实验课安排

3，4周，周一晚7点-10点 逸夫楼901

5，6周，周二晚7点-10点 逸夫楼901

第二章 状态空间搜索问题

2.1搜索问题

2.2状态空间表示法

2.3无信息搜索

2.4启发式搜索

2.5本章小结

2.1搜索问题

* 问题提出

人工智能要解决的问题是非结构化问题；

难以获得求解所需的全部信息；

更没有现成的算法可供求解使用。

* 应用

搜索需要解决的问题

* 知识表示（状态空间表示）
* 搜索策略（如何搜索，知识的使用）
* 最优搜索(如何找到最优路径)

2.2状态空间表示法

* 表示方法
  * (1)状态(State): $S_k=[S_{k1}, S_{k2}, \ldots S_{kn}]$
  * (2)操作(Operator): 操作描述了状态之间的关系
    表示: F:{f1, f2, ... fn}
  * (3)状态空间(State Space)

    三元组表示〈S, F, G〉

    其中: S初始状态集

    G: 目标状态集合

    F: 操作的集合。

状态空间图

* 可有相应的赋值有向图
* 节点表示状态，有向边表示操作
* 问题求解过程转化为在图中寻找从初始状态S出发到达目标状态G的路径问题，也就是寻找操作序列的问题

* 举例（用状态空间方法）
2阶“梵塔”问题(Tower of Hanoi Problem):
  * 有三个柱子(1，2和3)和两个不同尺寸的圆盘(A，B)。在每个圆盘的中心有个孔，所以圆盘可以堆叠在柱子上,最初,全部两个圆盘都堆在柱子1上(最大的在底部,最小的在顶部)。要求把所有 圆盘都移到另一个柱子上，搬动规则为:

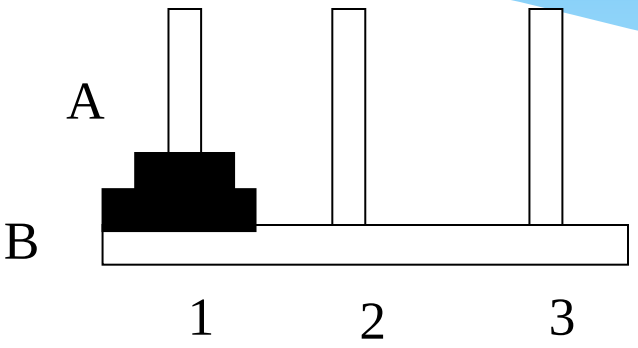
    (1)一次只能搬一个圆盘

    (2)不能将大圆盘放在小圆盘上

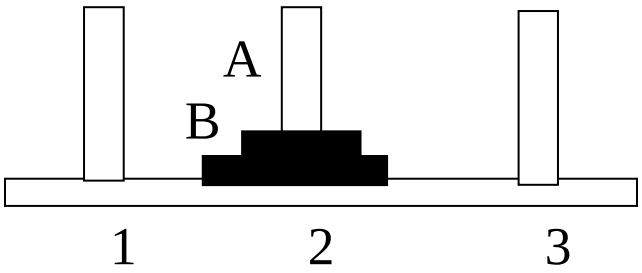
    (3)可以利用空柱子。(example,hono)

* 用状态空间方法来描述问题:
* 状态的表示
  * 柱的编号用i,j来代表
    S(i,j)表示问题的状态其中:i代表A(小）所在的柱子, j 代表B（大）所在的柱子
  * 状态集合（可能的）
    s0=(1,1), s1=(1,2), s2=(1,3)
    s3=(2,1), s4=(2,2), s5=(2,3)
    s6=(3,1), s7=(3,2), s8=(3,3)

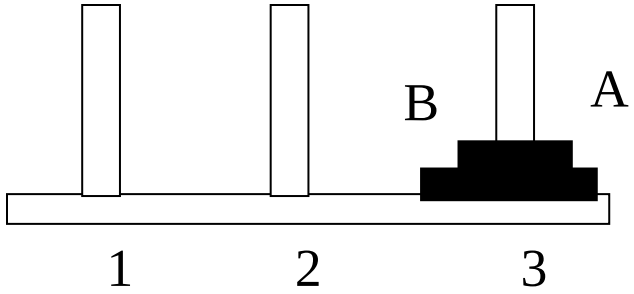
* 初始状态S={s0}，目标状态G={s4,s8}



$S_0=(1,1)$



$S_4=(2,2)$

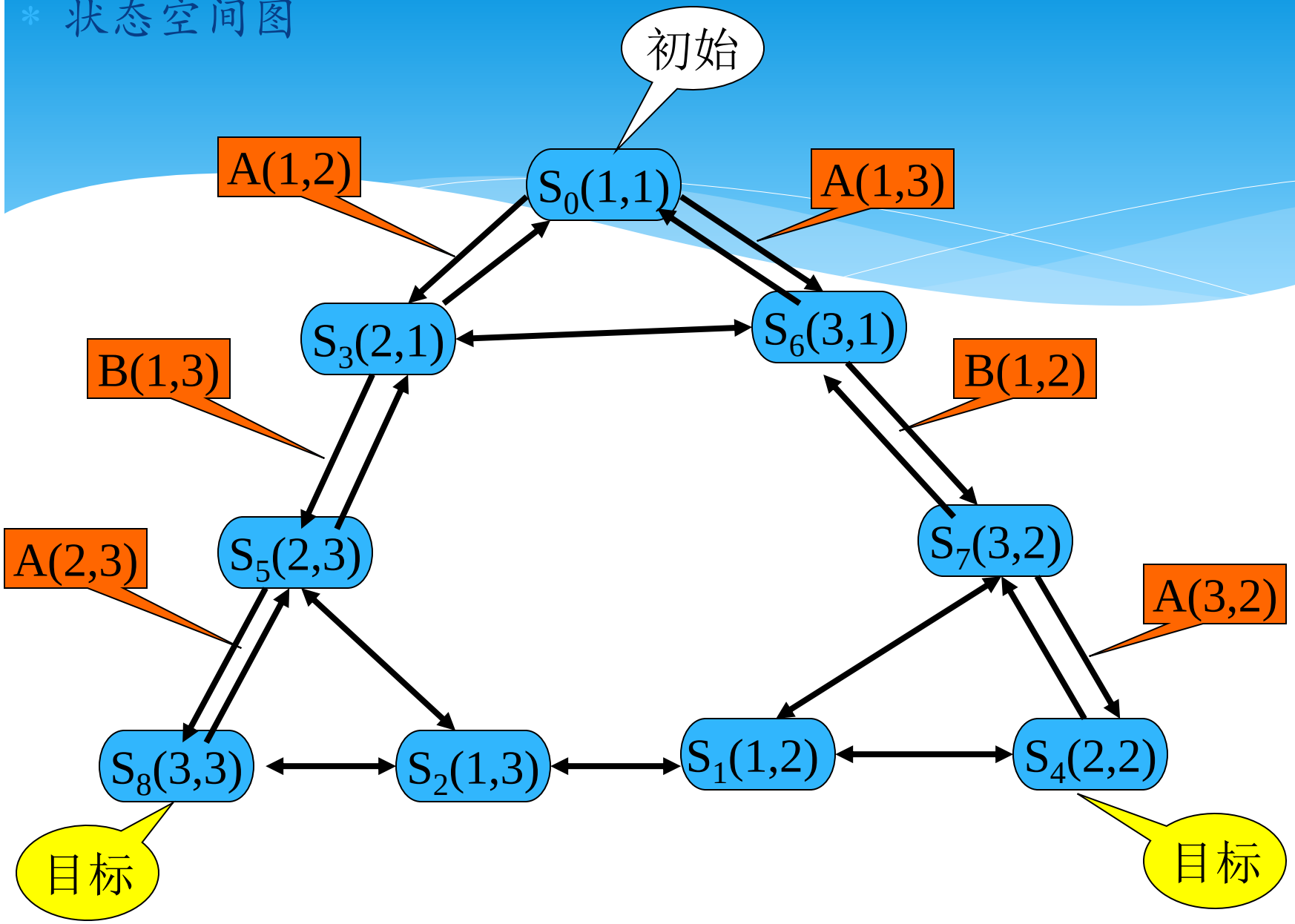


$S_8=(3,3)$

* 操作（算符）
  * 定义操作A(i,j)，B(i,j)
  * 操作集合(12种操作）：
    A(1,2),A(1,3),A(2,1),A(2,3),A(3,1),A(3,2)
    B(1,2),B(1,3),B(2,1),B(2,3),B(3,1),B(3,2)
  * 约束：

