ACM程序设计培训

计算机与信息学院 刘必雄

搜索技术专题

内 容 提 要

SEARCH TOPIC

SEARCH TOPIC

1、引言

2、深度优先搜索

3、广度优先搜索

4、启发式搜索

SEARCH TOPIC

SEARCH TOPIC

1、引言

统计信息

根据“信息学初学者之家”网站对俄罗斯的Ural州立大学的Ural Online Problem Set 的题目类型的统计结果，可知搜索是最基本、最常用的算法之一。

搜索	动态规划	贪心	构造	图论
约10%	约15%	约5%	约5%	约10%
计算几何	纯数学题	数据结构	其它	
约5%	约20%	约5%	约25%	

1、引言

1、引言

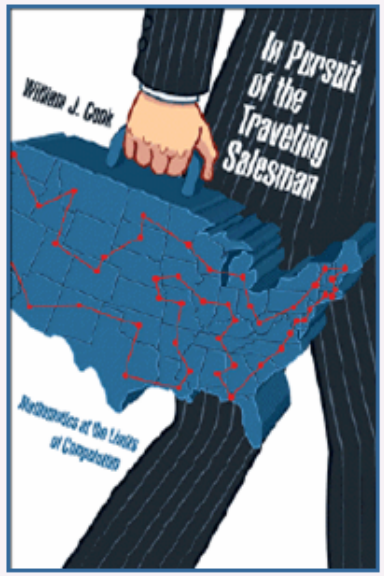
TSP

Home

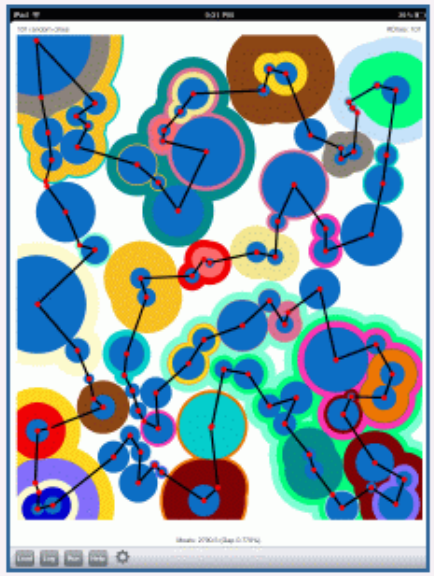
- Home
- The Problem
- History
- Applications
- Solving a TSP
- World Records
- Gallery
- TSP Games
- Google Maps
- Concorde
- Test Data
- News
- TSP Book
- Search Site

The Traveling Salesman Problem

The Traveling Salesman Problem is one of the most intensively studied problems in computational mathematics. These pages are devoted to the history, applications, and current research of this challenge of finding the shortest route visiting each member of a collection of locations and returning to your starting point.



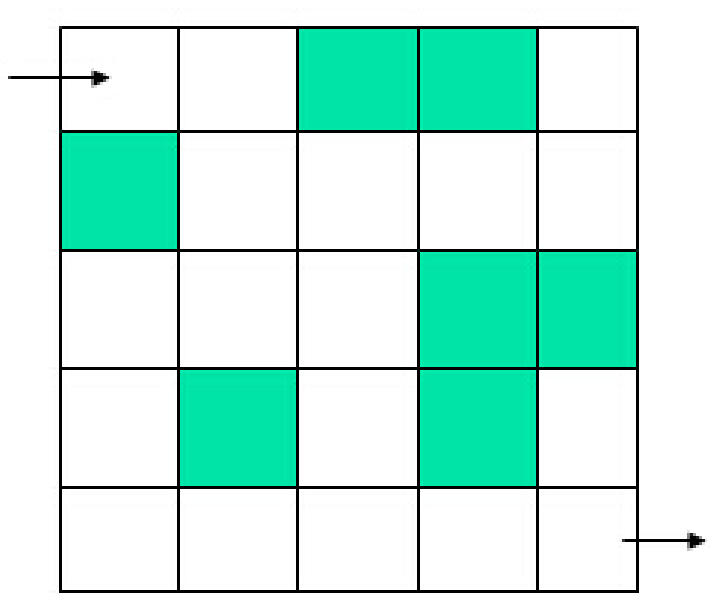
New TSP Book
$16.83 at Amazon.com
Chapter 1 as pdf file
Facebook Page

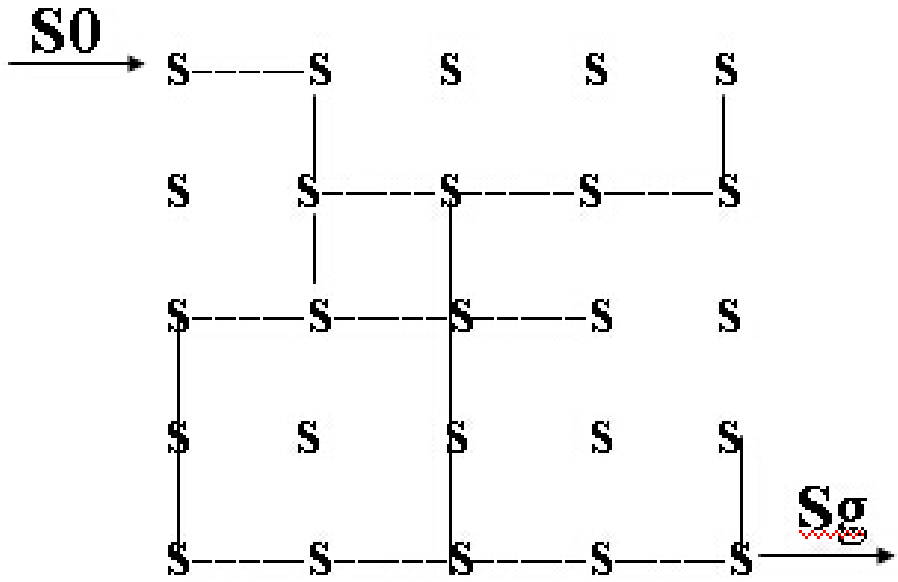


TSP iPhone/iPad App
iTunes Preview
App Support Page
NY Times Article

1、引言

迷宫问题





1、引言

- 什么是搜索算法？

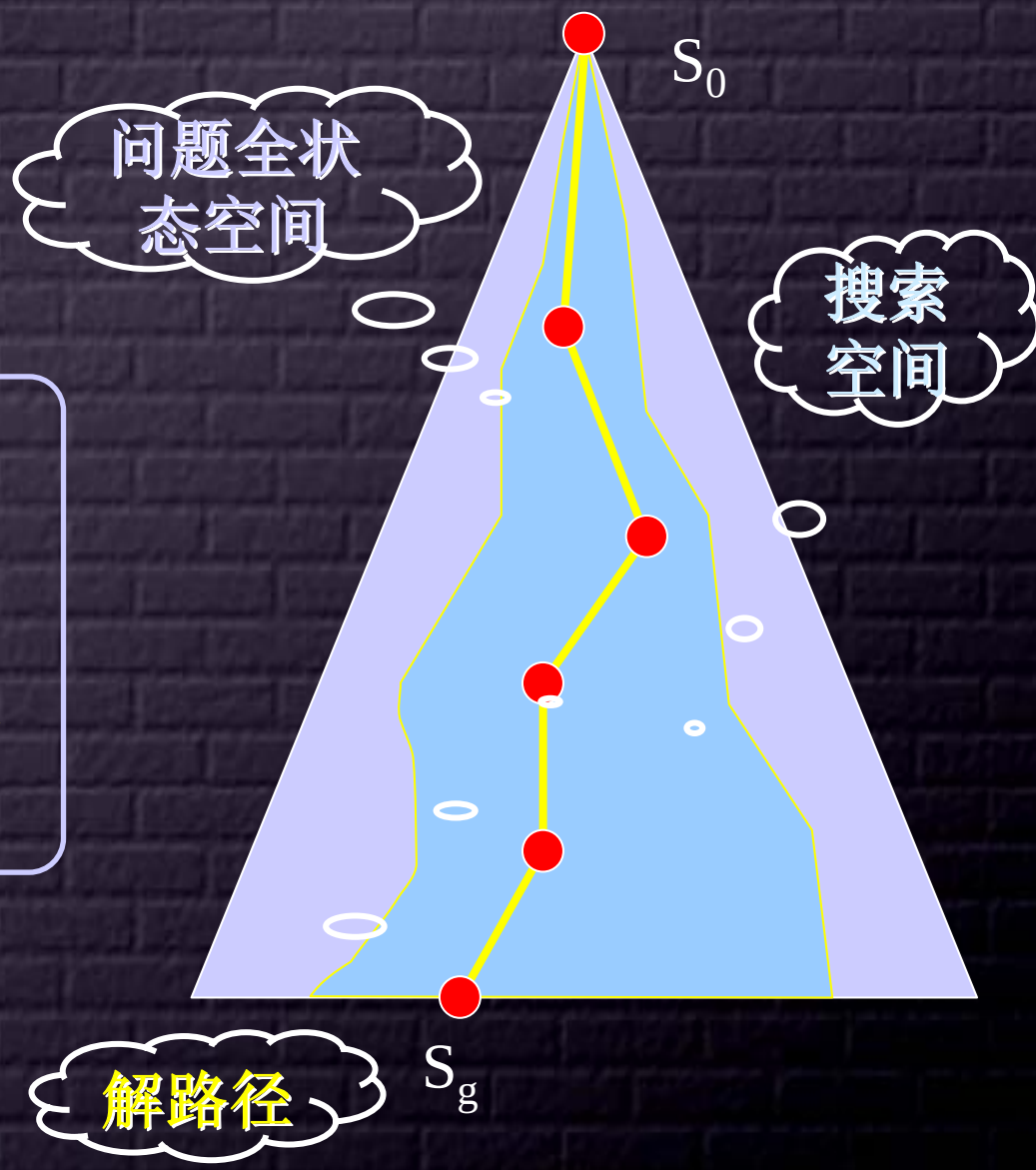
  - 人工智能所要解决的问题大部分**不具备明确的解题步骤**，而只能是利用已有的知识一步一步地摸索前进。
  - 根据问题的实际情况不断寻找可利用的知识，从而构造一条代价较少的**推理路线**，使问题得到圆满解决的过程称之为**搜索**。

搜索算法是利用计算机的高性能来有目的地穷举一个问题的部分或所有的可能情况，从而求出问题的解的一种方法。

1、引言

搜索过程

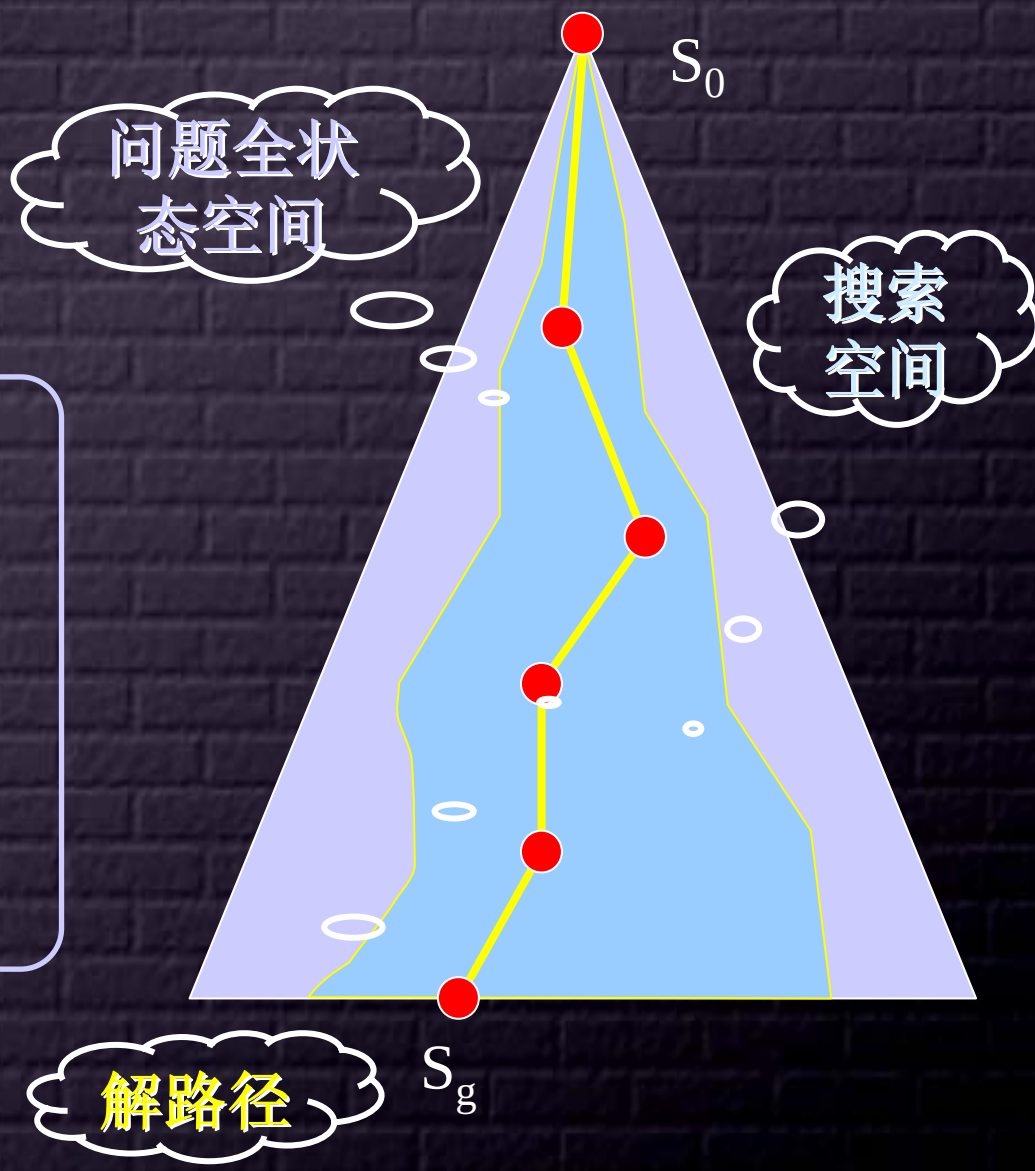
搜索过程实际上是根据初始条件和扩展规则构造一棵解答树并寻找符合**目标状态的节点**的过程。



