盲目式搜索
--深度优先搜索

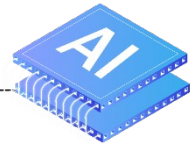
万永权

DFS

盲目搜索

深度优先搜索策略

新节点优先扩展，直到达到一定的深度限制。若找不到目标或无法在扩展时，回溯到另一节点继续扩展。



特点　需要深度限制，需要回溯控制，节省空间。

盲目搜索

深度优先搜索策略

搜索中使用的数据结构

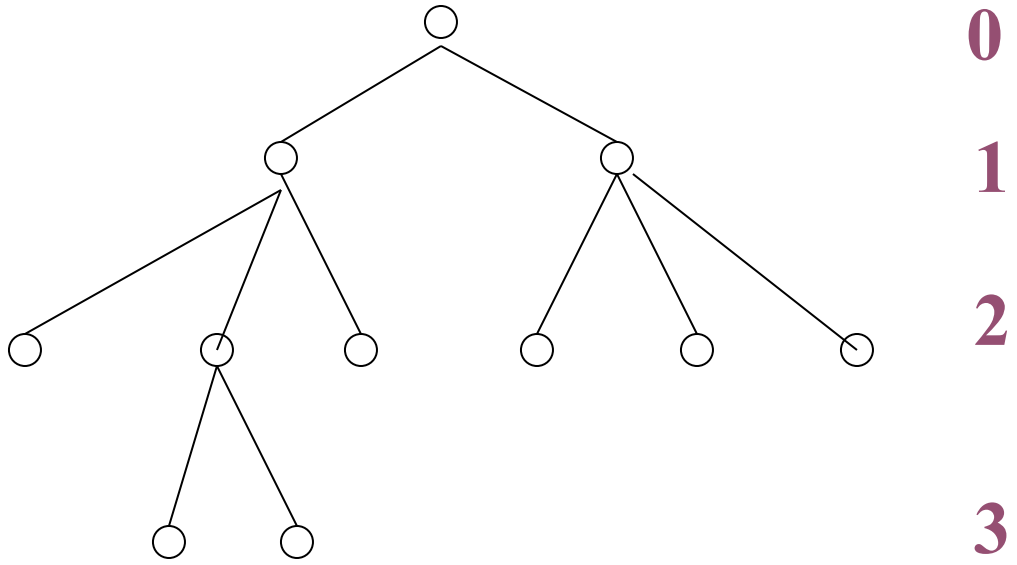
OPEN表　后进先出队列，存放待扩展的节点。

CLOSED表　存放已被扩展过的节点

除扩展后的子节点应放入到OPEN表的最前面之外，其它与宽度优先搜索算法一样。

深度优先搜索---DFS

- **深度优先搜索**(Depth-first Search Algorithm) 是从**图搜索算法**变化来的。
- 深度优先搜索的**基本思想**是**优先扩展当前深度最深的节点，是树的先根遍历**。
- 在一个图中，**初始节点的深度定义为0**，其他节点的深度定义为其父节点的深度加 1。

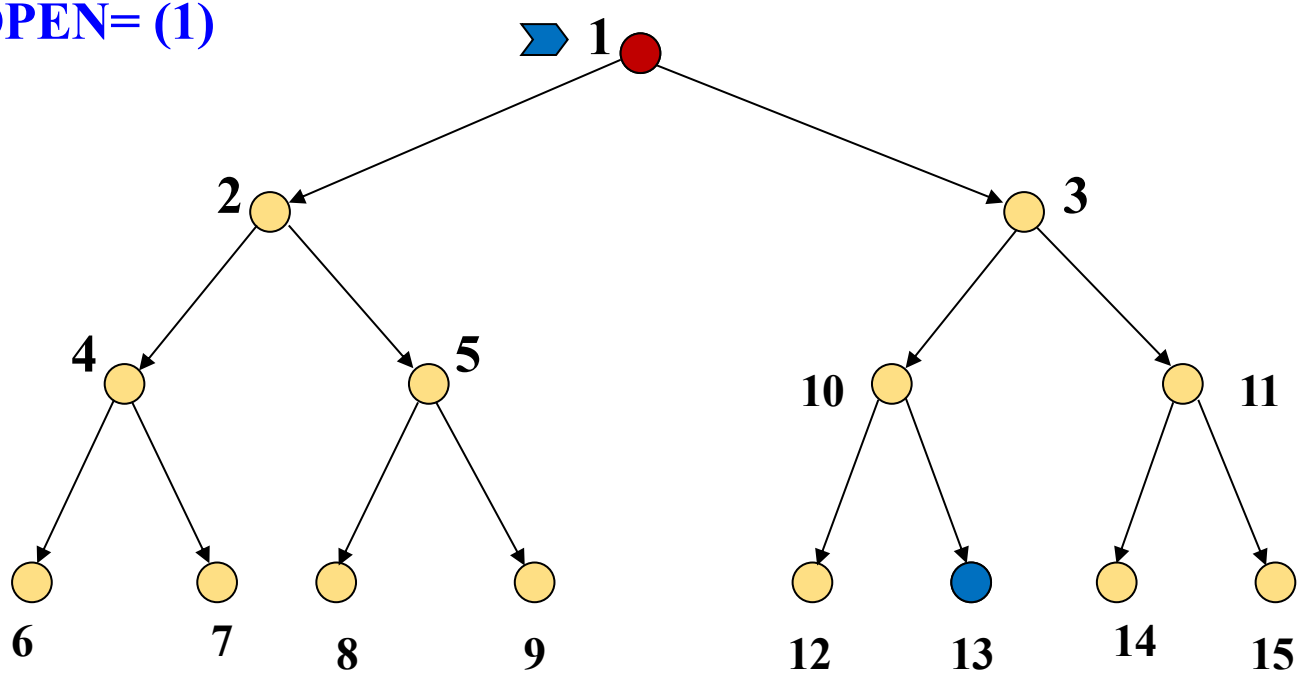


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=( )