不能将大圆盘（B）放在小圆盘（A）上

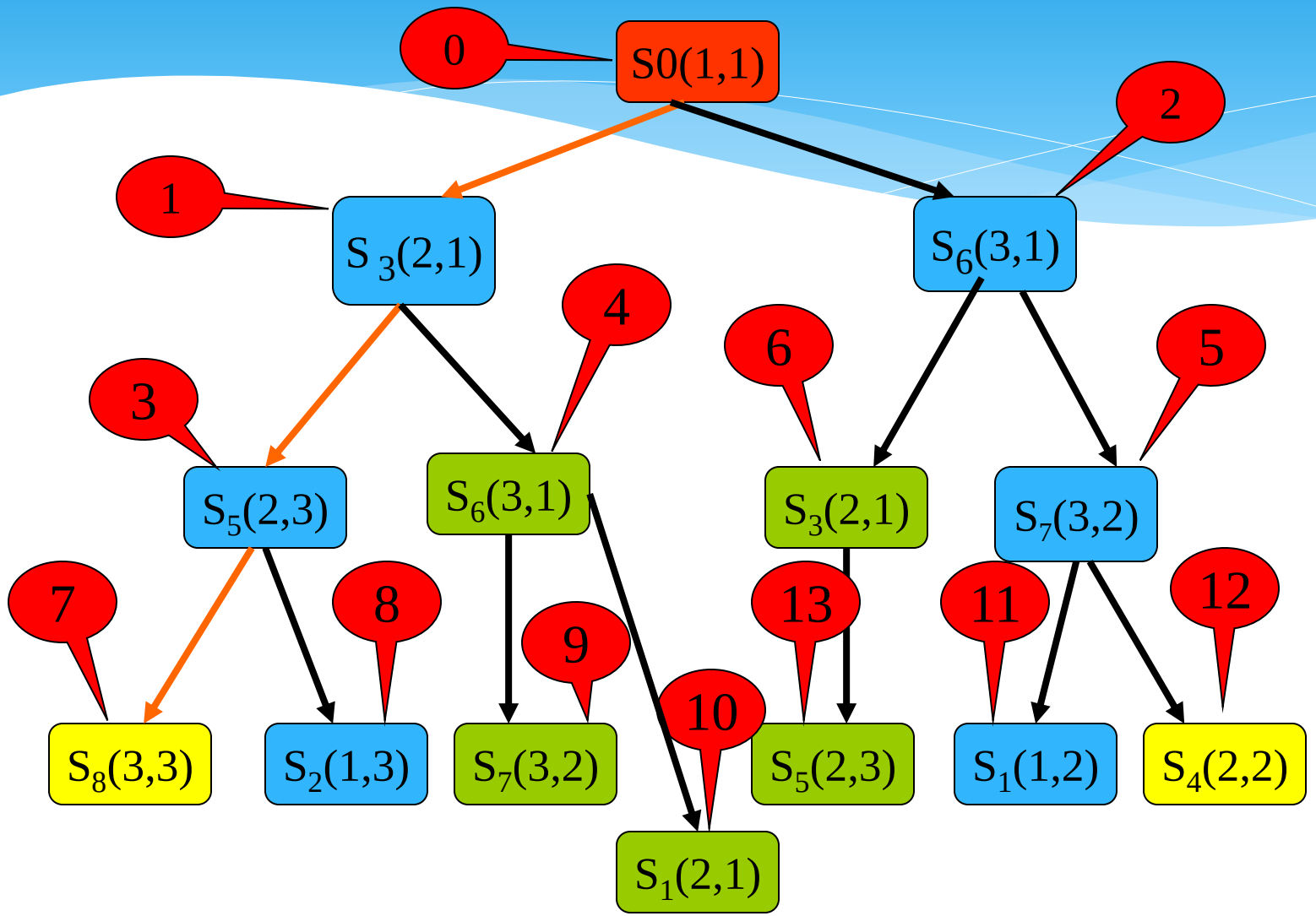
* 状态空间图


初始
A(1,2)
A(1,3)
$S_0(1,1)$
$S_3(2,1)$
$S_6(3,1)$
B(1,3)
B(1,2)
A(2,3)
$S_5(2,3)$
$S_7(3,2)$
A(3,2)
$S_8(3,3)$
$S_2(1,3)$
$S_1(1,2)$
$S_4(2,2)$
目标
目标

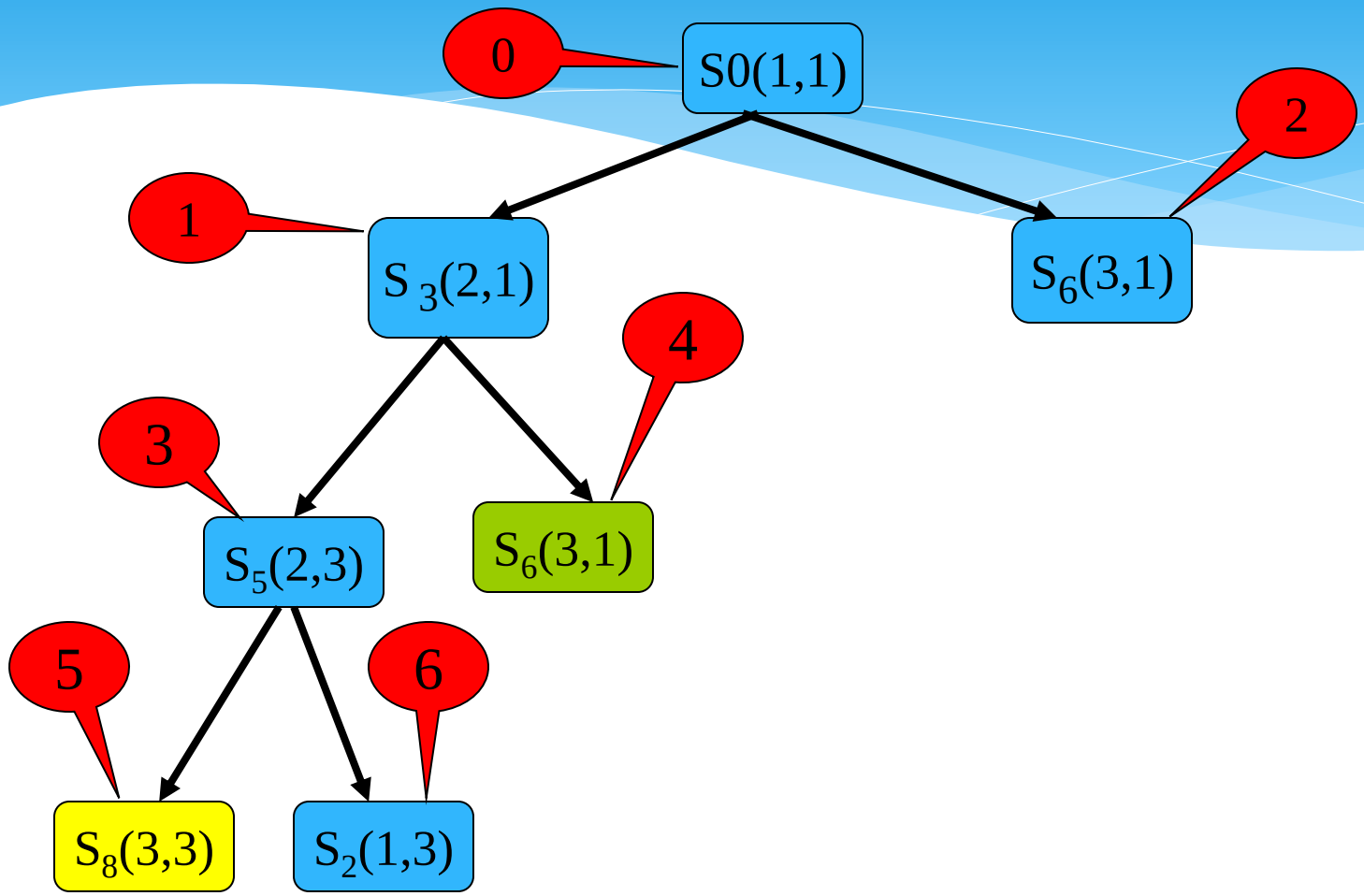
搜索要解决的问题

* **搜索策略**：如何找到解的路径
  * 即如何生成进一步的状态
* **约定**：不可走回头路
* **搜索图**：问题求解过程中经过的所有路径
* **最优解**：使用操作（算符）最少的解
* **例**

