

BUSS 3620.人工智能导论

搜索 I

刘佳璐

安泰经济与管理学院

上海交通大学

BUSS 3620.人工智能导论

#1. 搜索问题的一些定义


刘佳璐

安泰经济与管理学院

上海交通大学

什么是搜索?



Jim Fan  @DrJimFan · 14h 英伟达高级科学家

OpenAI Strawberry (o1) is out! We are finally seeing the paradigm of inference-time scaling popularized and deployed in production. As Sutton said in the Bitter Lesson, there're only 2 techniques that scale indefinitely with compute: learning & search. It's time to shift focus to

[Show more](#)



@DrJimFan

93

1K

4.9K

490K

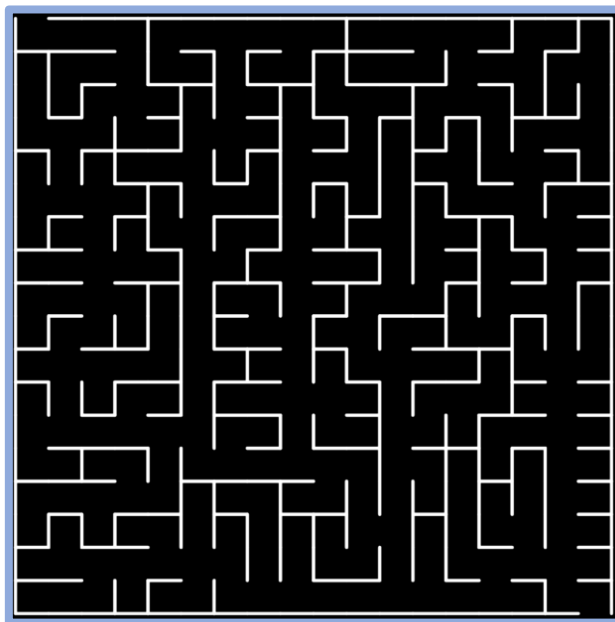
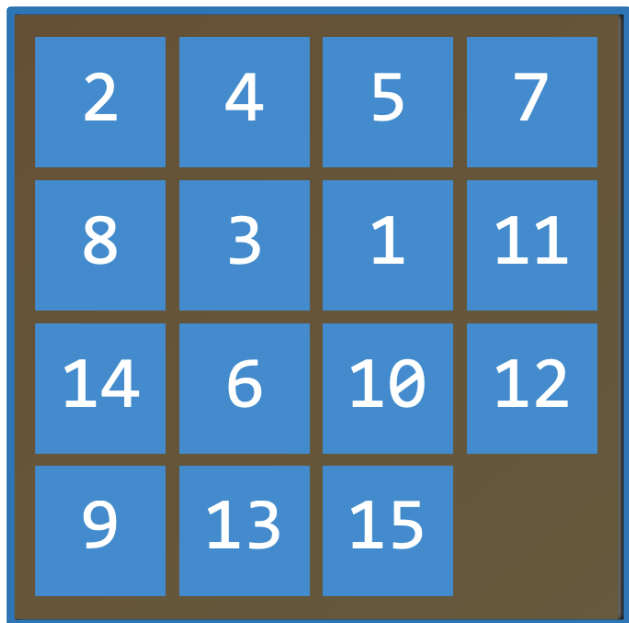


什么是搜索?



搜索

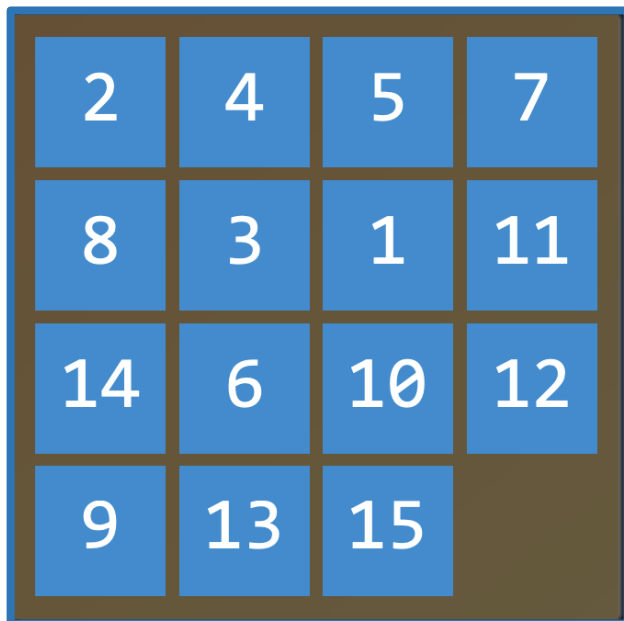
- 寻找问题的**答案**
 - 游戏：下一步
 - 迷宫：最佳路线



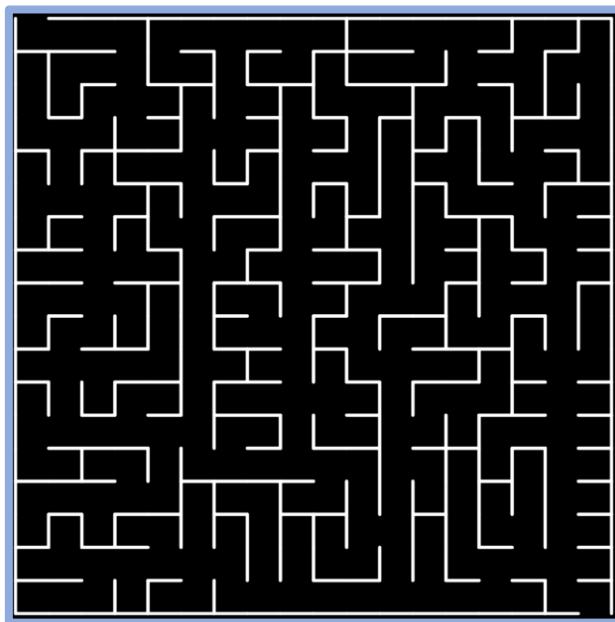
搜索问题

- 智能体 Agent
 - 感知其环境并对该环境采取行动的实体

玩家



玩家



车 / 人



搜索问题

- 状态 State
 - 智能体及其环境的一种配置

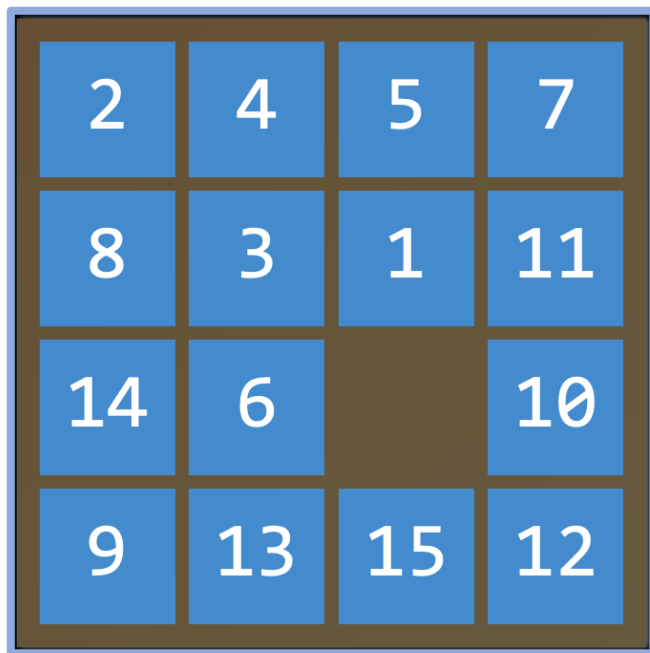
2	4	5	7
8	3	1	11
14	6		10
9	13	15	12

12	9	4	2
8	7	3	14
	1	6	11
5	13	10	15

15	4	10	3
13	1	11	12
9	5	14	7
6	8		2

搜索问题

- 初始状态Initial State
 - 智能体开始的状态、搜索的开始



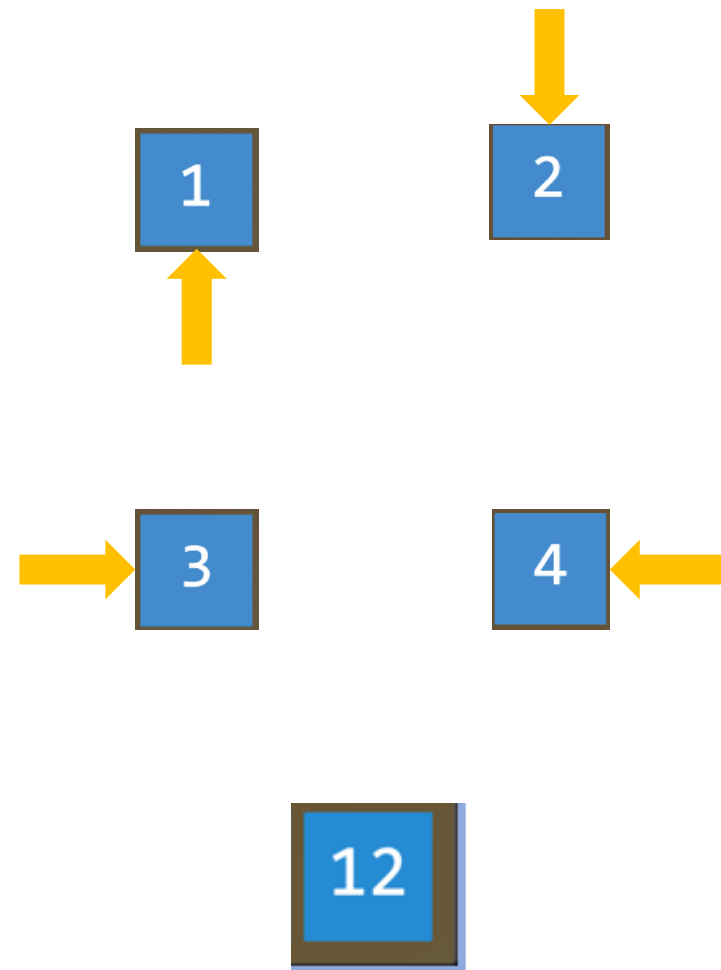
搜索问题

- 行动 Actions

- 在某种状态下可以执行的选择

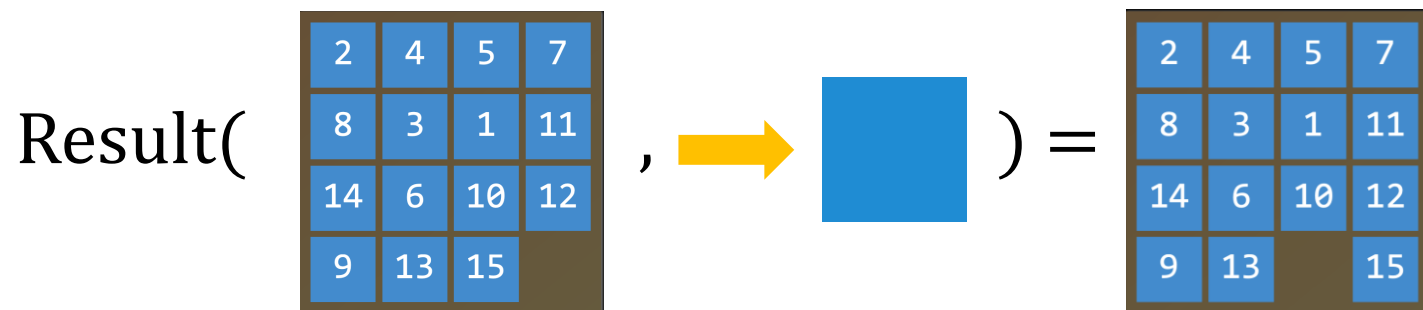
- Action(s) 函数

- 输入：状态 s
 - 功能：找到状态 s 下可以执行的所有（合法）行动
 - 输出：行动集合(Action set)



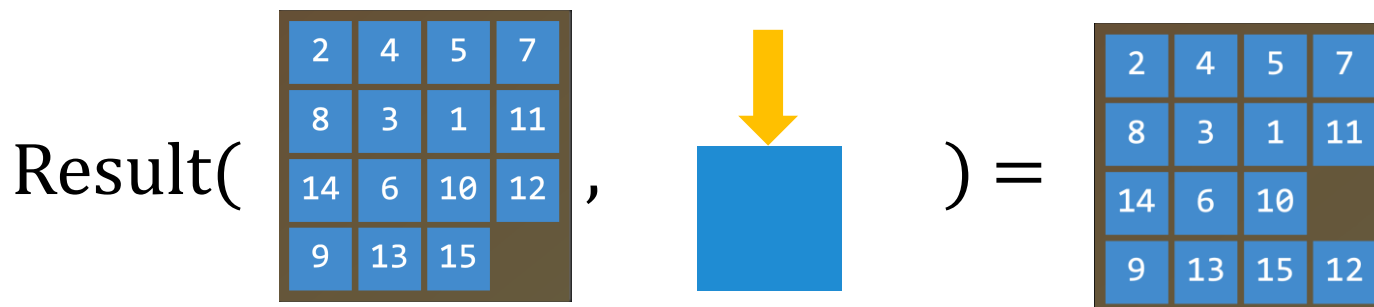
搜索问题

- 转换模型 Transitional model
 - 描述在某种状态下执行合法行动后到达的状态
 - $\text{Result}(s, a)$ 函数（后继函数）
 - 输入：状态 s ，行动 a
 - 功能：找到在状态 s 执行行动 a 后到达的新的状态
 - 输出：新的状态 s



搜索问题

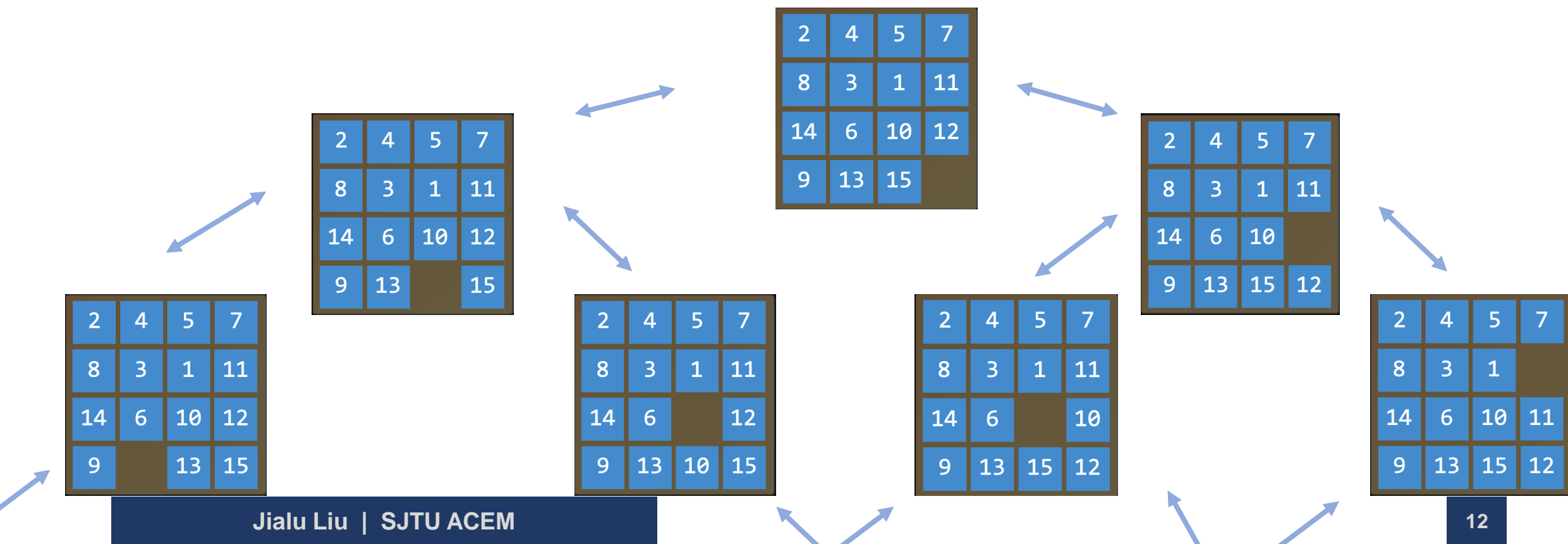
- 转换模型 Transitional model
 - 描述在某种状态下执行合法行动后到达的状态
 - $\text{Result}(s, a)$ 函数（后继函数）
 - 输入：状态 s ，行动 a
 - 功能：找到在状态 s 执行行动 a 后到达的新的状态
 - 输出：新的状态 s