OPEN= (1)

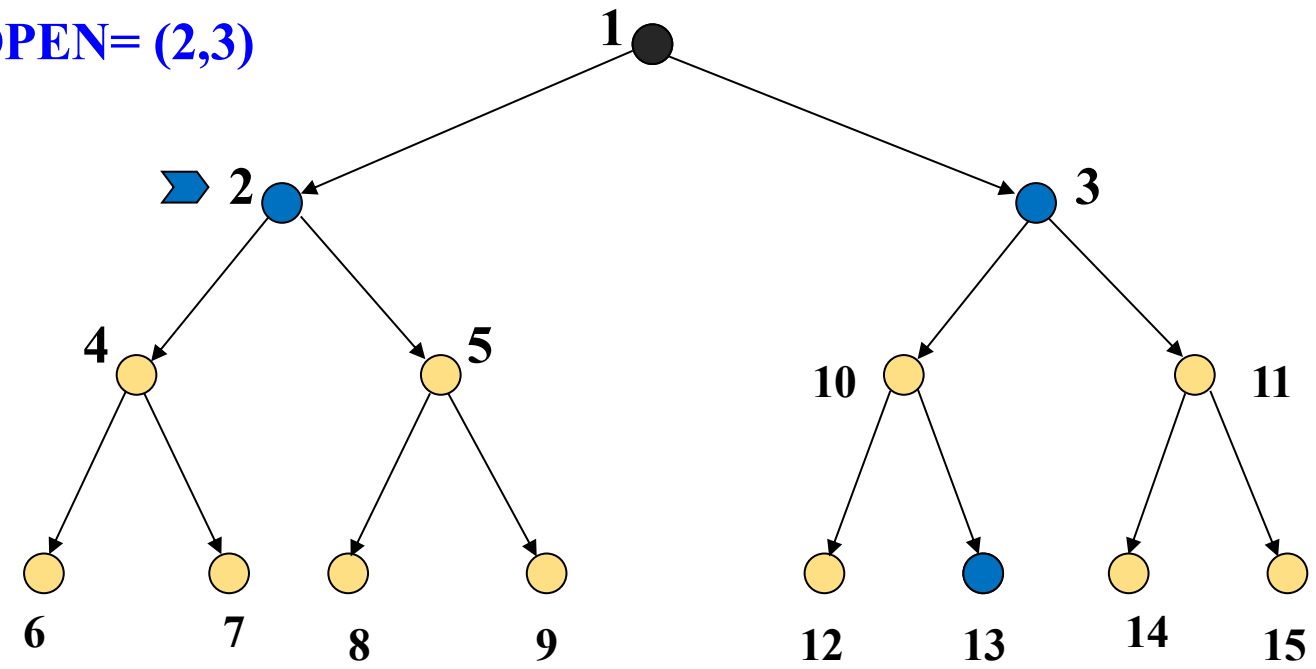


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1)

OPEN= (2,3)

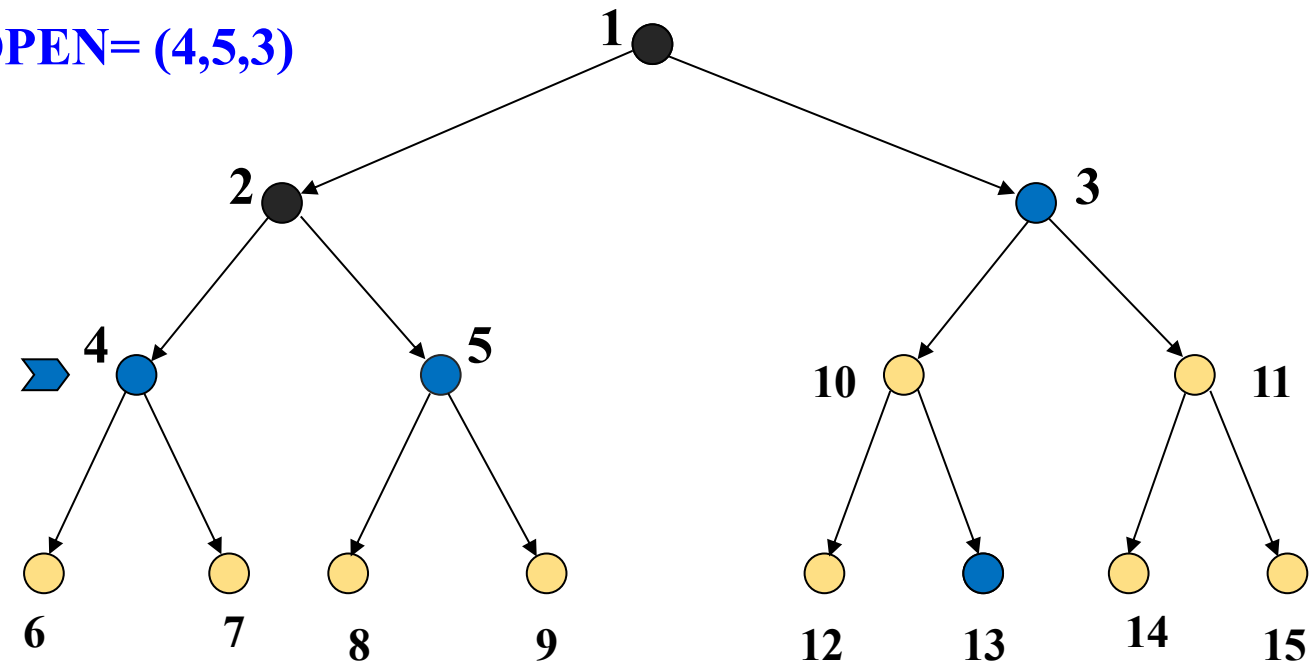


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2)