1、引言

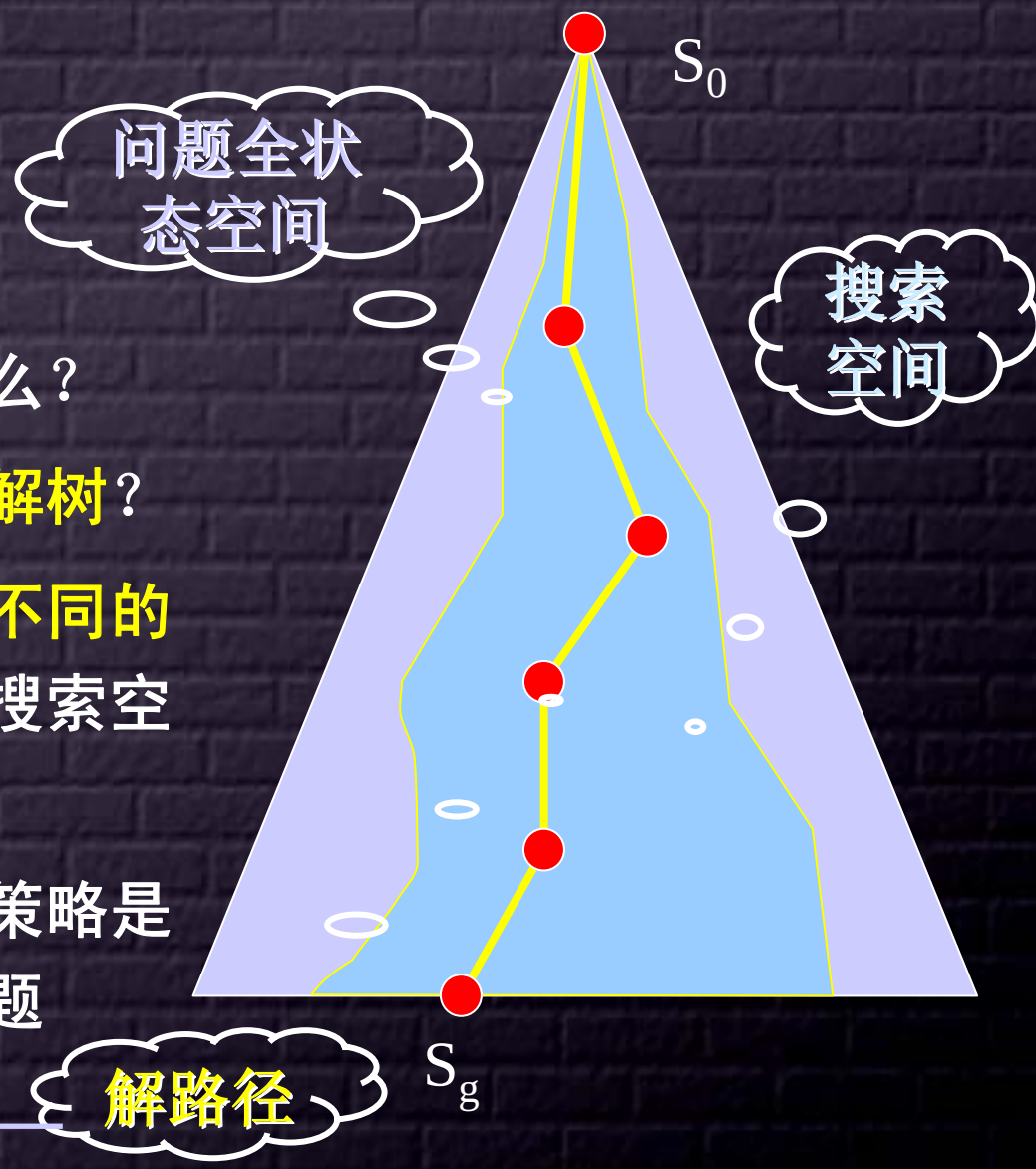
问题的解

搜索**问题的解**，就是一个合法状态的序列，其中序列中第一个状态是问题的**初始状态**，而最后一个状态则是问题的**结束状态**。



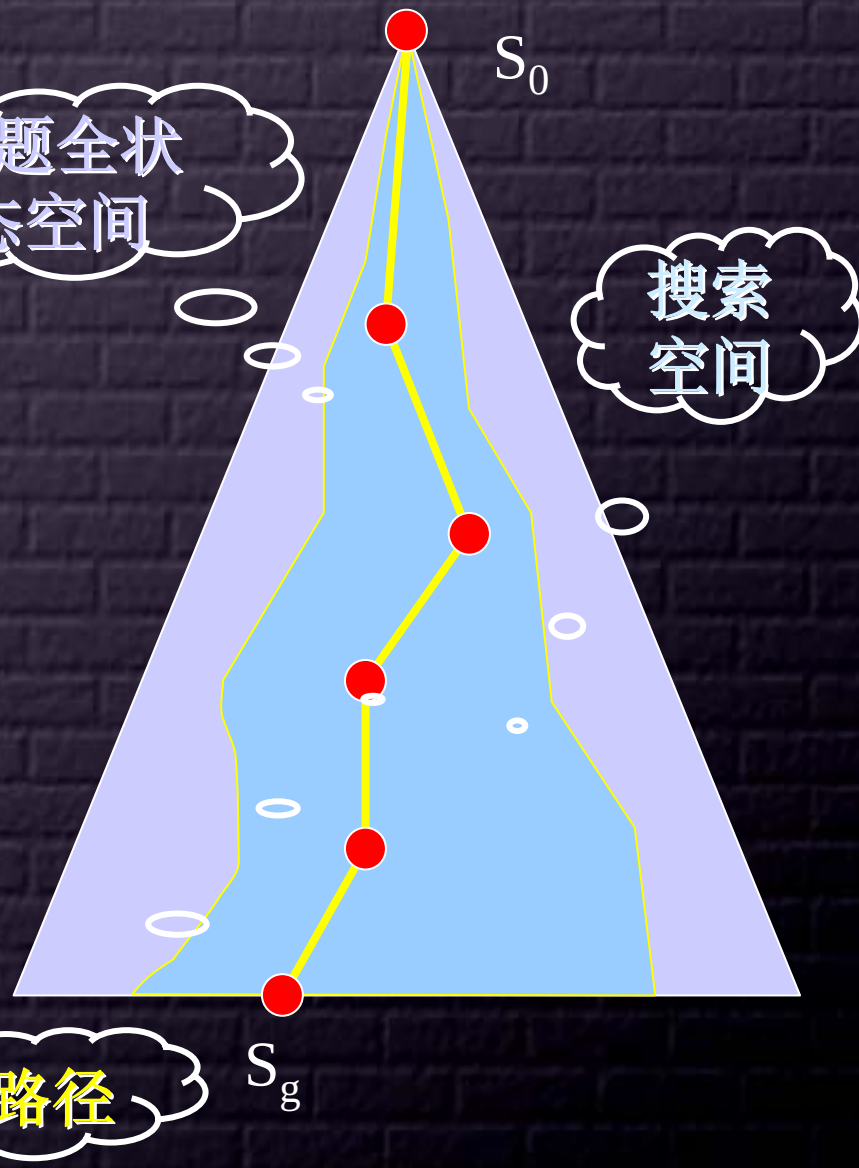
1、引言

- 需要思考的问题？
  - 以什么做为状态？
  - 状态之间的关系是什么？
  - 如何在搜索树中找到解树？
  - 当问题有解时，使用不同的搜索策略，找到解的搜索空间范围不同
  - 对大空间问题，搜索策略是要解决组合爆炸的问题



1、引言

- 状态图的概念
  - 状态图（状态空间图）实际上是一类问题的抽象表示。
  - 许多智力问题（八数码问题、梵塔问题、旅行商问题、八皇后问题、农夫过河问题等）；实际问题（如路径规划、定理证明、演绎推理、机器人行动规划等）都可以归结为在某一状态图中寻找目标或路径的问题。



1、引言

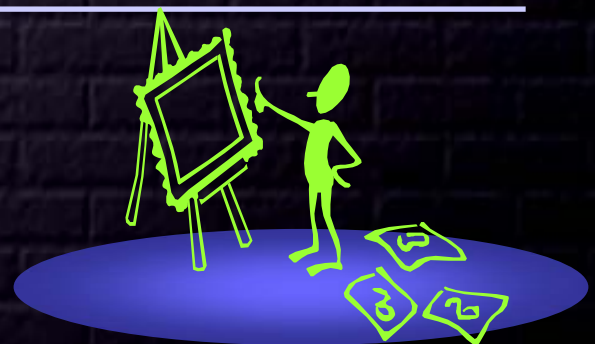
农夫过河问题

有一个农夫带一条狼、一只羊和一棵白菜过河。如果没有农夫看管，则狼要吃羊，羊要吃白菜。但是船很小，只够农夫带一样东西过河。问农夫该如何解此难题？

1、引言

农夫过河问题

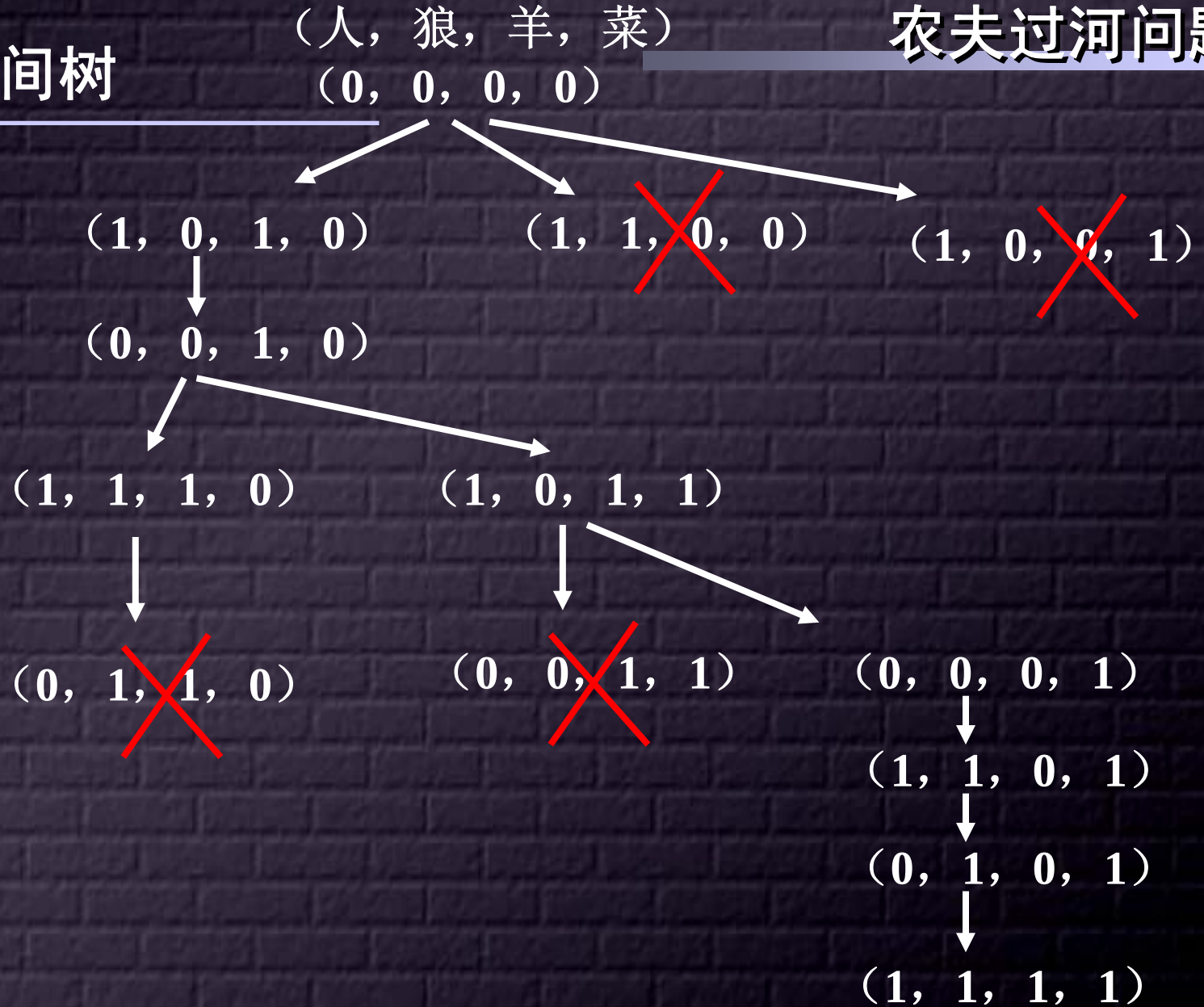
- 状态空间法表示
  - 以向量（人，狼，羊，菜）表示状态，其中每个变元可取0或1，取0表示在左岸（出发点），取1表示在右岸。
  - 初态：（0，0，0，0）
  - 终态：（1，1，1，1）
  - 非法中间状态有：（0，0，1，1），（0，1，1，0），（0，1，1，1），（1，1，0，0），（1，0，0，1），（1，0，0，0）



1、引言

农夫过河问题

- 解空间树



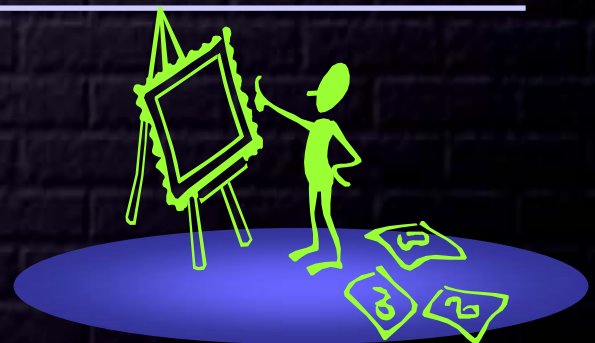
1、引言

- 状态空间法
  - 问题的状态空间表示通常采用状态空间的三元组(S, O, G)表示。

    S：初始状态集合

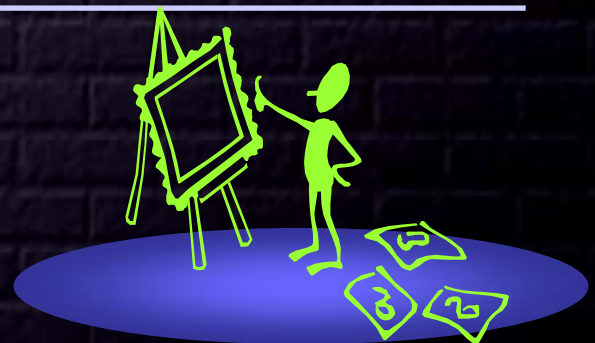
    O： 操作集合

    G：目标状态集合
  - 状态空间的搜索策略：广度优先搜索，深度优先搜索，启发式搜索。



1、引言

- 搜索策略
  - **盲目搜索**：又称无信息/穷举式搜索，只能按照预先规定的**搜索控制策略**进行搜索，没有任何中间信息来改变这些控制策略。
  - **启发式搜索**：在搜索求解过程中，根据问题本身的特性或搜索过程中所产生的一些与问题有关的**启发性信息**，指导搜索朝着最有希望的推理方向前进，**加速**问题的求解过程并找到最优解。



2、深度优先搜索

- 基本思想

从初始状态So开始，利用规则生成搜索树下一层任一个结点，检查是否出现目标状态Sg，若未出现，以此状态利用规则生成再下一层任一个结点，再检查是否为目标节点Sg，若未出现，继续以上操作过程，一直进行到叶节点（即不能再生成新状态节点），当它仍不是目标状态Sg时，回溯到上一层结果，取另一可能扩展搜索的分支。生成新状态节点。若仍不是目标状态，就按该分支一直扩展到叶节点，若仍不是目标，采用相同的回溯办法回退到上层节点，扩展可能的分支生成新状态，…，一直进行下去，直到找到目标状态Sg为止。

2、深度优先搜索

八格码问题

- 问题描述

在3×3的方格棋盘上，分别放置了标有数字1、2、3、4、5、6、7和8的八个棋子，其中空出一个位置使棋子可以移动，形成不同的局面。要使棋盘进入某种预定局面应**如何移动**棋子？