* 搜索策略1：宽度优先


0
S0(1,1)
2
1
S 3(2,1)
S6(3,1)
4
6
5
3
S5(2,3)
S6(3,1)
S3(2,1)
S7(3,2)
7
8
9
13
11
12
10
S8(3,3)
S2(1,3)
S7(3,2)
S5(2,3)
S1(1,2)
S4(2,2)
S1(2,1)

搜索策略2：深度优先



状态空间法求解问题的基本过程

* 首先为问题选择适当的“状态”及“操作”的形式化描述方法；
* 然后从某个初始状态出发，每次使用一个“操作”，递增地建立起操作序列，直到达到目标状态为止；
* 此时，由初始状态到目标状态所使用的算符序列就是该问题的一个解。

状态空间求解问题的关键

* 选择合适的状态描述形式
* 定义一组算符（操作）
* 搜索策略：决定算符生成进一步状态的顺序

搜索策略分类

* 无信息搜索过程
  * 宽度优先
  * 深度优先
* 启发式搜索过程
  * 代价树的广度优先搜索
  * 动态规划法（改进的代价树广度优先搜索)
  * 代价树的深度优先搜索(局部优先搜索)
  * 代价树有界深度优先搜索
  * 局部择优A算法
  * A算法(全局优先搜索)

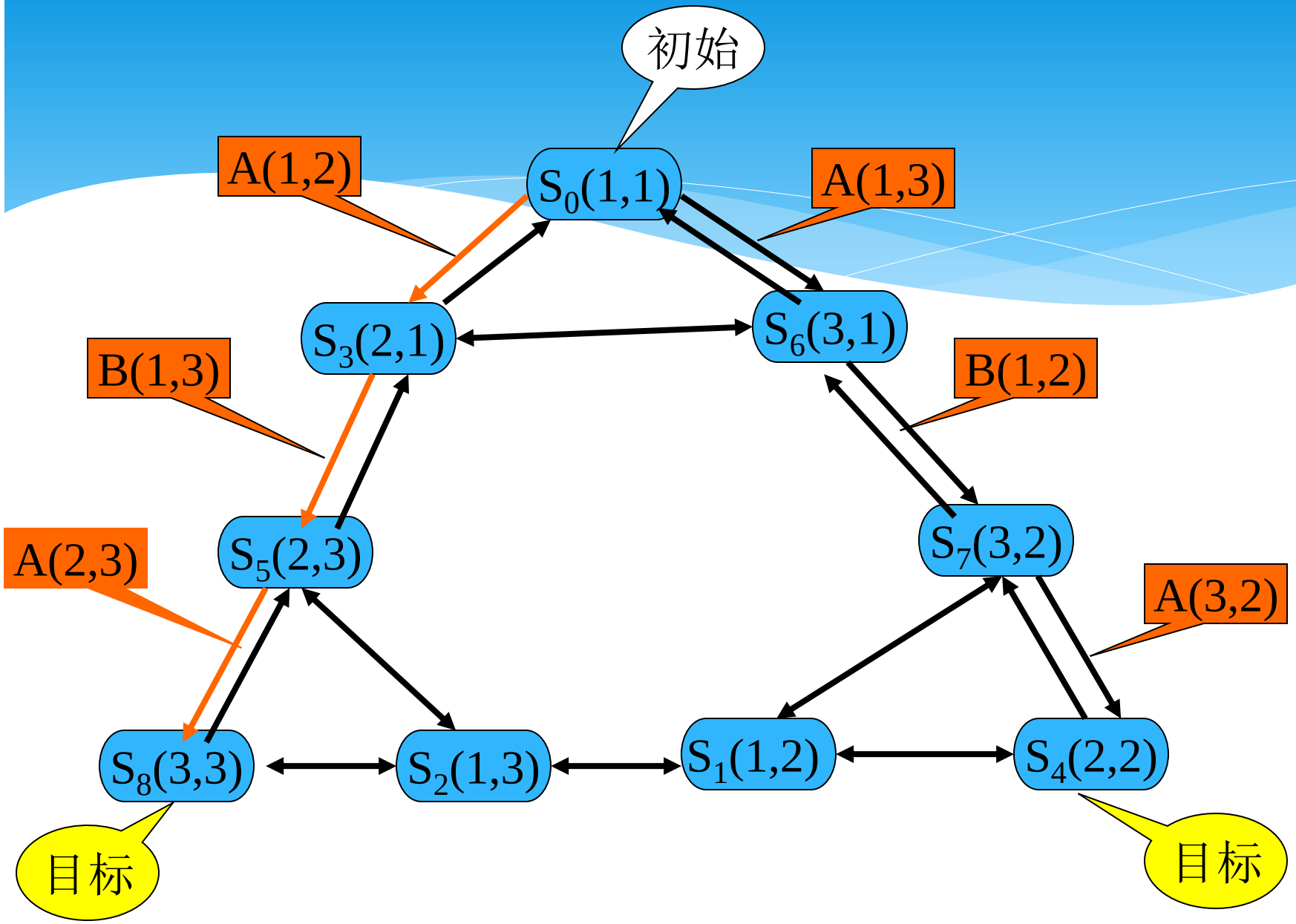
2.3 无信息搜索

* 基本术语
* 广（宽）度优先搜索
* 深度优先搜索
* 有界深度优先搜索

基本术语

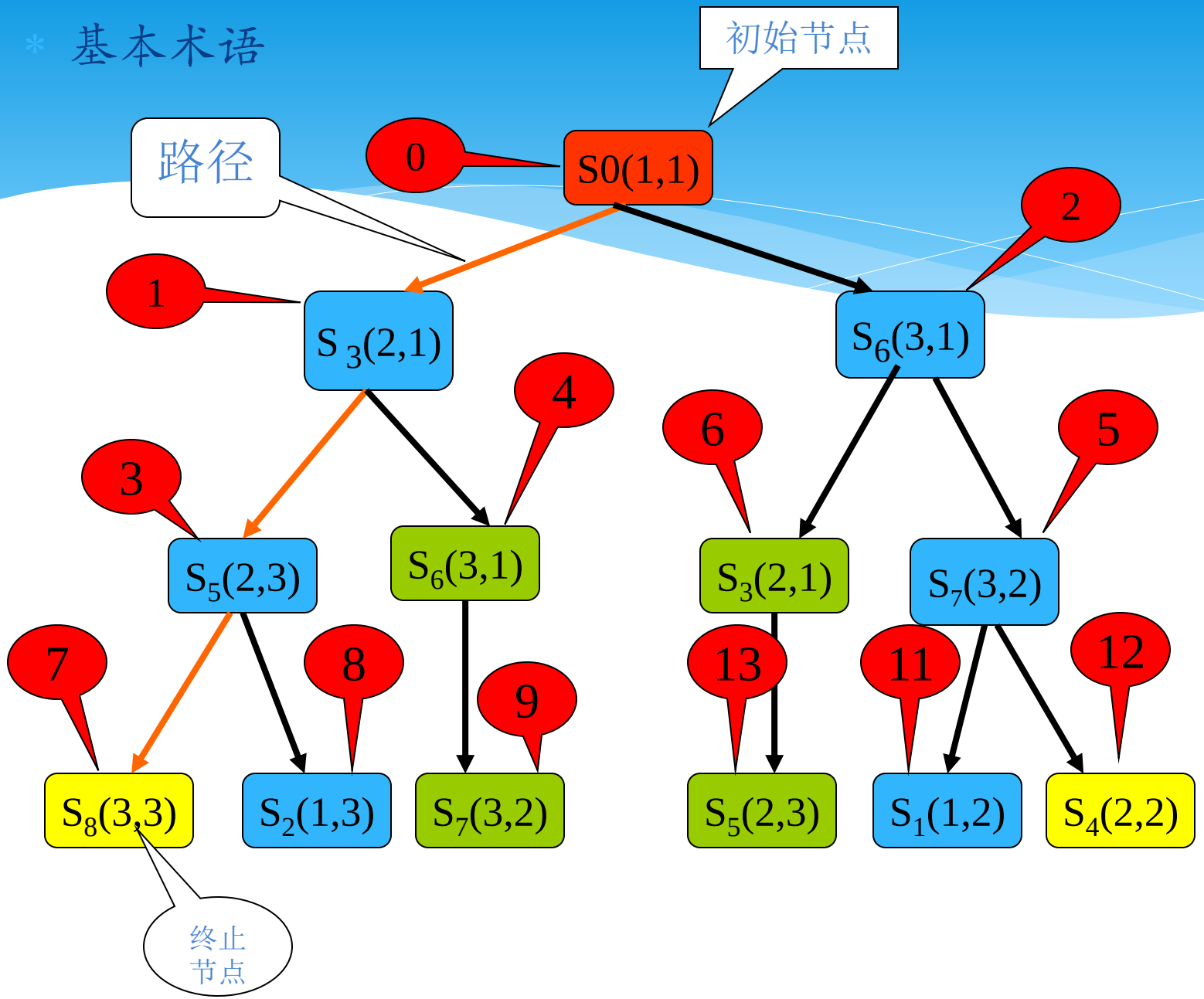
* 图搜索：实现从一个隐含图中,生成出一部分确实含有一个目标节点的显式表示子图的搜索过程.
* 例： 2阶“梵塔”问题

状态空间图


初始
A(1,2)
A(1,3)
$S_0(1,1)$
$S_3(2,1)$
$S_6(3,1)$
B(1,3)
B(1,2)
A(2,3)
$S_5(2,3)$
$S_7(3,2)$
A(3,2)
$S_8(3,3)$