OPEN= (4,5,3)

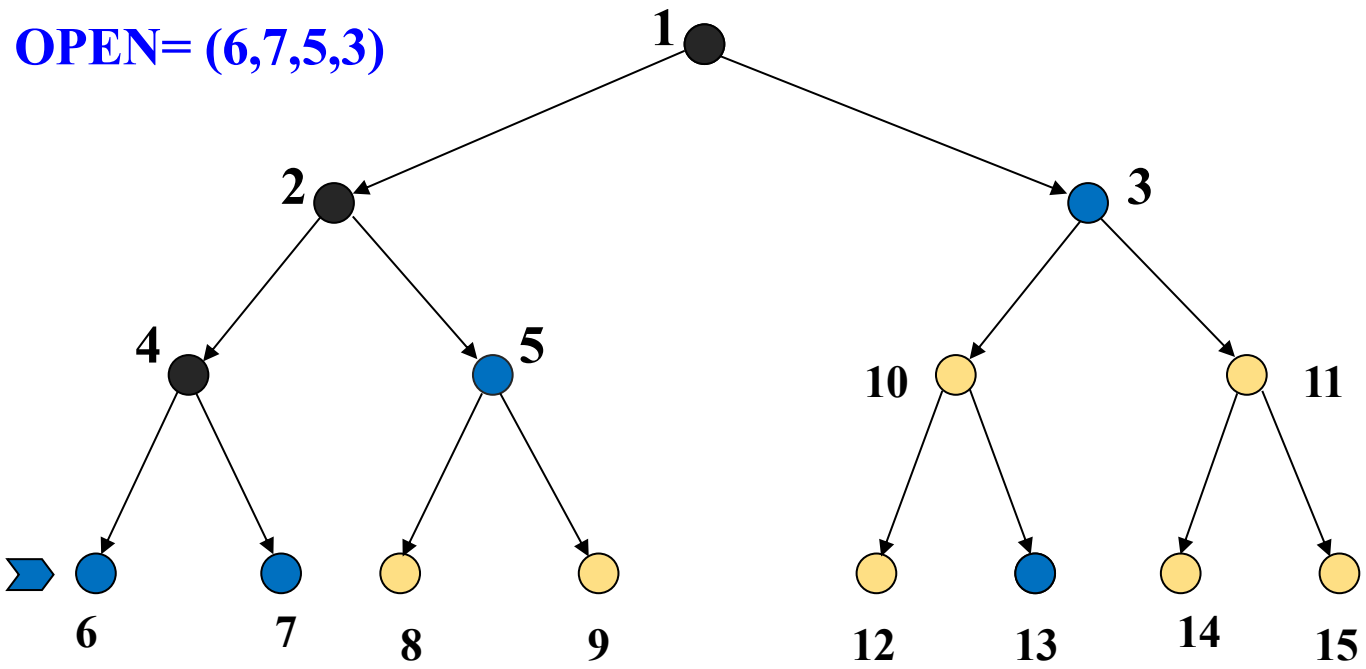


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,4)

OPEN= (6,7,5,3)

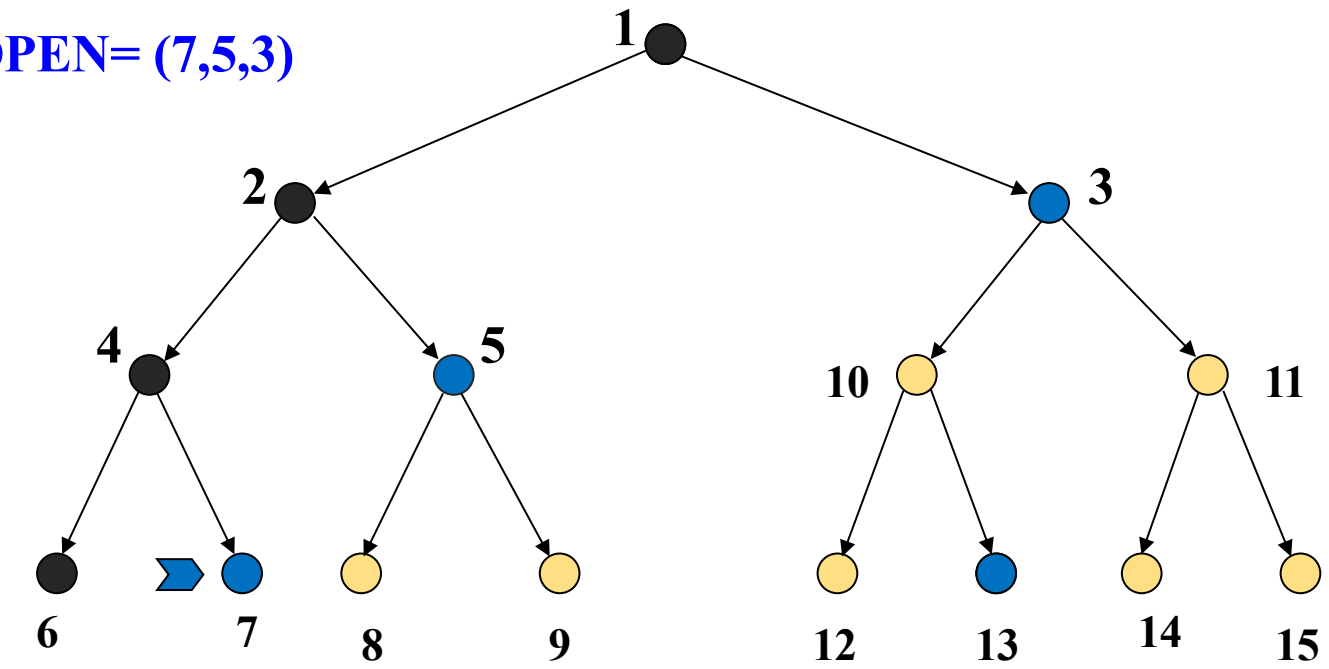


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,4,6)