S_0

2	8	3
1	□	4
7	6	5

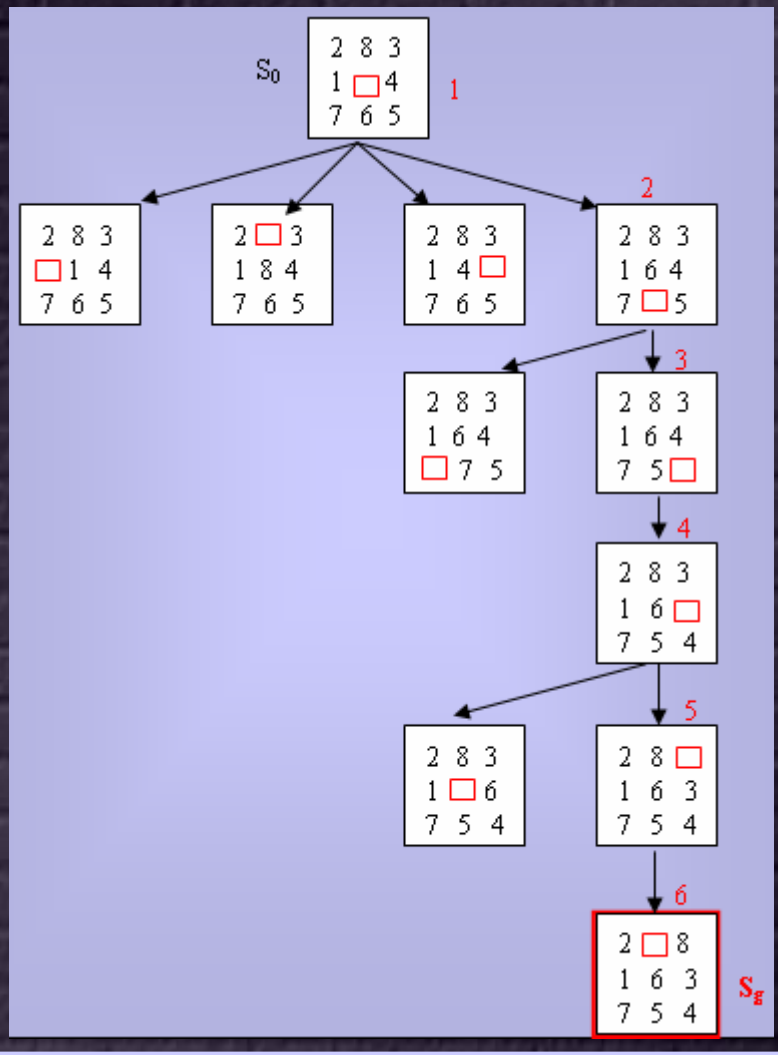
S_g

2	□	8
1	6	3
7	5	4

2、深度优先搜索

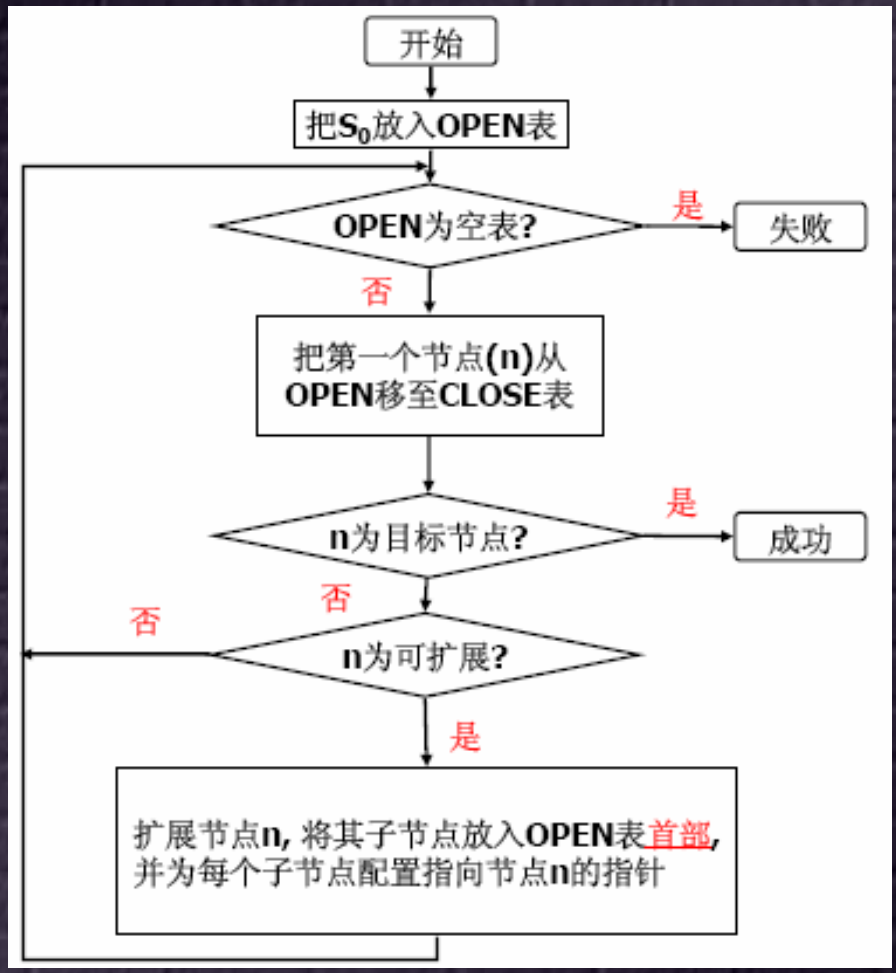
八格码问题

- 深度优先搜索树

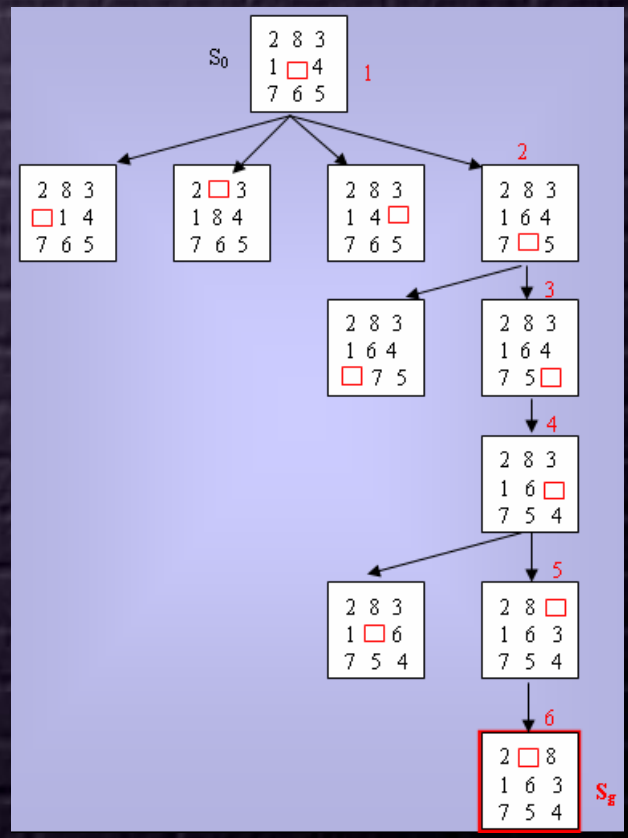


2、深度优先搜索

深度优先搜索算法



Open表是一种**栈结构**



2、深度优先搜索

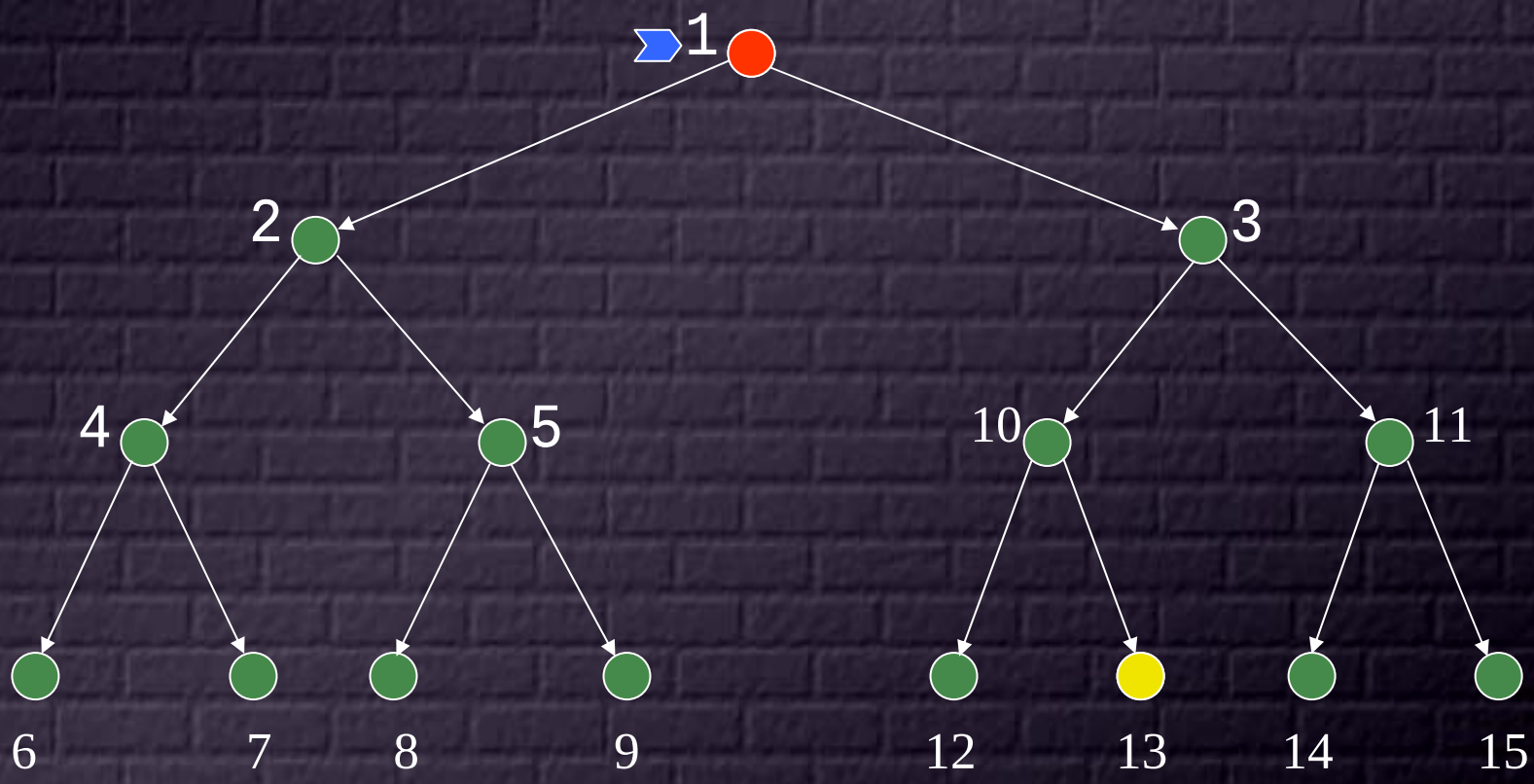
深度优先搜索算法

- Step1: 把初始节点S_0放入OPEN表中；
- Step2: 若OPEN表为空，则搜索失败，退出。
- Step3: 移出OPEN中第一个节点n放入CLOSED表中，并标以顺序号n；
- Step4: 若目标节点S_g=n，则搜索成功，结束。
- Step5: 若n不可扩展，则转Step2；
- Step6: 扩展n，将生成的一组子节点配上指向 n 的指针后，放入OPEN表首部，转 Step2。

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



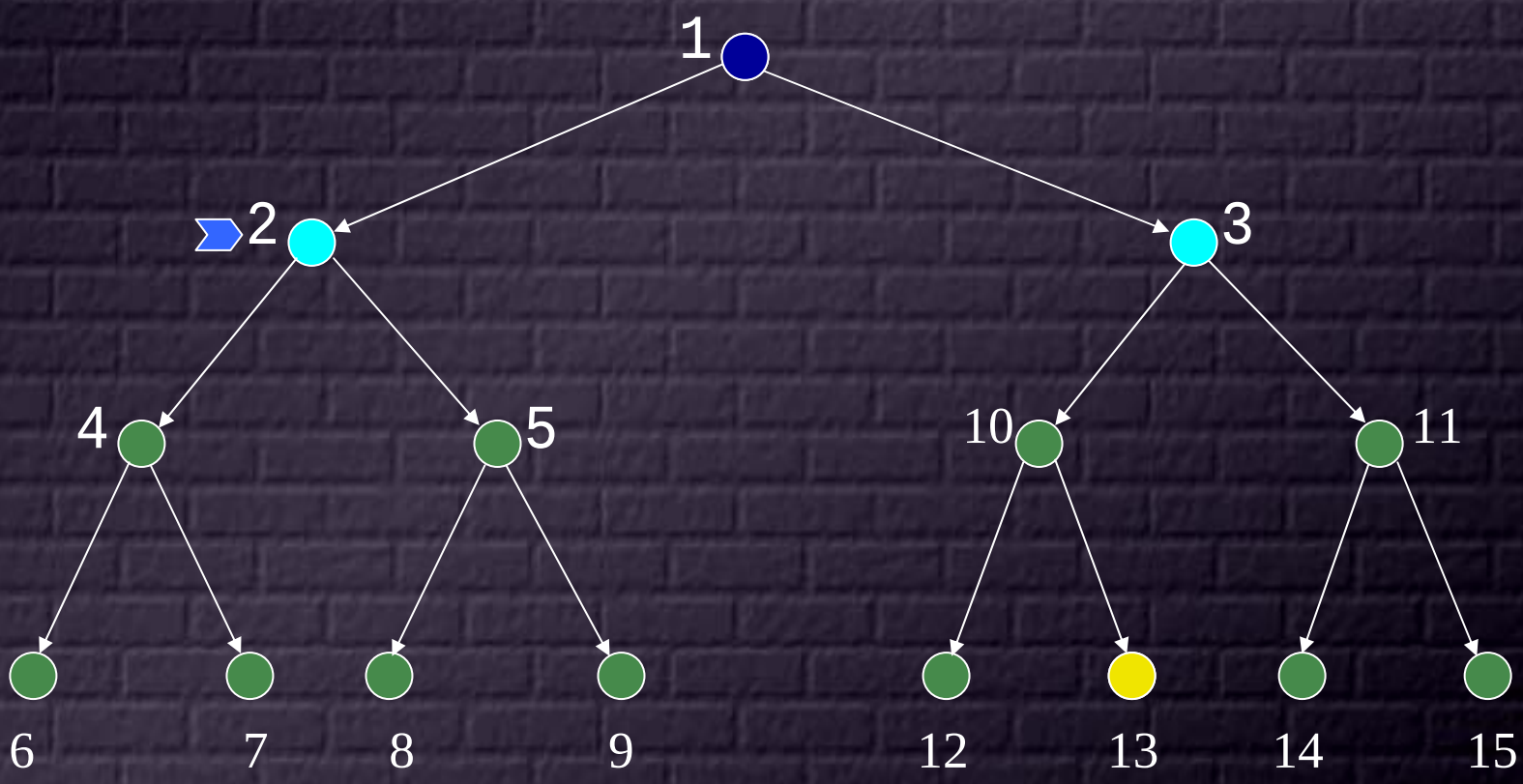
OPEN= (1)

CLOSED=()

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



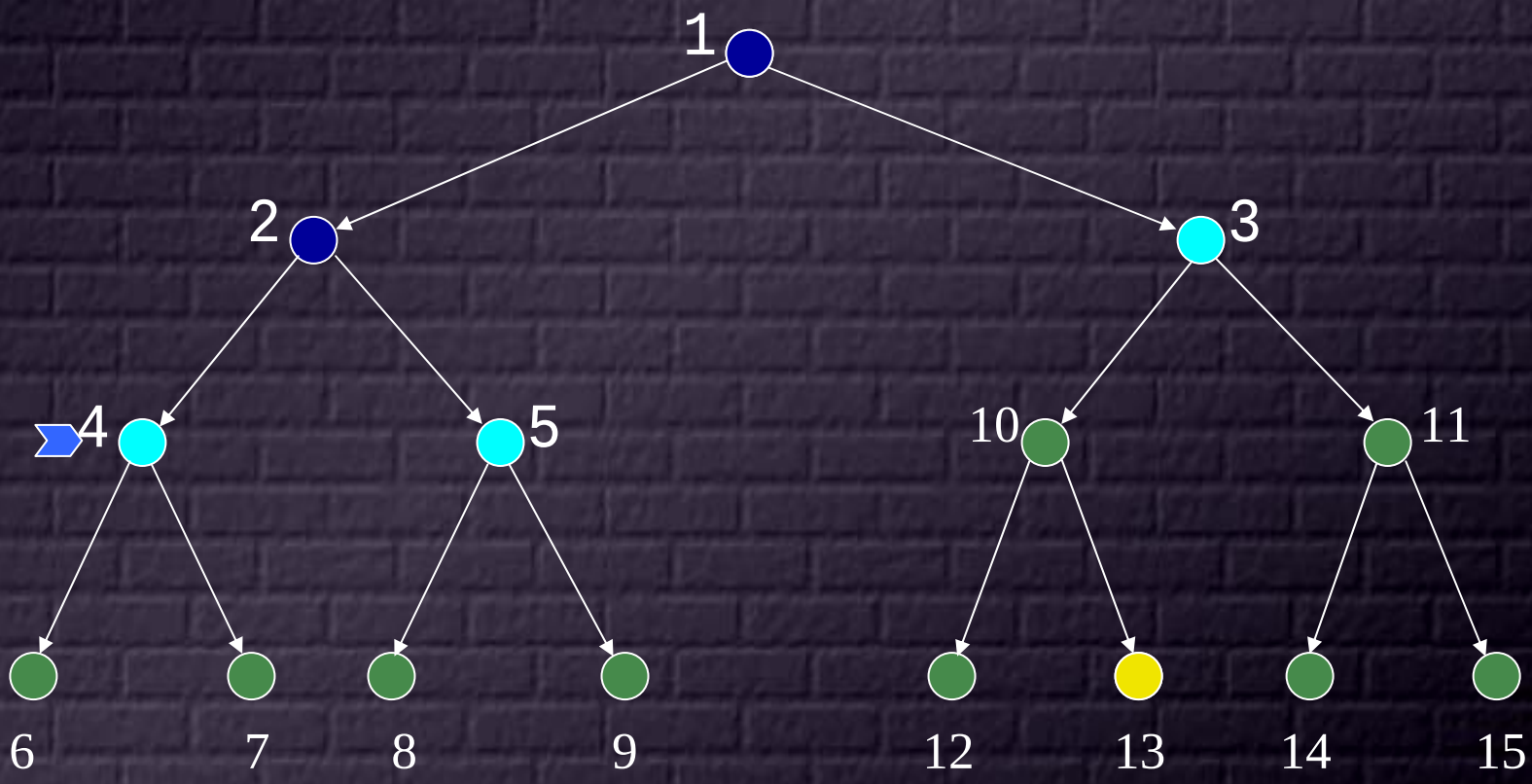
OPEN= (2、3)

