

3.12 队列的应用：农夫过河

1 1 1 1



0 0 0 0

TIPS1: 农夫只能一个人或者带一个物品坐船

TIPS2: 农夫必须和物品在同一岸才能带走

3.12 队列的应用：农夫过河

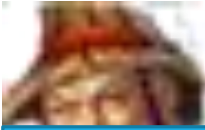


要找到可行的方案并输出，需要解决三个问题。

(1) 位置与状态

为了表示每个物品的位置，采用二进制位来区分南岸和北岸，
0表示在南岸，1表示在北岸。

用四个二进制位XXXX分别表示农夫、狼、菜和羊四个物品所在的位置。例如1110表示农夫、狼和菜在北岸，菜在南岸。

农夫过河问题的**初始状态为0000，结束状态为1111**

			
0	0	0	0
1	1	1	1

3.12 队列的应用：农夫过河

要找到可行的方案并输出，需要解决三个问题。

(2) 安全状态判断

在算法求解过程中，需要判断状态是否安全。


例如1110是安全的状态，而1000是不安全的状态。

分析所有的状态，其中不安全的状态有两个：

一个是羊和菜在同一岸但是农夫没有和它们在一起，

一个是狼和羊在同一岸但是农夫没有和它们在一起。

其他的都是安全状态。

			
0	0	0	0
1	1	1	1

3.12 队列的应用：农夫过河

要找到可行的方案并输出，需要解决三个问题。

(3) 中间状态记录

为了方便记录中间的状态，设置一维数据`status[16]`。

初始时`status[j] = -1 (j = 0, ..., 15)`。

如果状态`j`由状态`i`得到，则更新`status[j] = i`。

这样当到达结束状态时，可以通过`path[]`数组反向回推到初始状态0000，中间的状态则是农夫过河问题的一个解。

这里同队列解决迷宫类似。因为在结束状态时队列中并没有保留经过的中间状态。

			
0	0	0	0
1	1	1	1

3.12 队列的应用：农夫过河

(4) 其他细节问题

如何表示状态翻转？

0 异或 1 = 1

1 异或 1 = 0

`newstatus = nowstatus ^ (0x08 | movers);` //计算新状态

如何表示农夫带某个物品走呢？

0x08 或 0x01 农夫带着羊

0x08 或 0x02 农夫带着菜

0x08 或 0x04 农夫带着狼