OPEN= (7,5,3)

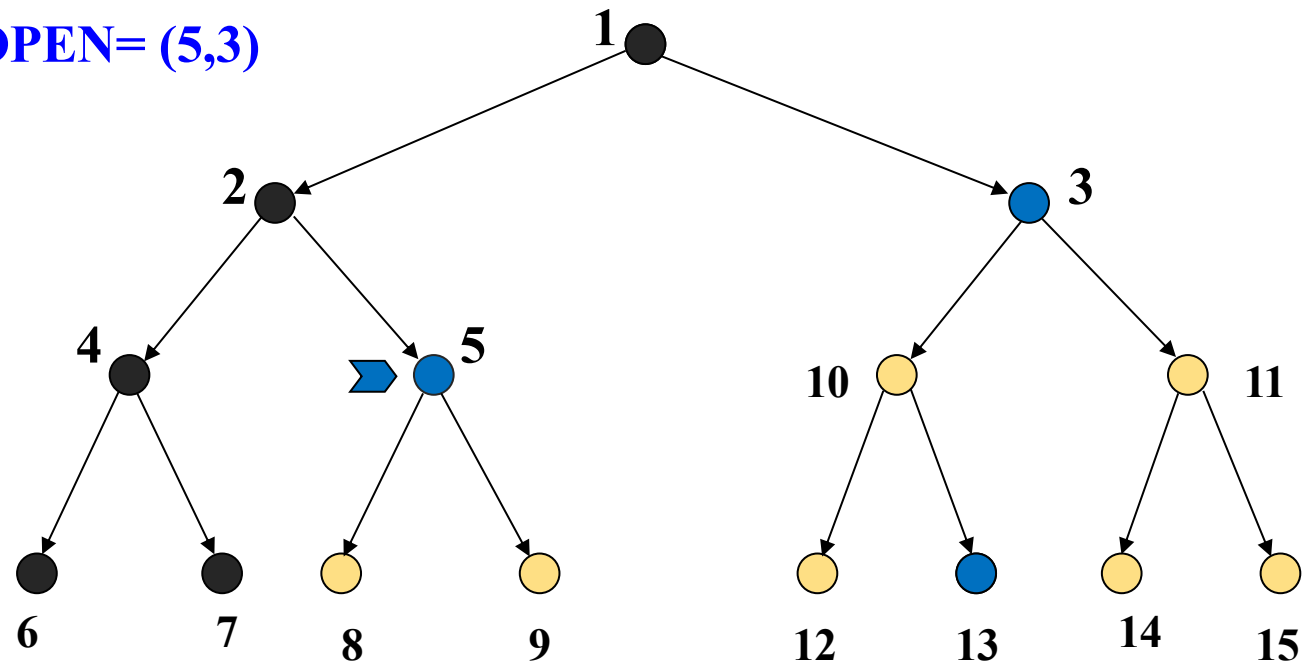


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,4,6,7)

OPEN= (5,3)

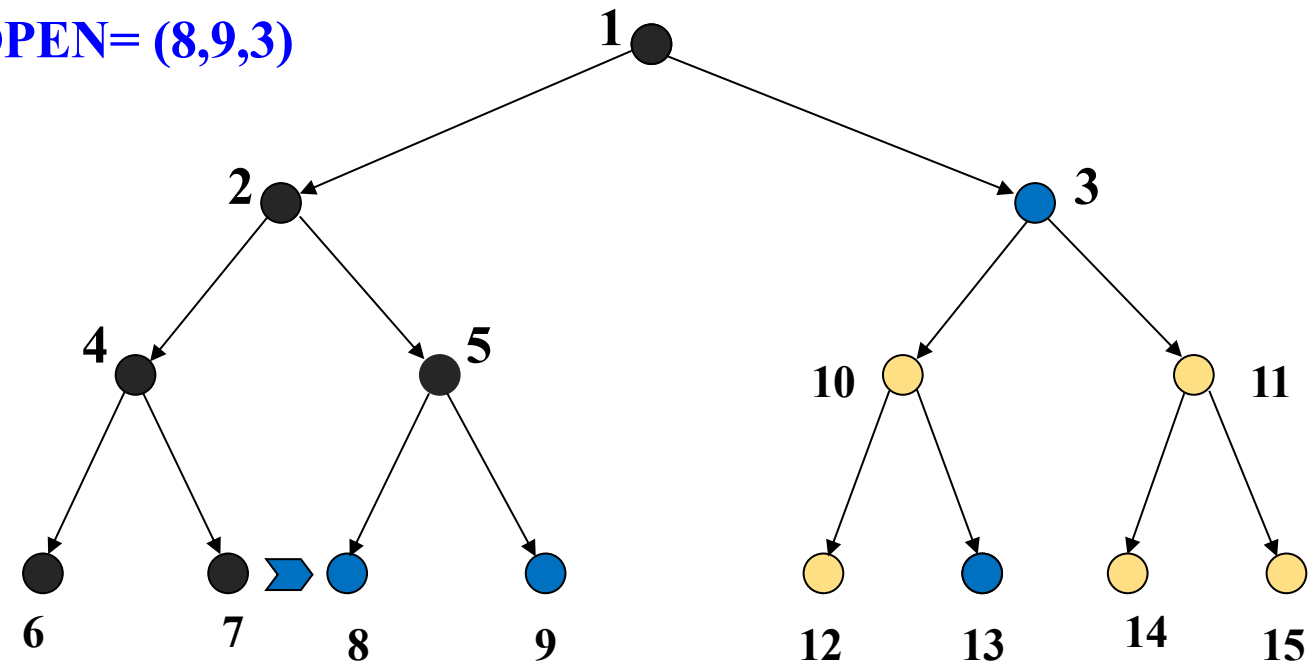


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,4,5,6,7)