CLOSED=(1)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



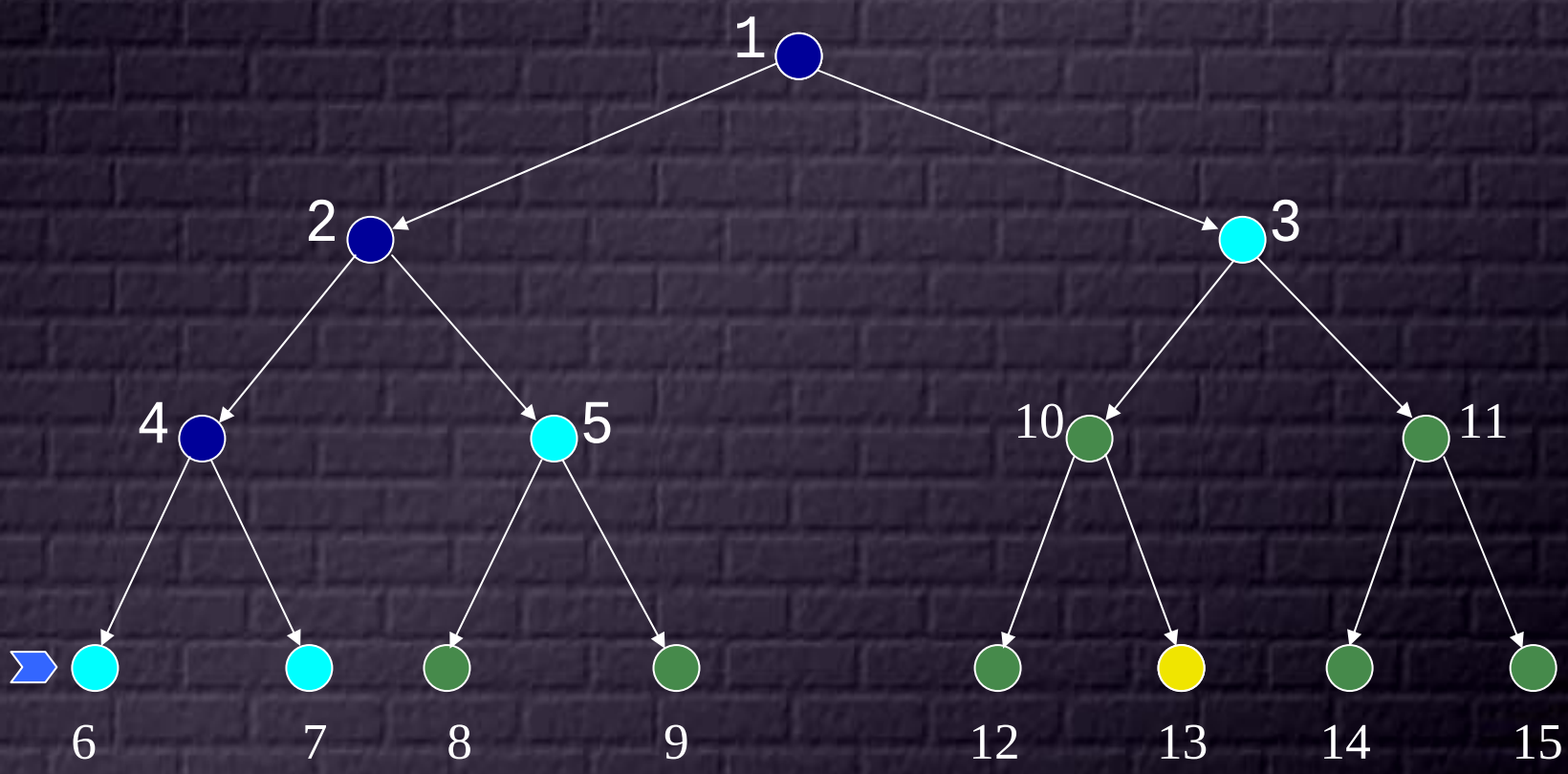
OPEN= (4、5、3)

CLOSED=(1、2)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



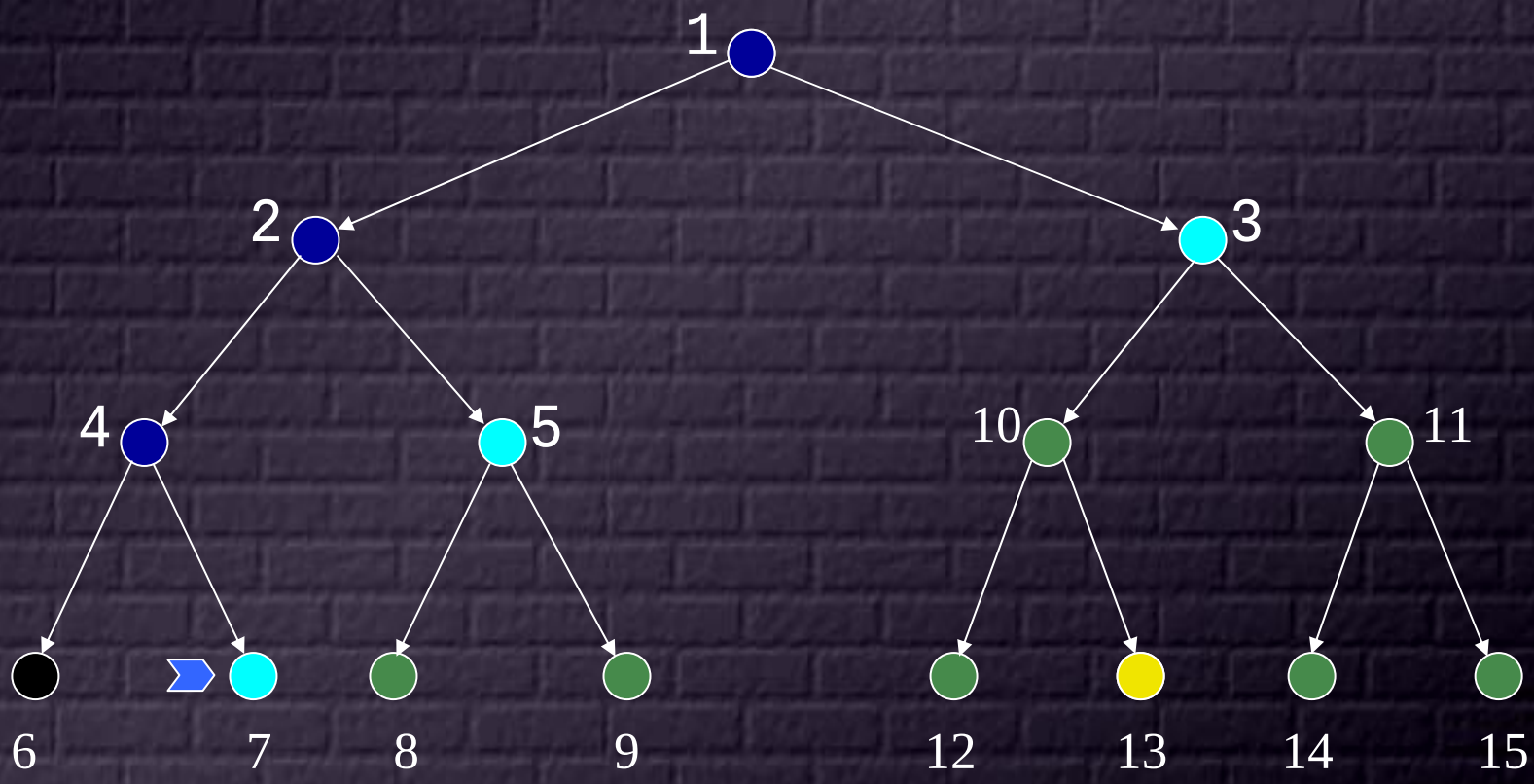
OPEN= (6、7、5、3)

CLOSED=(1、2、4)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



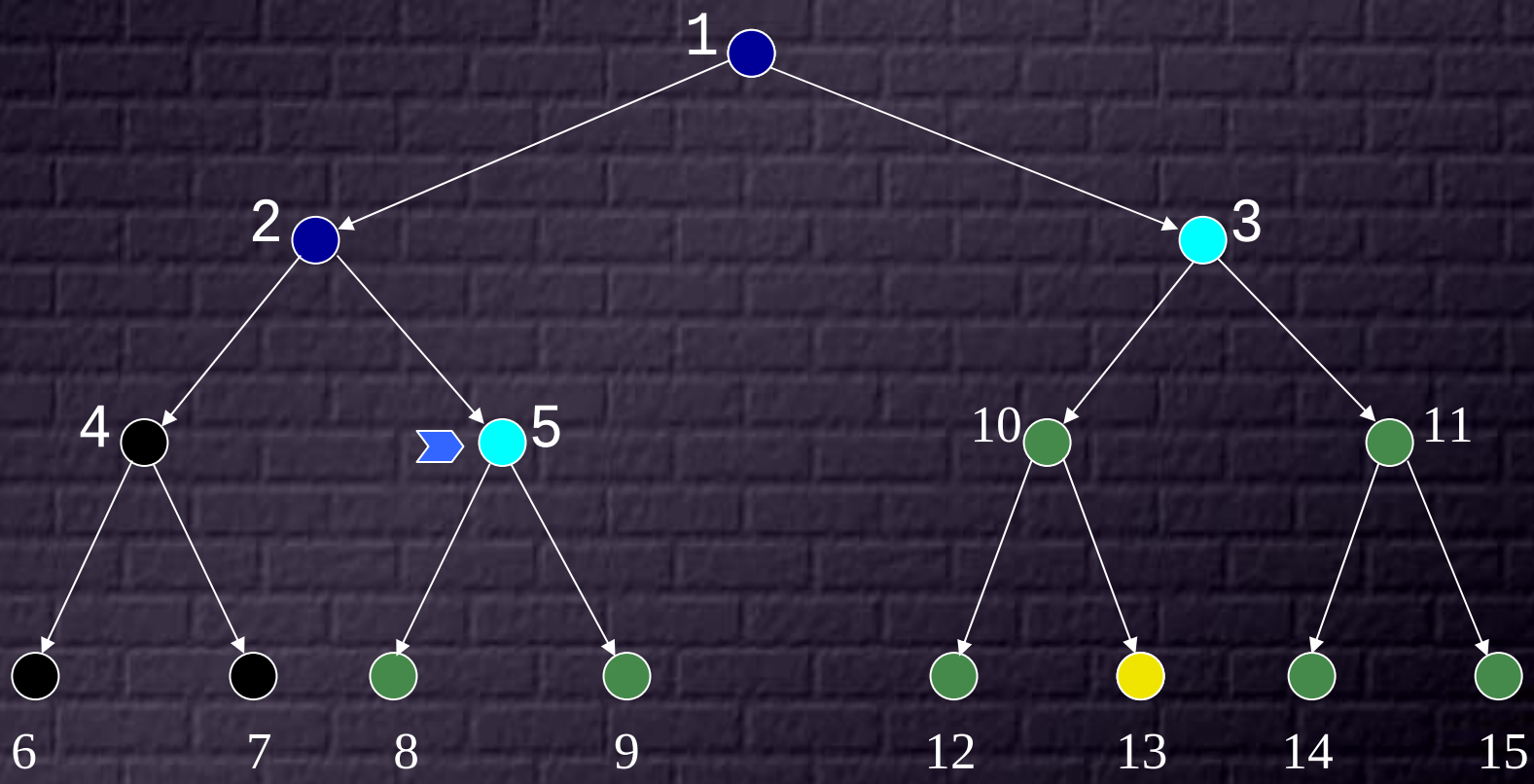
OPEN= (7、5、3)

CLOSED=(1、2、4、6)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



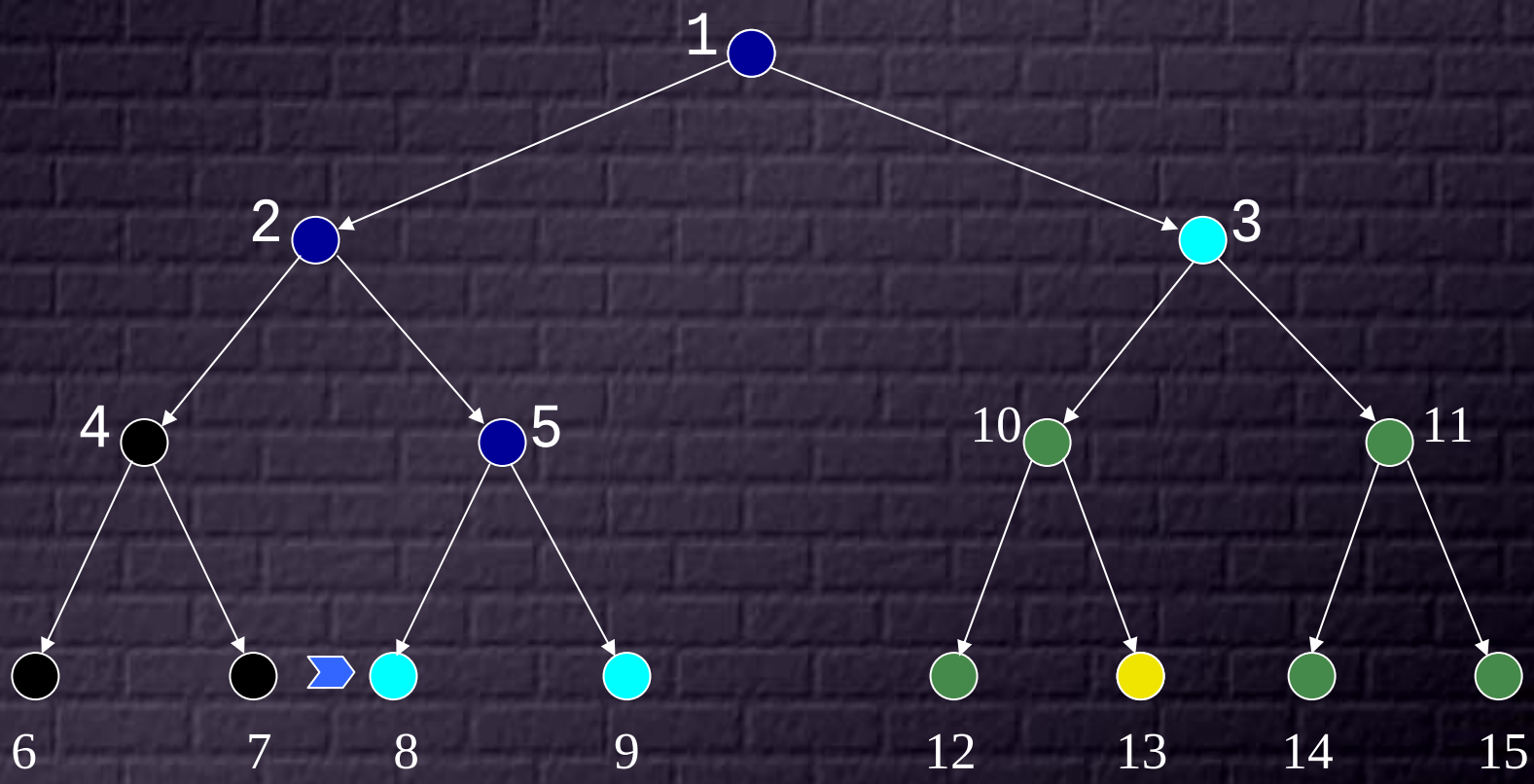
OPEN= (5、3)

CLOSED=(1、2、4、6、7)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



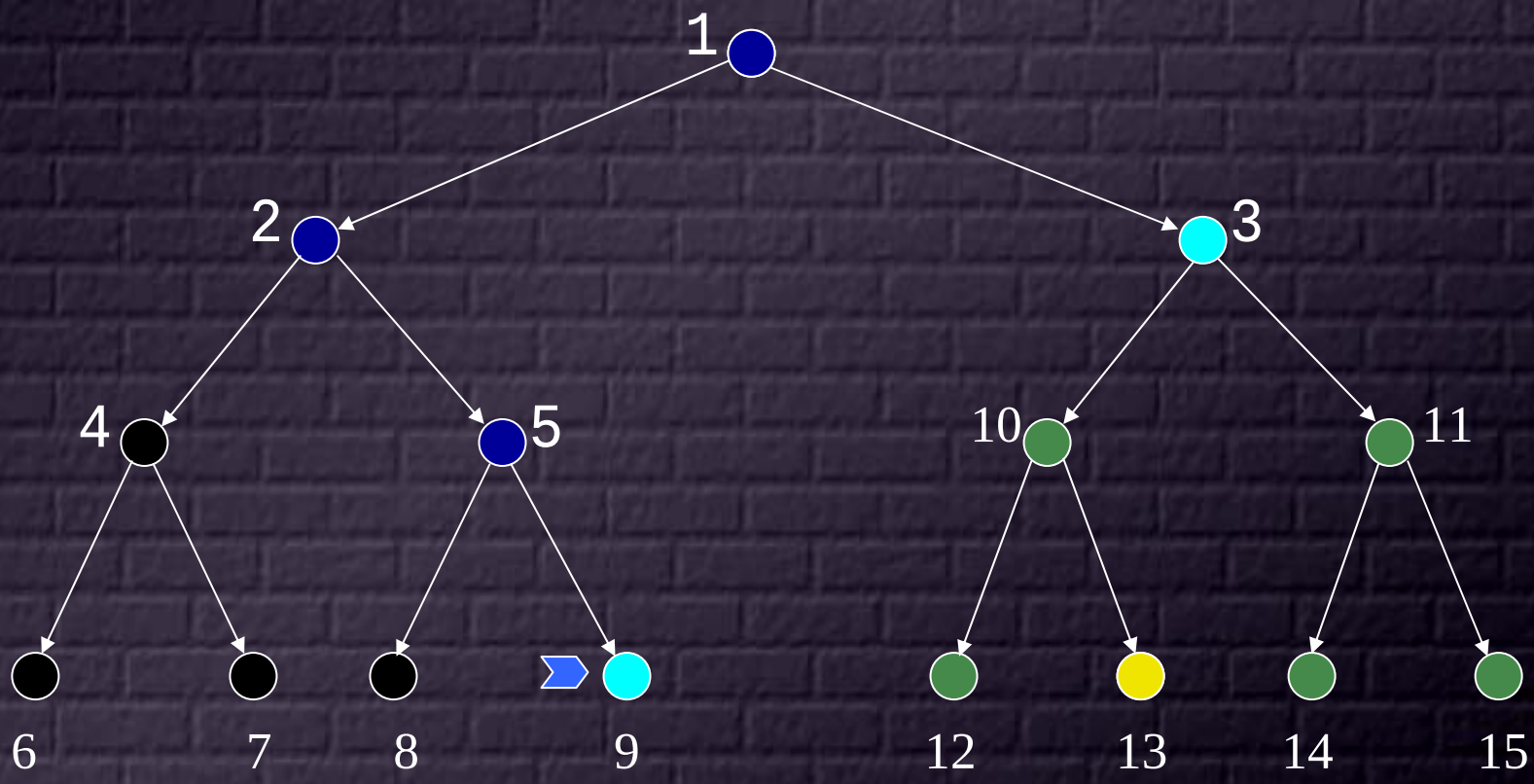
OPEN= (8、9、3)

