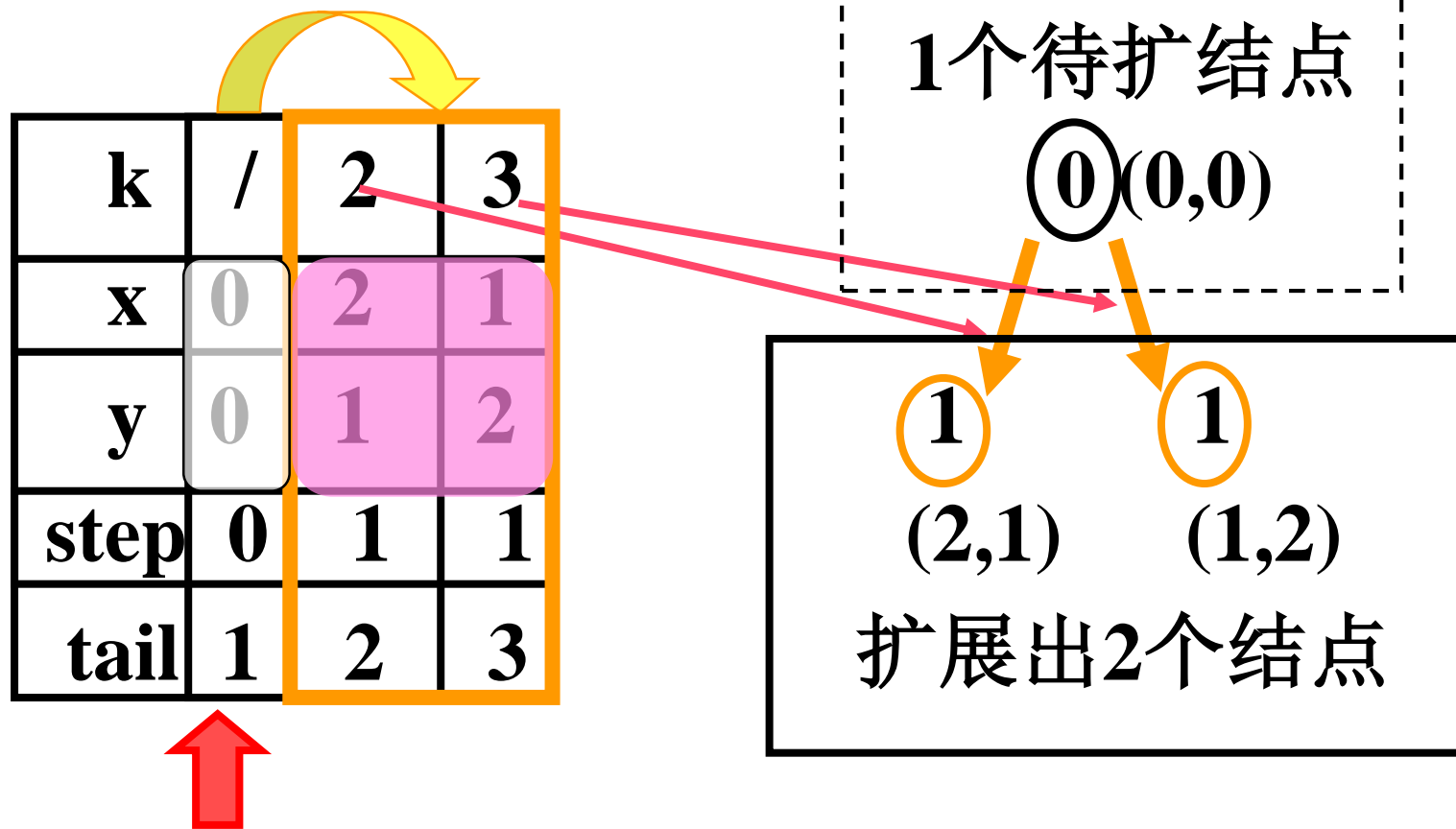
队列可用数组来实现，在队列操作时要设两个变量来记录队头和队尾。如：用**head**表示队头，**tail**表示队尾。在下图示例中，这两个变量都是数组q的下标值。