OPEN= (8,9,3)

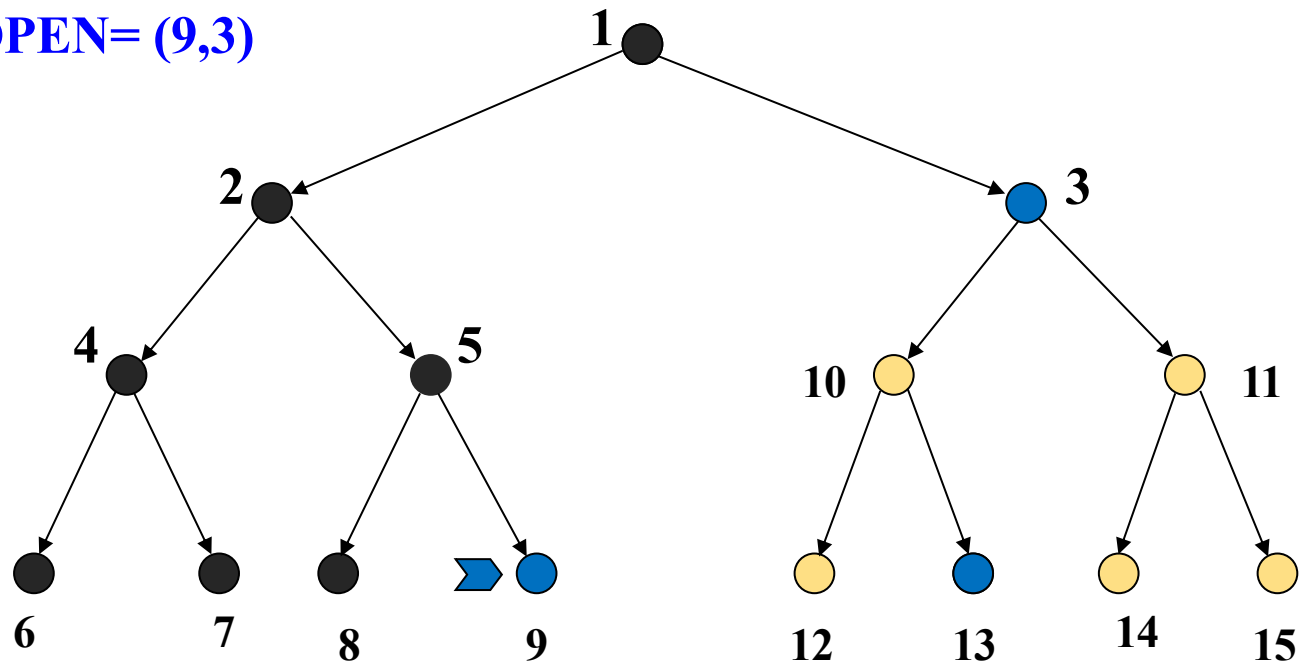


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,4,5,6,7,8)

OPEN= (9,3)

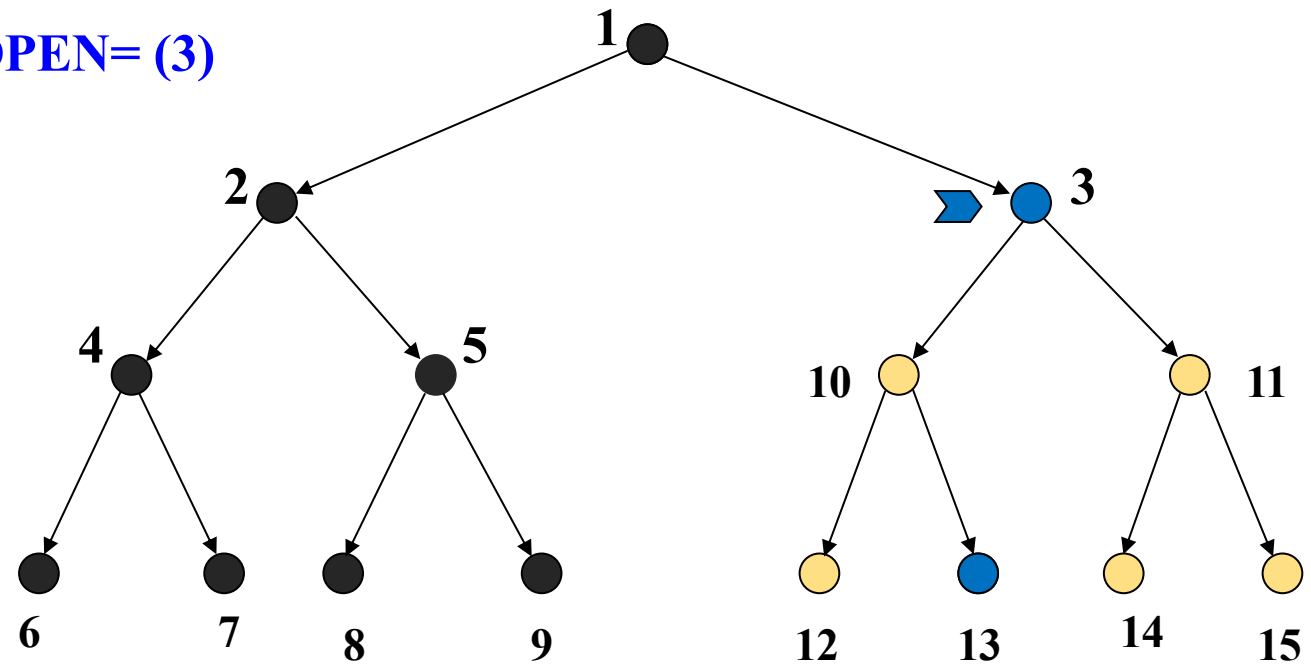


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,4,5,6,7,8,9)