搜索问题

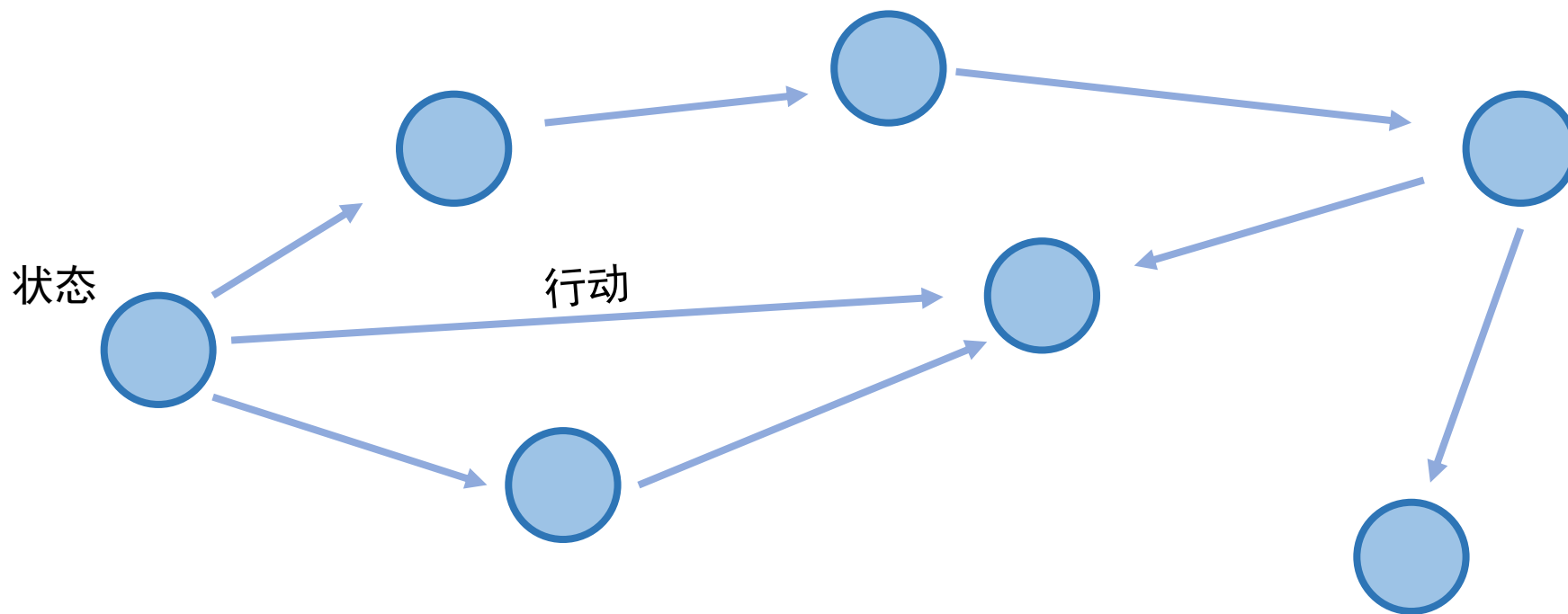
- 状态空间 State space

- 从初始状态开始，经过一系列的行动，所到达的全部状态集合



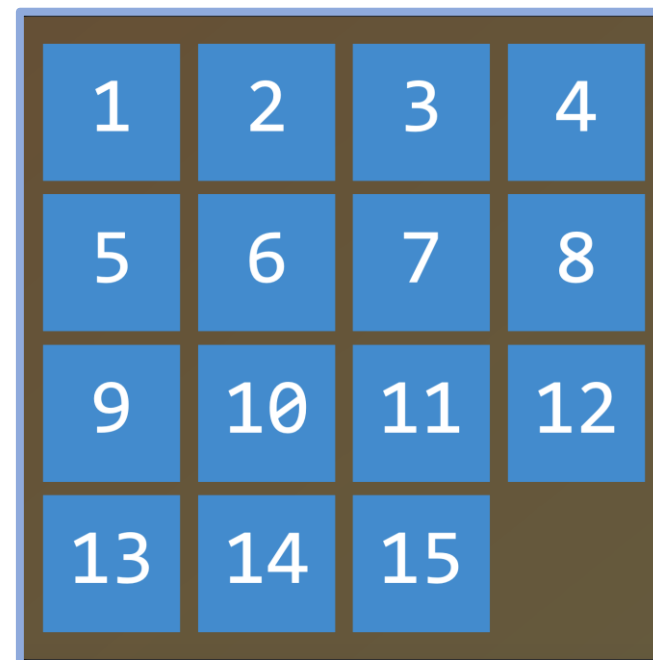
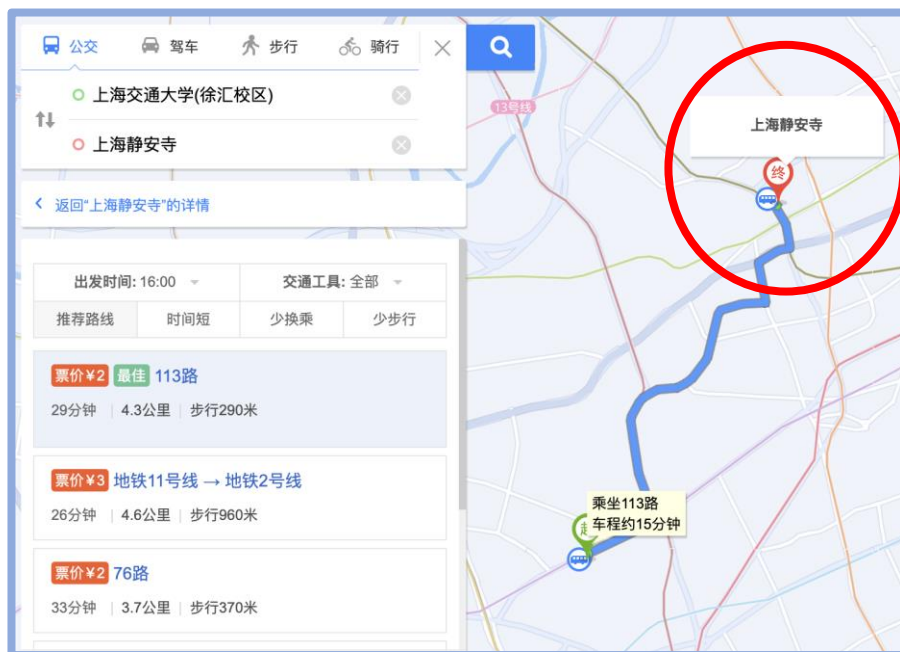
搜索问题

- 状态空间 State space
 - 从初始状态开始，经过一系列的行动，所到达的全部状态集合



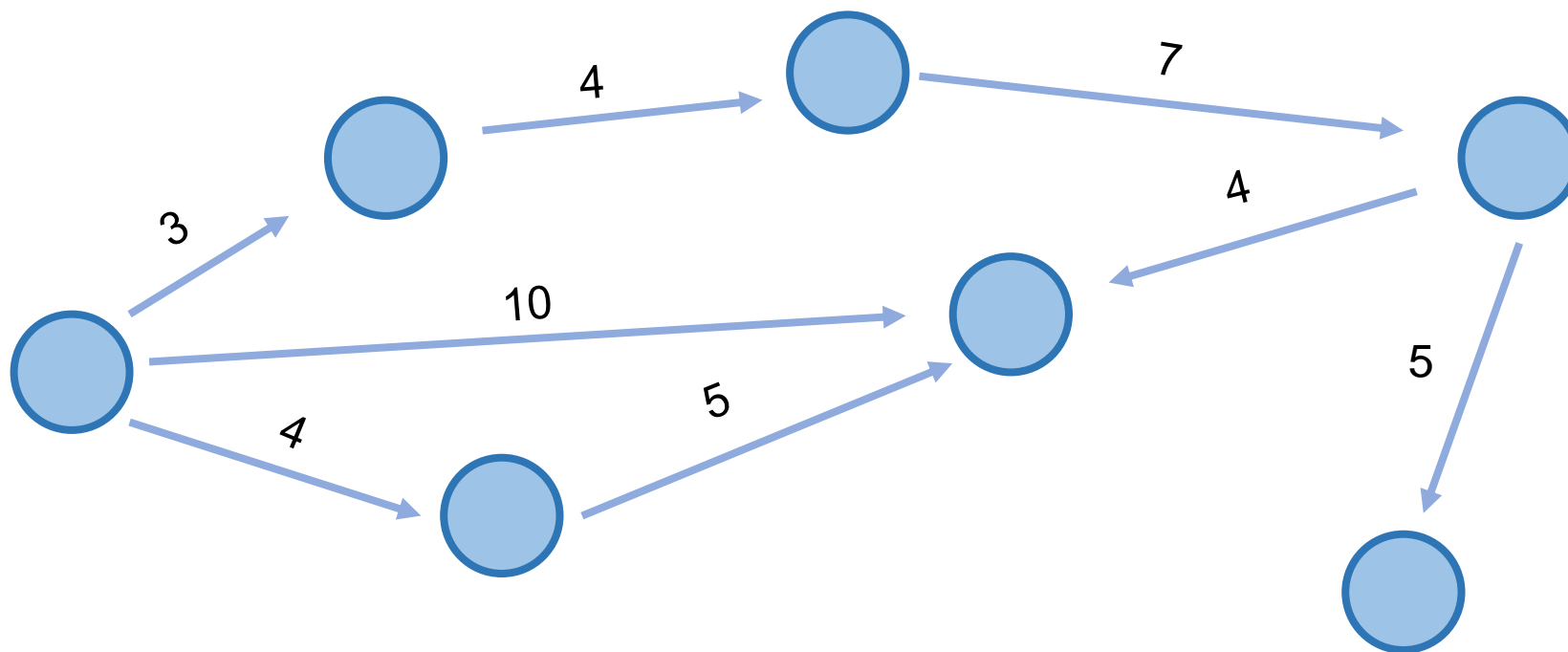
搜索问题

- 目标测试 Goal test
 - 确定给定状态是否为目标状态



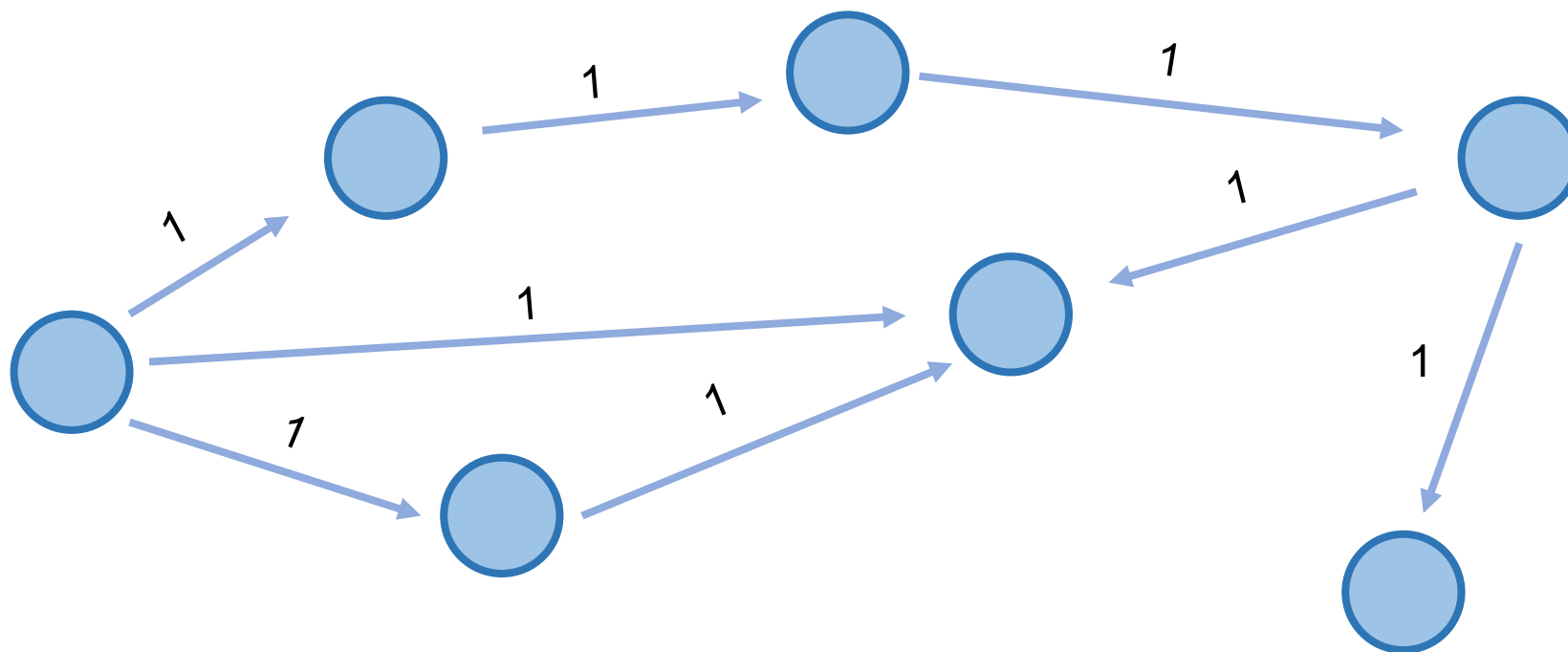
搜索问题

- 路径耗散 Path cost
 - 在状态之间进行转移所需的代价



搜索问题

- 路径耗散 Path cost
 - 在状态之间进行转移所需的代价



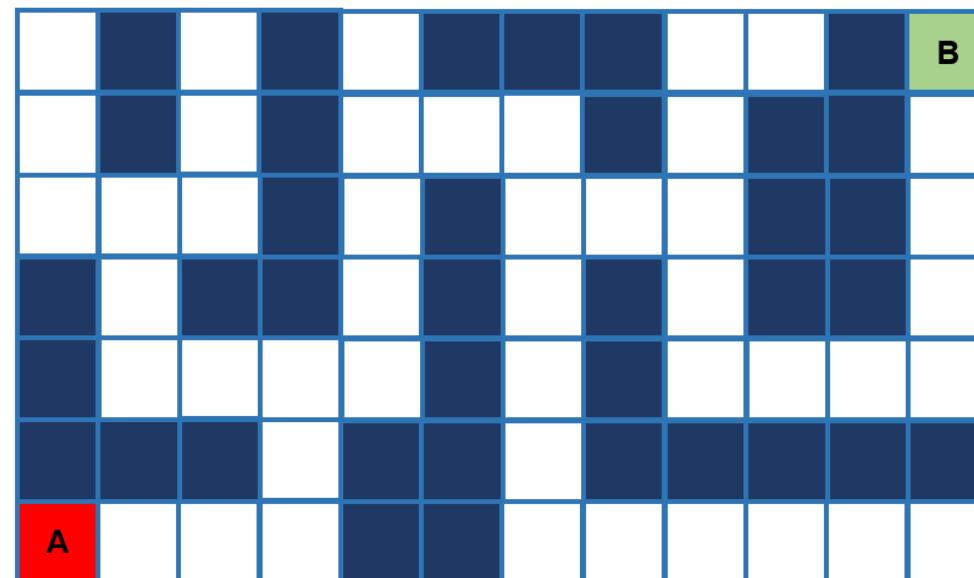
搜索问题

- 输出一个能够让智能体从初始状态(initial state)到目标状态(goal state)的解(solution)
 - 初始状态 Initial state
 - 行动 Actions
 - 转换模型 Transition models
 - 目标测试 Goal test
 - 路径耗散函数 Path cost function

练习 #1

- 将如图所示的迷宫问题抽象成搜索问题

- 初始状态 Initial state
- 行动 Actions
- 转换模型 Transition models
- 目标测试 Goal test
- 路径耗散函数 Path cost function



有问题吗？

- 请随时举手提问。



BUSS 3620.人工智能导论

#2. 搜索问题的求解

刘佳璐

安泰经济与管理学院

上海交通大学

求解

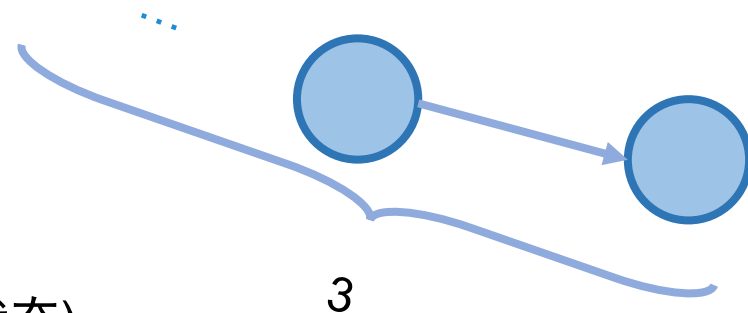
- 解 Solution
 - 从初始状态到目标状态之间的行动序列
- 最优解 Optimal solution
 - 路径耗散最低的解

求解

- 节点 Node

- 储存如下信息的数据结构

- 一个状态 A **state**
 - 父节点 A **parent** (产生当前节点/状态的那个节点/状态)
 - 一个行动 An **action** (从父节点/状态到当前节点/状态所执行的那个动作)
 - 路径耗散 A **path cost** (从初始节点/状态到当前节点所需的代价)



- 探索边界(活结点集合) Frontier

- 所有下一步可以探索但尚未探索的节点集合

求解

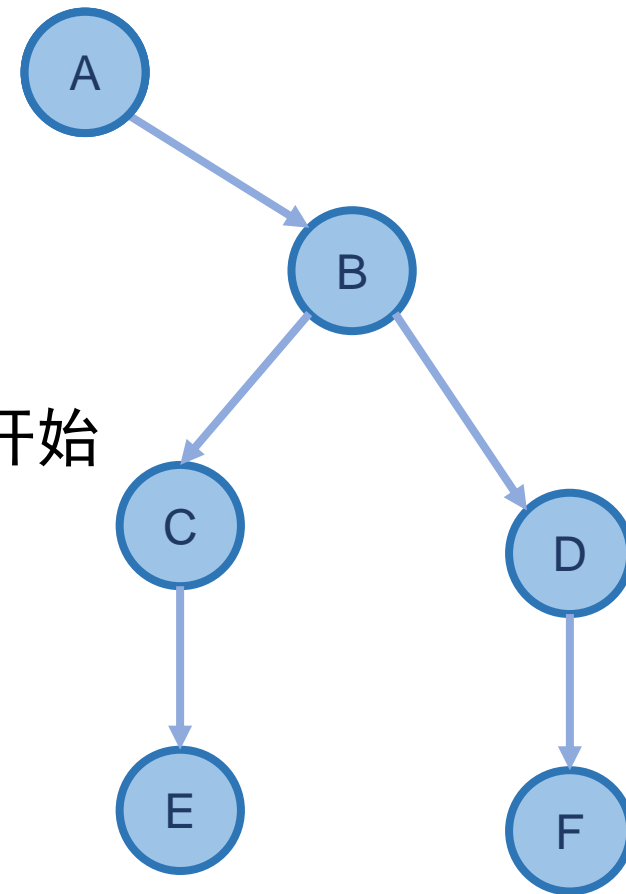
- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中

求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:

- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



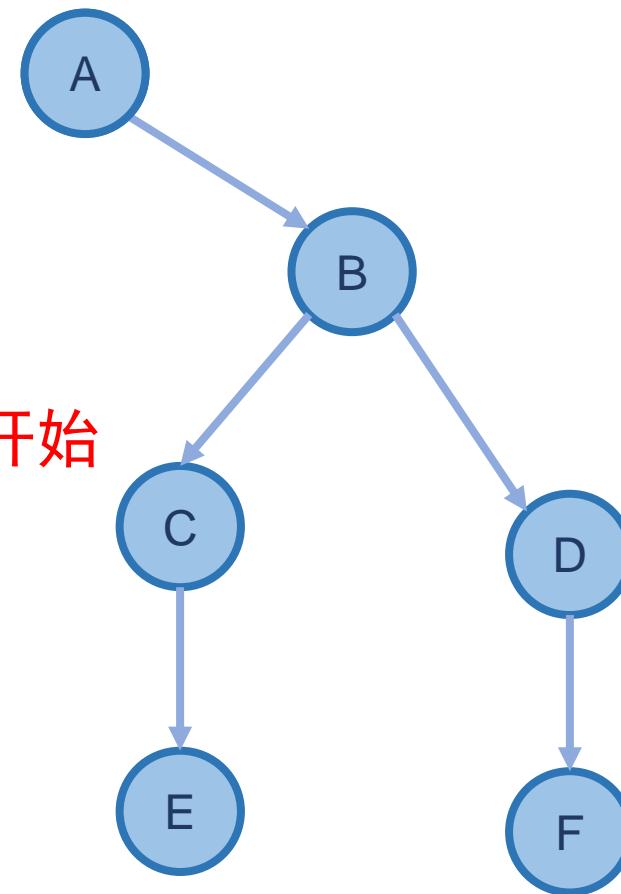
求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:



- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中

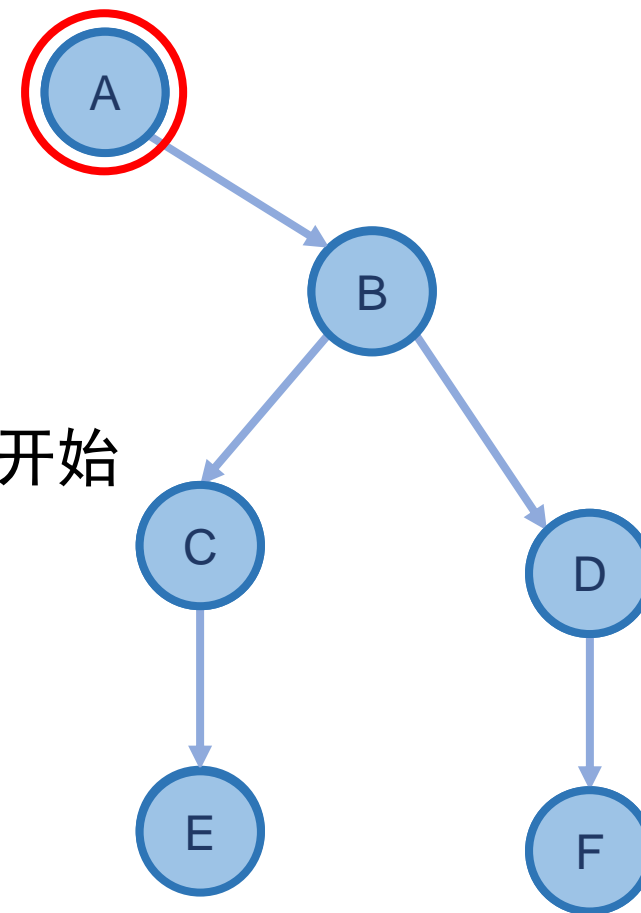


求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:

- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



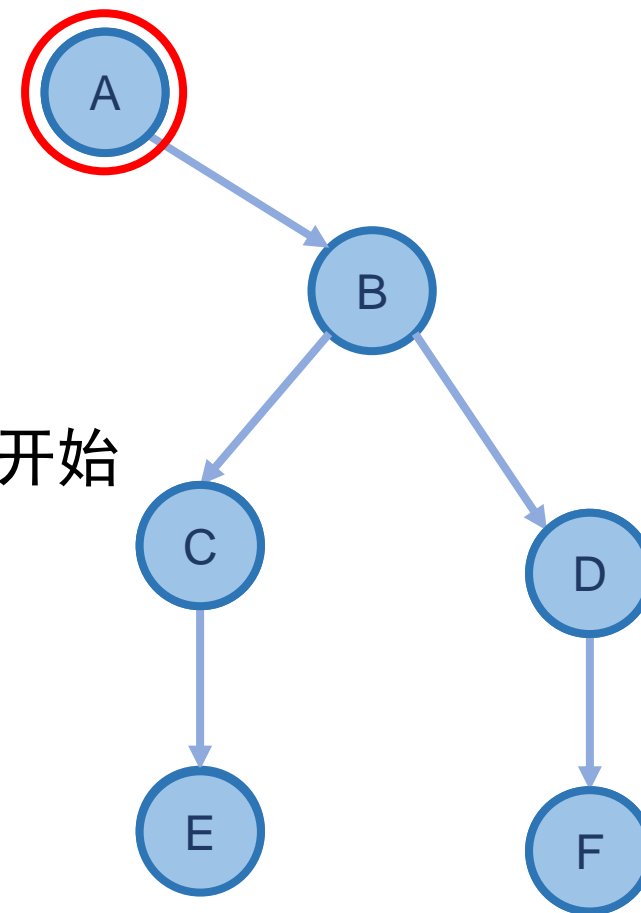
求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:



- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中

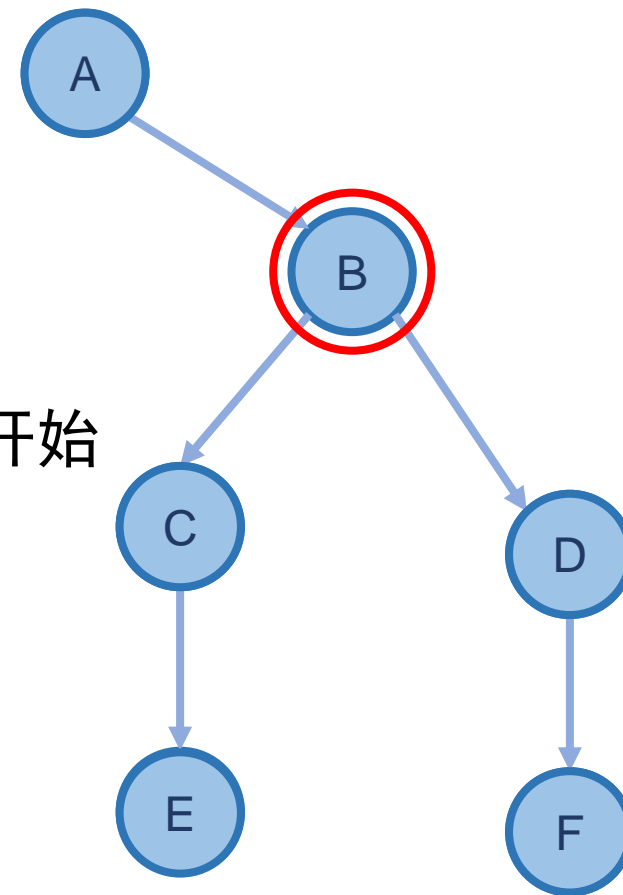


求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:

- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



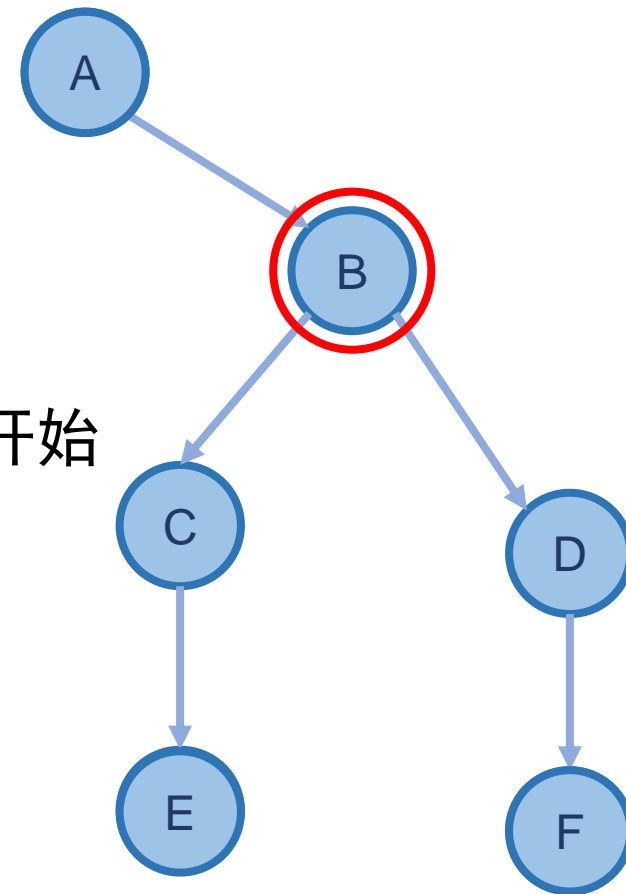
求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:



- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



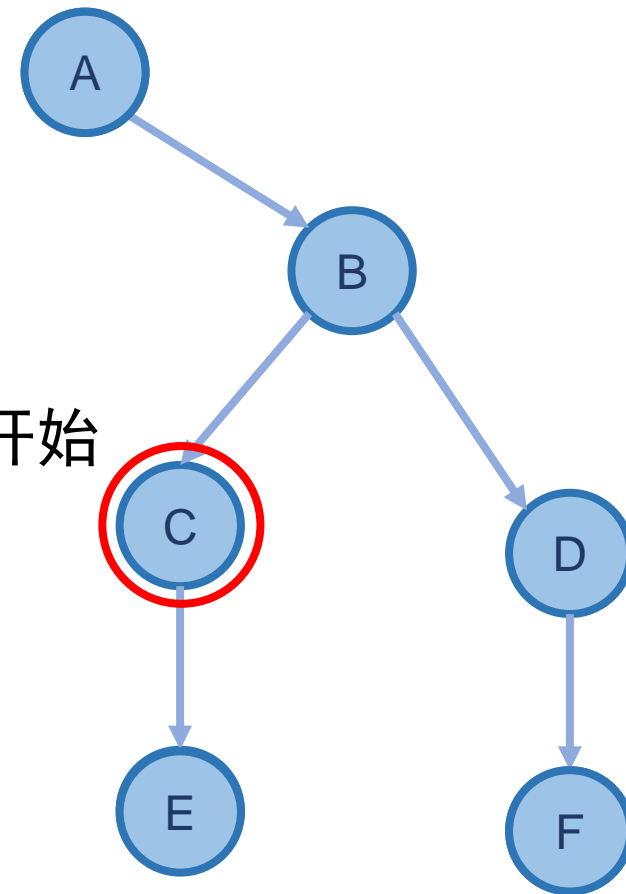
求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:



- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



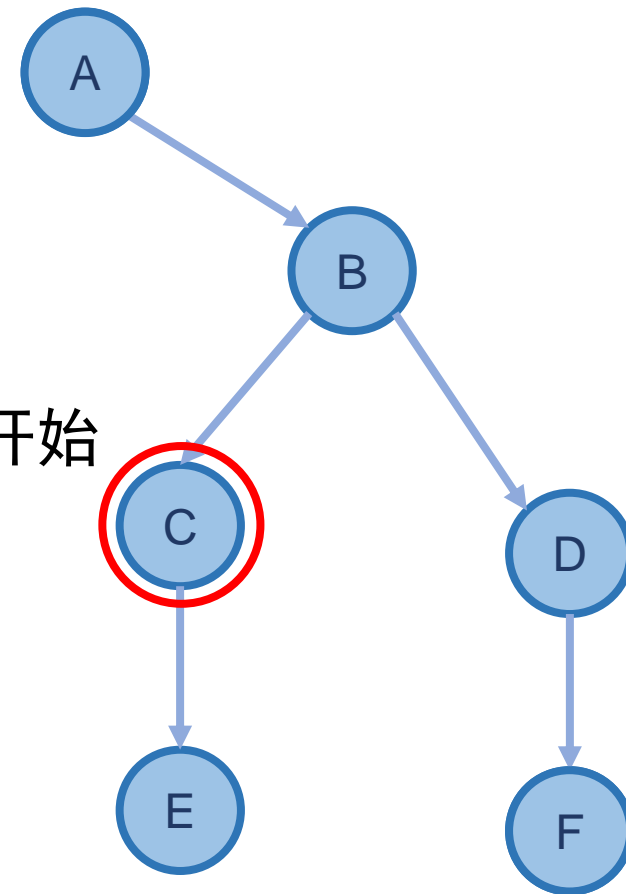
求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:



- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



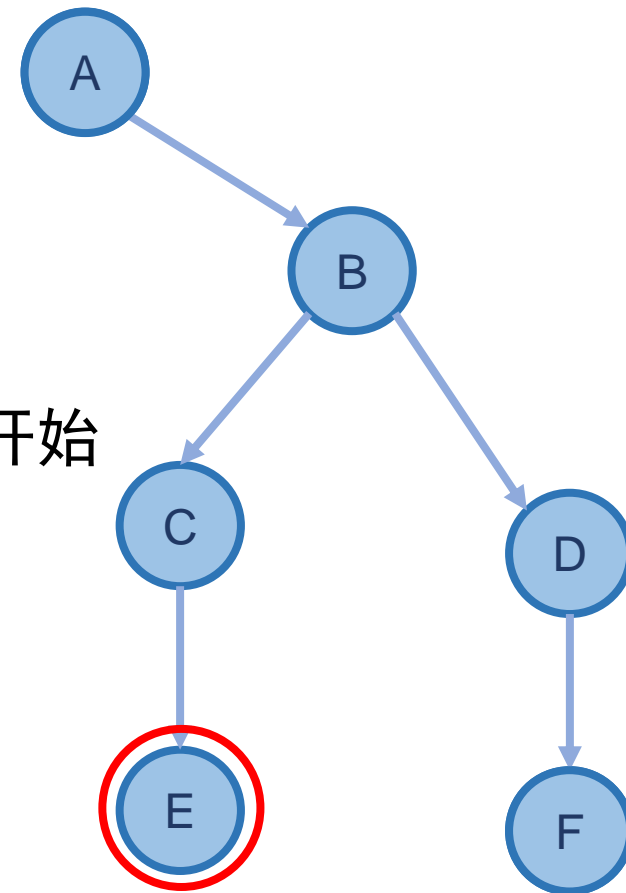
求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:



- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



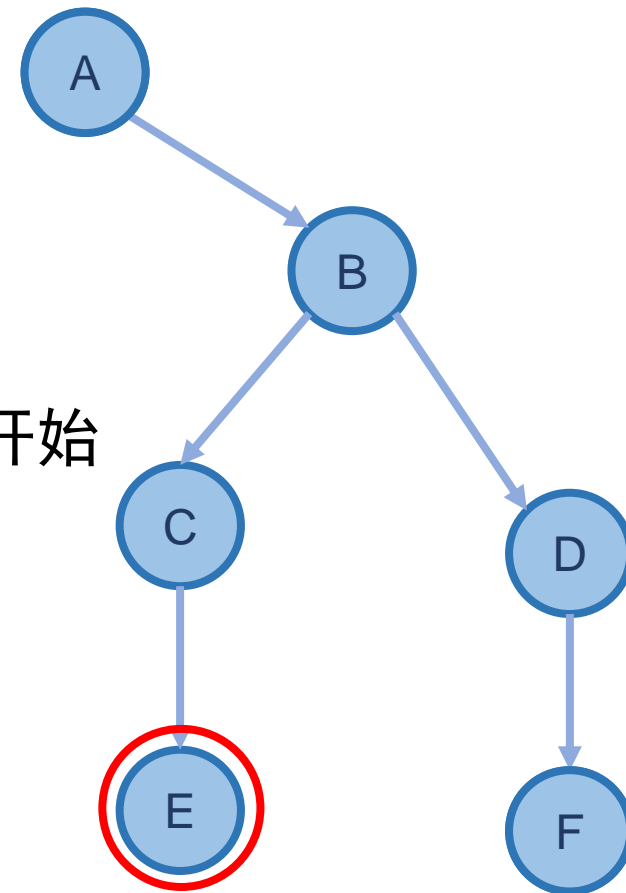
求解 - 例子

- 找到从 A 到 E 的路径

搜索边界 Frontier:



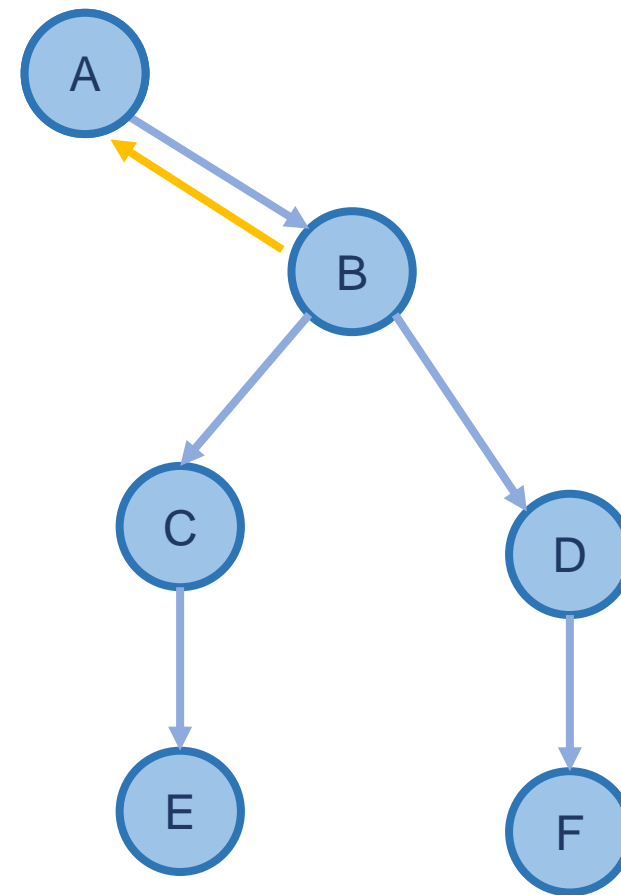
- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 重复:
 - 如果探索边界(frontier)是空的, 那么问题就无解
 - 否则: 从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态, 那么返回问题的解
 - 否则: 扩展这个节点, 将生成的节点添加到探索边界(frontier)中



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

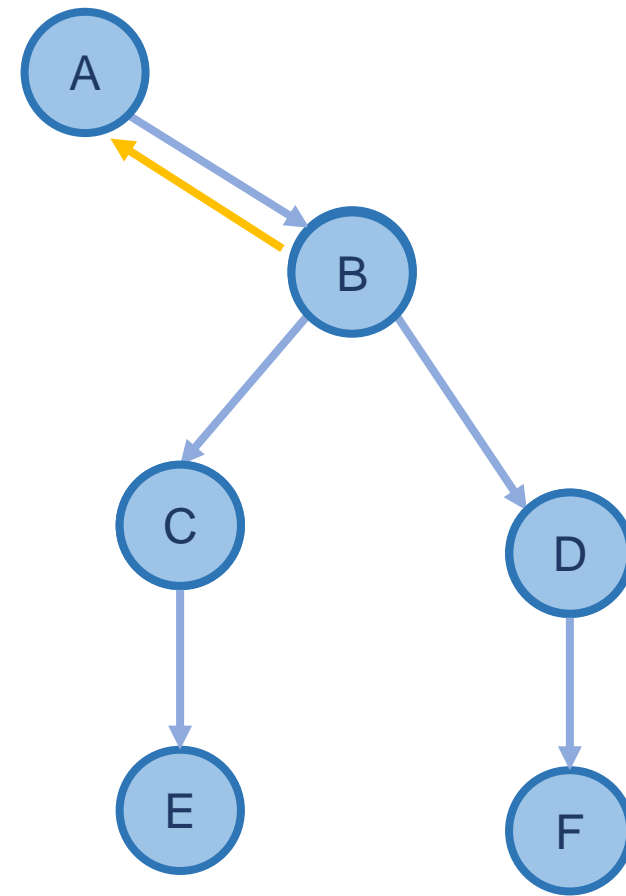
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

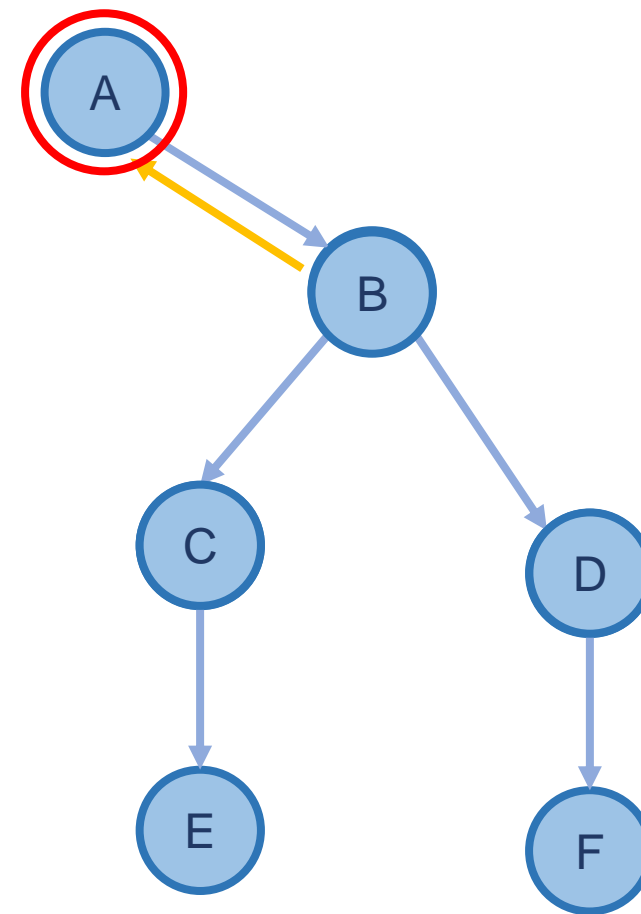
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