若有语句： **$\text{tail}=\text{tail}+1; q[\text{tail}]=8;$** 则：



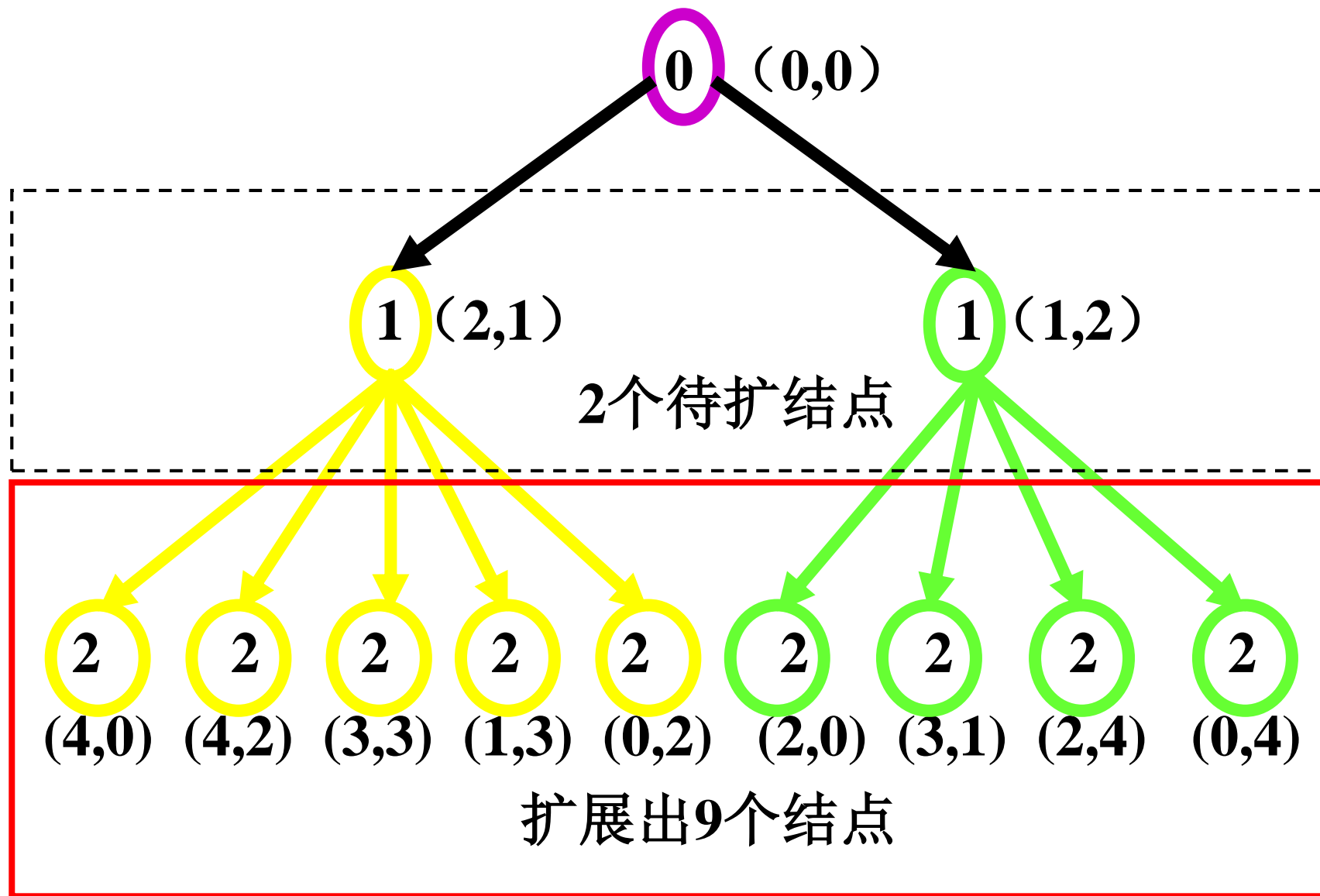
“大学水平”