OPEN= (3)

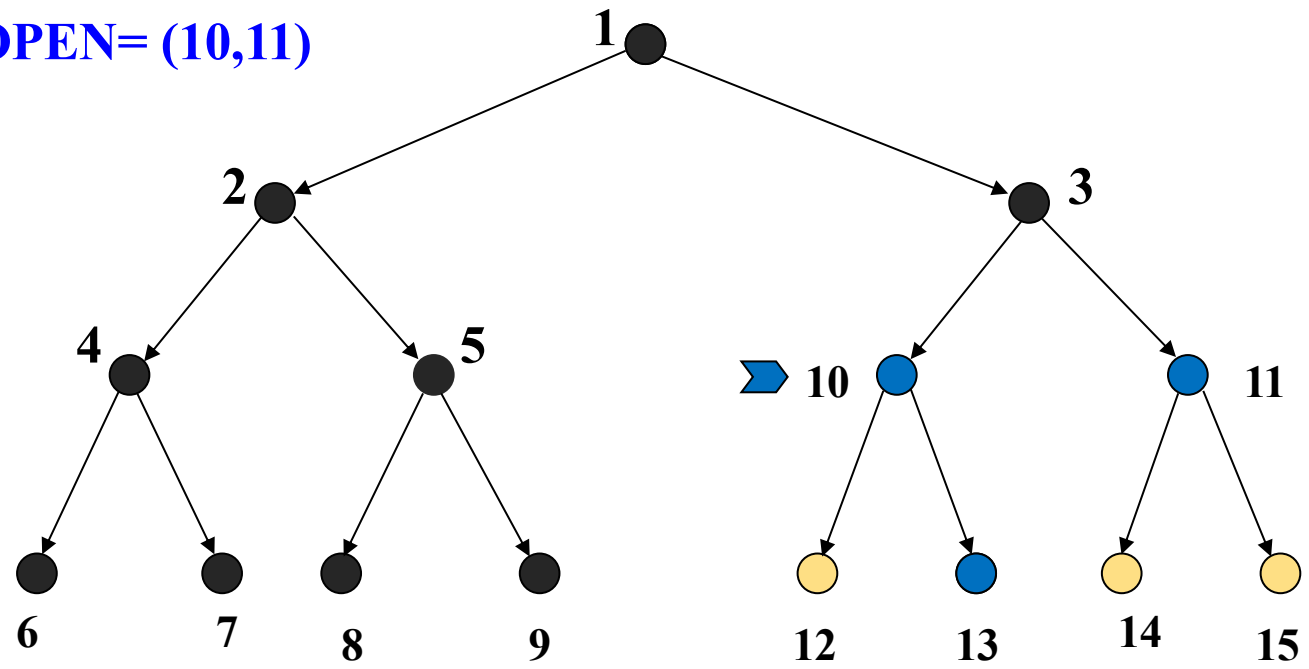


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,3,4,5,6,7,8,9)

OPEN= (10,11)

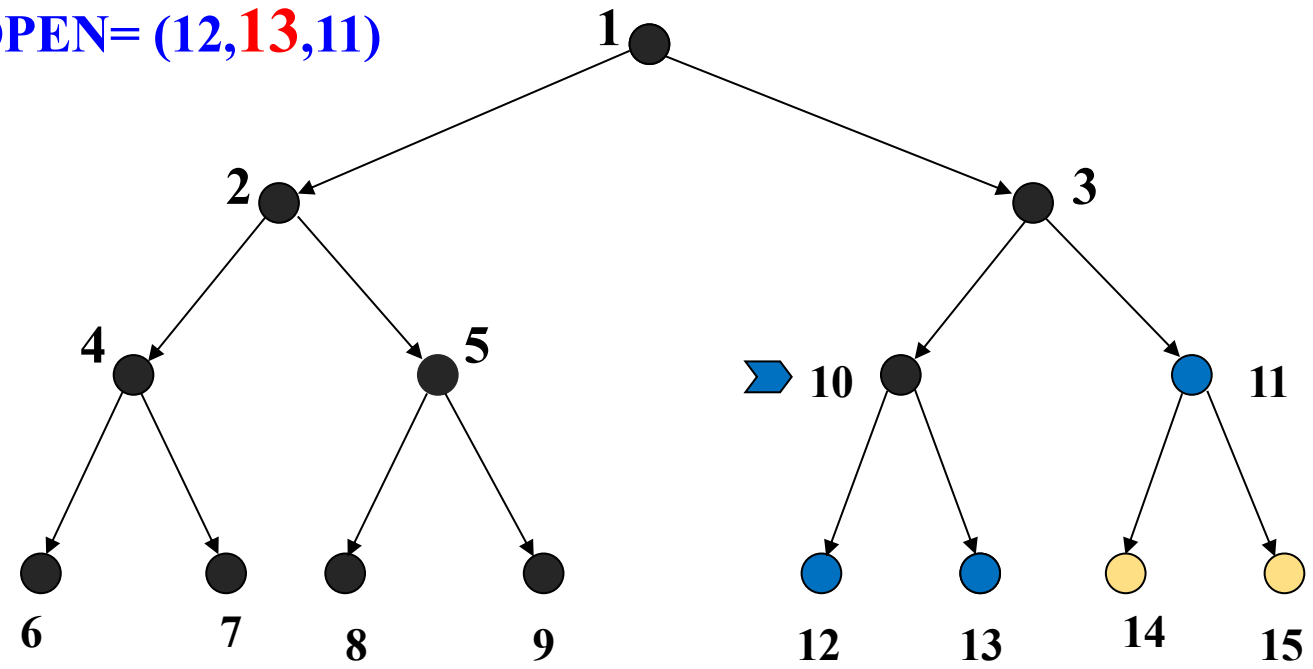


深度优先搜索

新的节点被插入到 OPEN 表的 最前面

CLOSED=(1,2,3,4,5,6,7,8,9,10)