CLOSED=(1、2、4、6、7、5)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



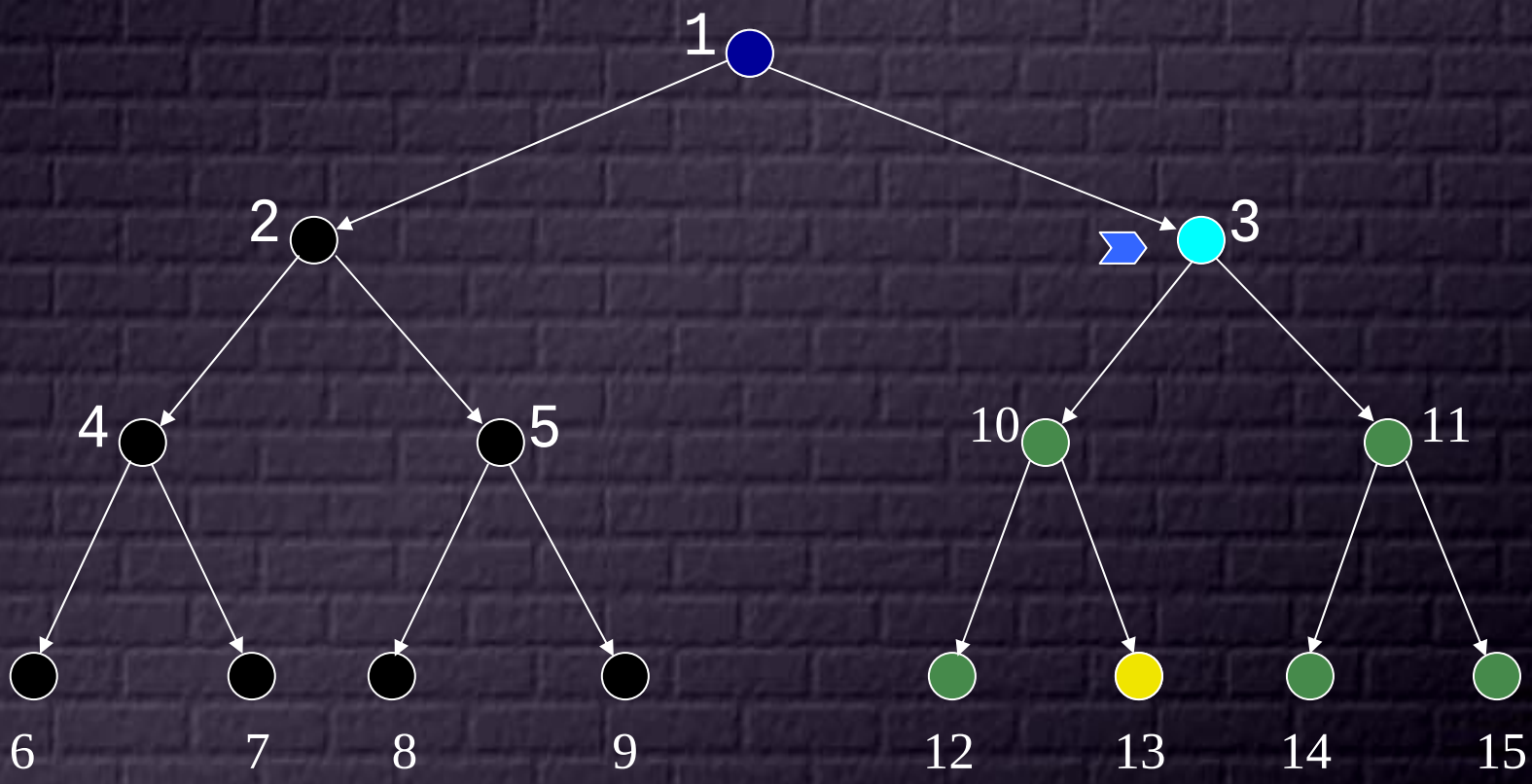
OPEN= (9、3)

CLOSED=(1、2、4、6、7、5、8)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



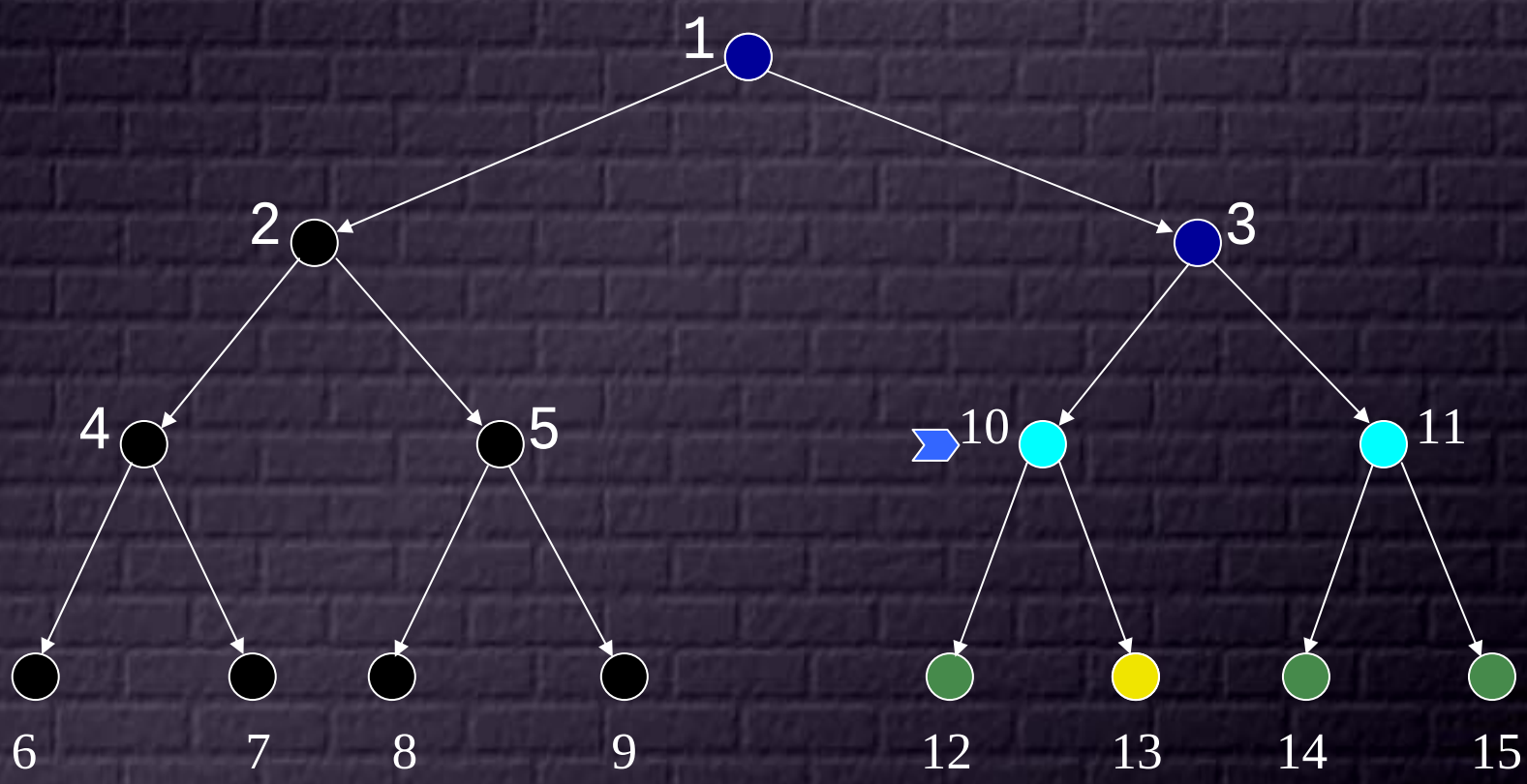
OPEN= (3)

CLOSED=(1、2、4、6、7、5、8、9)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



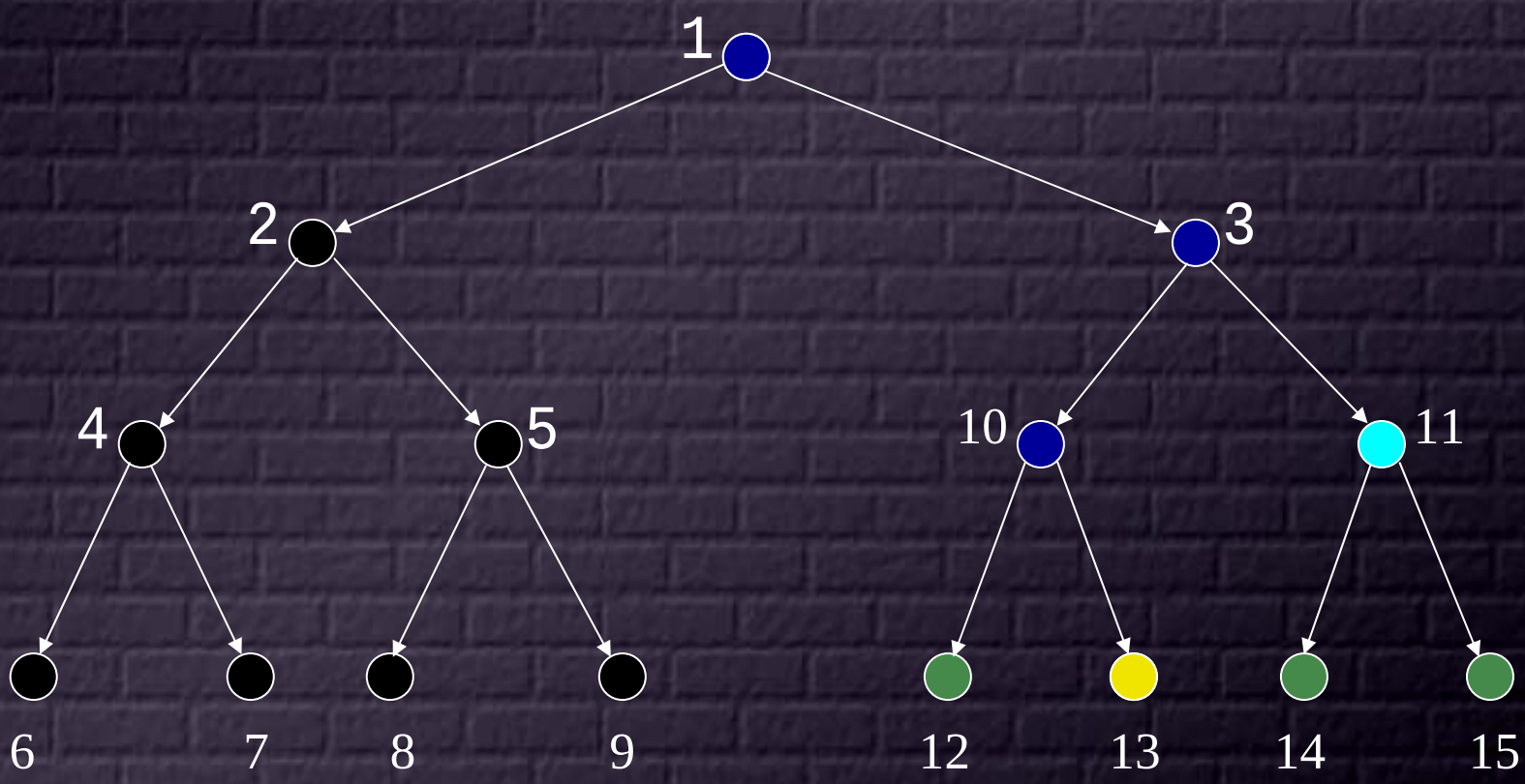
OPEN= (10、11)

CLOSED=(1、2、4、6、7、5、8、9、3)

2、深度优先搜索

搜索过程

新的节点被插入到栈OPEN的前部



OPEN= (12、13、11)

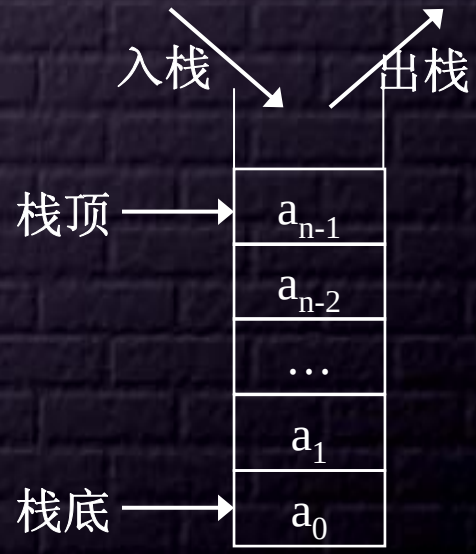
CLOSED=(1、2、4、6、7、5、8、9、3、10)

2、深度优先搜索

预备知识——栈

栈（Stack）是一种**特殊**的线性表，只能在表的**某一端**进行插入和删除运算 。

- 栈顶：进行插入和删除的一段。
- 栈底：与栈顶相对应的另一端。
- 入栈：在栈中插入一个元素。
- 出栈：从栈中删除一个元素。
- 空栈：栈中没有元素。



2、深度优先搜索

分书方案

- 问题描述

有A、B、C、D、E 5本书，要分给张、王、刘、赵、钱5位同学，每人只能选1本。每个人都将自己喜爱的书填写在表中。请你设计一个程序，统计让每个人都满意的所有分书方案数。

输入	输出
0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1	2

2、深度优先搜索

全排列问题

- 问题描述

对n个元素进行全排列，输出全部无重复的全排列。 其中n大于0，小于9。

输入	输出
3 a b c	a b c a c b b a c b c a c a b c b a

2、深度优先搜索

背包问题

- 问题描述

设有一个背包，可以放入的重量为s。现有n件物品，重量分别为t_1、t_2、t_3、…、t_i、…t_n，其中t_i ($1 \leqslant i \leqslant n$)均为正整数。从n件物品中挑选若干件（每种物品仅限挑选一次），使得放入背包的重量之和正好为s。如果有符合条件，则输出物品的序号，否则输出“No Solution”。

输入	输出
5 1 6 2 7 5 10	1 3 4

2、深度优先搜索

自然数分解

- 问题描述

给出一个自然数n，把n分解为若干个大于1的自然数之乘积。请编写程序找出所有的分解方案。

输入	输出
20	2 2 5 2 10 4 5

2、深度优先搜索

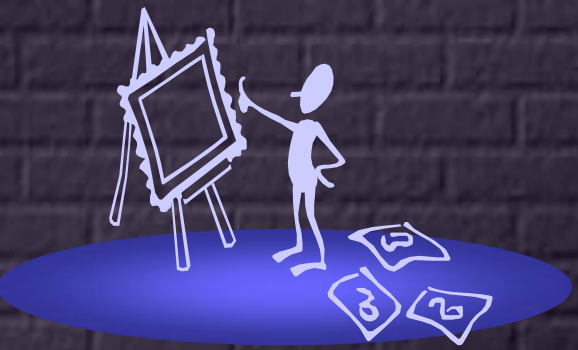
特点分析

- 搜索一旦进入某个分支，就将沿着这个分支一直进行下去；如果目标恰好在这个分支上，则它可以很快找到解。但是，如果目标不在这个分支上，且该分支是一个无穷分支，则搜索过程就**不可能找到解**。
- 深度优先搜索是一种**不完备**的盲目搜索策略。即使问题有解，它也**不一定**能够找到解；即使在能找到解的情况下，按深度优先找到的解也不一定是最优解。
- 搜索所用的时间取决于目标位于**解空间树的位置**。