head=1, cur_tail=1

对head到cur_tail区间内的结点进行扩展

第2步的扩展过程如何让计算机来表示呢？



核心问题：每种跳法及其落点记录到哪里？ —— 数组！

说明：变量k表示跳法编号

k	/	2	3								
x	0	2	1								
y	0	1	2								
step	0	1	1								
tail	1	2	3								



两种可能性（两种跳法）

head=2, cur_tail=3

对 head 到 cur_tail 区间内的结点进行扩展

head=3

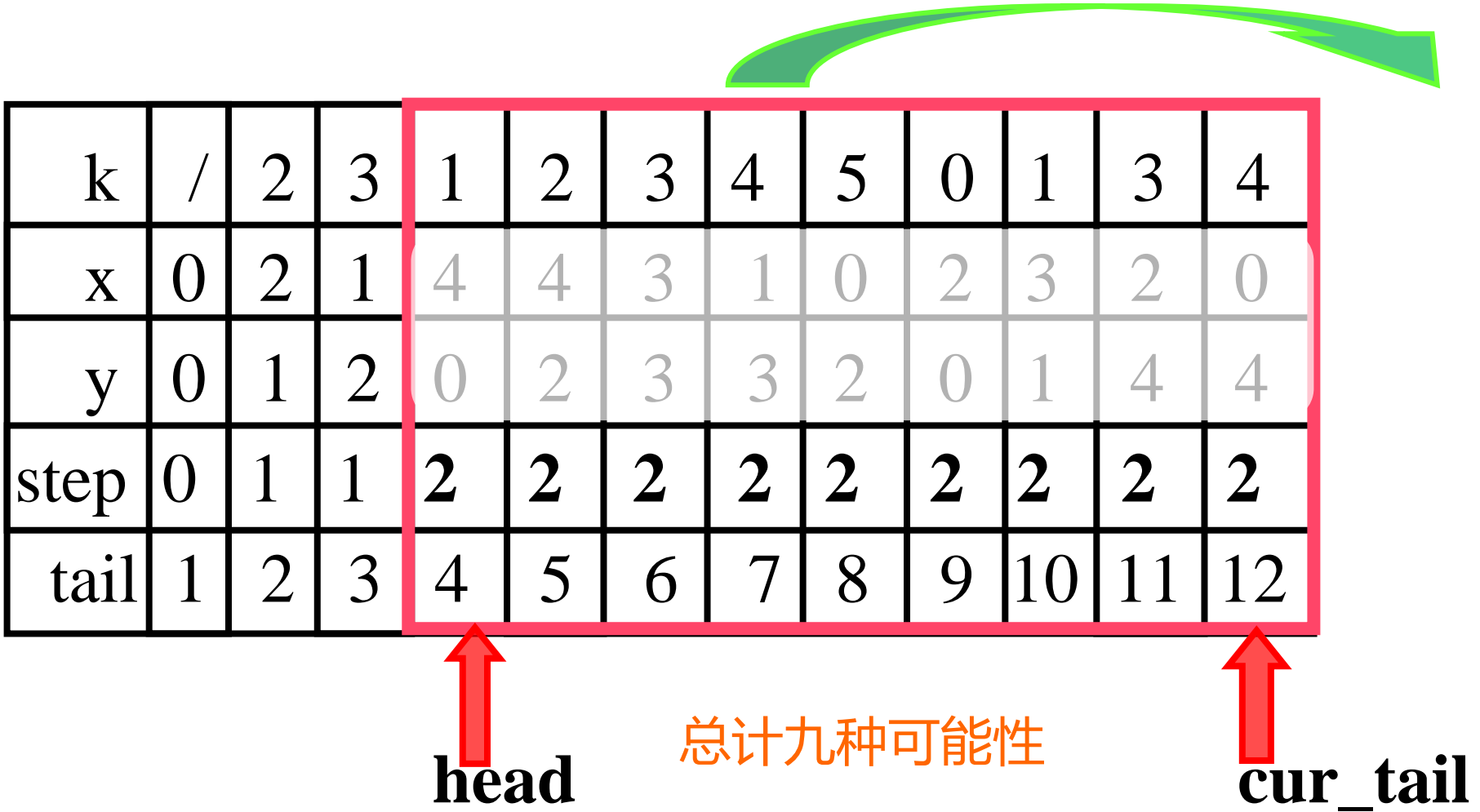
四种可能性

k	/	2	3	1	2	3	4	5	0	1	3	4
x	0	2	1	4	4	3	1	0	2	3	2	0
y	0	1	2	0	2	3	3	2	0	1	4	4
step	0	1	1	2	2	2	2	2	2	2	2	2
tail	1	2	3	4	5	6	7	8	9	10	11	12

五种可能性

head=2, cur_tail=3

依此类推，可进行第3步跳步扩展



k	/	2	3	1	2	3	4	5	0	1	3	4
x	0	2	1	4	4	3	1	0	2	3	2	0
y	0	1	2	0	2	3	3	2	0	1	4	4
step	0	1	1	2	2	2	2	2	2	2	2	2
tail	1	2	3	4	5	6	7	8	9	10	11	12