0x08 或 0x08 农夫自己走

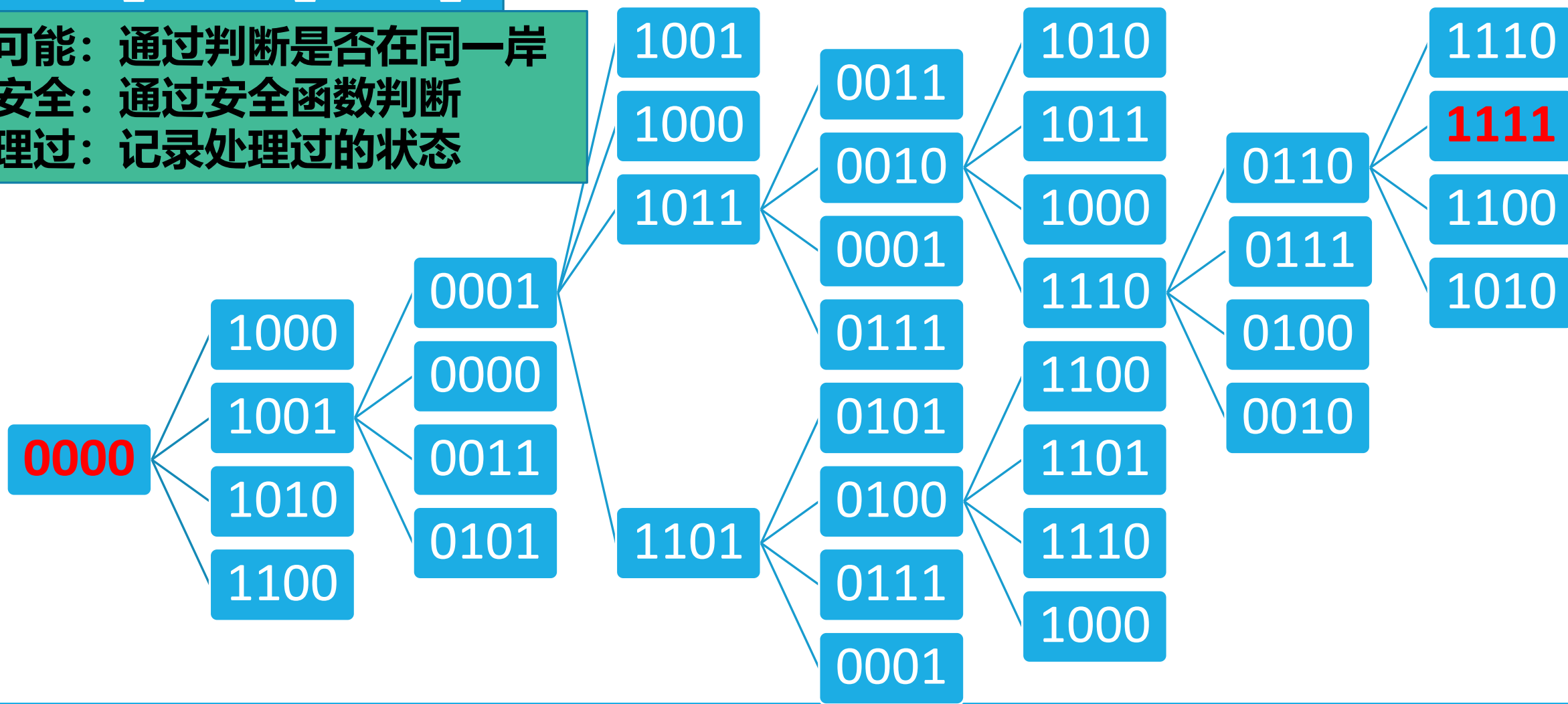
			
0	0	0	0
1	1	1	1



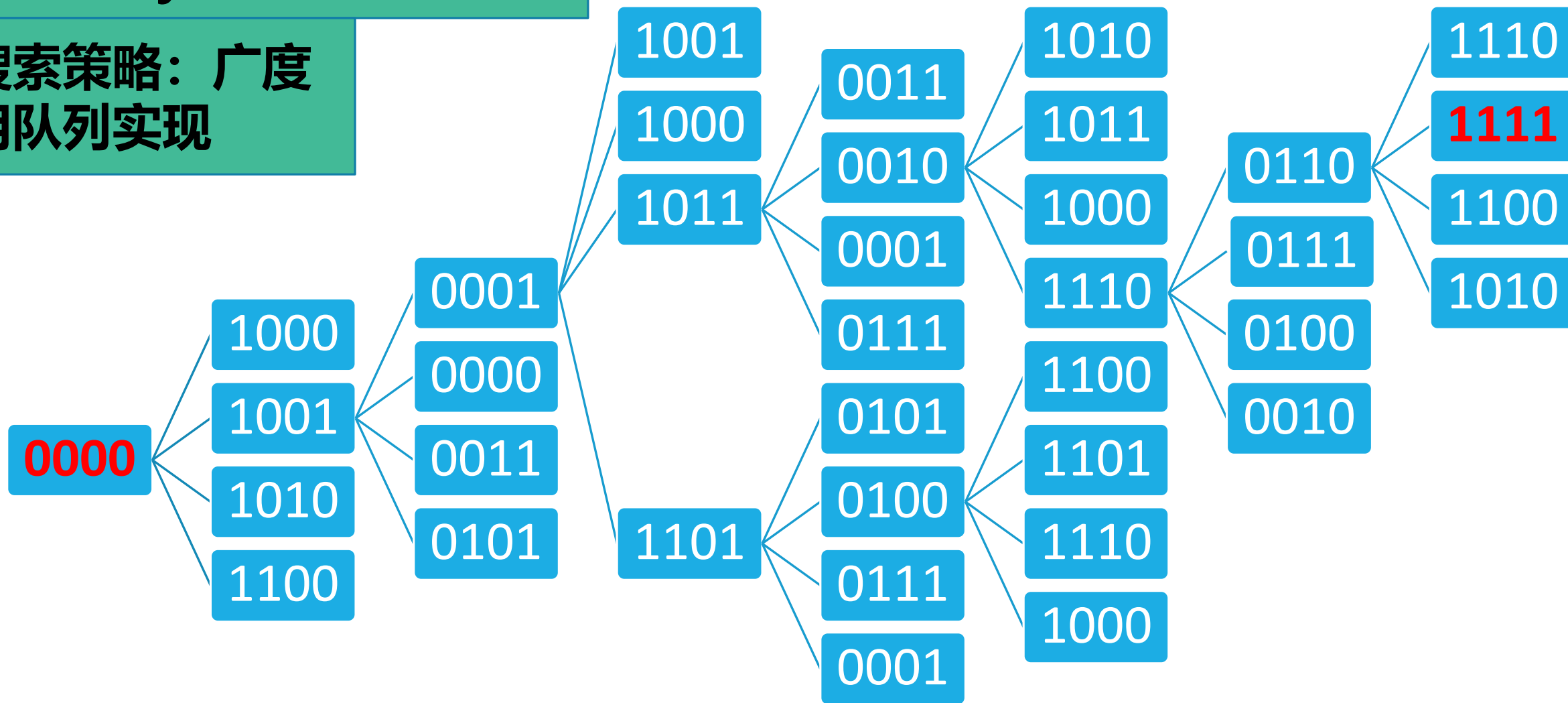
0	0	0	0
1	1	1	1

1不可能：通过判断是否在同一岸
2不安全：通过安全函数判断
3处理过：记录处理过的状态

TIPS1:农夫只能一个人或者带一个物品坐船
TIPS2: 农夫必须和物品在同一岸才能带走



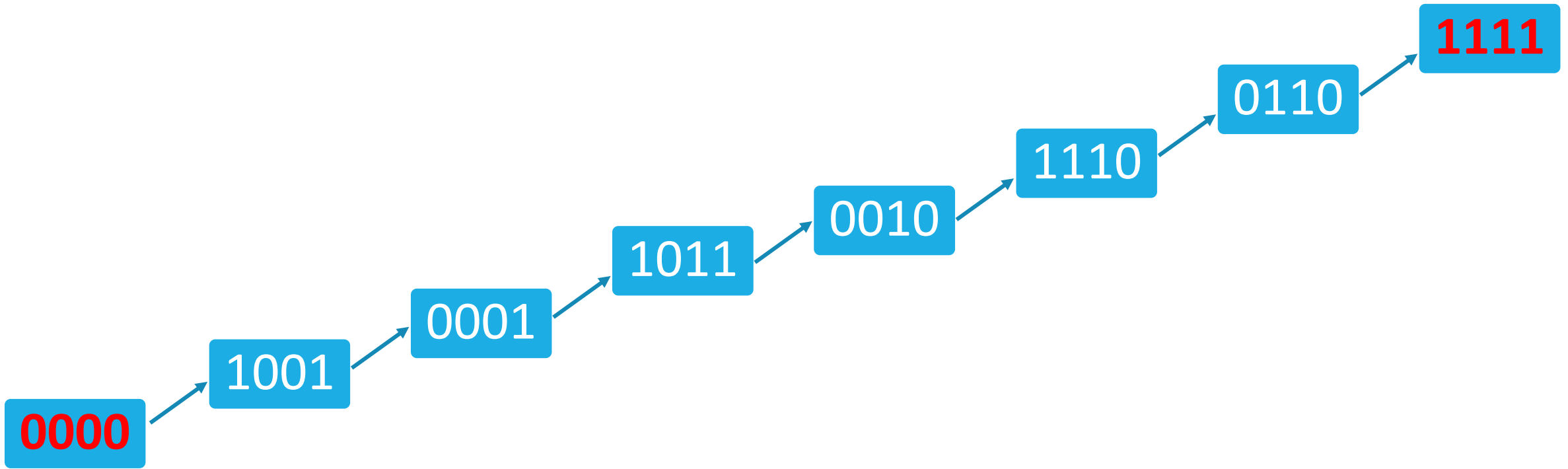
搜索策略：广度用队列实现



3.12 队列的应用：农夫过河

具体实现见算法3-30

- 初始状态0000入队
- 当队列不空且没有到达结束状态1111时，循环以下操作：
 - 队头状态出队
 - 按照农夫一个人走、农夫分别带上三个物品走，循环以下操作：
 - 农夫和物品如果在同一岸，则计算新的状态
 - 如果新状态是安全的并且是没有处理过的，则更新状态数组，并将新状态入队
- 当状态为1111时，逆向输出状态数组



思考：如果物品的位置如下，写出算法的状态变化