$S_2(1,3)$
$S_1(1,2)$
$S_4(2,2)$
目标
目标

* 搜索树：由初始状态出发进行试探，以期找到一条通往目标状态的路径. 记录这搜索过程的状态的树
* 初始节点，目标节点，子节点
* 节点深度
* 路径
* 例： 2阶“梵塔”问题

* 基本术语


初始节点
路径
0
S0(1,1)
2
1
S 3(2,1)
S6(3,1)
4
6
5
3
S5(2,3)
S6(3,1)
S3(2,1)
S7(3,2)
7
8
9
13
11
12
S8(3,3)
S2(1,3)
S7(3,2)
S5(2,3)
S1(1,2)
S4(2,2)
终止
节点

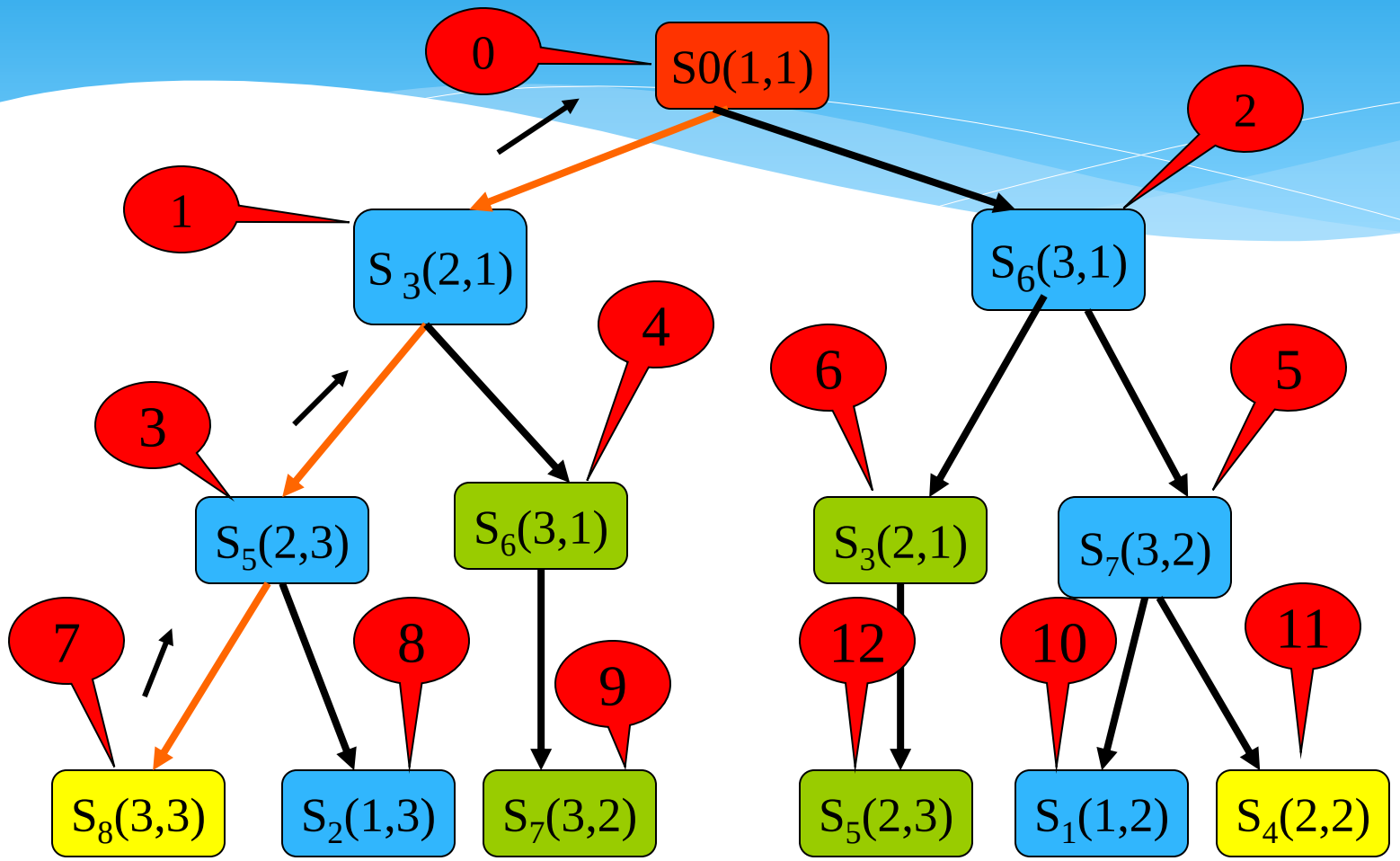
数据结构

* 记录搜索过程：OPEN表，CLOSED表
  - OPEN表:存放刚生成的节点
  * OPEN表形式：　状态节点，父节点

  - CLOSED表:存放将要扩展或已扩展的节点
  * CLOSED表形式：编号，状态节点，父节点

* 例：　2阶“梵塔”问题

搜索树：宽度优先


0
S0(1,1)
1
S 3(2,1)
2
S6(3,1)
3
S5(2,3)
4
S6(3,1)
6
S3(2,1)
5
S7(3,2)
7
8
9
12
10
11
S8(3,3)
S2(1,3)
S7(3,2)
S5(2,3)
S1(1,2)
S4(2,2)

* 初始
* Open表　　CLOSE表

状态节点	父节点
S0(1,1)	

编号	状态节点	父节点

* 第一次扩展

* Open表

状态节点	父节点
S3(2,1)	S0(1,1)
S6(3,1)	S0(1,1)

CLOSE表

编号	状态结点	父结点
1	$S_0(1,1)$	

* 第二次扩展

* OPEN表

状态节点	父节点
S_6 (3,1)	S0 (1,1)
S_5 (2,3)	S_3(2,1)
S_6 (3,1)	S_3(2,1)