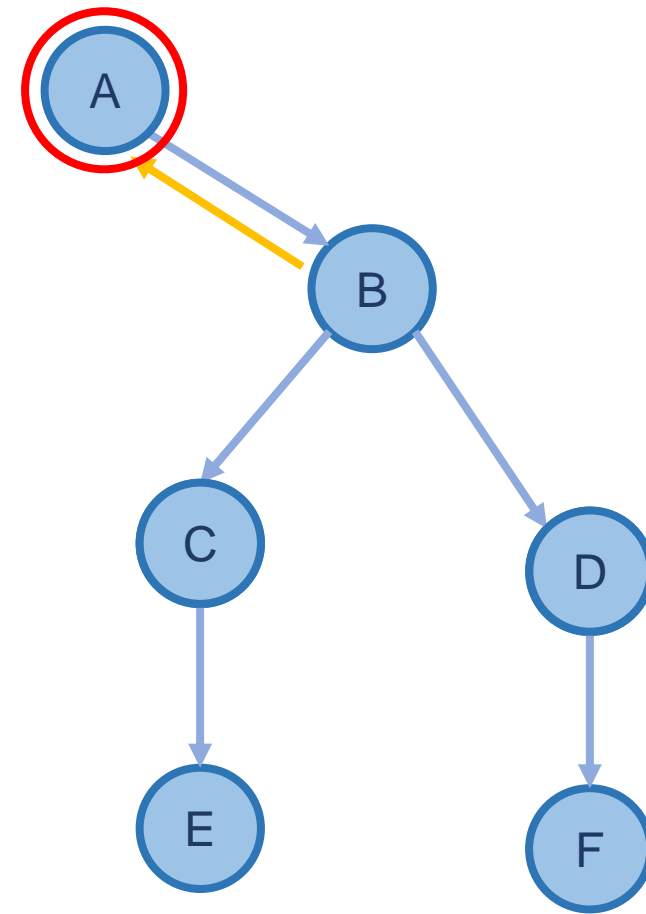
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

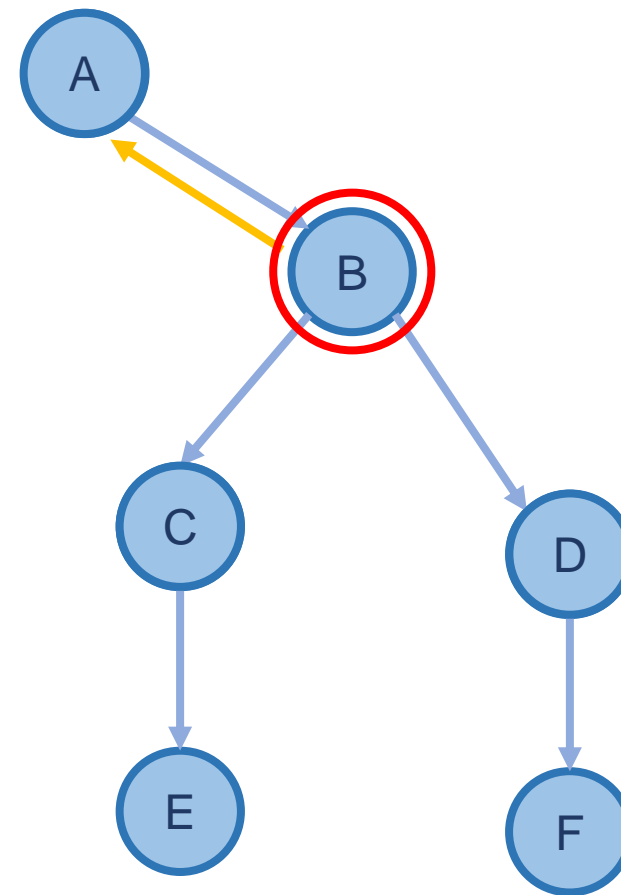
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

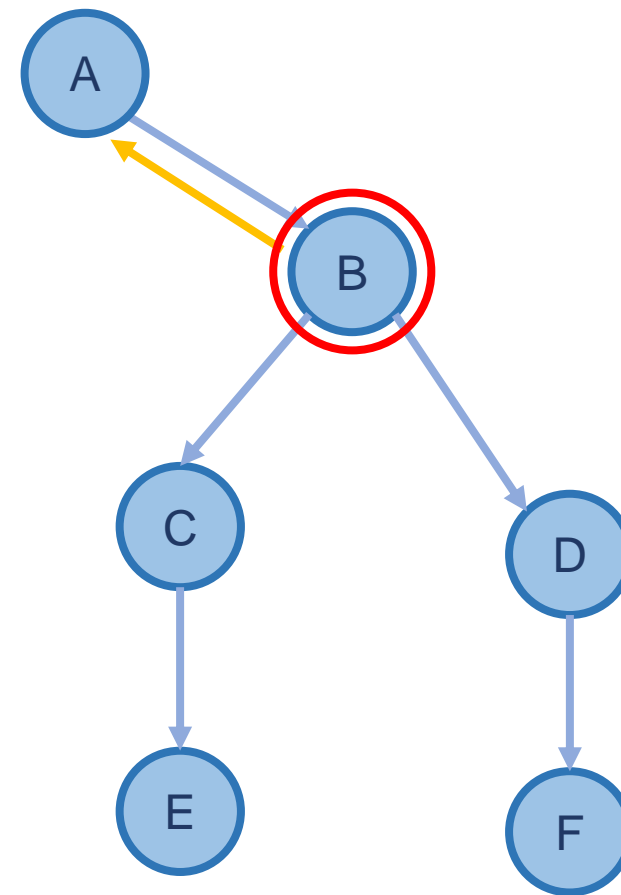
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

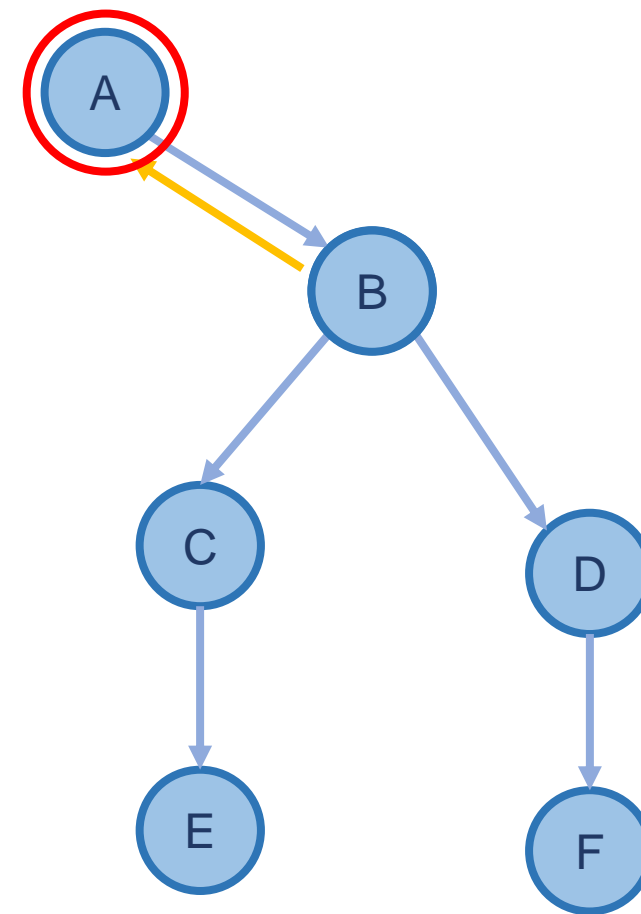
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

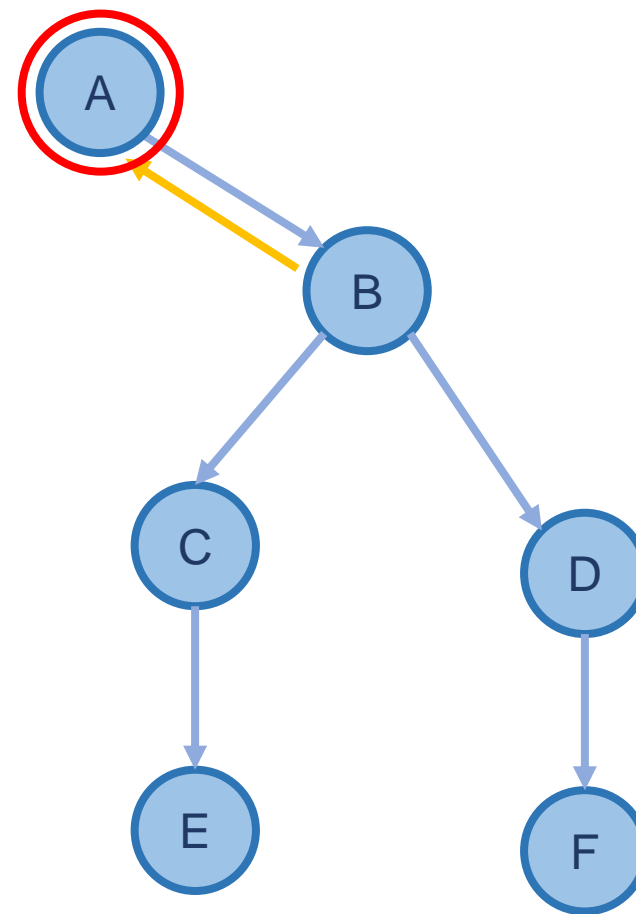
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

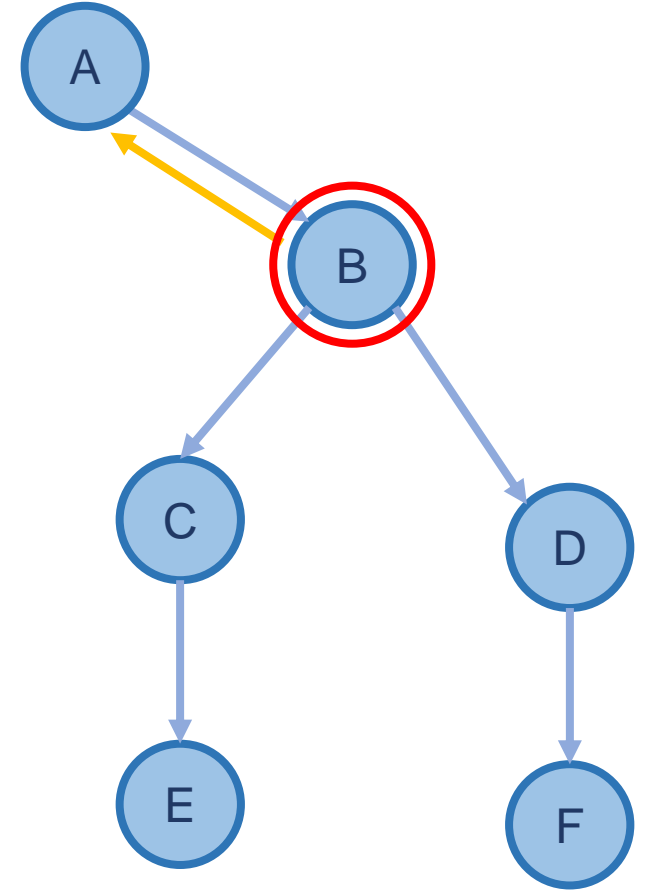
搜索边界 Frontier:



这个方法会出现什么问题？

- 找到从 A 到 E 的路径

搜索边界 Frontier:



求解 – 修改版本

- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 初始化一个空的**已探索集合(empty explore set)**
- 重复:
 - 如果探索边界(frontier)是空的，那么问题就无解
 - 否则：从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态，那么返回问题的解
 - 否则：
 - **将这个节点加入到已探索集合**
 - 扩展这个节点，如果生成的节点不在已探索集合，那么添加这个节点到探索边界(frontier)中

应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈数据结构 stack data structure)

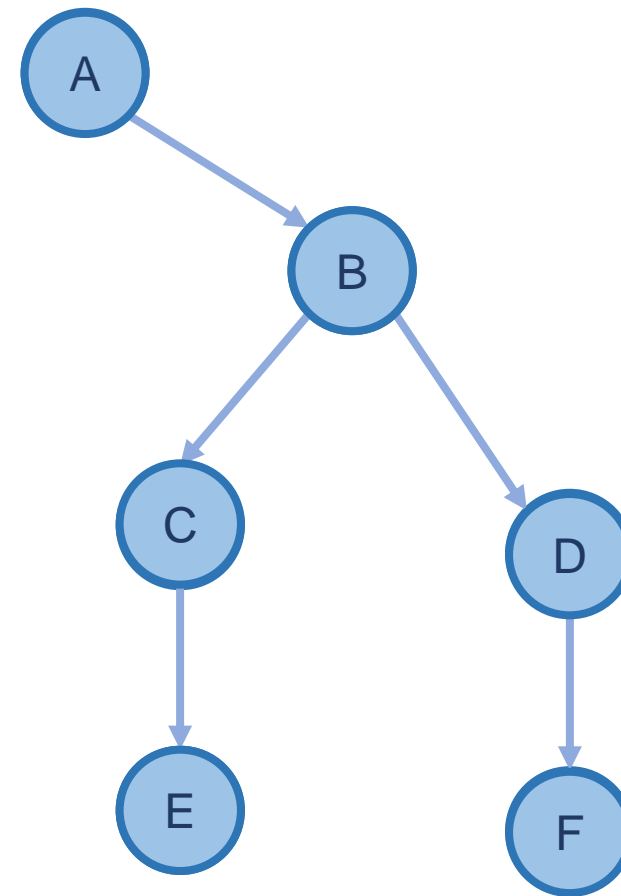


应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

已探索集合 Explored set:



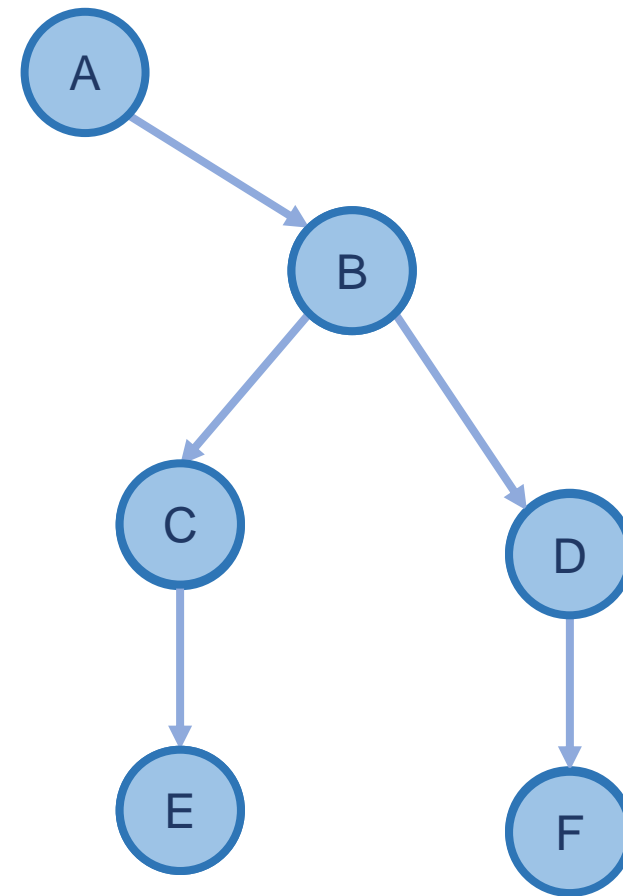
应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:

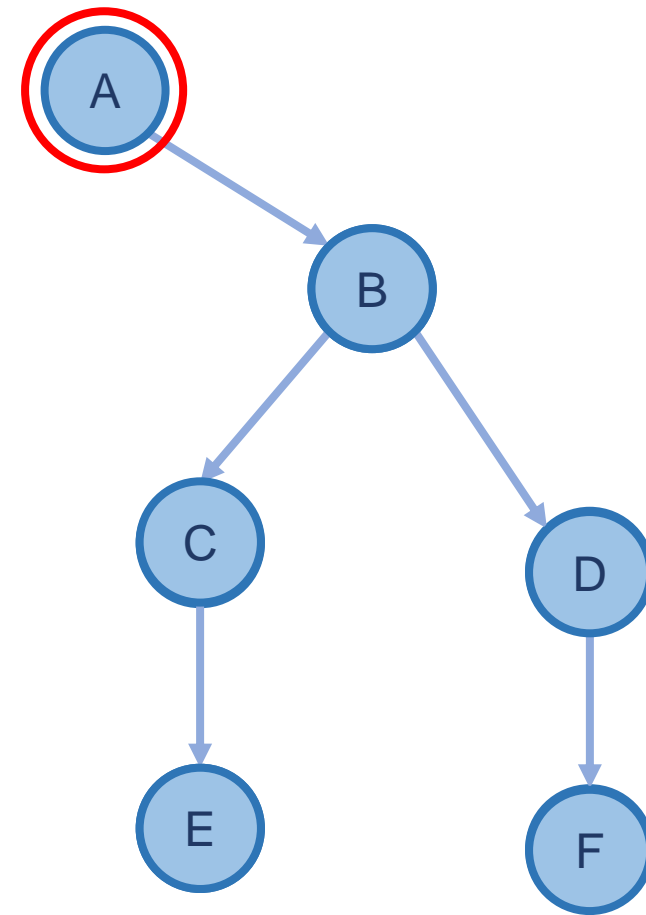
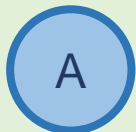


应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

已探索集合 Explored set:



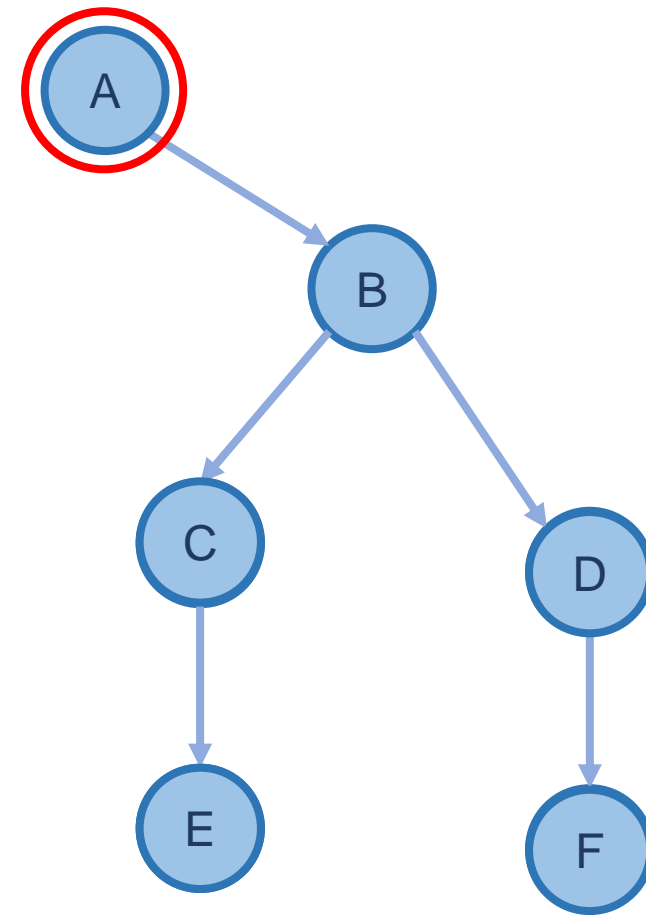
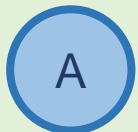
应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:

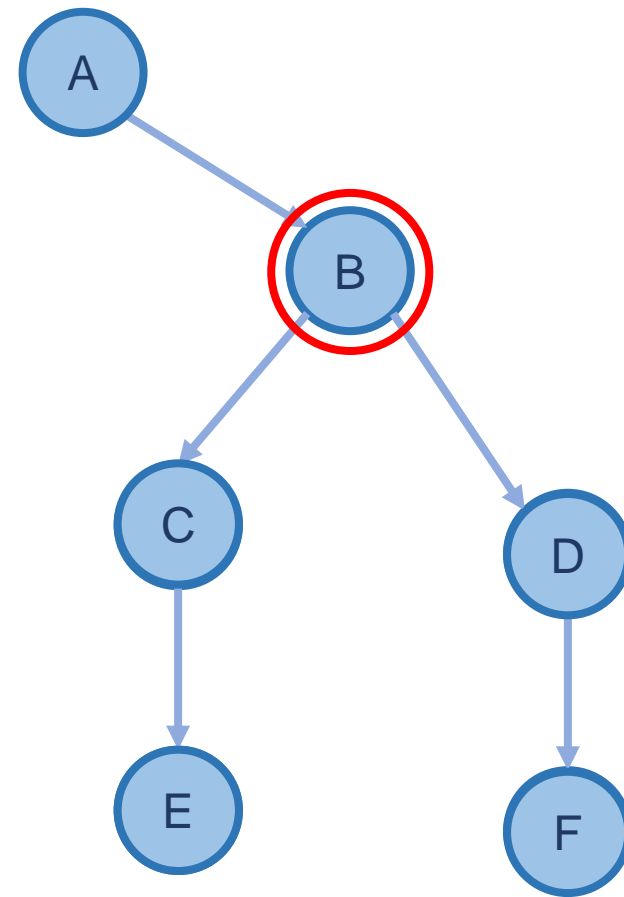
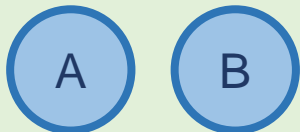


应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

已探索集合 Explored set:



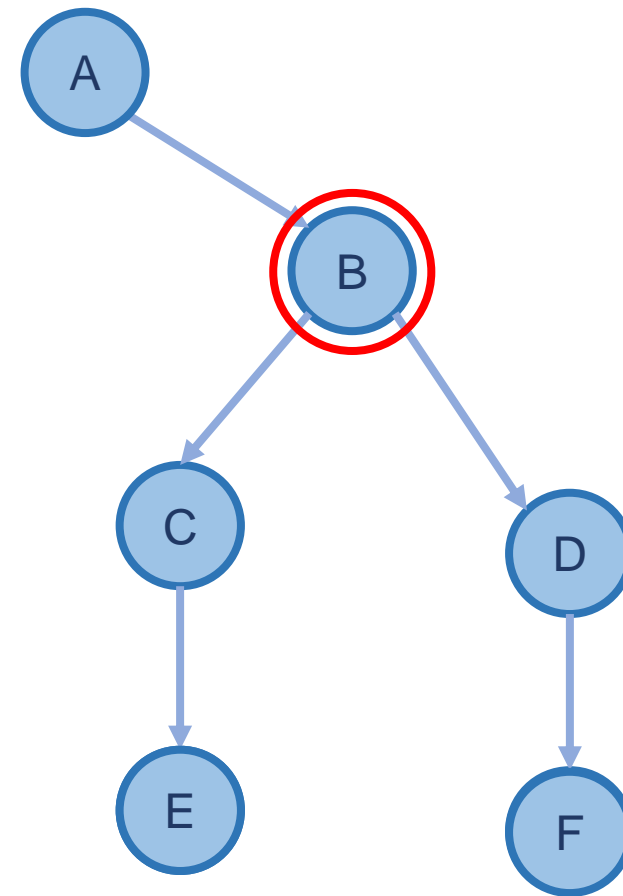
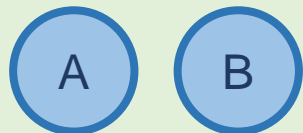
应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



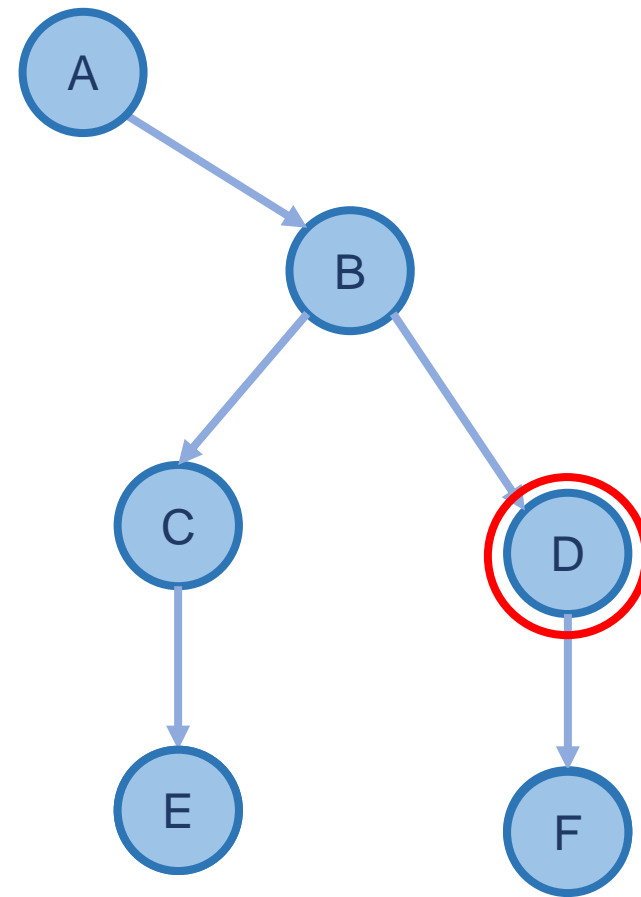
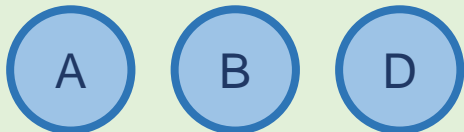
应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



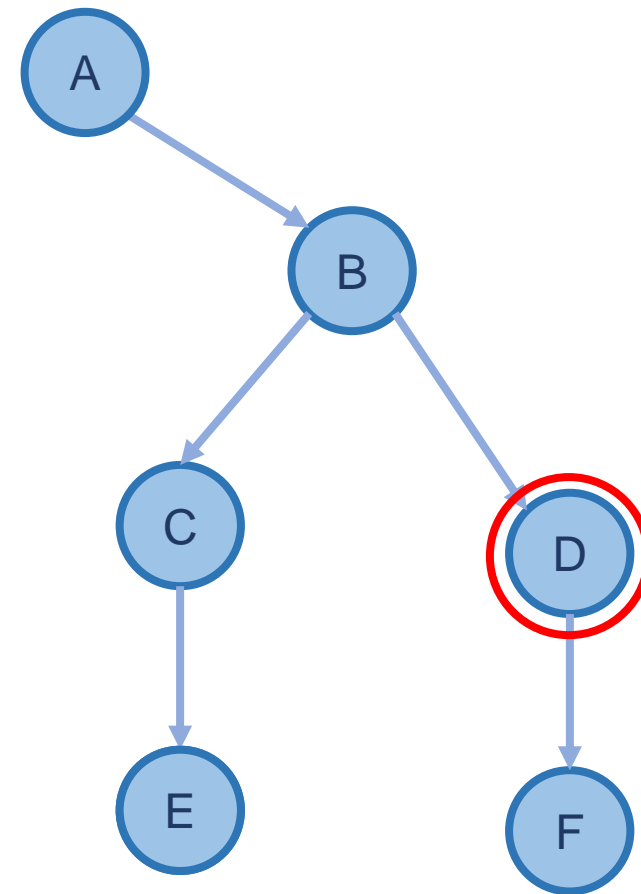
应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



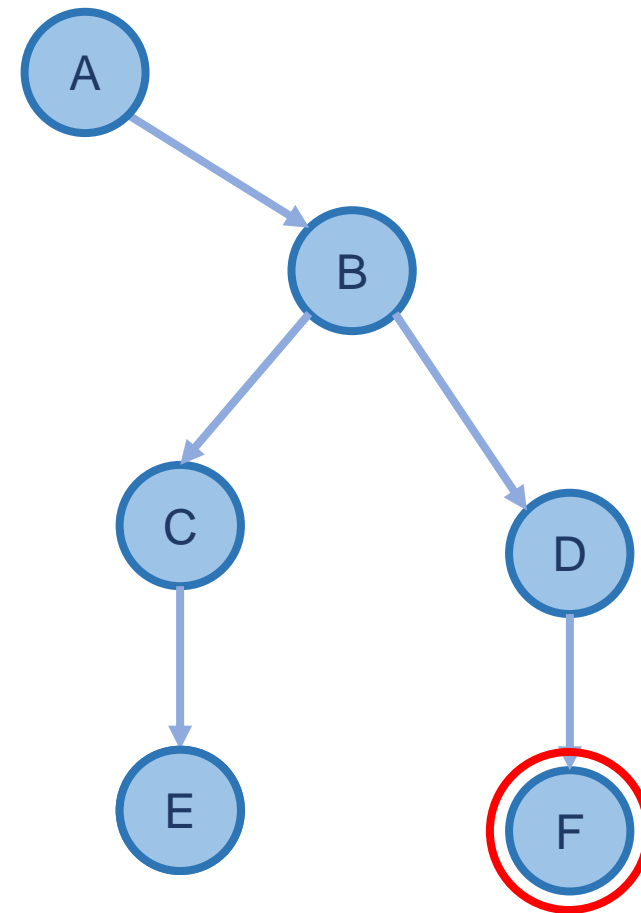
应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:

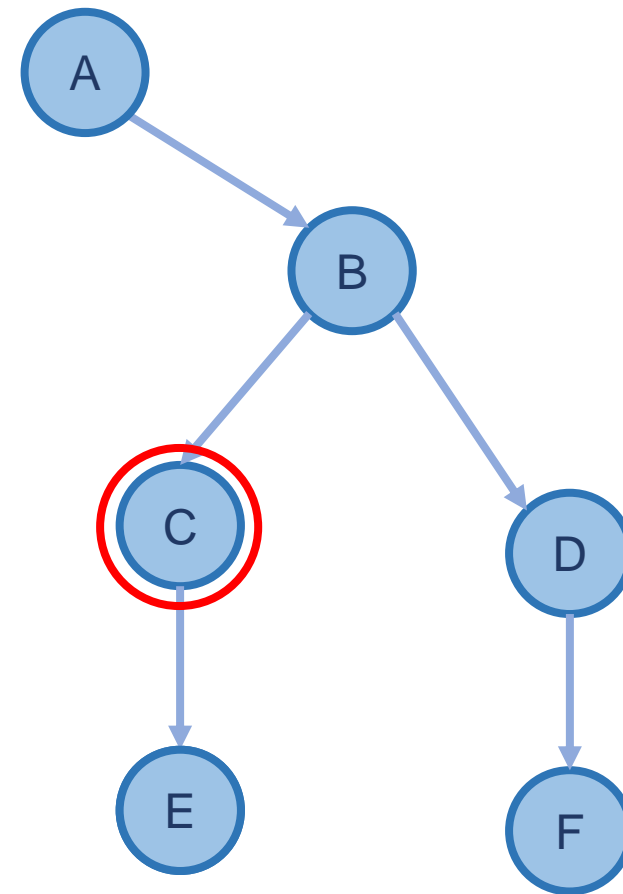


应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

已探索集合 Explored set:



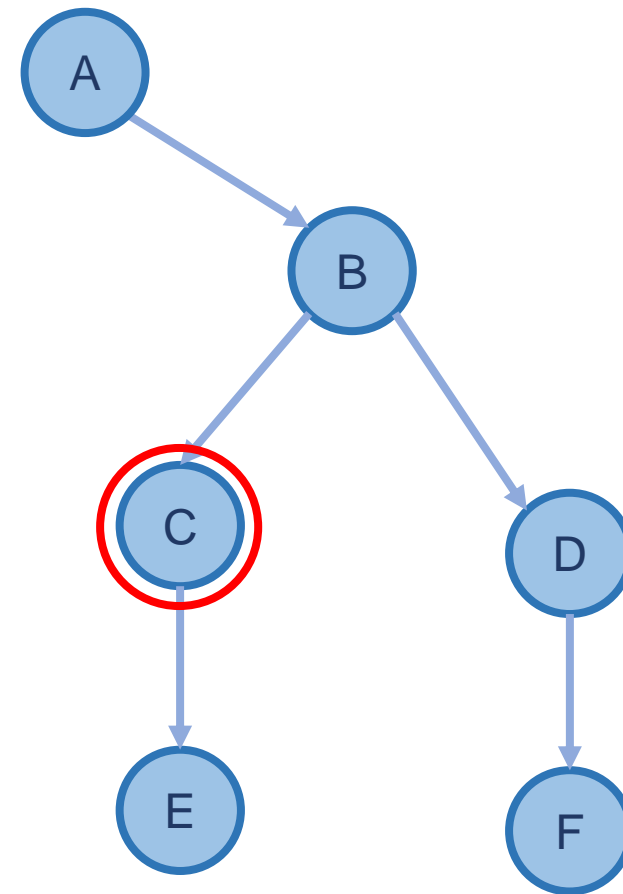
应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



应该先选择哪个节点？

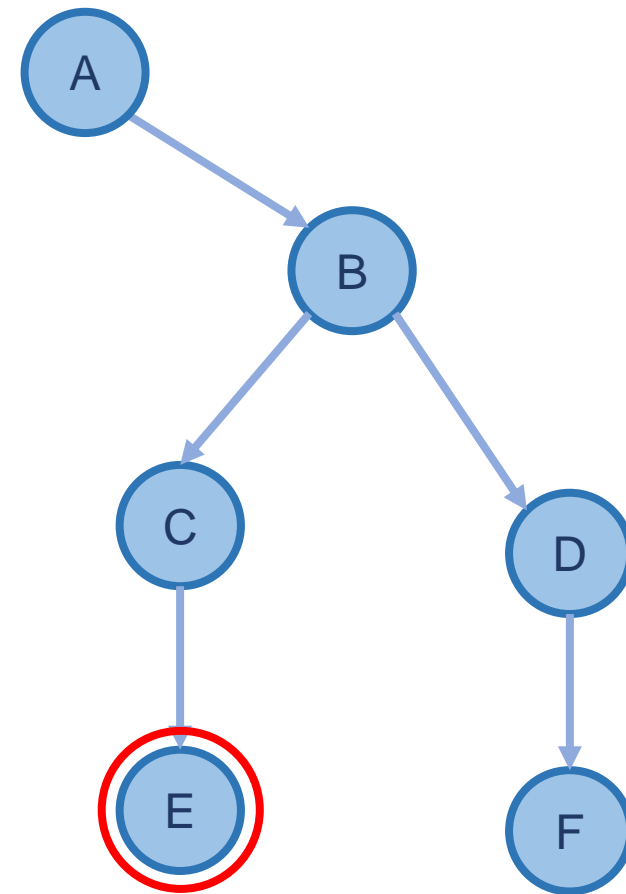
- 后进先出 Last-in first-out (堆栈 stack)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

已探索集合 Explored set:



- 深度优先搜索 Depth-first search (DFS)

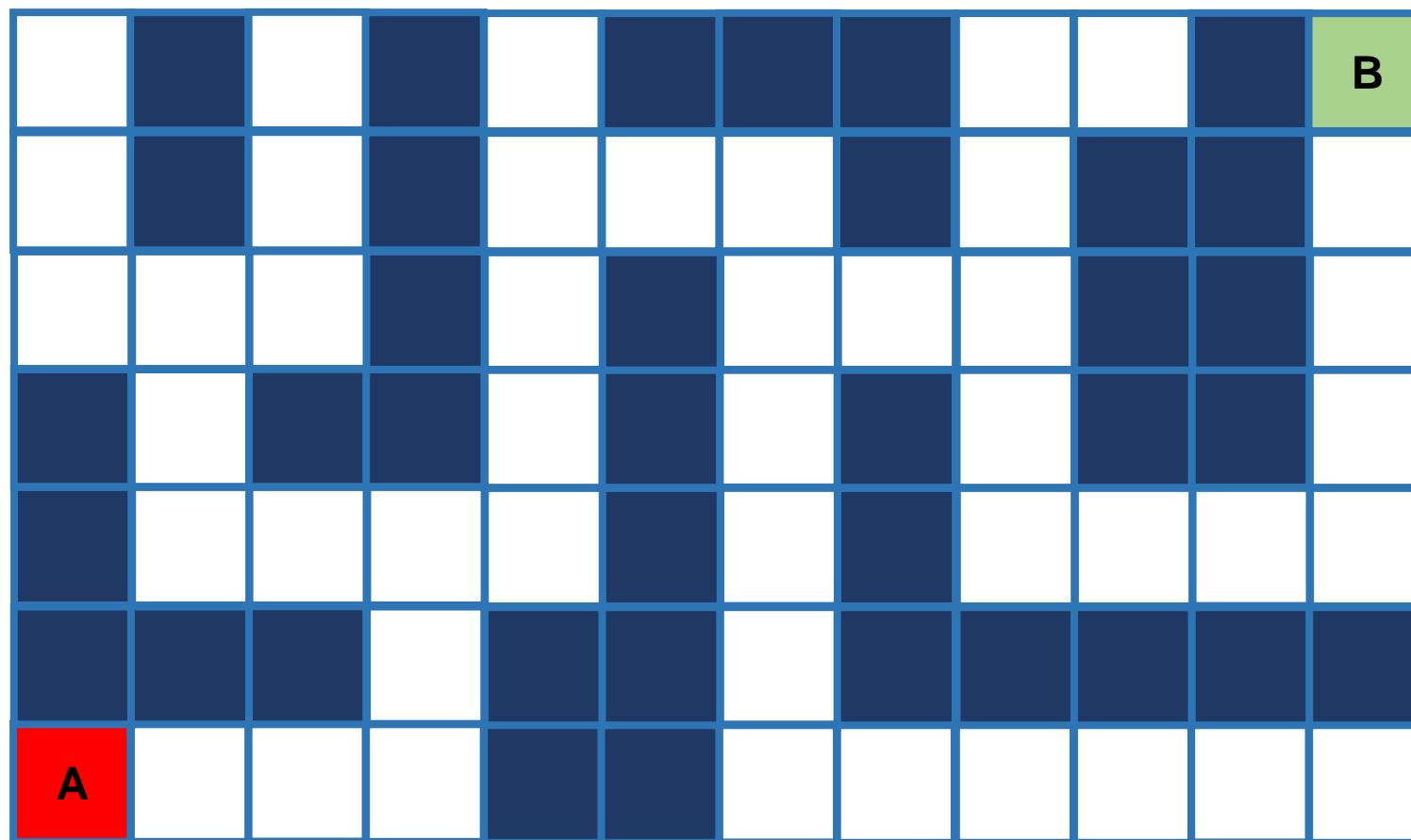


练习 #2

- 使用DFS找到从A到B的路径

- 多种选择:

- 右, 上, 左, 下

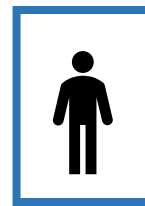


深度优先搜索 Depth-first search

- 总是扩展边界中**最深的**节点的搜索算法
 - 只要状态是**有限的**，就能找到解决方案

应该先选择哪个节点？

- 后进先出 Last-in first-out (堆栈数据结构 stack data structure)
- 先进先出 First-in first out (队列数据结构 queue data structure)

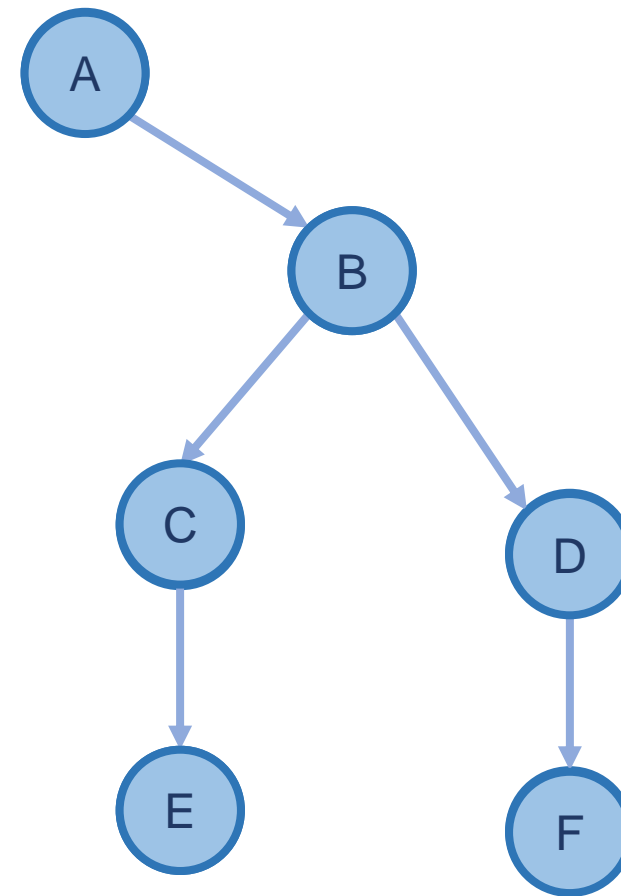


应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

已探索集合 Explored set:



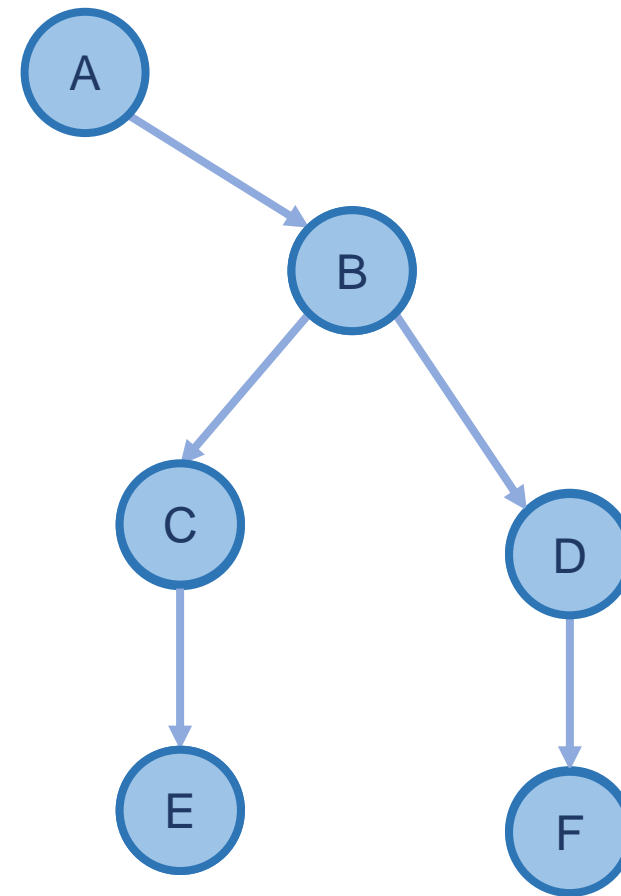
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:

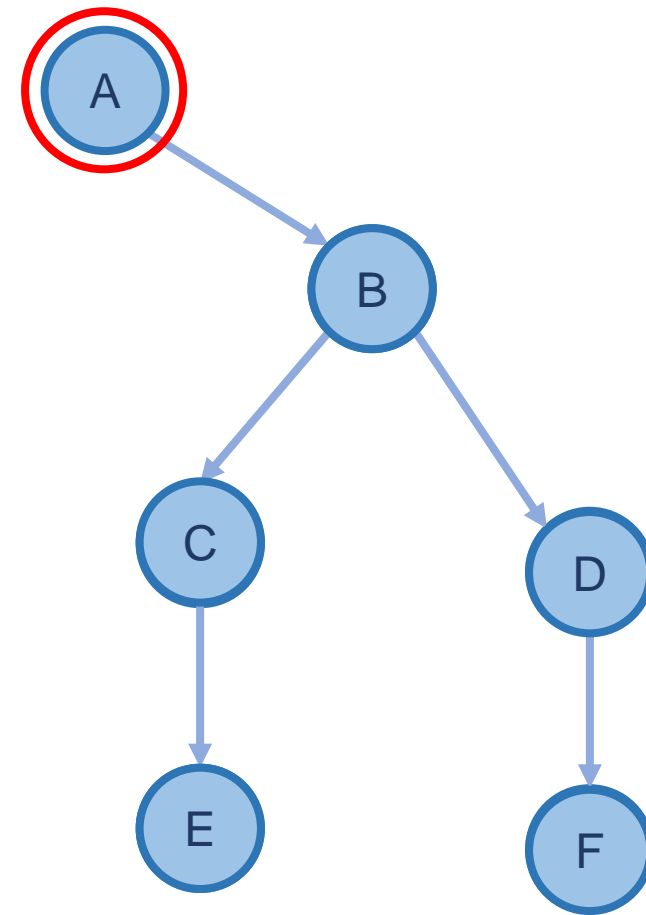
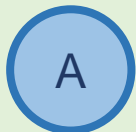


应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

已探索集合 Explored set:



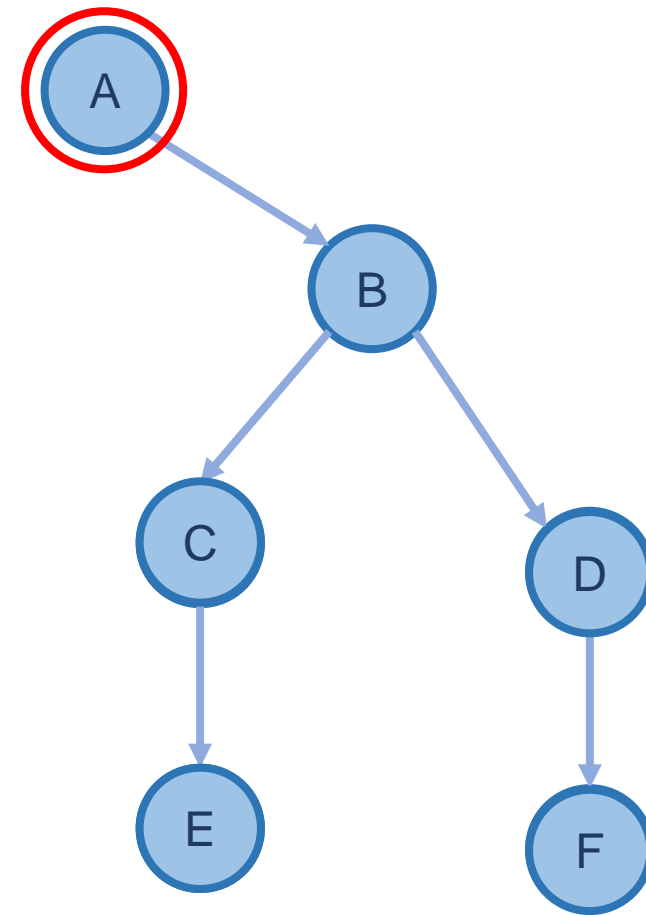
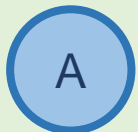
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:

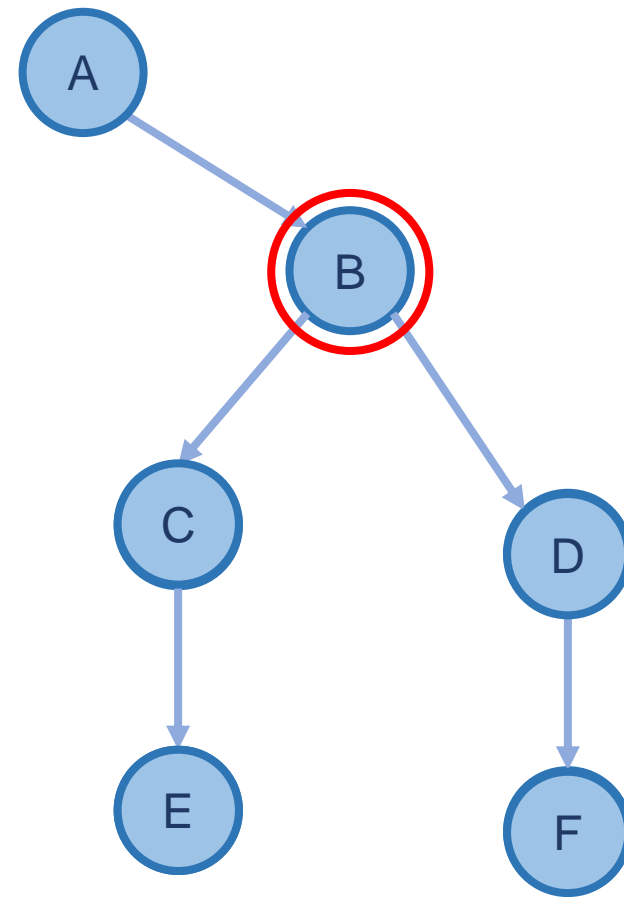
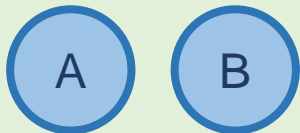


应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:

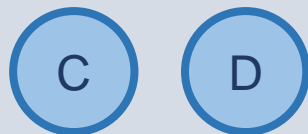
已探索集合 Explored set:



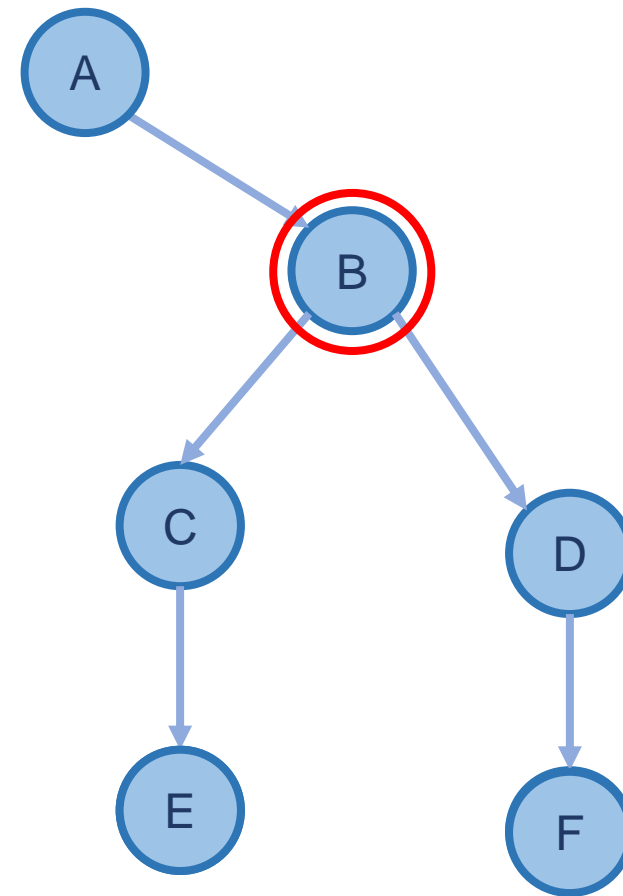
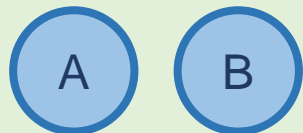
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



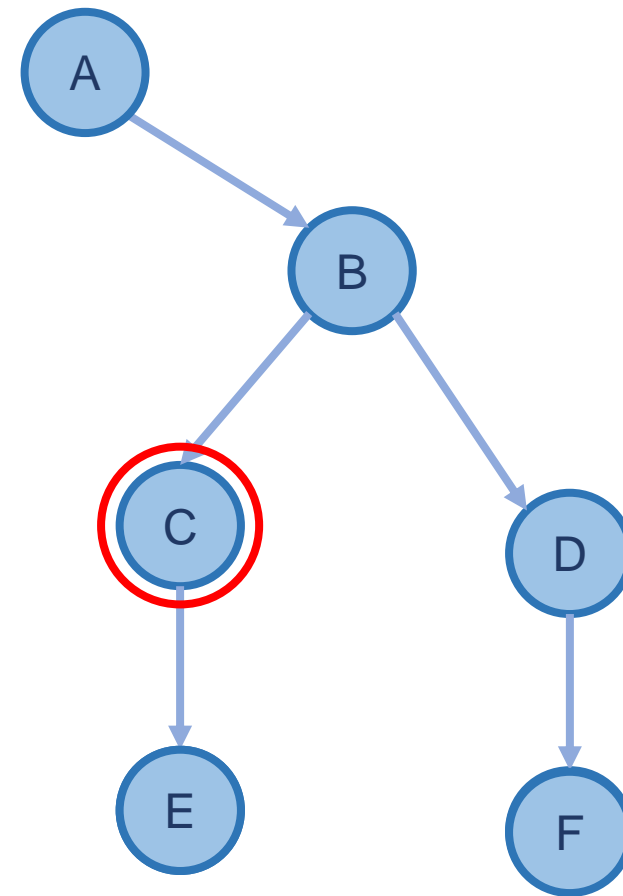
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



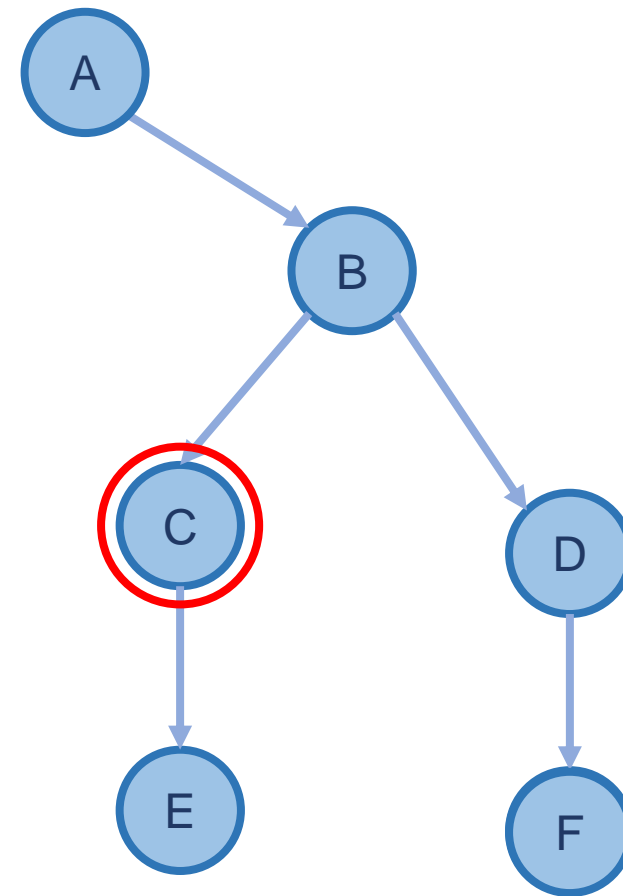
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



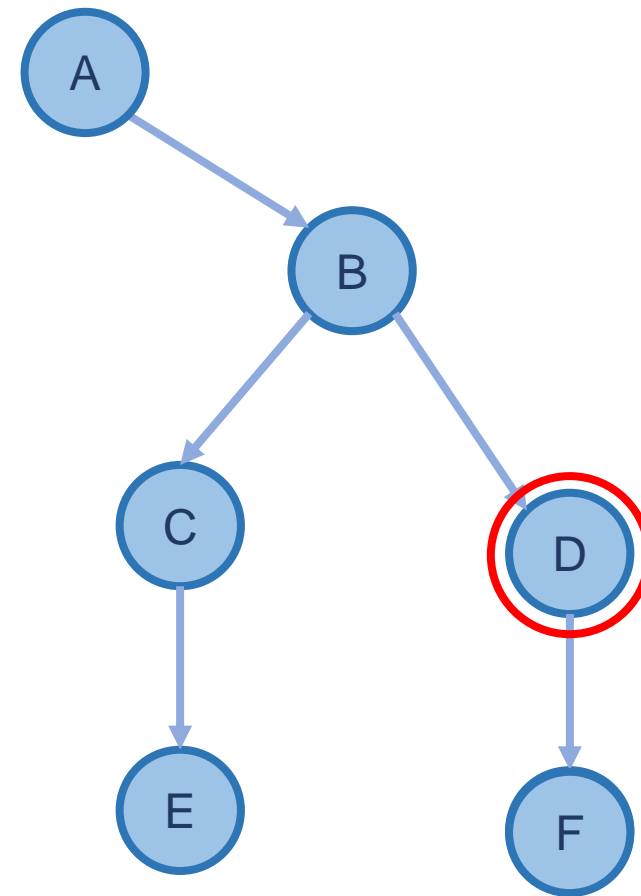
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