3、广度优先搜索

- 基本思想

从初始状态So开始，利用规则生成所有可能的状态。构成树的下一层节点，检查是否出现目标状态Sg；若未出现，就对该层所有状态节点，分别顺序利用规则，生成再下一层的所有状态节点，对这一层的所有状态节点检查是否出现Sg；若未出现，继续按上面思想生成再下一层的所有状态节点，这样一层一层往下展开，直到出现目标状态为止。



3、广度优先搜索

八格码问题

- 问题描述

在3×3的方格棋盘上，分别放置了标有数字1、2、3、4、5、6、7和8的八个棋子，其中空出一个位置使棋子可以移动，形成不同的局面。要使棋盘进入某种预定局面应**如何移动**棋子？

S_0

2	8	3
1	□	4
7	6	5

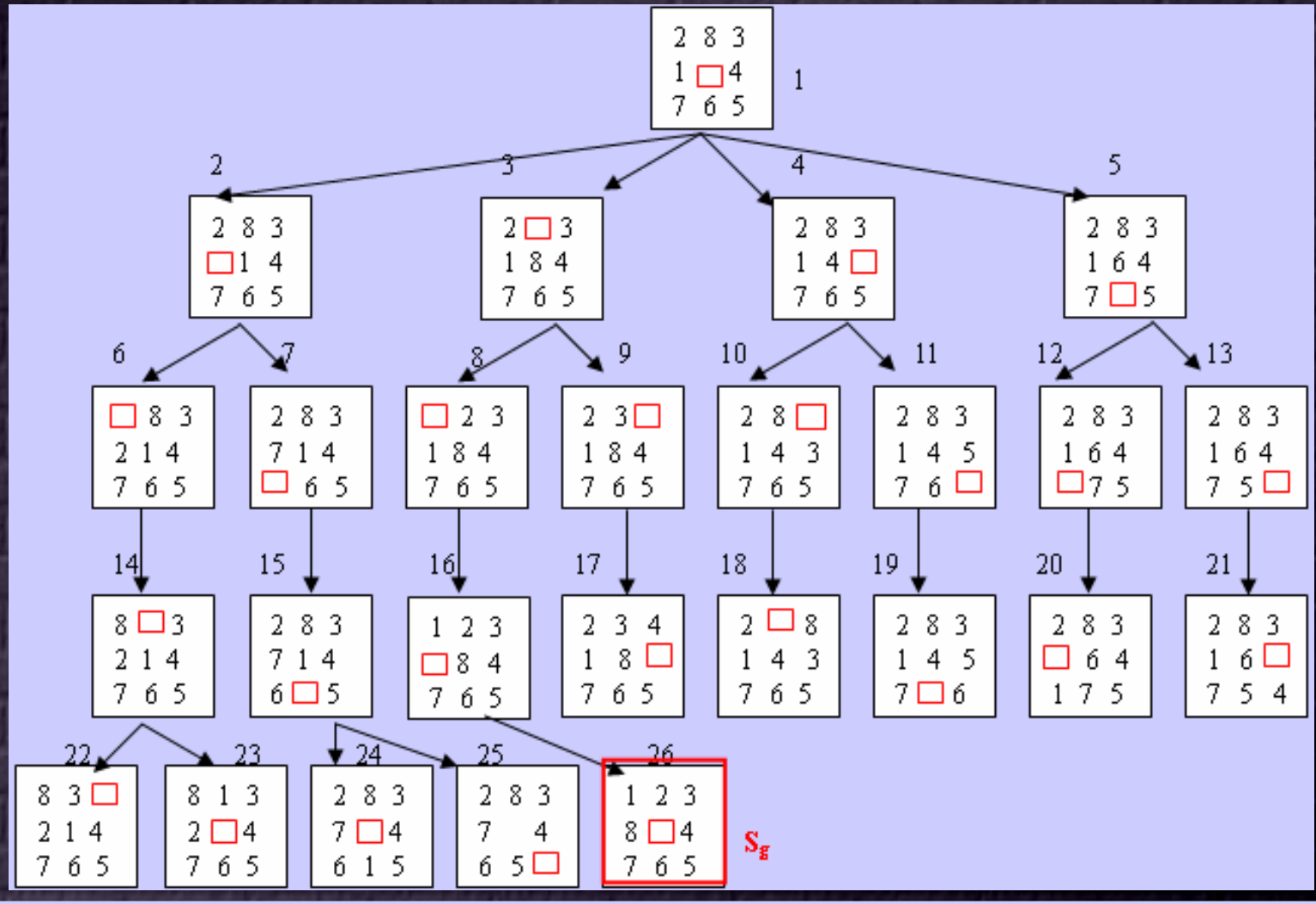
S_g

1	2	3
8	□	4
7	6	5

3、广度优先搜索

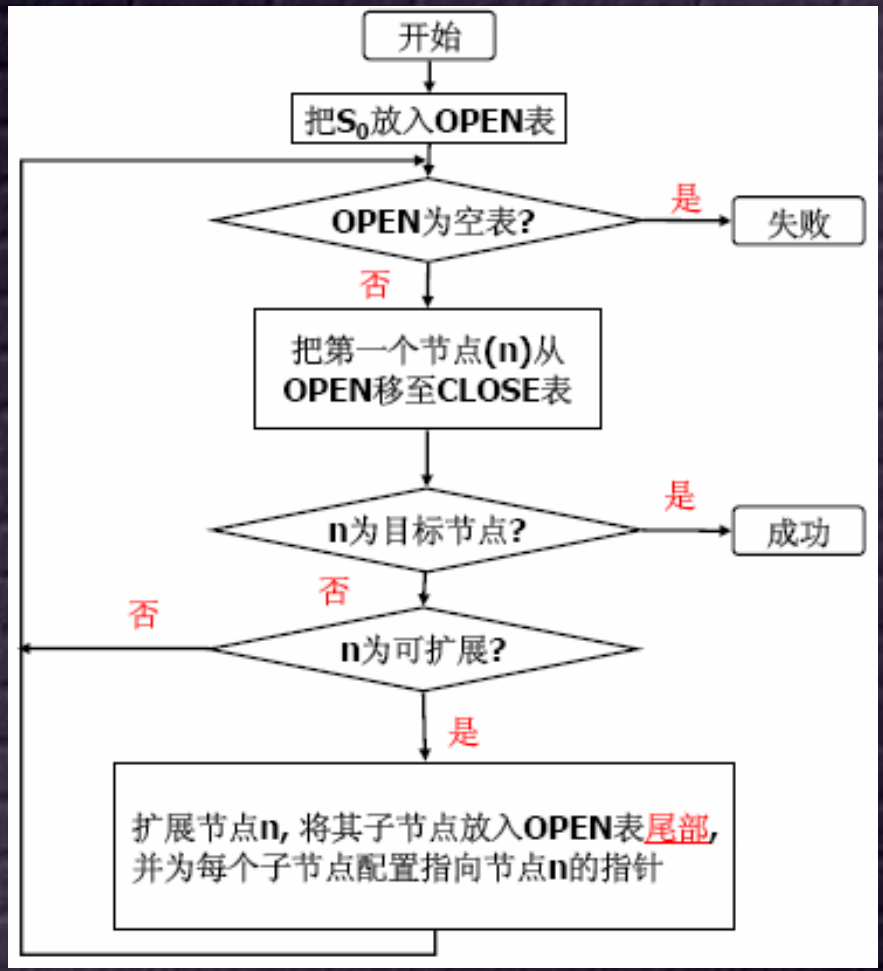
八格码问题

- 广度优先搜索树

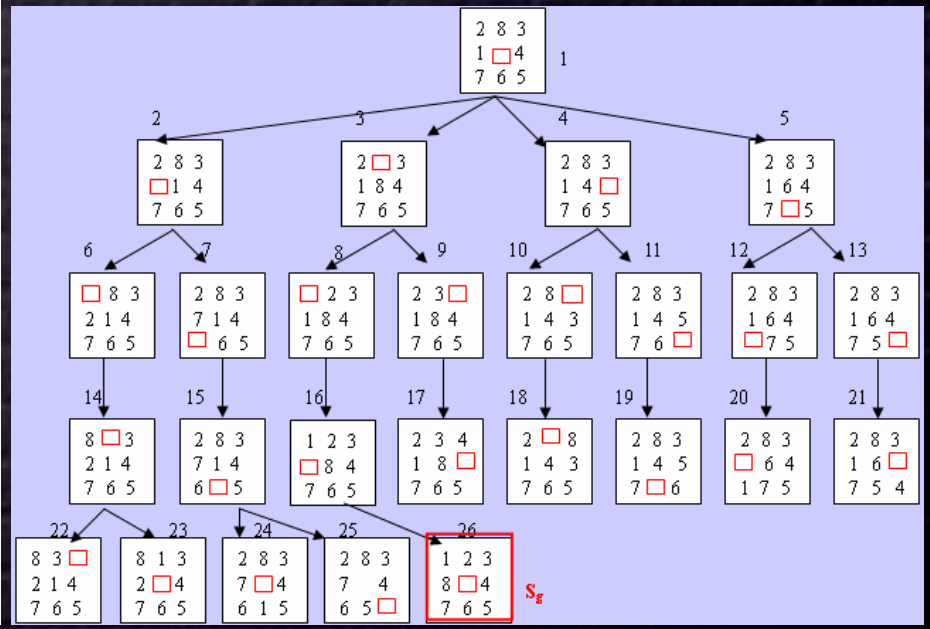


3、广度优先搜索

广度优先搜索算法



Open表是一种**队列**



3、广度优先搜索

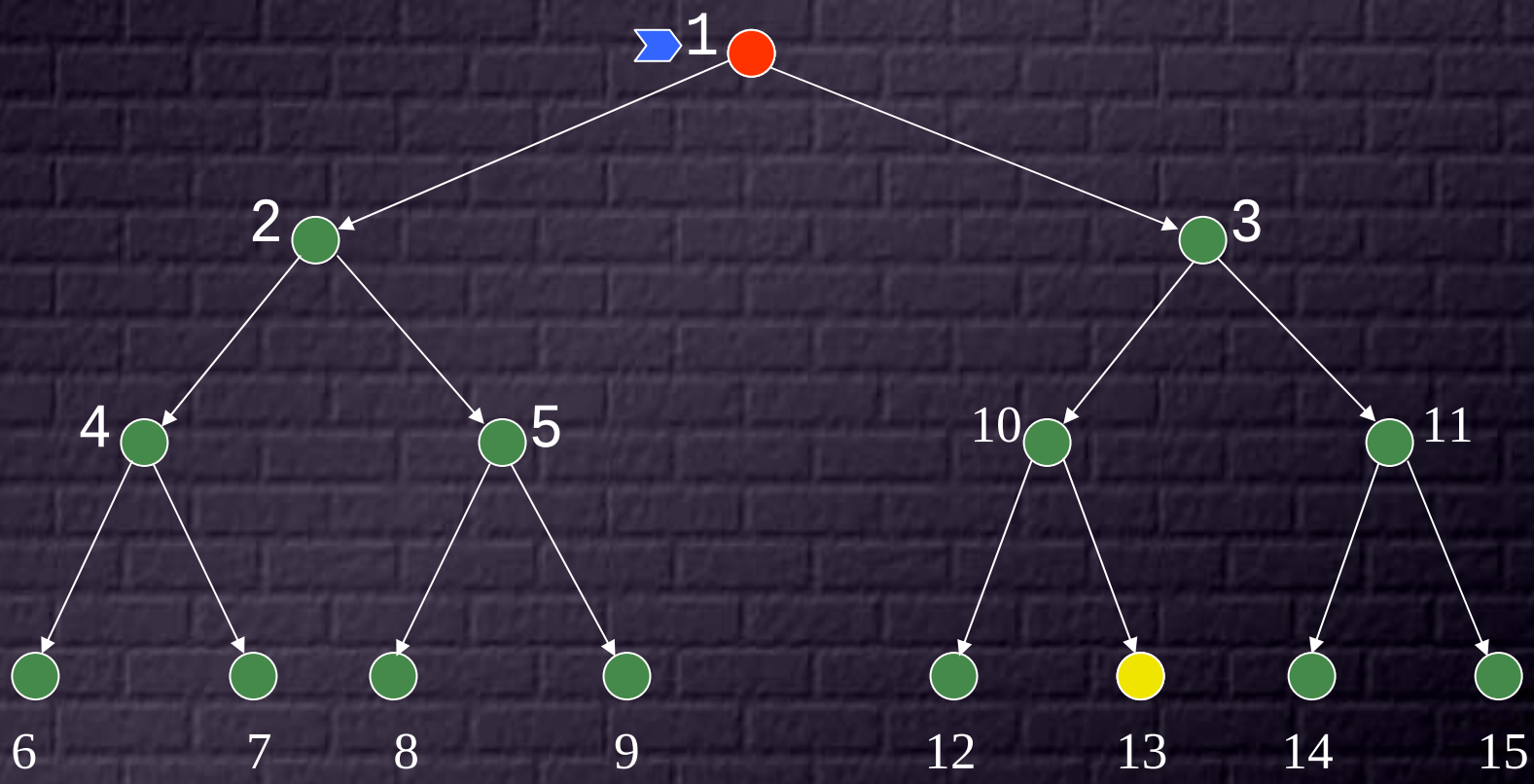
广度优先搜索算法

- Step1: 把初始节点S_0放入OPEN表中；
- Step2: 若OPEN表为空，则搜索失败，退出。
- Step3: 移出OPEN中第一个节点n放入CLOSED表中，并标以顺序号n；
- Step4: 若目标节点S_g=n，则搜索成功，结束。
- Step5: 若n不可扩展，则转Step2；
- Step6: 扩展n，将生成的一组子节点配上指向 n 的指针后，放入OPEN表尾部，转 Step2。

2、广度优先搜索

搜索过程

新的节点被插入到队列OPEN的尾部



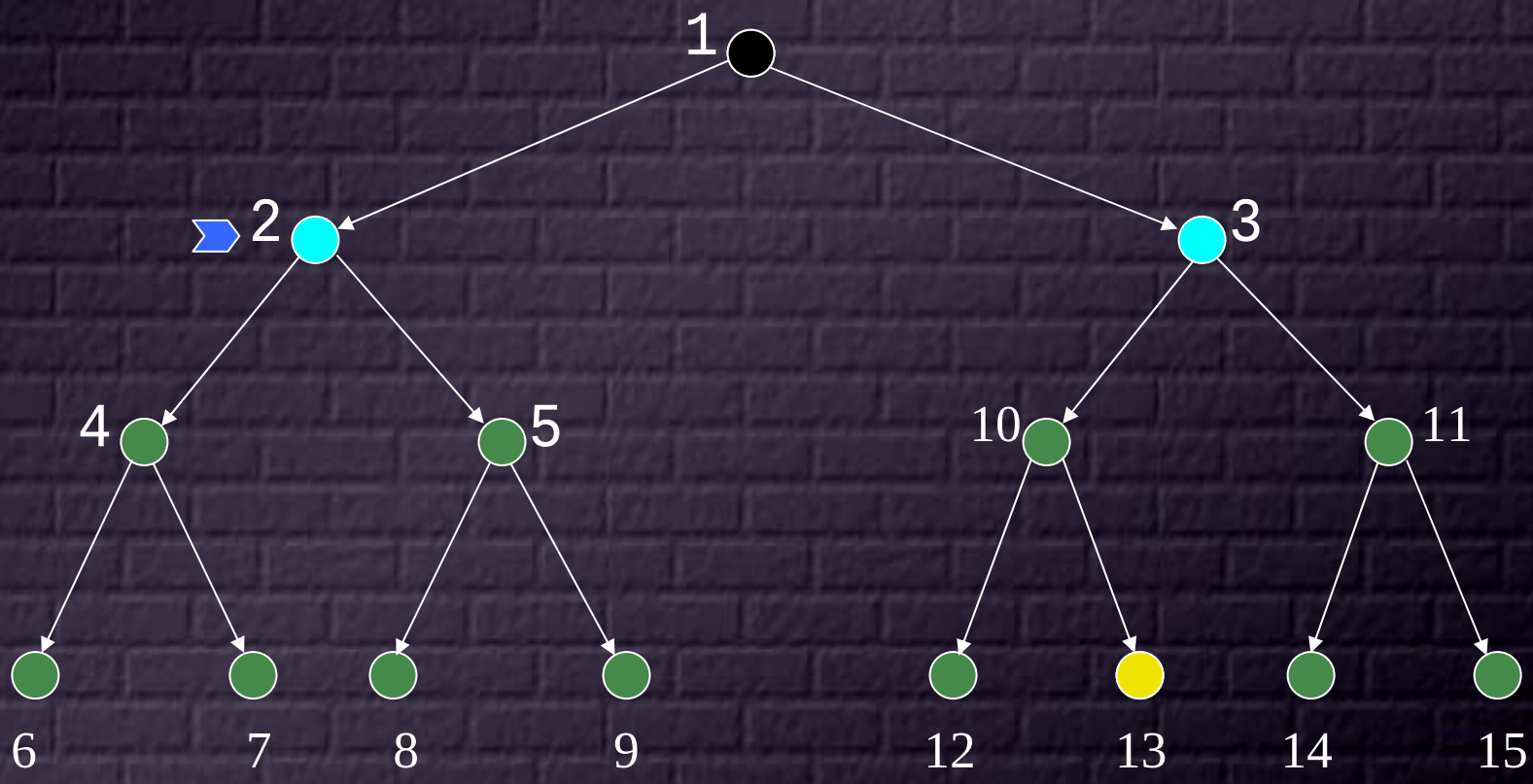
OPEN= (1)