head

总计九种可能性

cur_tail

对 head 到 cur_tail 区间内的结点进行扩展

骑士跳步的扩展过程

骑士的初始位置（队头） 结点1

由结点1 扩展

第1步跳到结点2， 结点3

由结点2， 3 扩展

第2步跳到结点4,结点5,...结点12

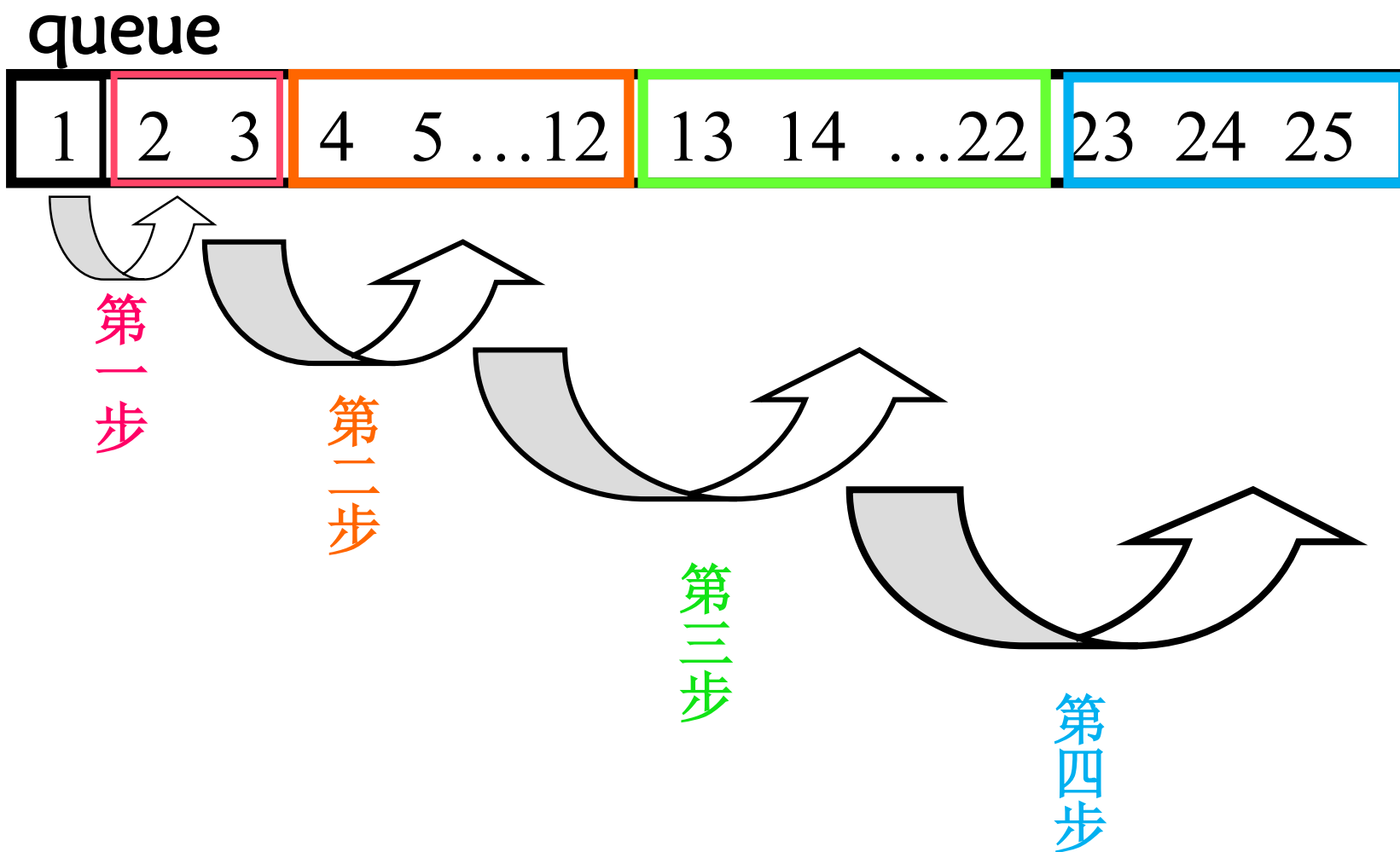
由结点4,5...12 扩展

第3步跳到结点13,结点14,...结点22

由结点13,14,...22 扩展

第4步跳到结点23， 结点24， 结点25

根据上图可构造如下队列
以序号来表示结点

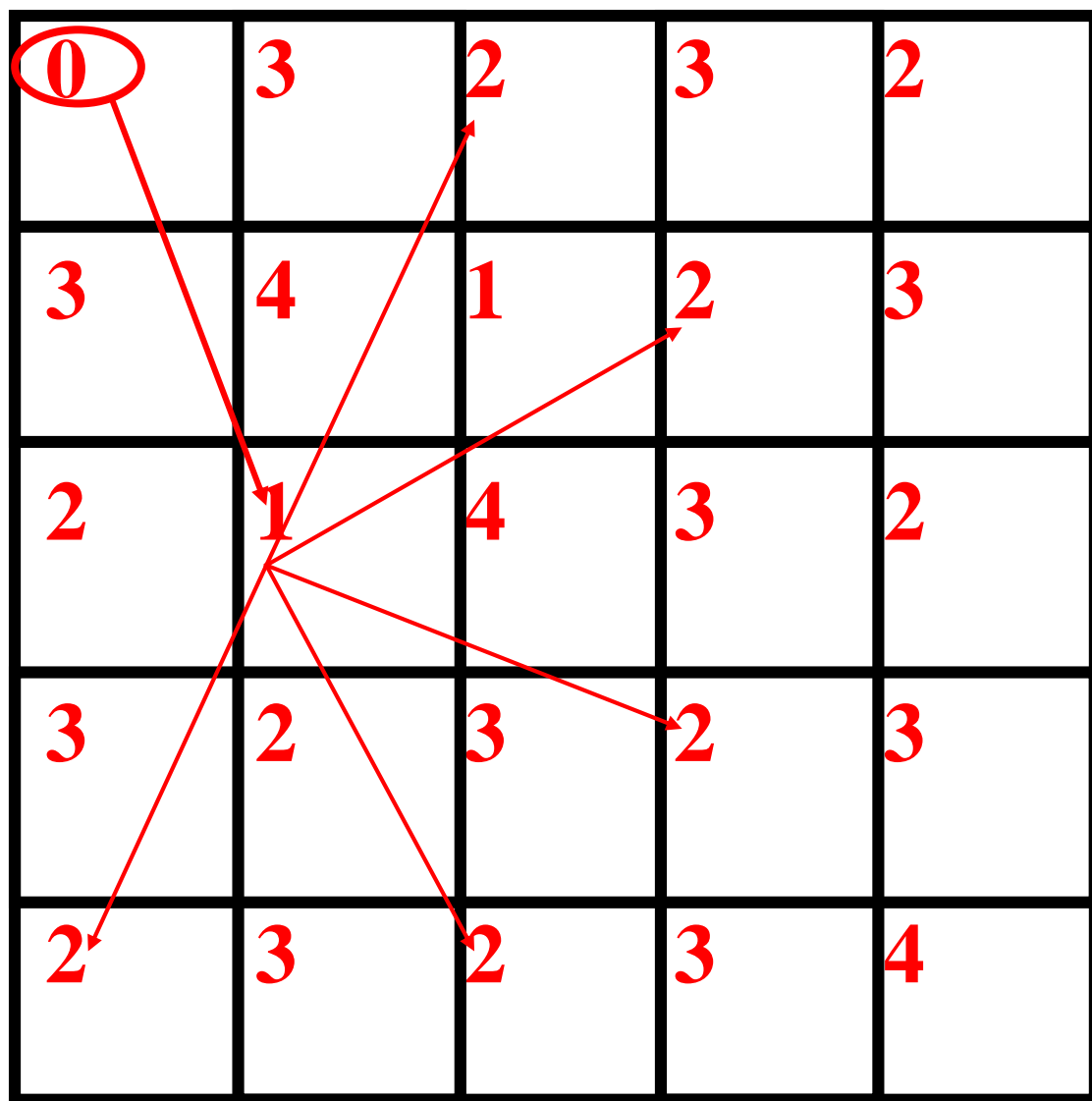


从图看出

- 1 队列中有25个结点，每个结点对应棋盘中的 25个格子；
- 2 队列的序号，表示结点扩展过程中的先后顺序；
- 3 扩展是分层的，处于同一层的结点从左至右依次扩展，这种策略被称为“宽度优先”；
- 4 标记同层元素的左右位置是编程的关键