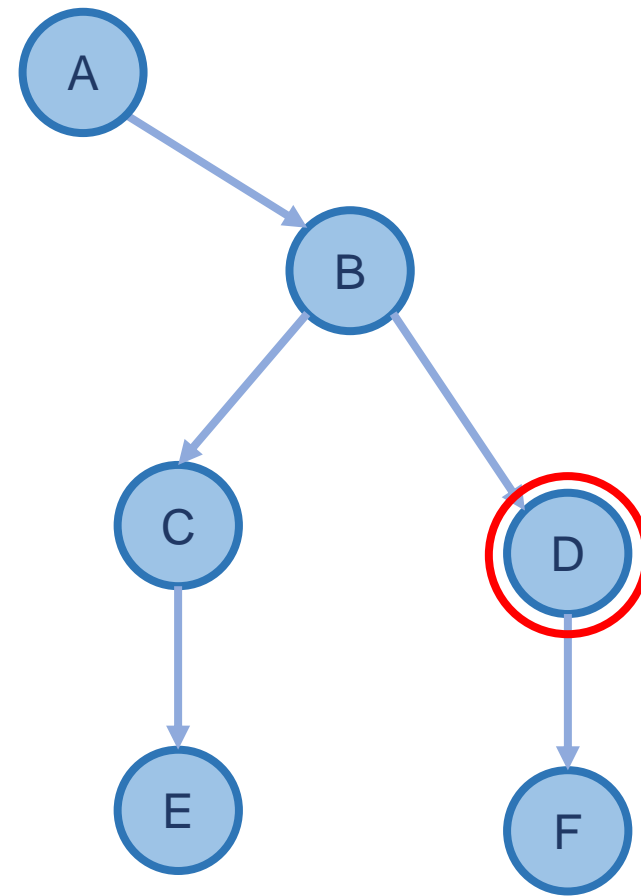
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



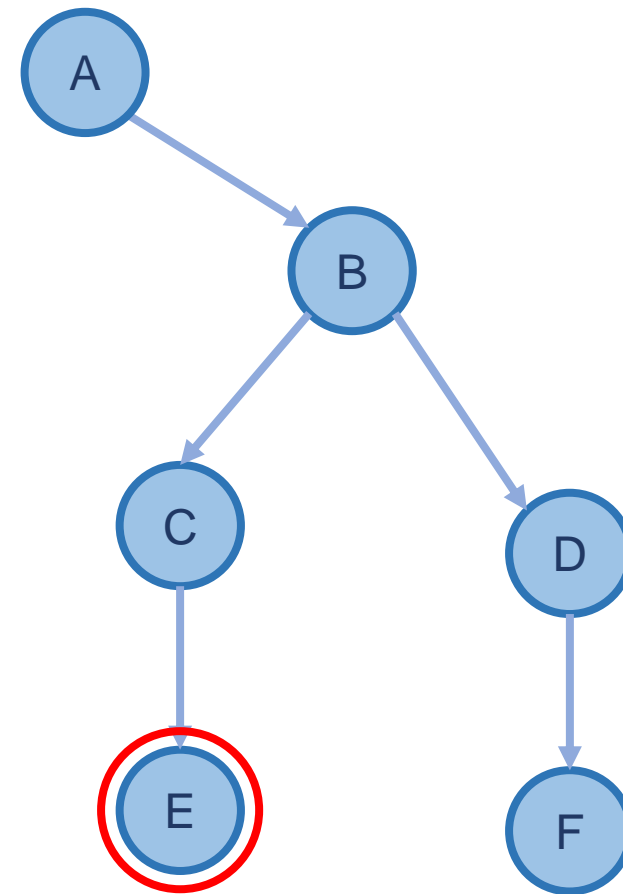
应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

搜索边界 Frontier:



已探索集合 Explored set:



应该先选择哪个节点？

- 先进先出 First-in first out (队列 queue)
- 找到从 A 到 E 的路径

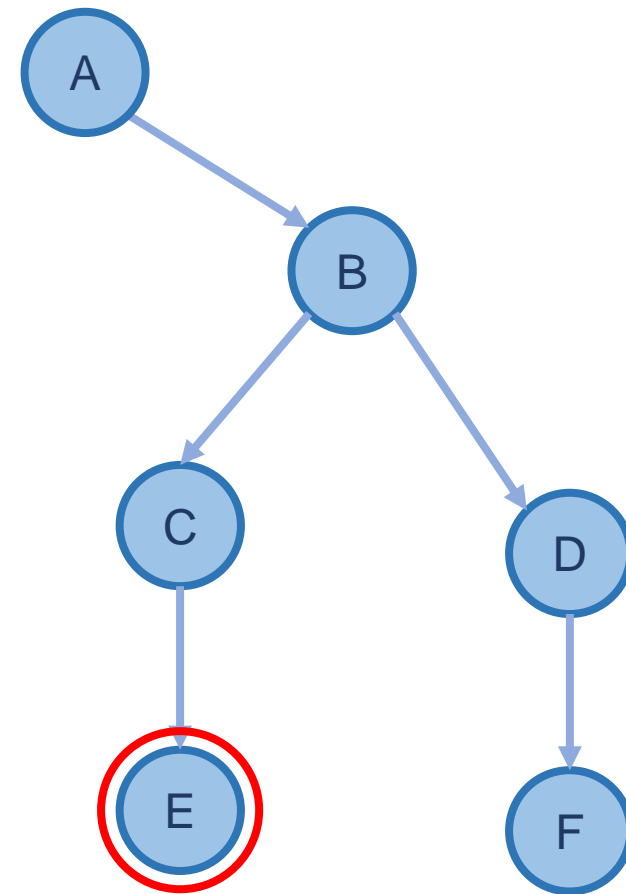
搜索边界 Frontier:



已探索集合 Explored set:



- 广度优先搜索 Breadth-first search

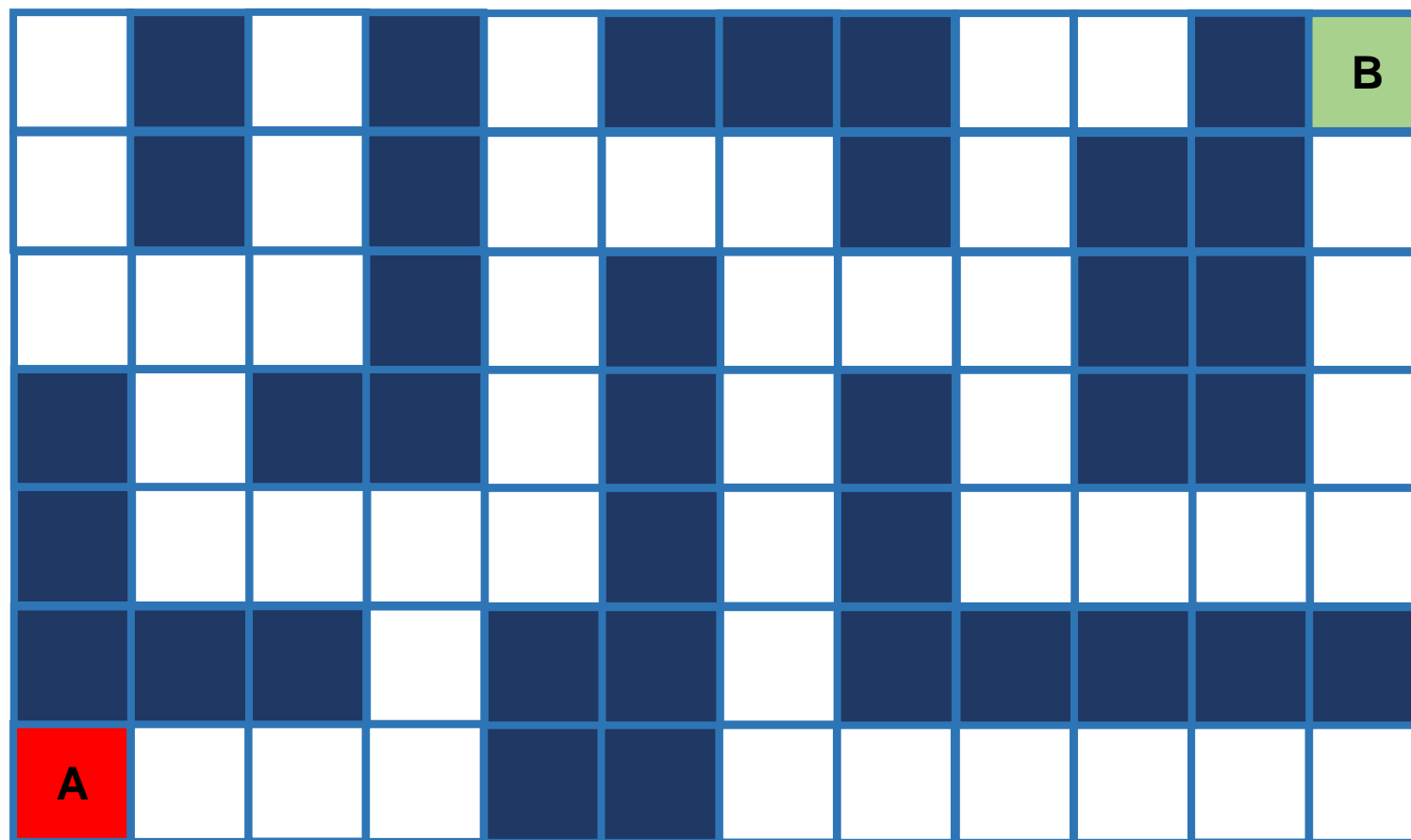


练习 #3

- 使用BFS找到从A到B的路径

- 多种选择:

- 左, 上, 右, 下

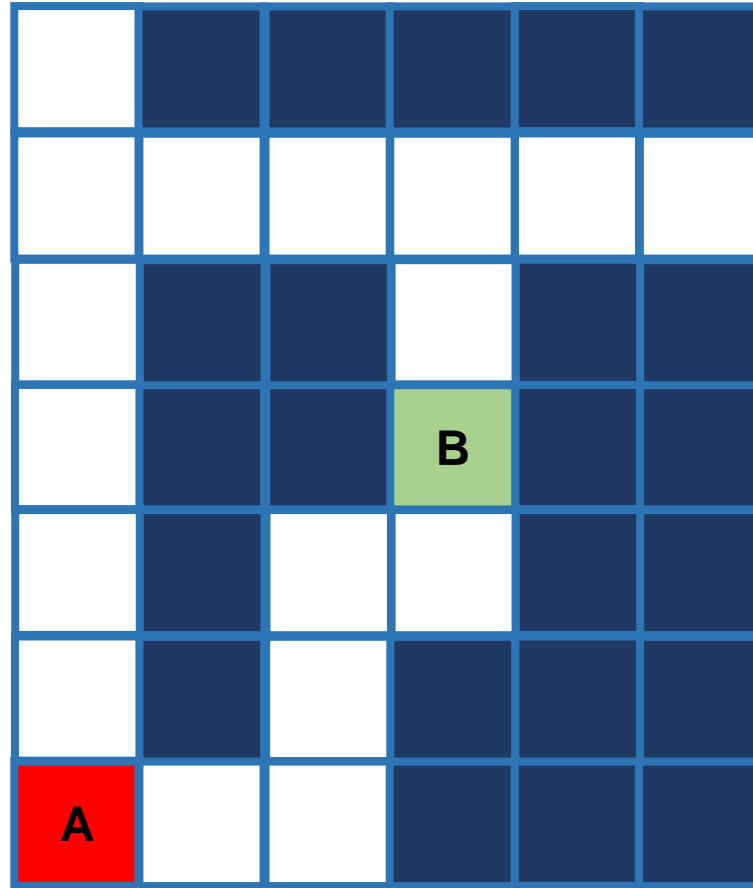


广度优先搜索 Breadth-first search

- 总是扩展边界中**最浅的**节点的搜索算法
 - 可能需要探索非常多的节点来找到解

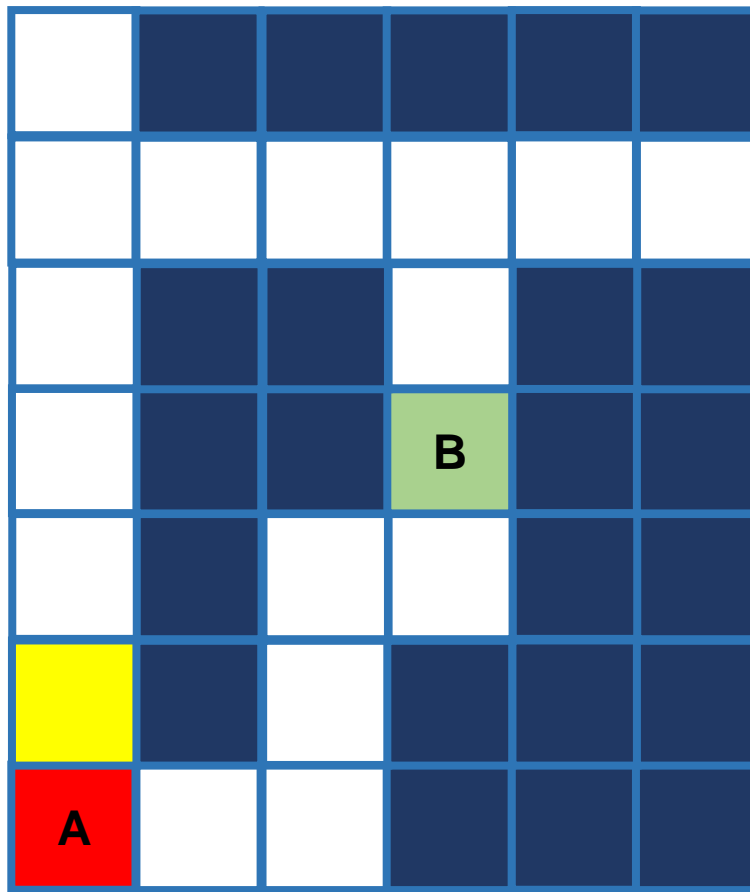
DFS会更好吗？

- 深度优先搜索



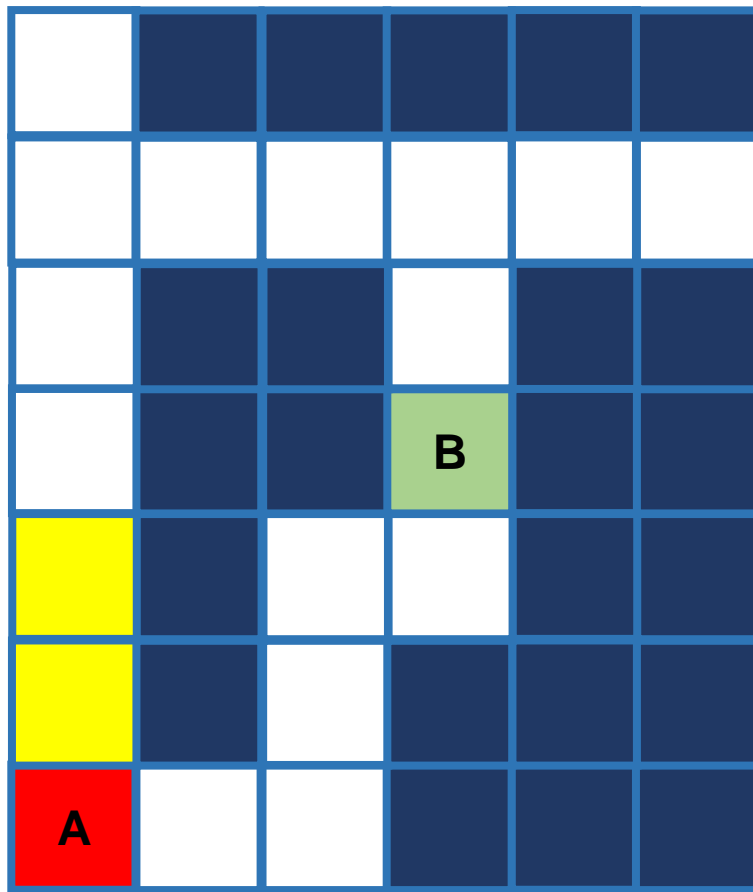
DFS会更好吗？

- 深度优先搜索



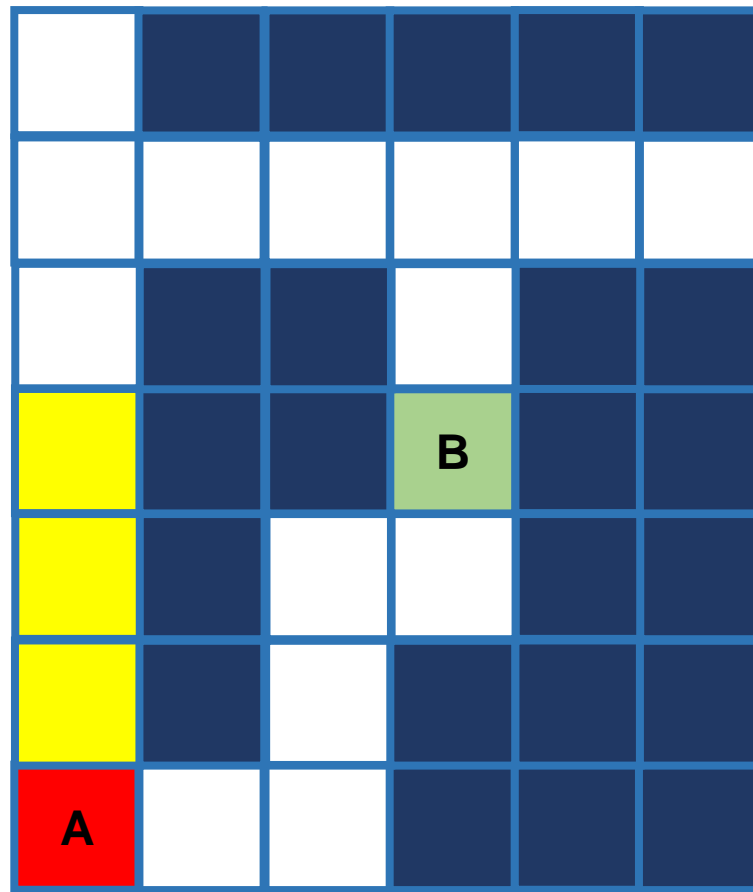
DFS会更好吗？

- 深度优先搜索



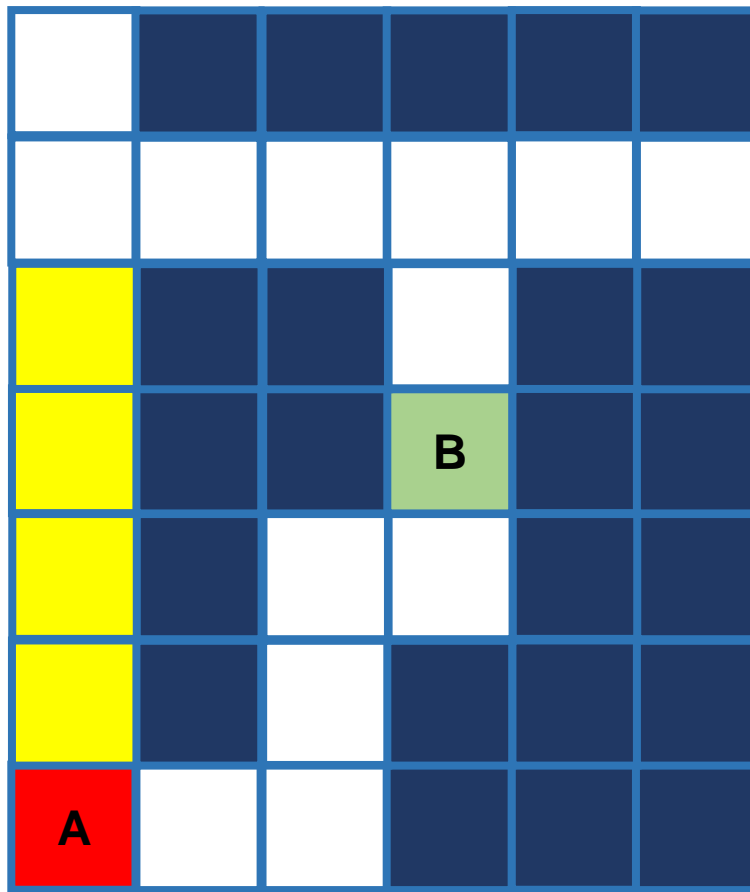
DFS会更好吗?