CLOSED=()

2、广度优先搜索

搜索过程

新的节点被插入到队列OPEN的尾部



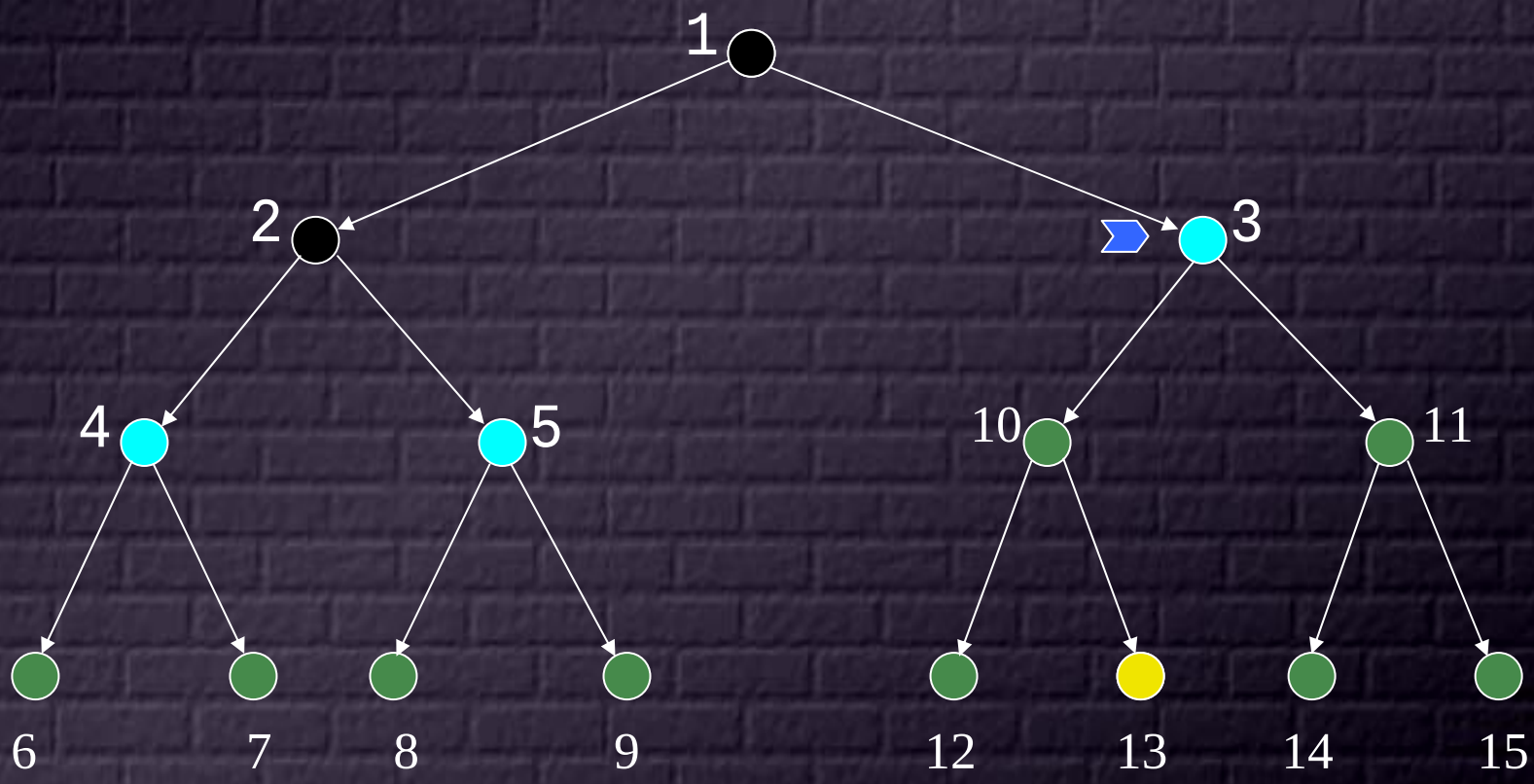
OPEN= (2、3)

CLOSED=(1)

2、广度优先搜索

新的节点被插入到队列OPEN的尾部

搜索过程



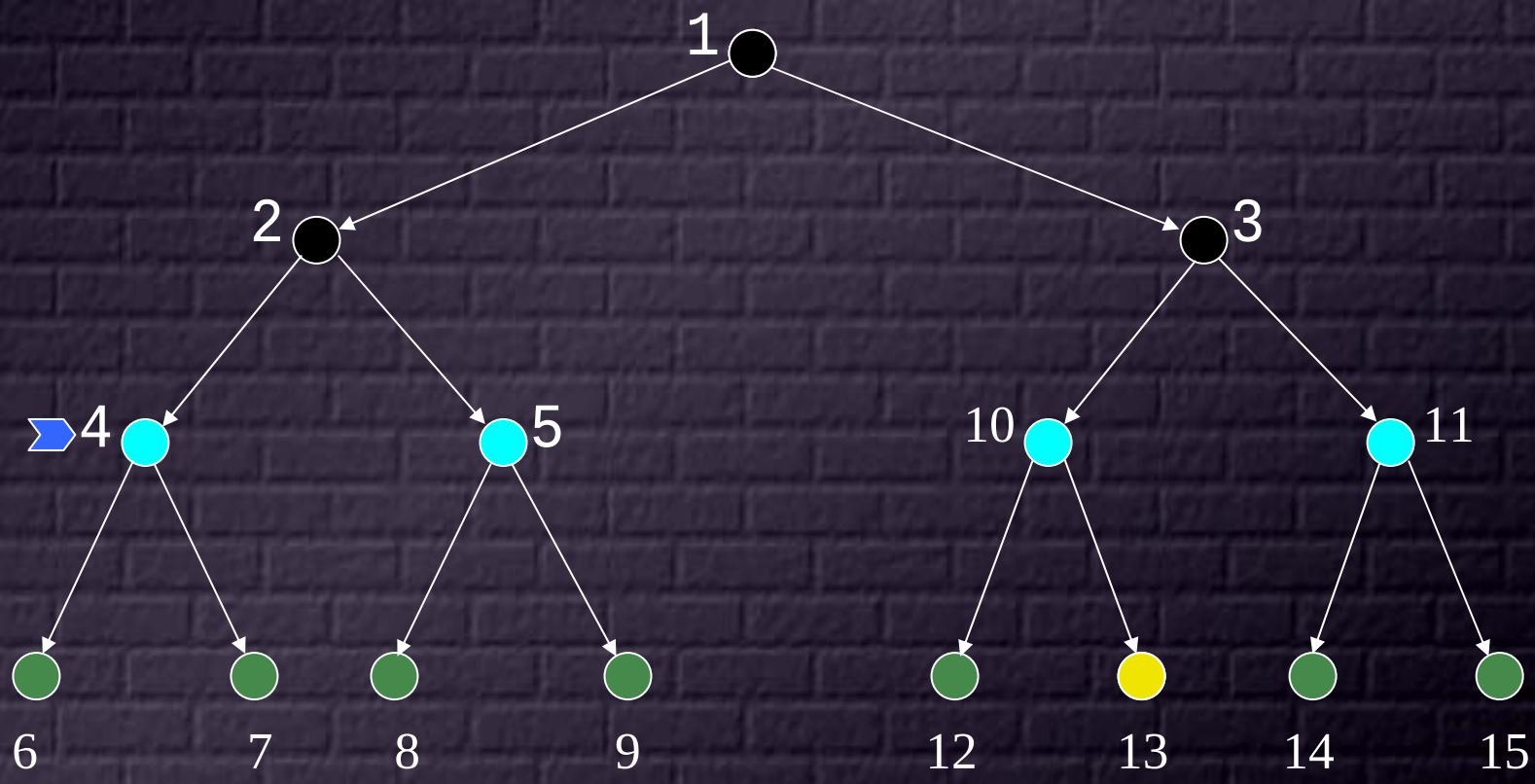
OPEN= (3、4、5)

CLOSED=(1、2)

2、广度优先搜索

搜索过程

新的节点被插入到队列OPEN的尾部



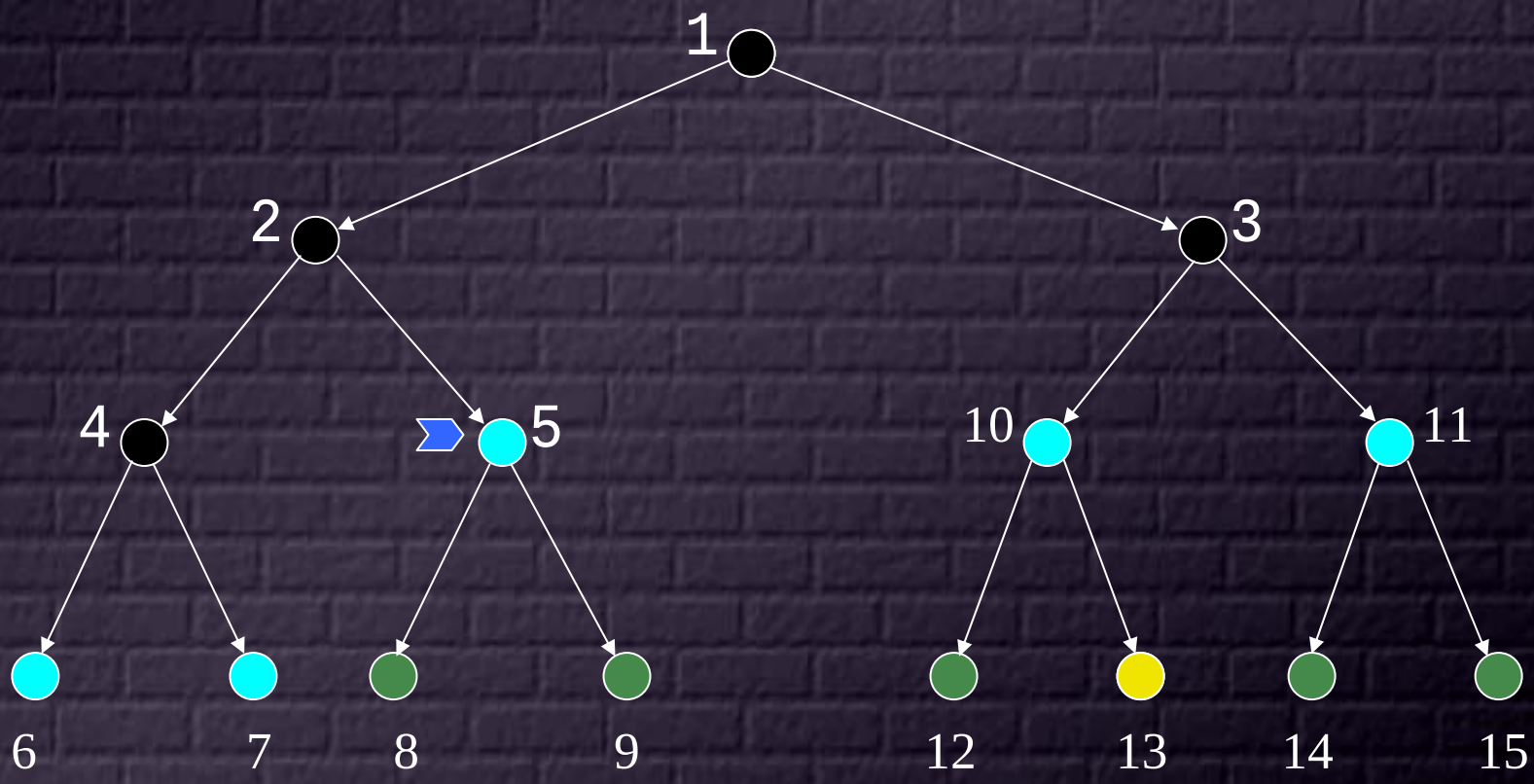
OPEN= (4、5、10、11)

CLOSED=(1、2、3)

2、广度优先搜索

搜索过程

新的节点被插入到队列OPEN的尾部



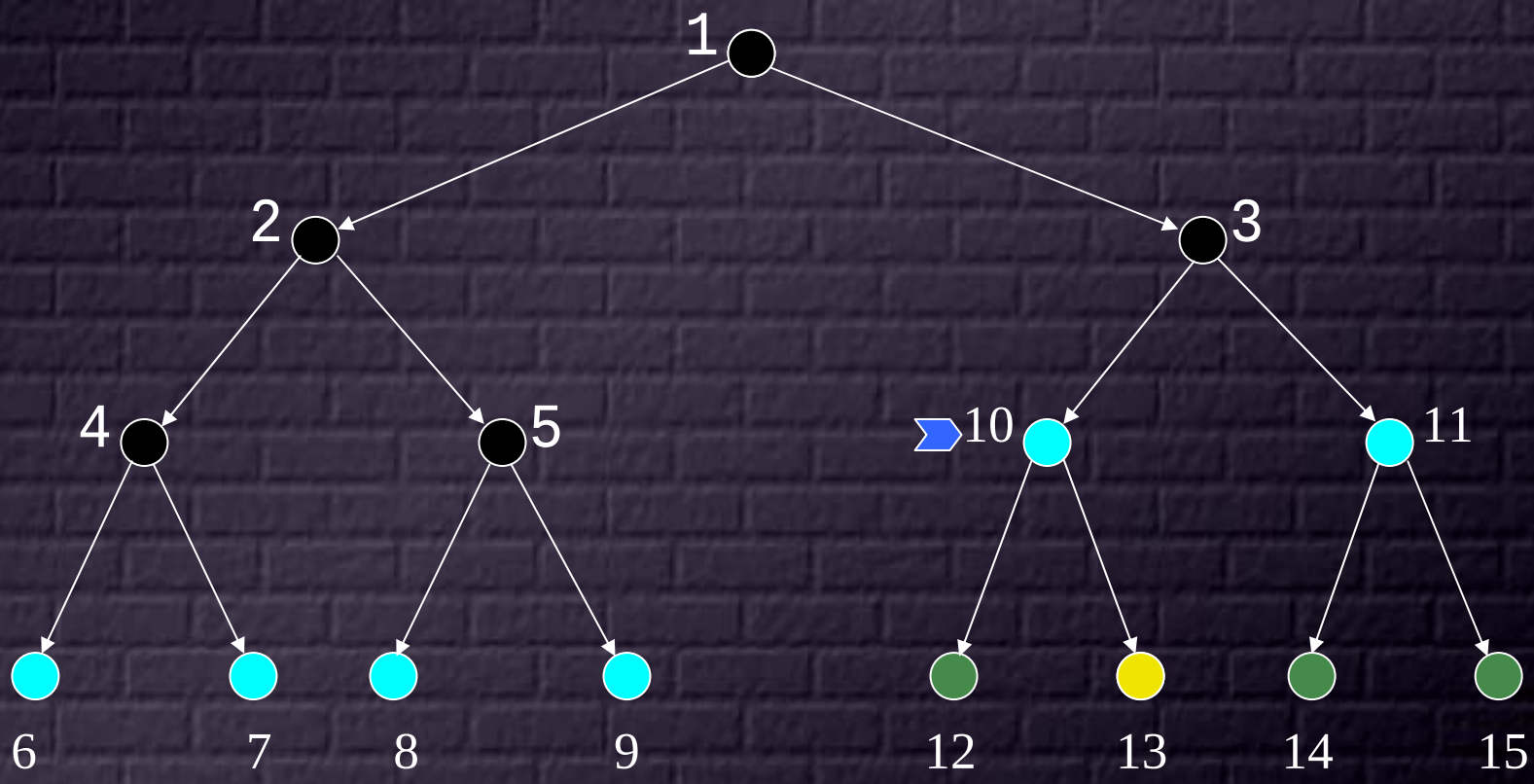
OPEN= (5、10、11、6、7)

CLOSED=(1、2、3、4)