CLOSED表

编号	状态结点	父结点
1	S_0(1,1)	
2	S_3(2,1)	S_0(1,1)

广（宽）度优先搜索（Breadth-First-Search）

* 基本思想
* 搜索过程
* 实例
* 算法讨论

宽度优先搜索基本思想

* 从初始节点So开始，逐层地对节点进行扩展并考察它是否为目标节点，在第n层节点没有全部扩展并考察前，不对第n+1层的节点进行扩展。
* 节点按进入open表的先后顺序排列

广（宽）度优先搜索过程

(1)把初始节点S_0放入Open表中；

(2)如果Open表为空，则问题无解，失败退出；

(3)把Open表的第一个节点取出放入Closed表，并记该节点为n；

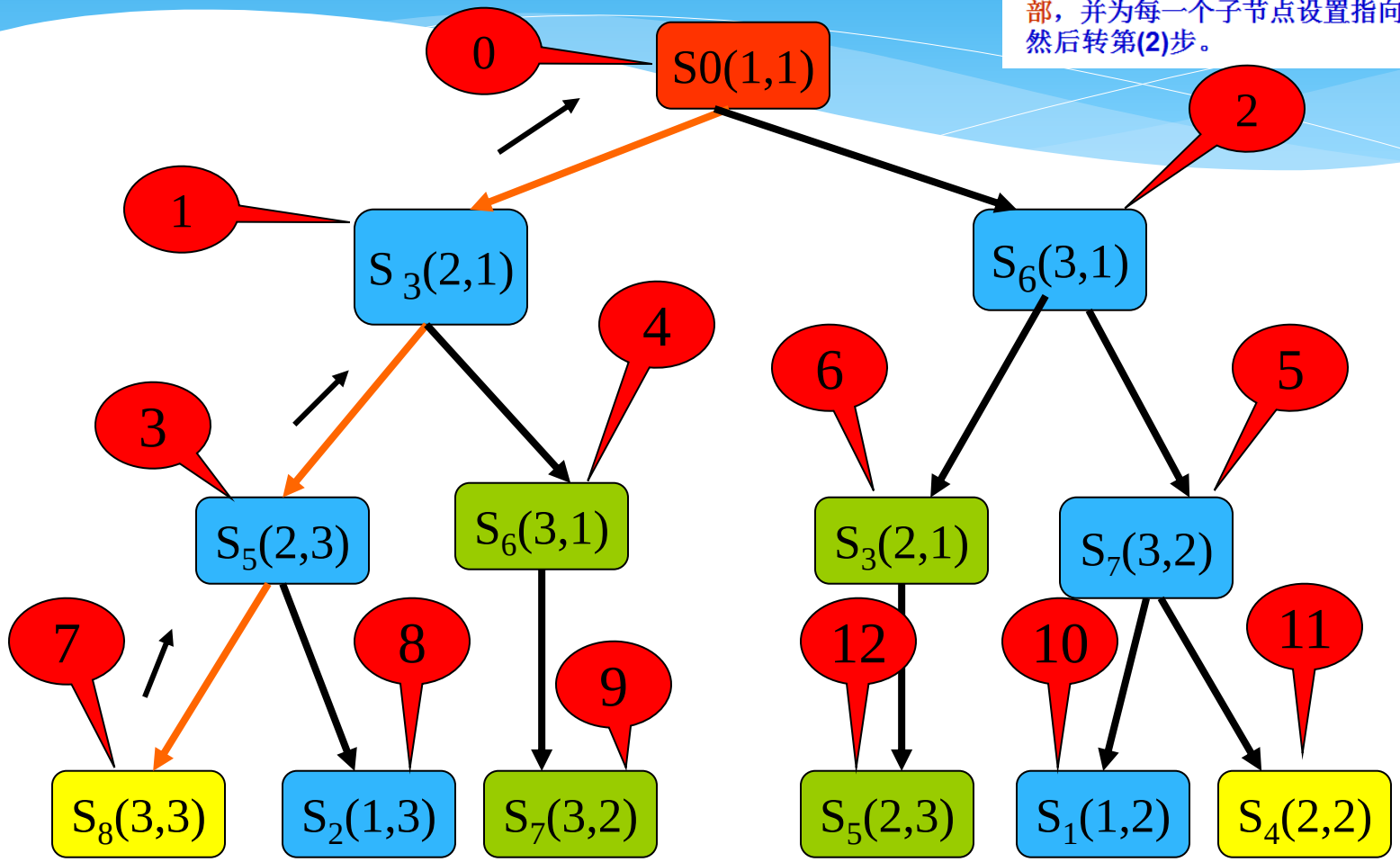
(4)考察节点n是否为目标节点。若是，则得到问题的解，成功退出；

(5)若节点n不可扩展，则转第(2)步；

(6)扩展节点n，将其子节点放入Open表的尾部，并为每一个子节点设置指向父节点的指针，然后转第(2)步。

例1 宽度优先（2级梵塔）

(1)把初始节点S_0放入Open表中；
(2)如果Open表为空，则问题无解，失败退出；
(3)把Open表的第一个节点取出放入Closed表，并记该节点为n；
(4)考察节点n是否为目标节点。若是，则得到问题的解，成功退出；
(5)若节点n不可扩展，则转第(2)步；
(6)扩展节点n，将其子节点放入Open表的尾部，并为每一个子节点设置指向父节点的指针，然后转第(2)步。