OPEN= (12,13,11)



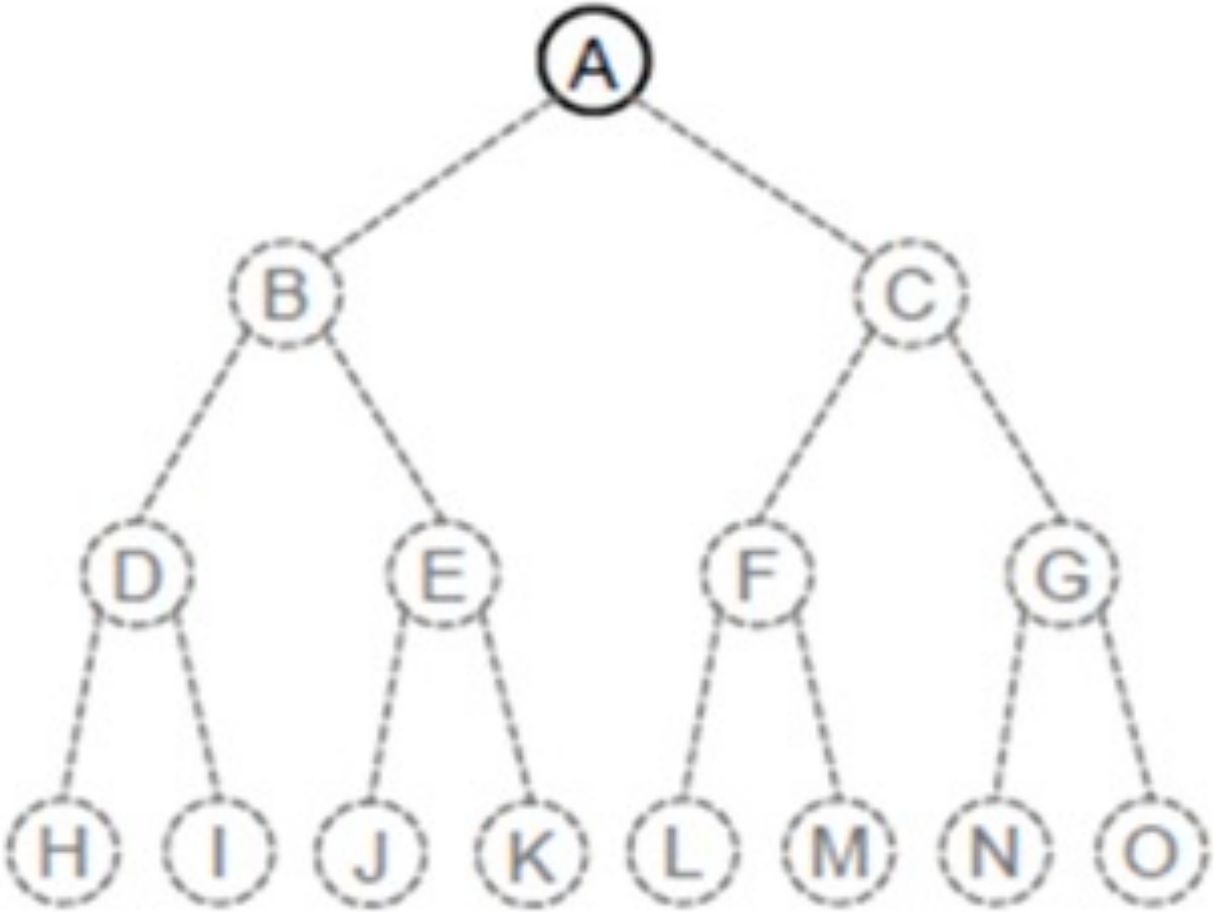
深度优先搜索

- DFS总是选择**深度最深**的节点进行扩展，
- 若有**多个**相同深度的节点，则按照指定的规则从中选择一个；
- 若该节点**没有子节点**，则选择一个除了该节点之外的深度最深的节点进行扩展。
- 以此类推，直到**找到问题的解**为止；或者直到**找不到可扩展的节点**，结束搜索，此种情况说明没有找到问题的解。

深度优先搜索

- DFS 的实现方法
  可以使用 **LIFO** (Last-In First-Out)的**栈**存储OPEN表，把后继节点放在**栈顶**。
- **DFS**是将OPEN表中的节点按搜索树中节点**深度**的**降序**排序，**深度最大的节点排在栈顶，深度相同的节点可以任意排列。**
- DFS总是扩展搜索树中当前OPEN表中**最深**的节点**（即栈顶元素）**。
- 搜索很快推进到搜索树的最深层，那里的节点没有后继。当那些节点被扩展完之后，就从表OPEN中去掉（**出栈**），然后搜索算法回溯到下一个还有未扩展后继的深度稍浅的节点。

DFS：类似于树的先根遍历



DFS 访问的顺序：即扩展顺序，为 {A, B, D,H, I, E,J, K, C, F, L, M,G, N, O}.