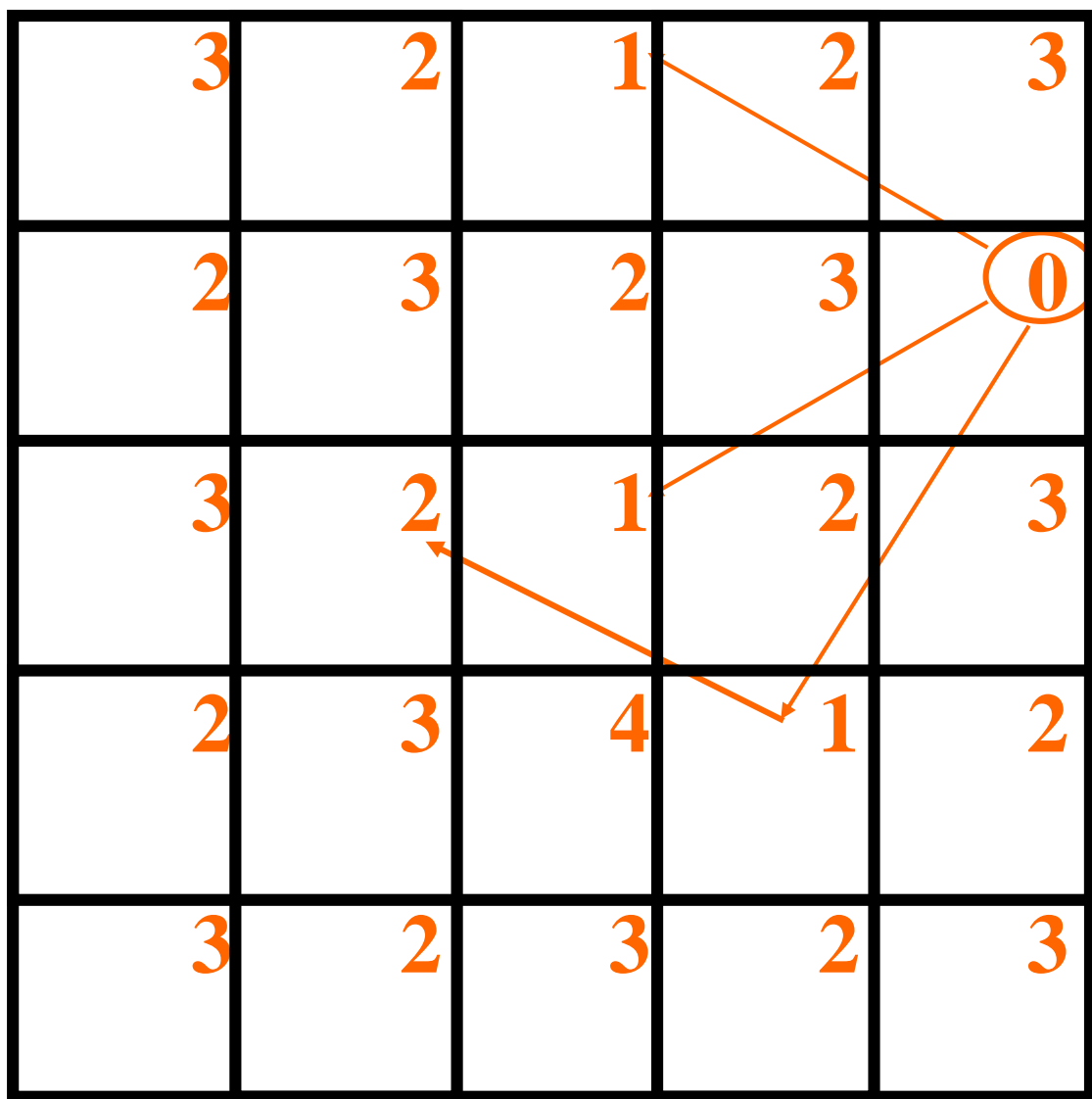
在完成了前面的子任务后，对于 5×5 的棋盘上的“4骑士聚会”问题，所有骑士到达各个位置的时间（步数）就都计算出来了。剩下的问题是：如何确定他们聚会的“场所”。