例2：重排九宫问题

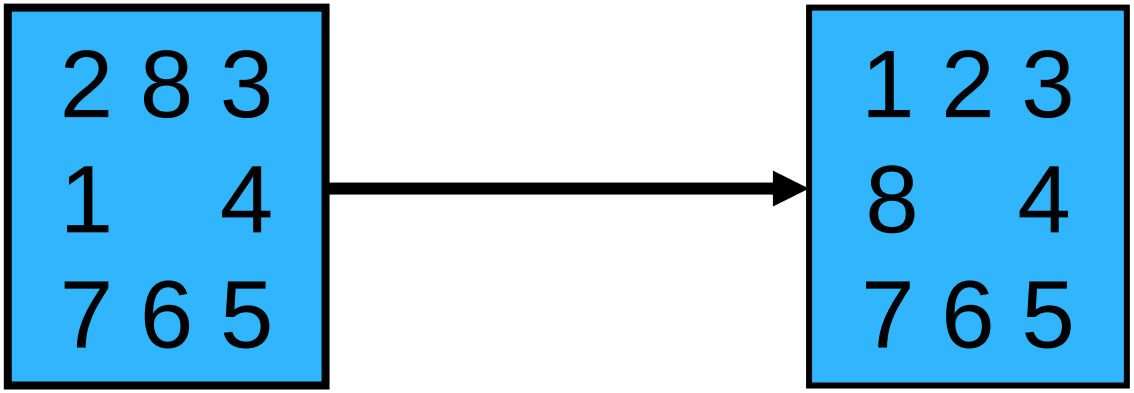
例：重排九宫问题。在3×3的方格棋盘上放置八张牌，初始状态和目标状态如右图

2	8	3
1		4
7	6	5

1	2	3
8		4
7	6	5


三国华容道
黄忠
曹操
赵云
张飞
关羽
马超
卒
卒
卒
卒

宽度优先搜索演示演示.ppt
（九宫图）

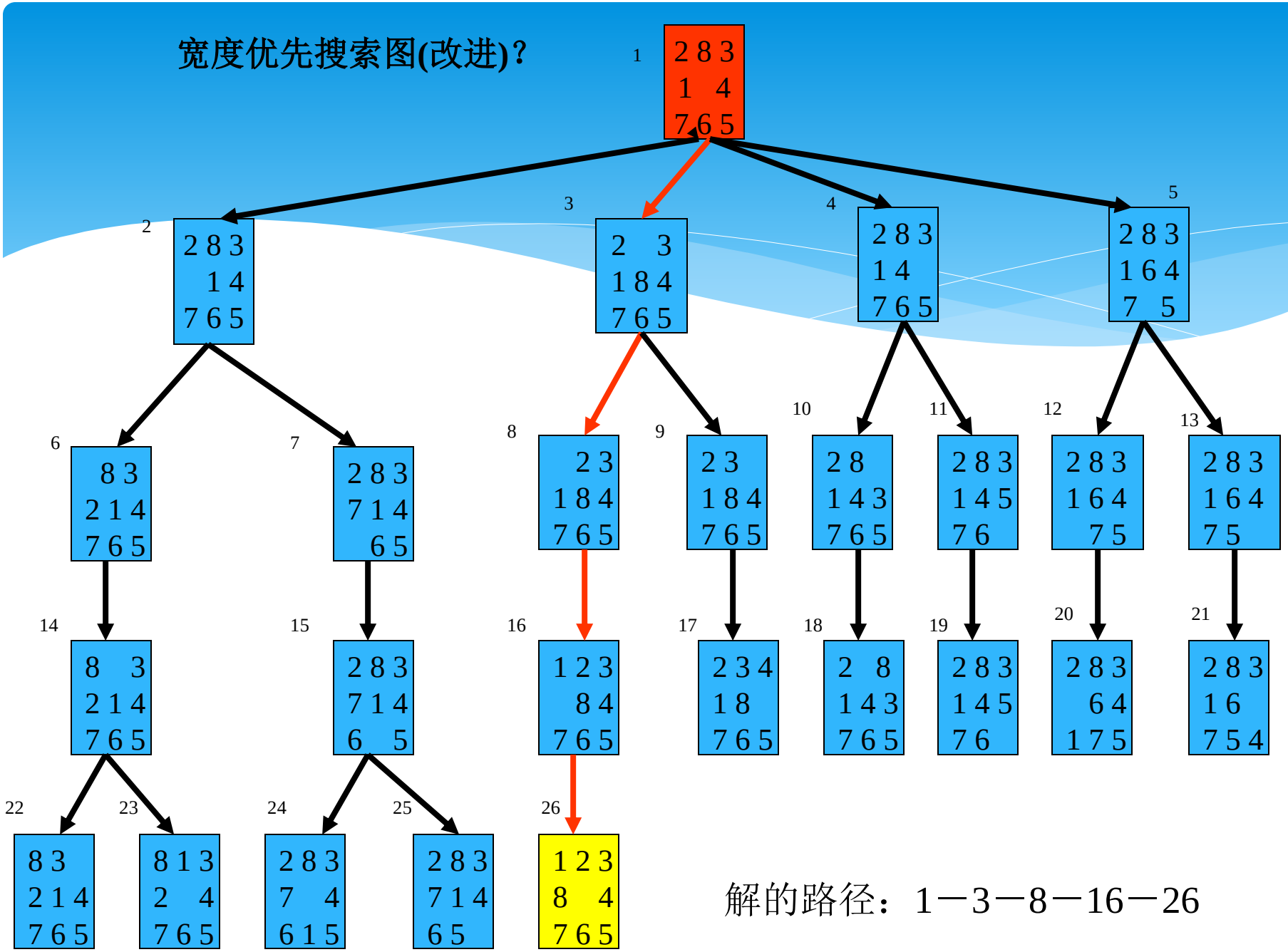


初始状态　　目标状态

算法讨论

* 存在问题吗?
* 如何改进?
* 算法特点

宽度优先搜索图(改进)?



解的路径：1—3—8—16—26

* Open表的变化(改进的宽度优先搜索法)

初始 (1)

1 (2,3,4,5)

2 (3,4,5,6,7)

3 (4,5,6,7,8,9)

4 (5,6,7,8,9,10,11)

5 (6,7,8,9,10,11,12,13)

6 (7,8,9,10,11,12,13,14)

7 (8,9,10,11,12,13,14,15,)

8 (9,10,11,12,13,14,15,16)

9 (10,11,12,13,14,15,16,17)

10 (11,12,13,14,15,16,17,18)

11 (12,13,14,15,16,17,18,19)

12 (13,14,15,16,17,18,19,20)

13 (14,15,16,17,18,19,20,21)

14 (15,16,17,18,19,20,21,22,23)

15 (16,17,18,19,20,21,22,23,24,25)

16 (17,18,19,20,21,22,23,24,25,) 26

深度优先搜索

* 基本思想
* 搜索过程
* 实例
* 算法讨论

深度优先搜索基本思想

* 从初始节点So开始，在其子节点中选择一个节点进行扩展并考察它是否为目标节点，若不是目标节点，则在该子节点的子节点中选择一个节点进行考察，一直如此向下搜索。当到达某个子节点，且该子节点即不是目标节点又不能继续扩展时，才选择其兄弟节点进行扩展。