2、广度优先搜索

新的节点被插入到队列OPEN的尾部

搜索过程



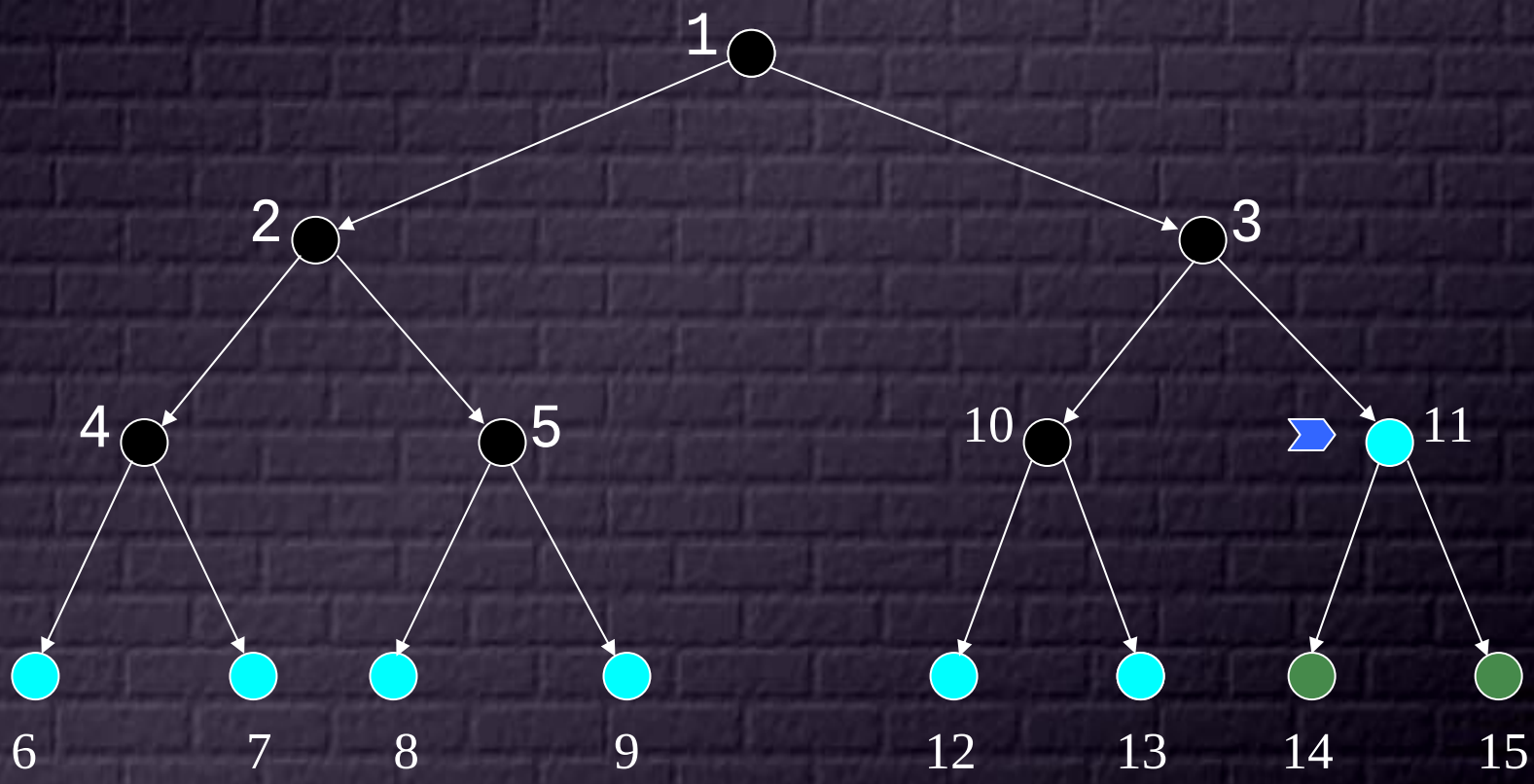
OPEN= (10、11、6、7、8、9)

CLOSED=(1、2、3、4、5)

2、广度优先搜索

搜索过程

新的节点被插入到队列OPEN的尾部



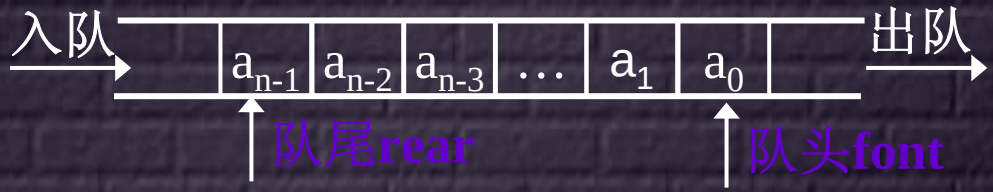
OPEN= (11、6、7、8、9、12、13)

CLOSED=(1、2、3、4、5、10)

2、广度优先搜索

预备知识——队列

队列（Queue）是一种特殊的线性表，限定插入操作只能在表的一端进行，而删除操作都只能在表的另一端进行。



- 队尾：进行插入的一端；队尾元素：位于队尾的元素
- 队头：进行删除的一端；队头元素：位于队头的元素
- 队列长度：队列中元素的个数；
- 空队列：队列中没有元素；
- 入队：在队列中插入一个元素；
- 出队：在队列中删除一个元素。

3、广度优先搜索

- 特点分析
  - 在产生新的结点时，深度越小的结点越先得到扩展。为了算法便于实现，一般采用队列来存储结点。
  - 如果当前结点到起始结点的费用和结点的深度成正比时，甚至就是费用等于深度，此时用广度优先搜索得到的第1个解一定是最优解。
  - BFS算法一般需要存储大量的结点状态值，而且空间是级数增长的，因此在设计时必须考虑溢出和节省空间的问题。
  - 与DFS相比较，BFS无回溯操作，运行速度比DFS要快些，但BFS占用内存空间大。

3、广度优先搜索

分油问题

- 问题描述

有3个容器，其容量分别为a、b、c，这3个容器的初始油量分别为x1、y1、z1，要求用这三个容器倒油，使得最后在这3个容器中的油量分别为x2、y2和z2，问最少的倒油次数是多少？

输入	输出
10 7 3 10 0 0 5 5 0	9

3、广度优先搜索

位置交换

- 问题描述

给出一个由1、2、3、4、5、6组成的6位数，相邻的两个数字可以交换位置，问最少经过多少次交换，可以到达另一个目标6位数。

输入	输出
123456 231456	2

3、广度优先搜索

硬币翻转

- 问题描述

在桌面上有一排硬币，共n枚，每一枚硬币均为正面向上。现在要把所有的硬币翻转成反面向上，规则是每次可翻转任意n-1枚硬币（正面向上的被翻转为反面向上，反之亦然）。求一个最短的操作序列（将每次翻转n-1枚硬币定为一次操作）。

输入	输出
4	4 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1

4、启发式搜索

启发式搜索是指在控制性知识中增加关于被解问题和相应任务的某些特性，利用启发性信息来确定节点的生成、扩展和搜索顺序，指导搜索朝着最有希望的方向前进的一类搜索方法。

- 特点
  - 多数是深度优先搜索的**改进**，即尽量沿着最有希望的路径，向**深度方向小**范围前进；
  - 在有多条路可走时，会给出该走哪条路径的建议，从而指导搜索过程朝**最有利**的方向前进；
  - 利用问题求解的先验知识，使之**尽快**找到问题的解；
  - 可采用**估值的方法**进行搜索指导；
  - 生成的状态空间小、搜索时间短且效率高、控制性好，**易于得到问题的解**。

4、启发式搜索

- 启发性信息的类型
  - 有效地帮助确定扩展节点的信息，即用于决定应先扩展哪一个节点，以免盲目扩展。
  - 有效地帮助决定哪些后继节点应被生成的信息，即用于决定应生成哪些后继节点，以免盲目地生成过多无用节点。
  - 能决定在扩展一个节点时哪些节点应从搜索树上删除的信息，即用于决定应删除哪些无用节点，以免造成时空浪费。

4、启发式搜索

估价函数：用来估价节点重要性的函数。

- f(n)=g(n)+h(n)
  - g(n)是从初始节点So到节点n的已经实际付出的代价；
  - h(n)是从节点n到目标节点Sg的最优路径的估计代价。

4、启发式搜索

八格码问题

- 问题描述

在3×3的方格棋盘上，分别放置了标有数字1、2、3、4、5、6、7和8的八个棋子，其中空出一个位置使棋子可以移动，形成不同的局面。要使棋盘进入某种预定局面应**如何移动**棋子？

S_0

2	8	3
1		4
7	6	5

S_g

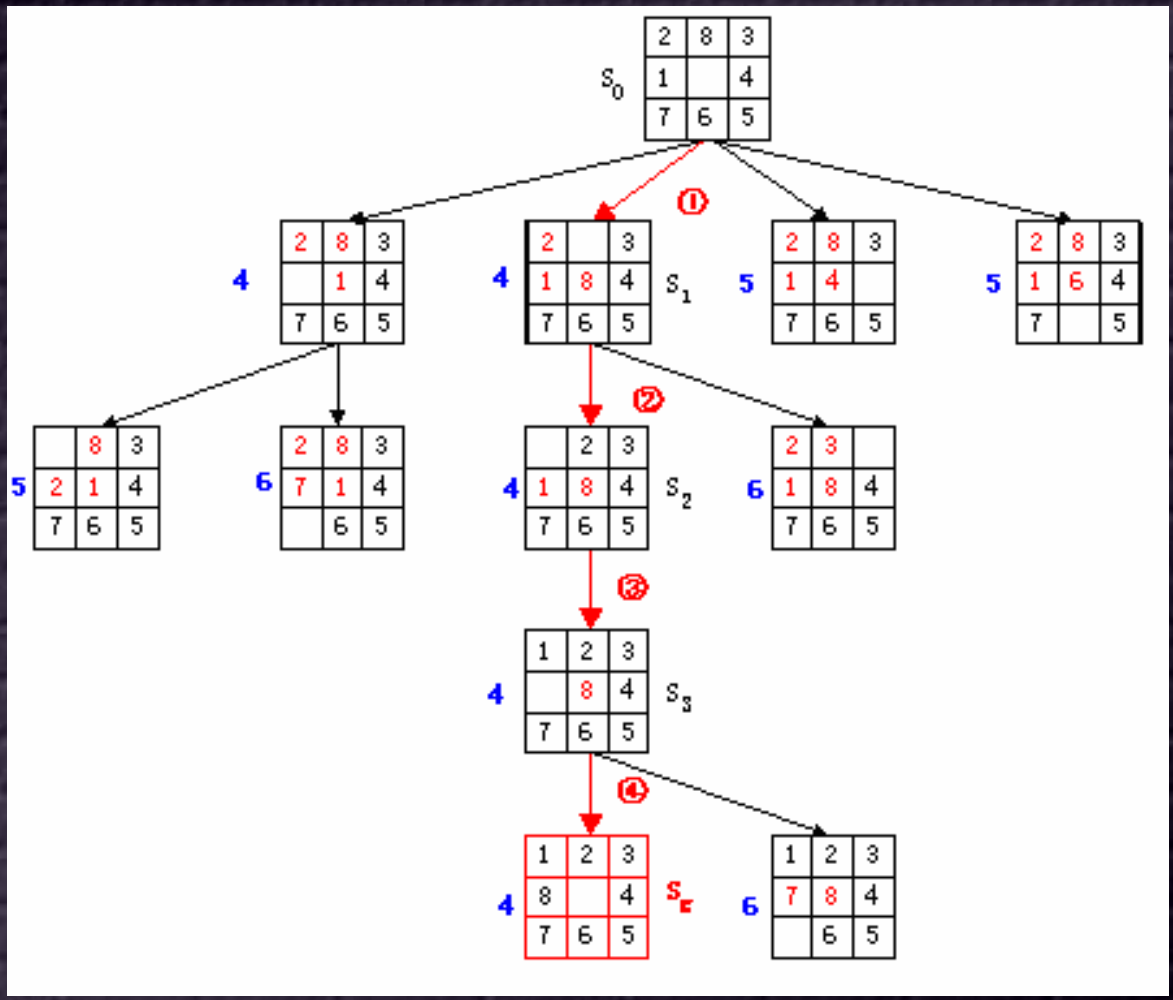
1	2	3
8		4
7	6	5

g(n)表示节点n的搜索深度

h(n)表示节点n与目标节点两个棋局之间位置不相同的棋子数

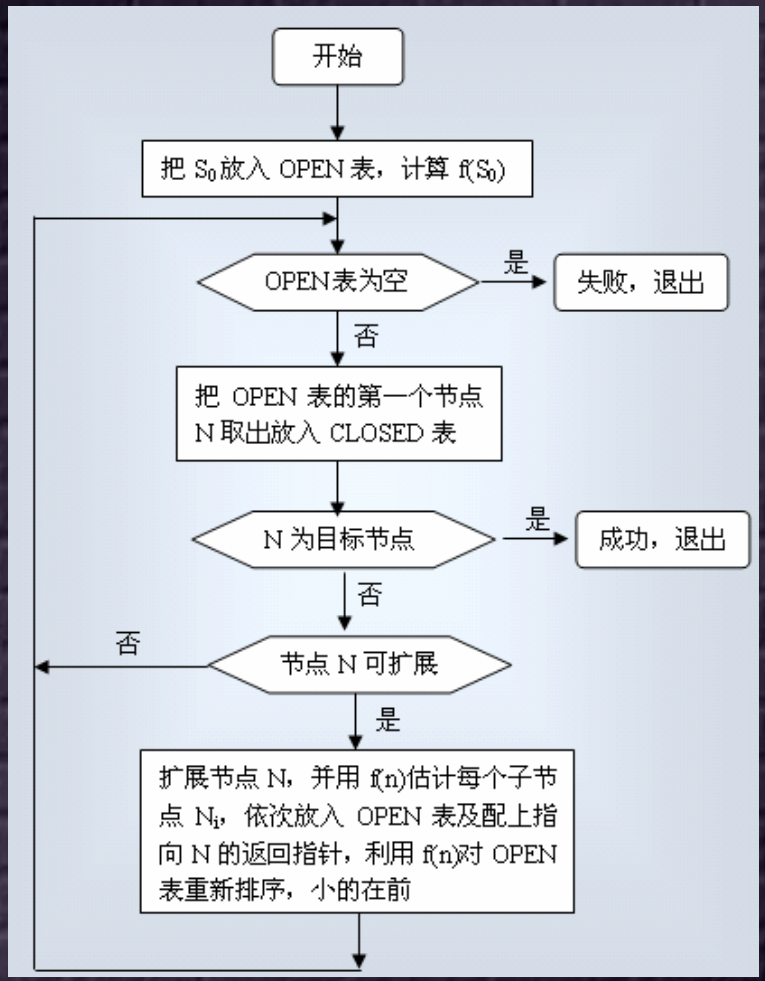
4、启发式搜索

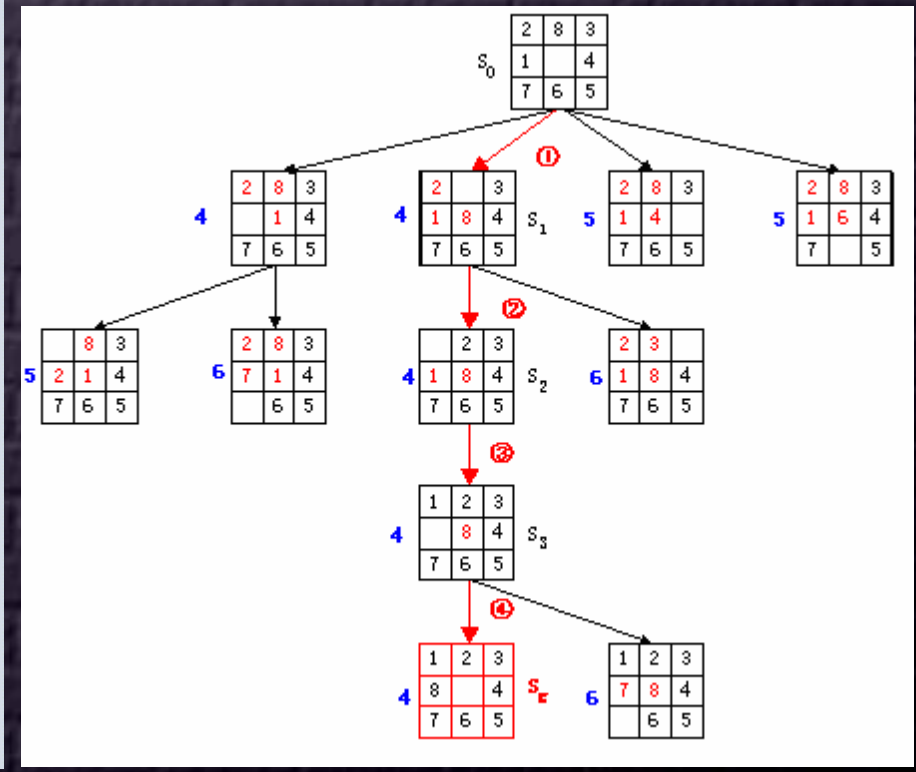
- 搜索树



4、启发式搜索

启发式搜索算法





谢 谢

欢迎批评指正