先来看看这些骑士到达各处的跳步信息表：



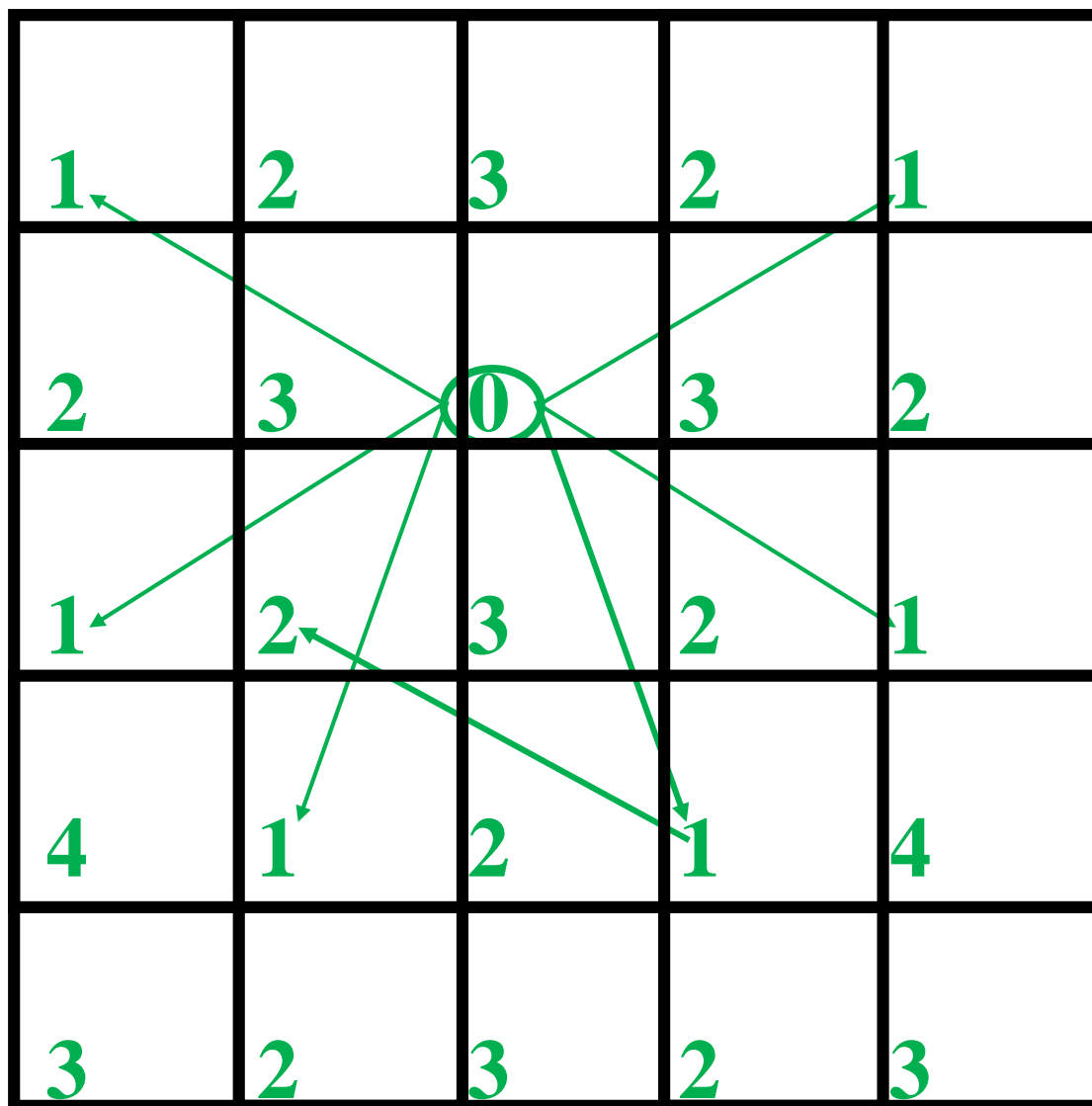
0号骑士的
跳步图

数字放在
左上角



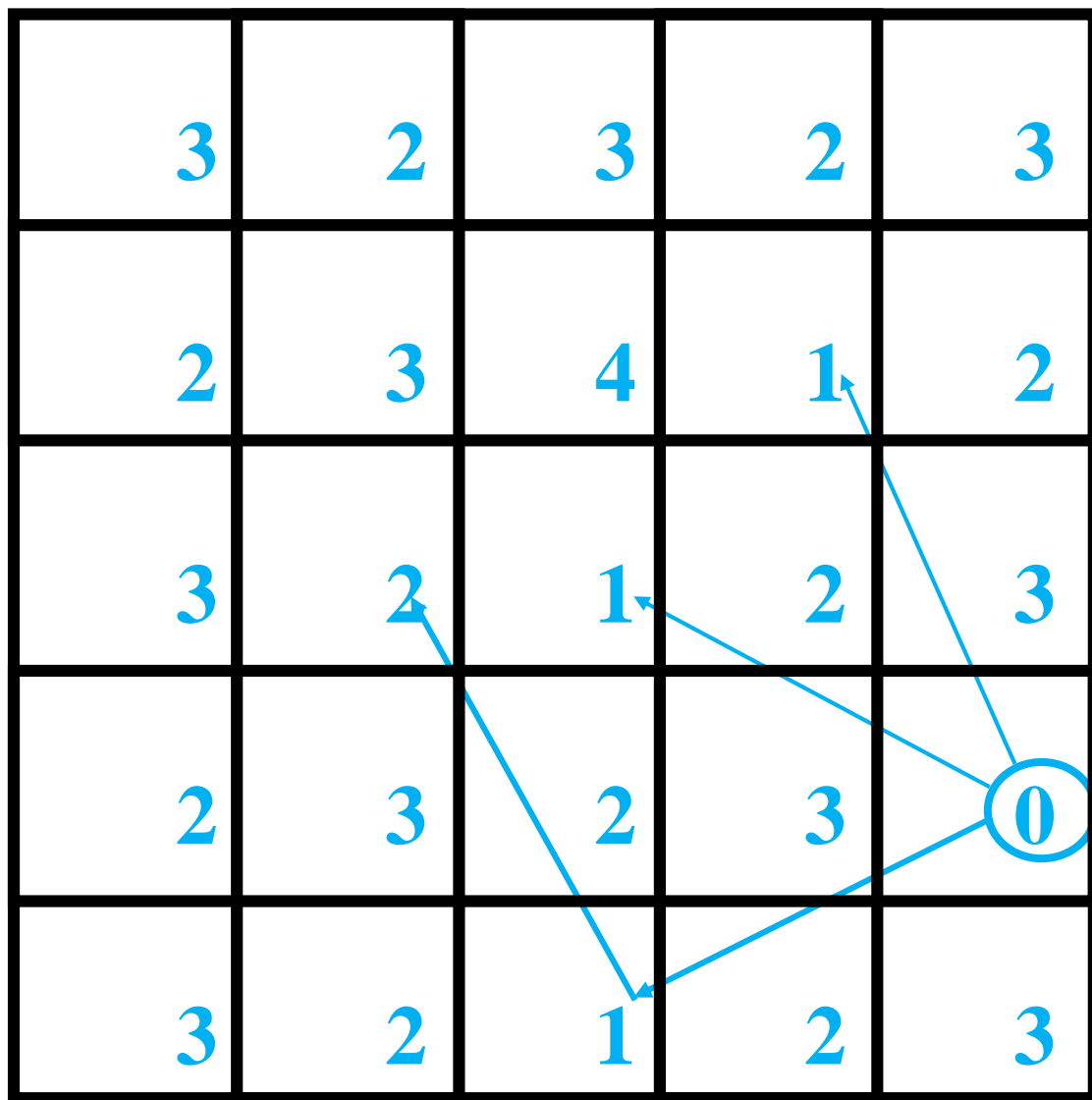
1号骑士的
跳步图

数字放在
右上角



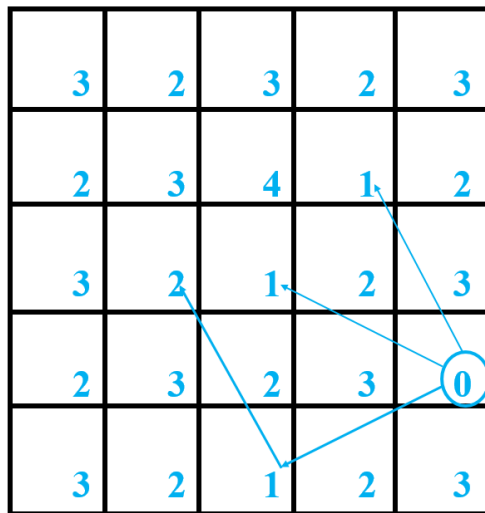
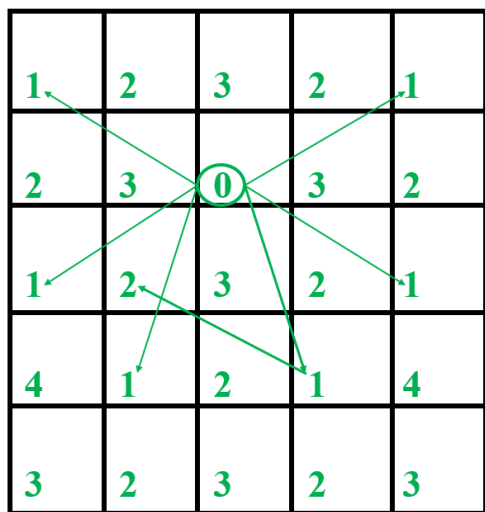
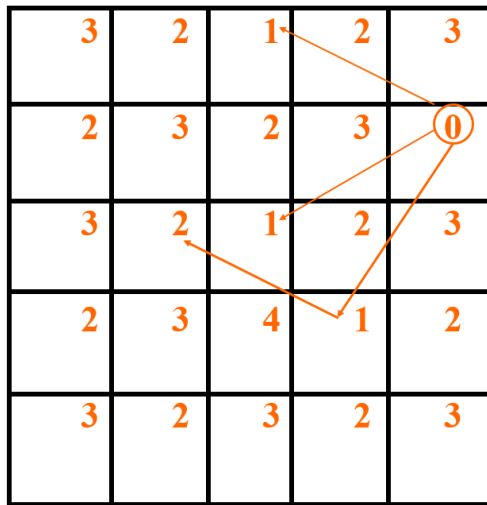
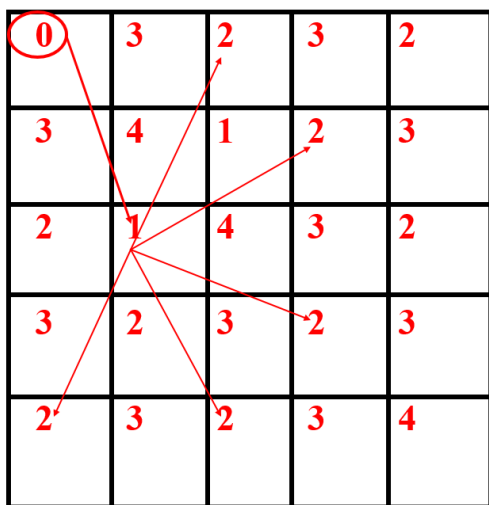
2号骑士的
跳步图

数字放在
左下角



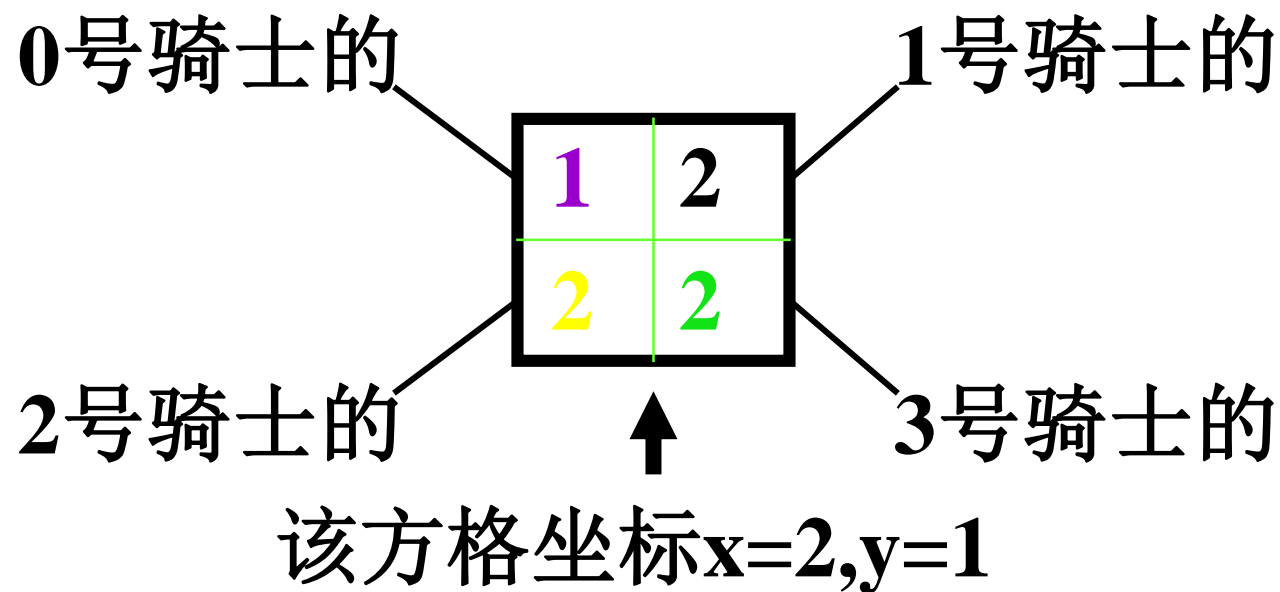
3号骑士的
跳步图

数字放在
右下角



如果从每个骑士角度
考虑问题，则计算聚
会地点变得十分复杂！
但是，若注意到他们
都“共用”同一张
“地图”，则……

发挥你的想象力，把4张图画在透明胶片上，然后把4张摞在一起，这时你可以看到每一个格子里有4个数据，分别属于4个骑士的跳步信息



0	3	3	2	2	1	3	2	2	3
1	3	2	2	3	3	2	2	1	3
3	2	4	3	1	2	2	3	3	0
2	2	3	3	0	4	3	1	2	2
2	3	1	2	4	1	3	2	2	3
1	3	2	2	3	1	2	2	1	3
3	2	2	3	3	4	2	1	3	2
4	2	1	3	2	2	1	3	4	0
2	3	3	2	2	3	3	2	4	3
3	3	2	2	3	1	2	2	3	3