节点按后进入open表的顺序排列，即后进入的节点排在表的最前面

深度优先搜索过程

(1) 把初始节点S_0放入Open表中；

(2) 如果Open表为空，则问题无解 ，失败退出；

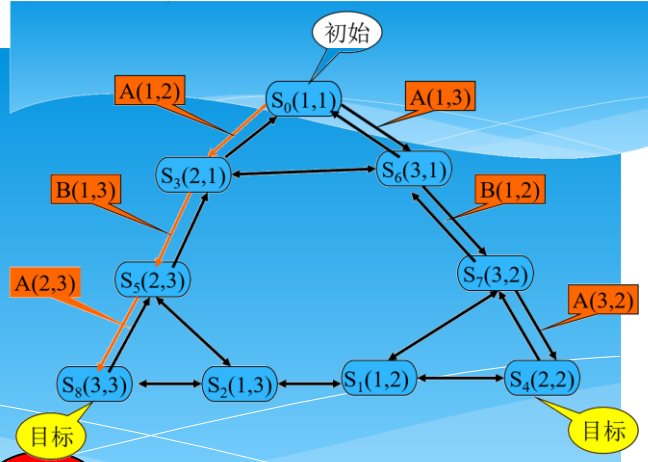
(3) 把Open表的第一个节点取出放入Closed表，并记该节点为n；

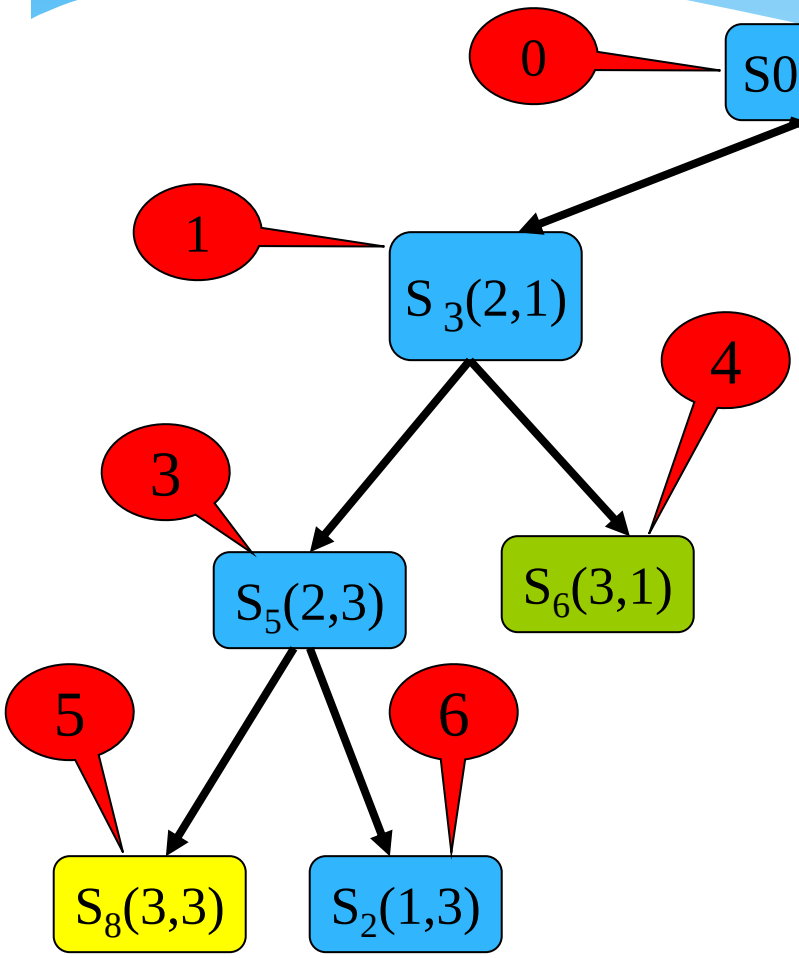
(4) 考察节点n是否为目标节点。若是，则得到问题的解，成功退出；

(5) 若节点n不可扩展，则转第(2)步；

(6) 扩展节点n，将其子节点放入Open表的首部，并为每一个子节点设置 指向父节点的指针，然后转第(2)步。

例1 深度优先（2级梵塔）





(1) 把初始节点S_0放入Open表中；

(2) 如果Open表为空，则问题无解 ，失败退出；

(3) 把Open表的第一个节点取出放入Closed表，并记该节点为n；

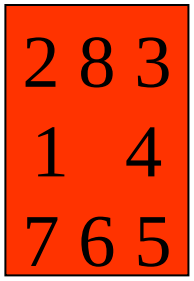
(4) 考察节点n是否为目标节点。若是，则得到问题的解，成功退出；

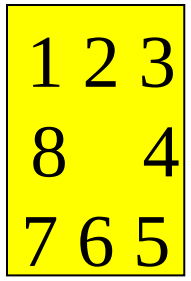
(5) 若节点n不可扩展，则转第(2)步；

(6) 扩展节点n，将其子节点放入Open表的首部，并为每一个子节点设置 指向父节点的指针，然后转第(2)步。

例2：重排九宫问题

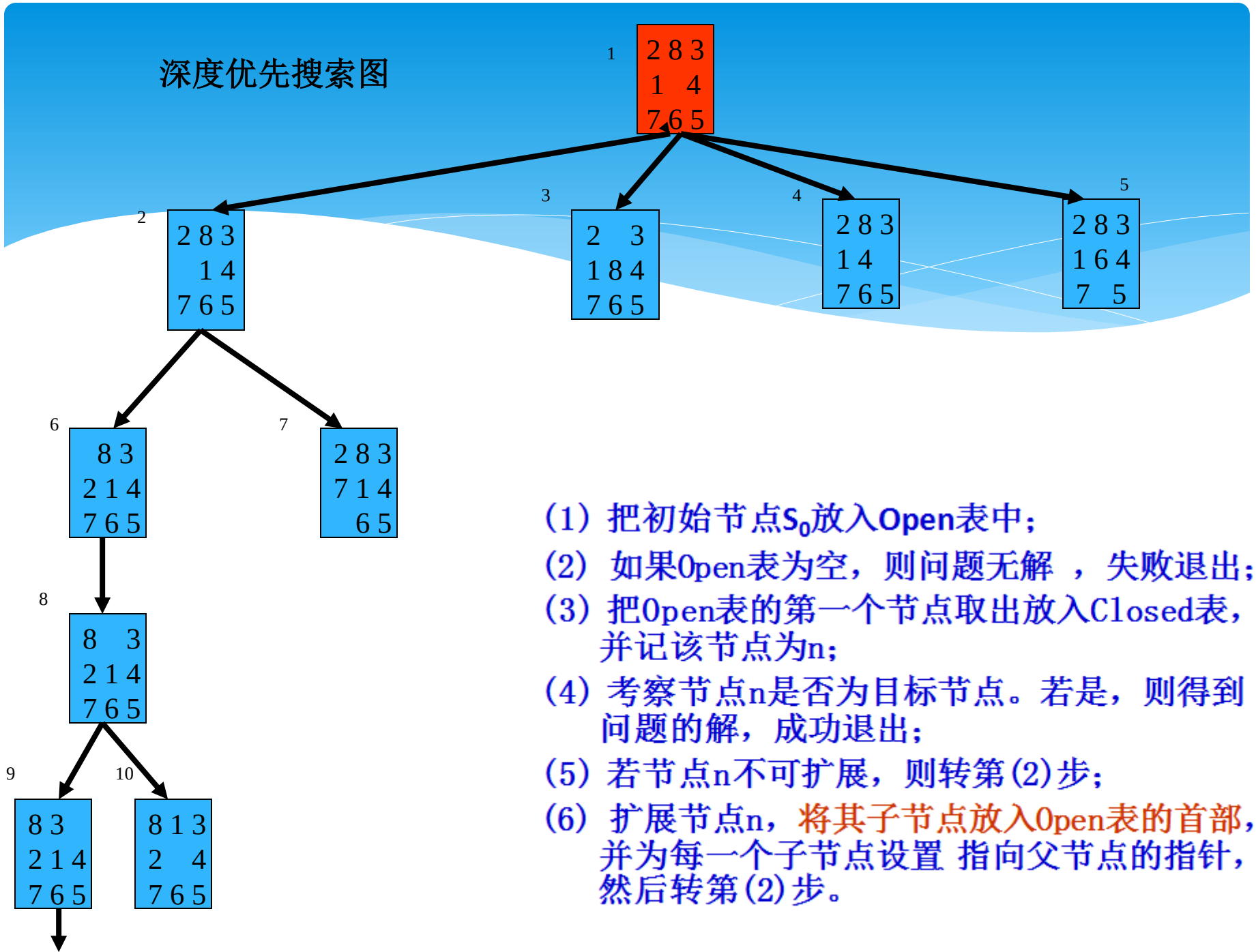
例：重排九宫问题。在3×3的方格棋盘上放置八张牌，初始状态和目标状态如右图





* 算符有：
  * R1: 如果满足条件 则 空格左移
  * R2: 如果满足条件 则 空格上移
  * R3: 如果满足条件 则 空格右移
  * R4: 如果满足条件 则 空格下移
  * 注：条件指有位置并且不重复
* 冲突解决方法：算符序号