DFS算法

深度优先搜索算法的过程如下：

（1）将初始节点*S*放入OPEN表的栈顶；

（2）若OPEN表为空，表示再也没有可扩展的节点，即未能找到问题的解，则算法结束；

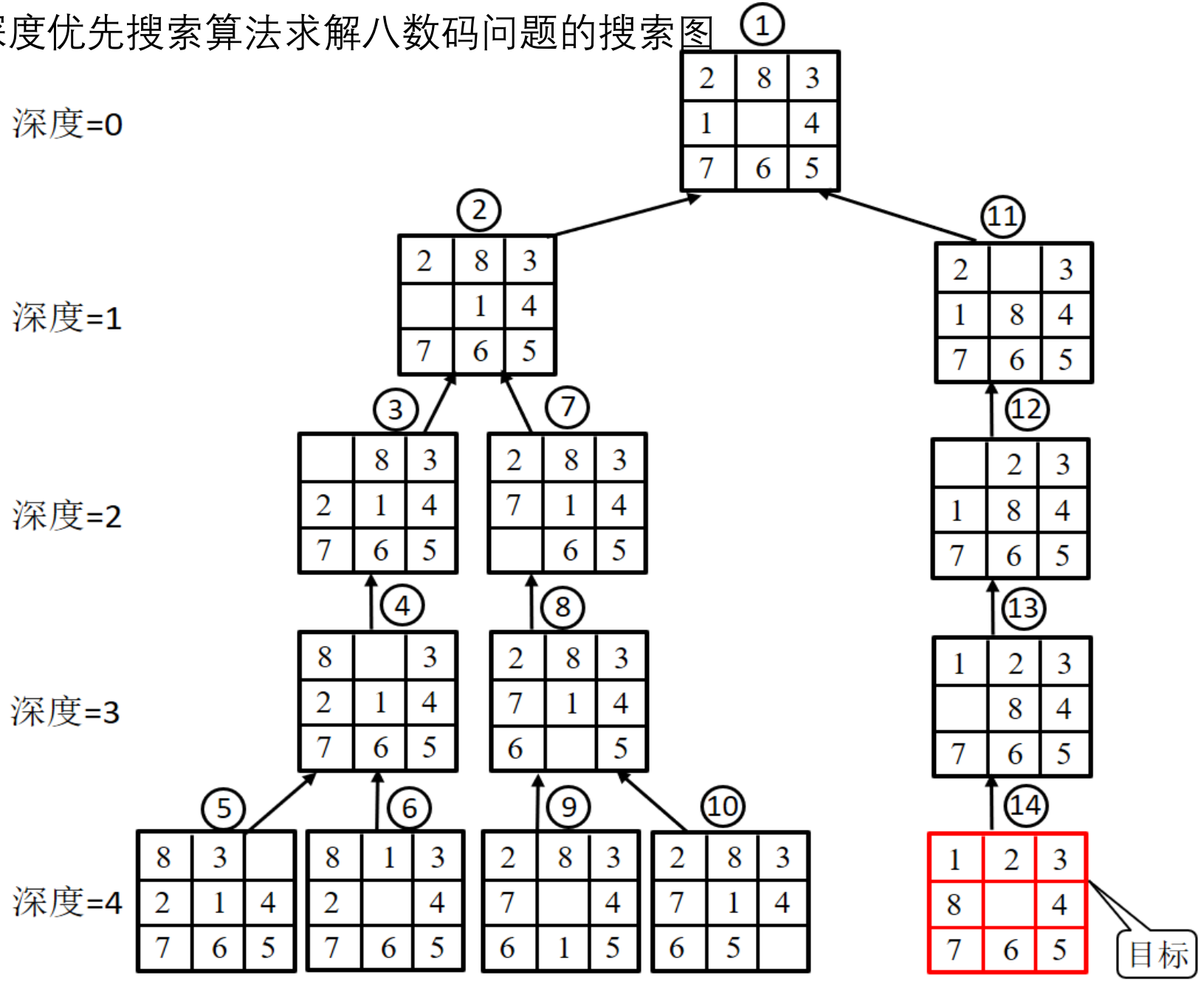
（3）将OPEN表的**栈顶元素**（记为节点*n*）取出，放入CLOSED表中；

（4）若节点*n*是目标节点，则已求得问题的解，算法结束；

（5）若节点*n*不可扩展，即*n*没有后继节点，则转至步骤（2）；

（6）扩展节点*n*，将**其所有未被访问过的子节点**依次**放入OPEN表的栈顶**，并将这些子节点的前驱指针设为指向父节点*n*，然后转至步骤（2）。

图3.2 采用深度限制为4的深度优先搜索算法求解八数码问题的搜索图



DFS无法避免冗余路径，不具有最优性。

总结

- 若**状态空间有限**，**DFS是完备的**，因为它最多扩展所有节点，直到找到一个解。
- 但在**无限状态空间**中，若沿着一个“错误”的路径搜索下去而陷入“深渊”，则会导致无法到达目标节点，在这种情况下，DFS**是不完备的**。
- 为避免此情况发生，在DFS中往往会加上一个**深度限制**，称为**深度受限的深度优先搜索**，即若一个节点的深度达到了事先指定的深度阈值k，强制进行回溯，选择一个比它浅的节点进行扩展，而不是沿着当前节点继续扩展。
- 当深度限制**过深**时，会陷入“**深渊**”，**求解效率低，未必会找到最优解**；

  若深度限制**过浅**，可能找不到解，即**不完备**。
- 所以，应该根据具体问题**合理地设定深度限制值**，或在**搜索过程中逐步加大深度限制值**，反复搜索，直到找到解。
- **DFS不一定是完备的，也未必能找到最优解**。
- **最坏情况时，DFS的搜索空间等同于穷举**。
- DFS是一个通用的、与问题无关的方法。