4 张
胶片叠放
每个格子
有 4 个数

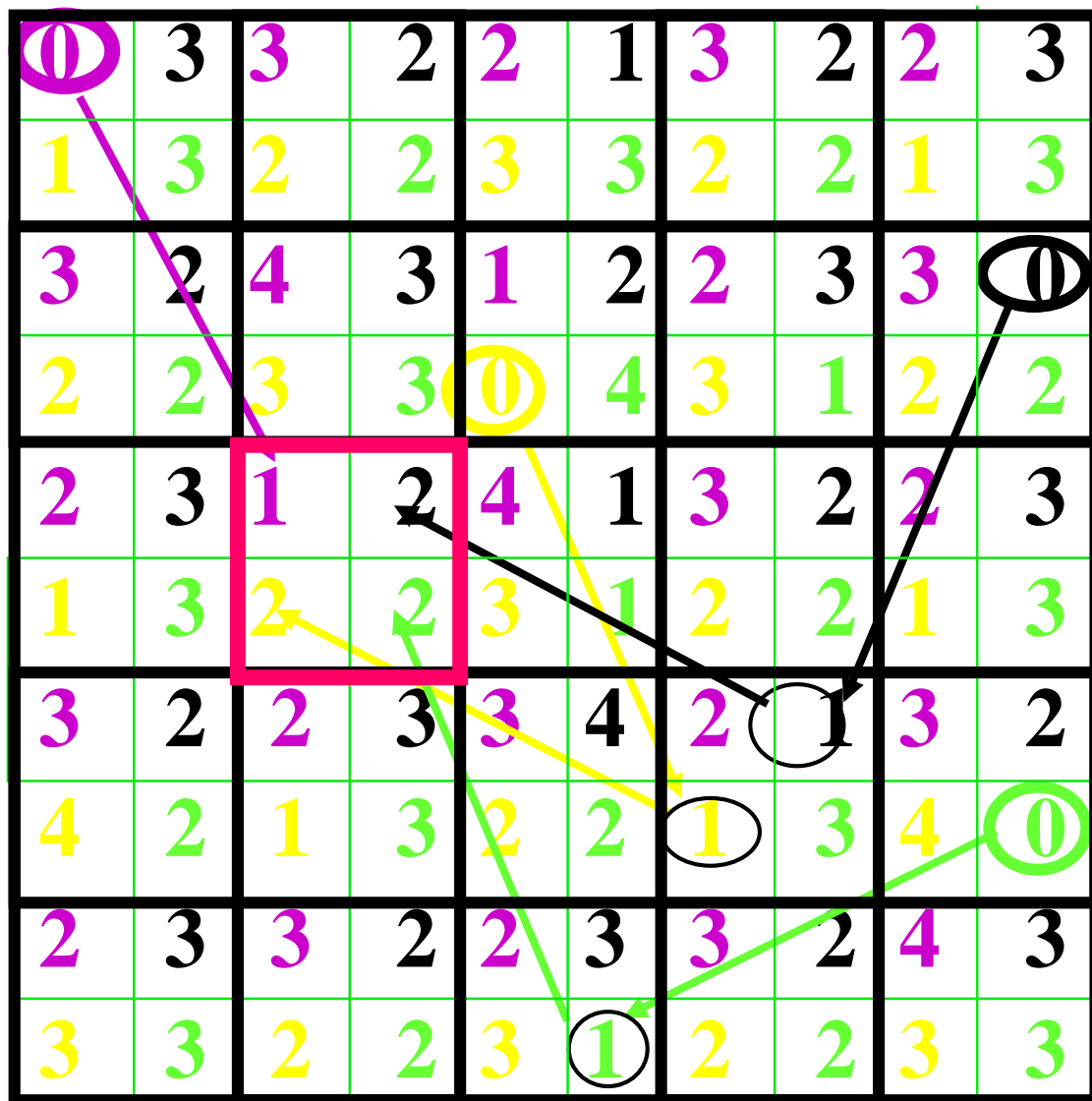
不是骑士说“我要到哪里去”，而是地图说“谁要到我这里来”

依次对每个格子进行观察：

格子里4个数中最大的那个数，就是在该位置聚会所需天数，这是该位置的标志性数字。

按照题目要求寻找尽早聚会位置，即标志性数字最小的格子。有可能这种格子不止一个，这时再比较4个骑士跳到这些位置上的总步数。最早聚会且4个人总步数也最少的位置即为所求。

下图位置(2,1)是最佳聚会位置：
用2天时间，4人共跳7步。



最佳聚会点
(2, 1)

搜索骑士最佳聚会位置的算法流程

- 1 初始化棋盘，输入n个骑士的初始位置
- 2 定义函数BFS ()，使用宽度优先策略，对每一个骑士扩展马的跳步信息
- 3 计算在位置(x,y)上的
 - n个人中的最多跳步 $\text{good}[x][y].\text{max}$
 - n个人的跳步总和 $\text{good}[x][y].\text{sum}$
 - $x, y = 0, 1, \dots, T-1$
- 4 枚举棋盘上的每个格子寻找 $\text{good}[x][y].\text{max}$ 的最小值

$$\min = \min \{ \text{good}[x][y].\max \}$$

$$\{x, y\}$$

①

$$x, y = 0, 1, \dots, T-1$$

5 在满足1式的前提下，寻找个人跳步总和最小的格子位置

$$\text{sum} = \min \{ \text{good}[x][y].\text{sum} \}$$

$$\{x, y\}$$

②

$$x, y = 0, 1, \dots, T-1$$

$$\text{good}[x][y].\max == \min$$

找到满足公式①和②的格子位置 x, y

记录并显示 $\min, \text{sum}, (x, y)$

任务即告完成

课后思考题

如果有多处最佳聚会场所满足要求，应如何修改程序，使之能输出所有的聚会方案？