深度优先搜索图



(1) 把初始节点S_0放入Open表中；

(2) 如果Open表为空，则问题无解 ，失败退出；

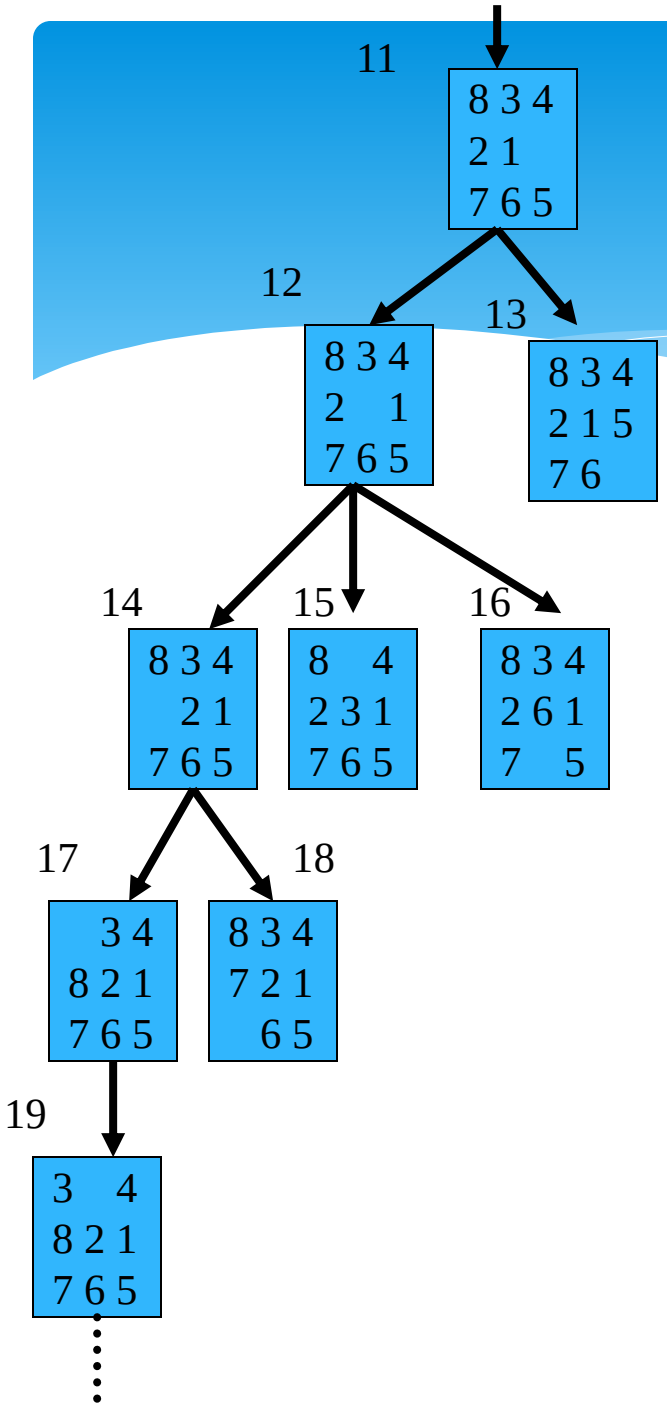
(3) 把Open表的第一个节点取出放入Closed表，并记该节点为n；

(4) 考察节点n是否为目标节点。若是，则得到问题的解，成功退出；

(5) 若节点n不可扩展，则转第(2)步；

(6) 扩展节点n，将其子节点放入Open表的首部，并为每一个子节点设置 指向父节点的指针，然后转第(2)步。

深度优先搜索图（续）



(1) 把初始节点S_0放入Open表中；

(2) 如果Open表为空，则问题无解 ，失败退出；

(3) 把Open表的第一个节点取出放入Closed表，并记该节点为n；

(4) 考察节点n是否为目标节点。若是，则得到问题的解，成功退出；

(5) 若节点n不可扩展，则转第(2)步；

(6) 扩展节点n，将其子节点放入Open表的首部，并为每一个子节点设置 指向父节点的指针，然后转第(2)步。

* Open表

初始（1）

1 （2,3,4,5）

2 (6,7,3,4,5)

3 (8,7,3,4,5)

4 (9,10,7,3,4,5)

5 (11,10,7,3,4,5)

6 (12,13,10,7,3,4,5)

7 (14,15,16,13,10,7,3,4,5)

8 (17,18,15,16,13,10,7,3,4,5)

9 (19,18,15,16,13,10,7,3,4,5)

……

……

算法讨论

* 存在问题
* 改进方法

有界深度优先搜索

* 基本思想
* 搜索过程
* 实例

有界深度优先搜索基本思想

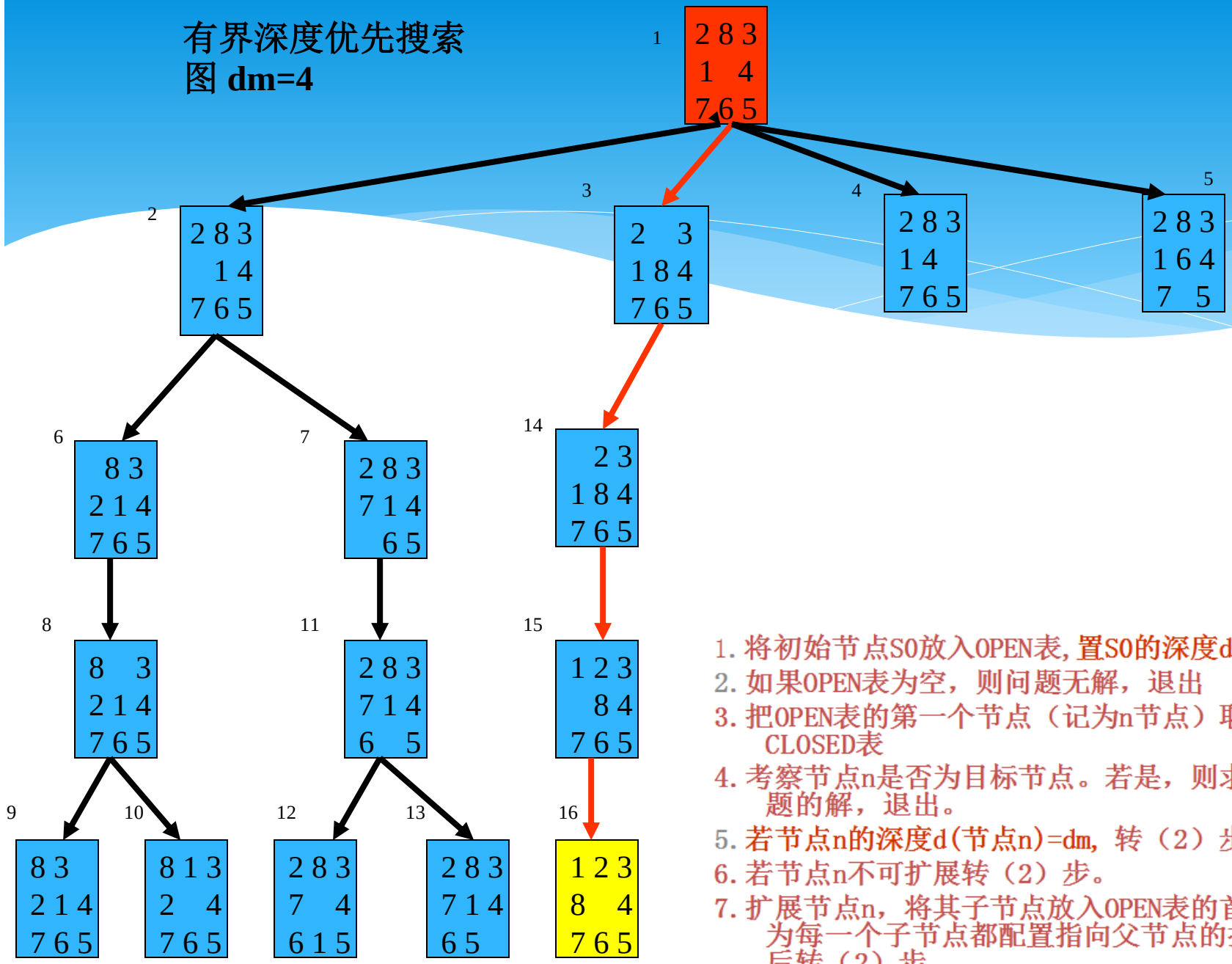
* 对深度优先搜索引入搜索深度的限制（设为dm），当搜索深度达到深度界限时，尚未出现目标节点，就选择其兄弟节点进行扩展。
* 节点按后进入open表的顺序排列，即后进入的节点排在前面
* 深度的确定：固定深度

  可变深度

有界深度搜索过程

1. 将初始节点S0放入OPEN表，置S0的深度d(S0)=0
2. 如果OPEN表为空，则问题无解，退出
3. 把OPEN表的第一个节点（记为n节点）取出放入CLOSED表
4. 考察节点n是否为目标节点。若是，则求得了问题的解，退出。
5. 若节点n的深度d(节点n)=dm，转（2）步。
6. 若节点n不可扩展转（2）步。
7. 扩展节点n，将其子节点放入OPEN表的首部，并为每一个子节点都配置指向父节点的指针，然后转（2）步。

有界深度优先搜索
图 dm=4



1. 将初始节点S0放入OPEN表，置S0的深度d(S0)=0
2. 如果OPEN表为空，则问题无解，退出
3. 把OPEN表的第一个节点（记为n节点）取出放入CLOSED表
4. 考察节点n是否为目标节点。若是，则求得了问题的解，退出。
5. 若节点n的深度d(节点n)=dm，转（2）步。
6. 若节点n不可扩展转（2）步。
7. 扩展节点n，将其子节点放入OPEN表的首部，并为每一个子节点都配置指向父节点的指针，然后转（2）步。

* Open表

初始（1）

1 （2,3,4,5）

2 (6,7,3,4,5)

3 (8,7,3,4,5)

4 (9,10,7,3,4,5)　　dm=5

5 (10,7,3,4,5)　　dm=5

6 (7,3,4,5)

7 (11,3,4,5)

8 (12,13,3,4,5)　　dm=5

9 (13,3,4,5)　　dm=5

10 (3,4,5)

11 (14,4,5)

12 (15,4,5)

13 (16,4,5)

14 16 为目标节点

.