- 深度优先搜索



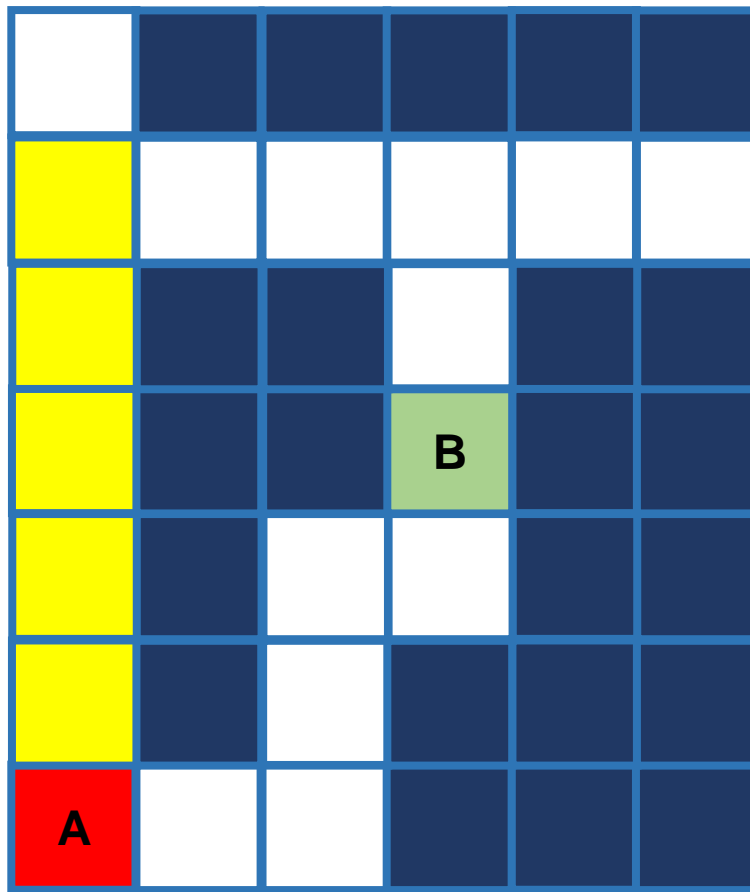
DFS会更好吗?

- 深度优先搜索



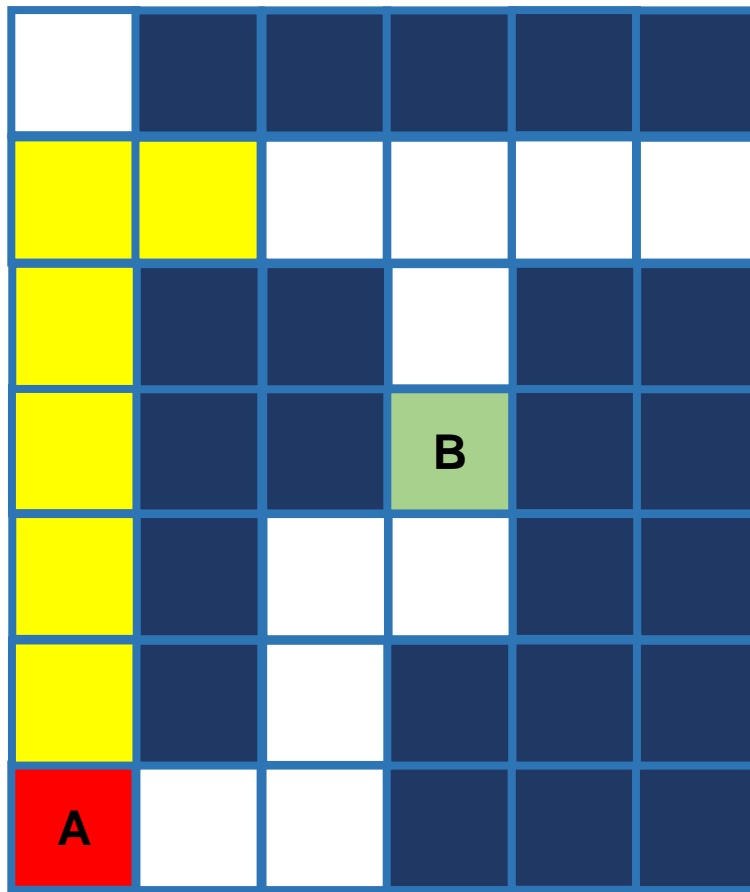
DFS会更好吗?

- 深度优先搜索



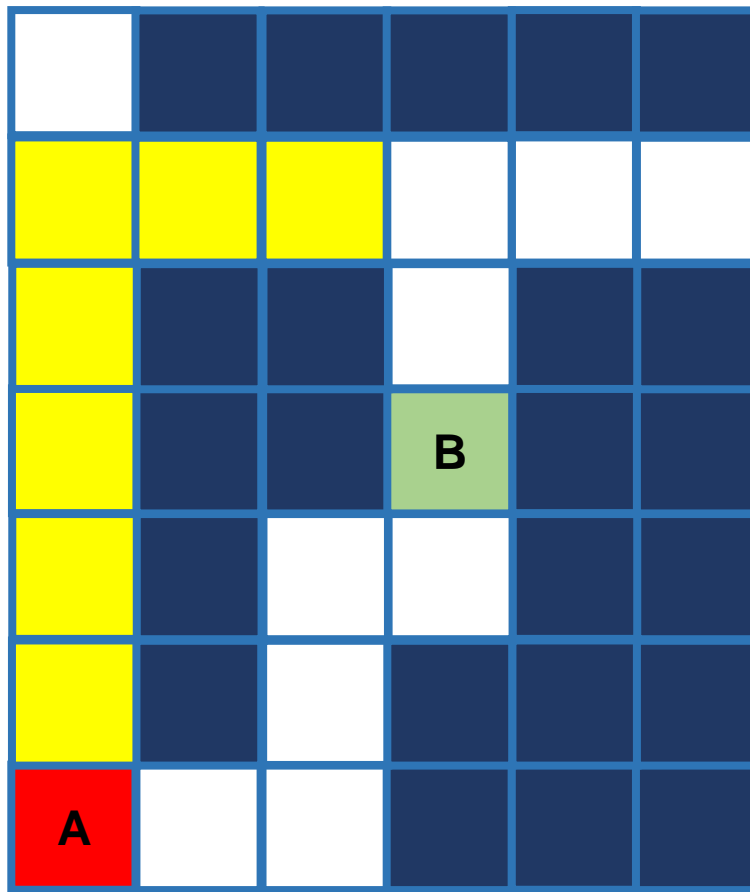
DFS会更好吗?

- 深度优先搜索



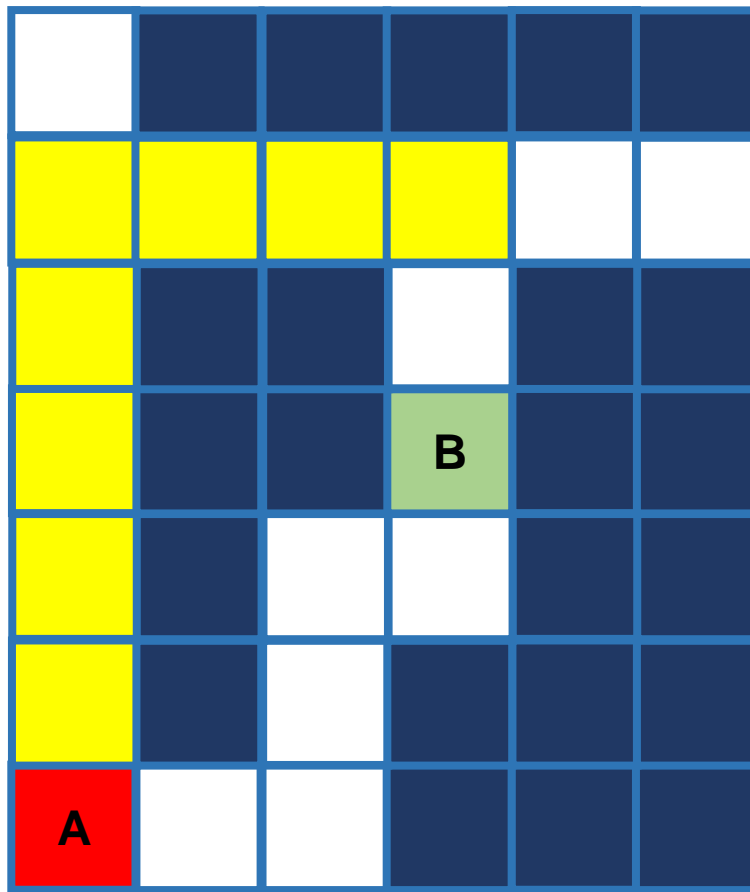
DFS会更好吗?

- 深度优先搜索



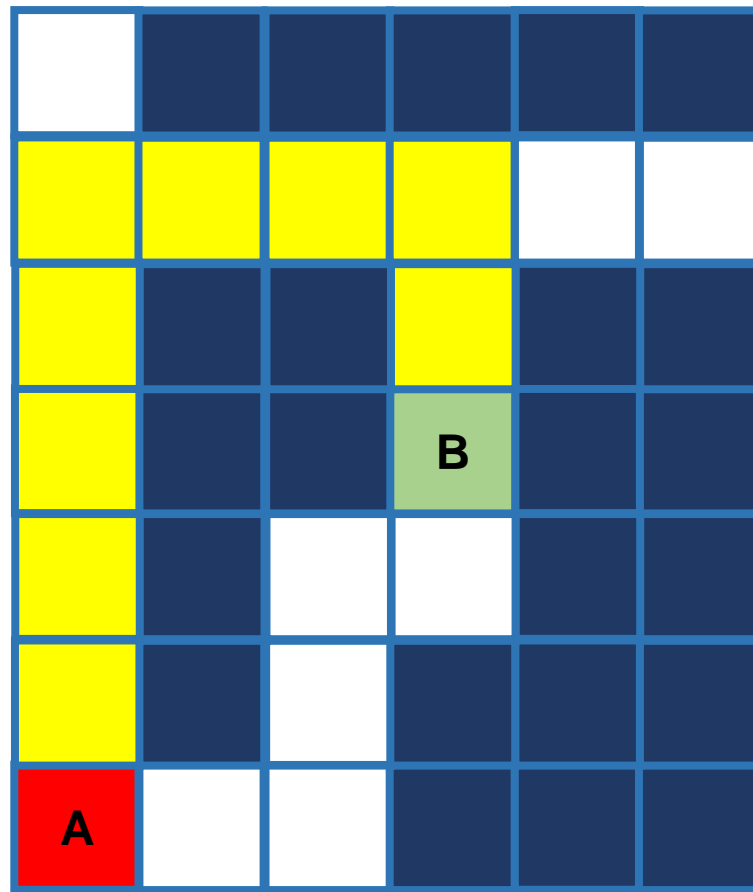
DFS会更好吗?

- 深度优先搜索



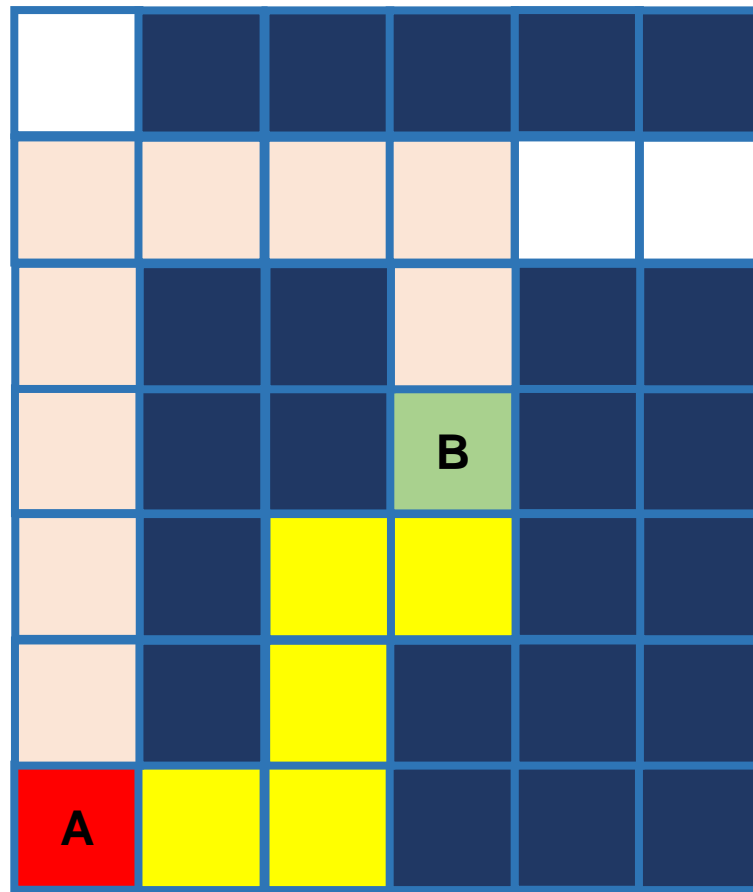
DFS会更好吗?

- 深度优先搜索



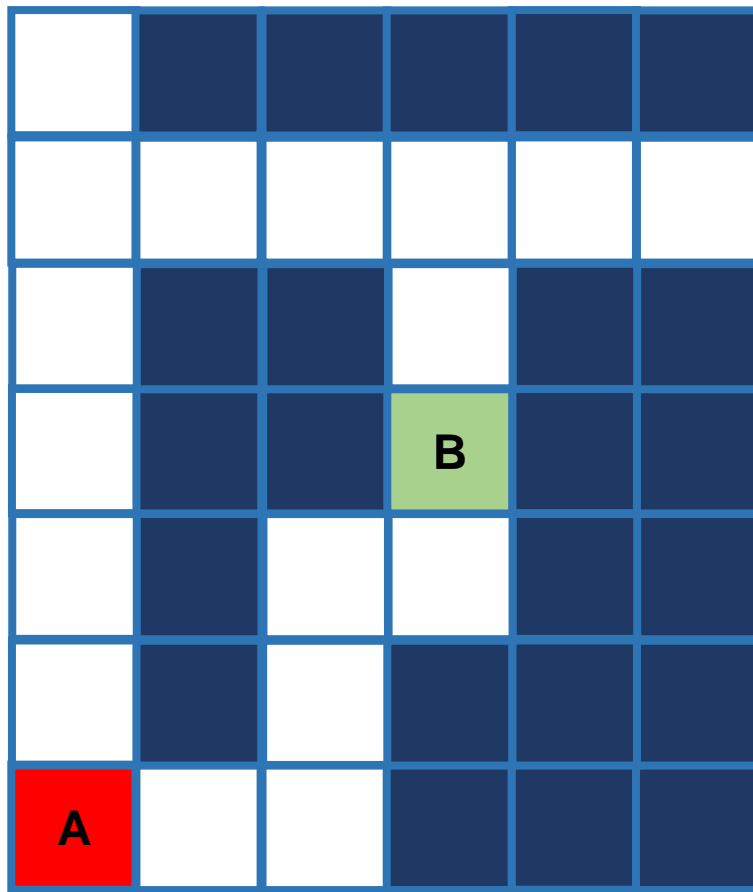
DFS会更好吗？

- 深度优先搜索
 - 可能并不总是找到最佳解决方案



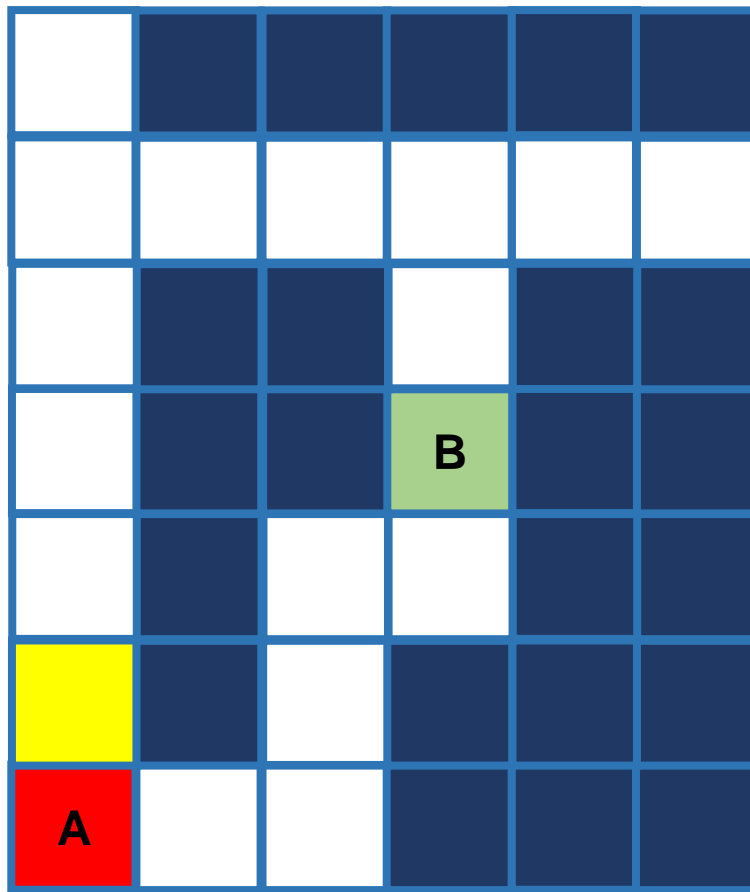
DFS会更好吗?

- 广度优先搜索



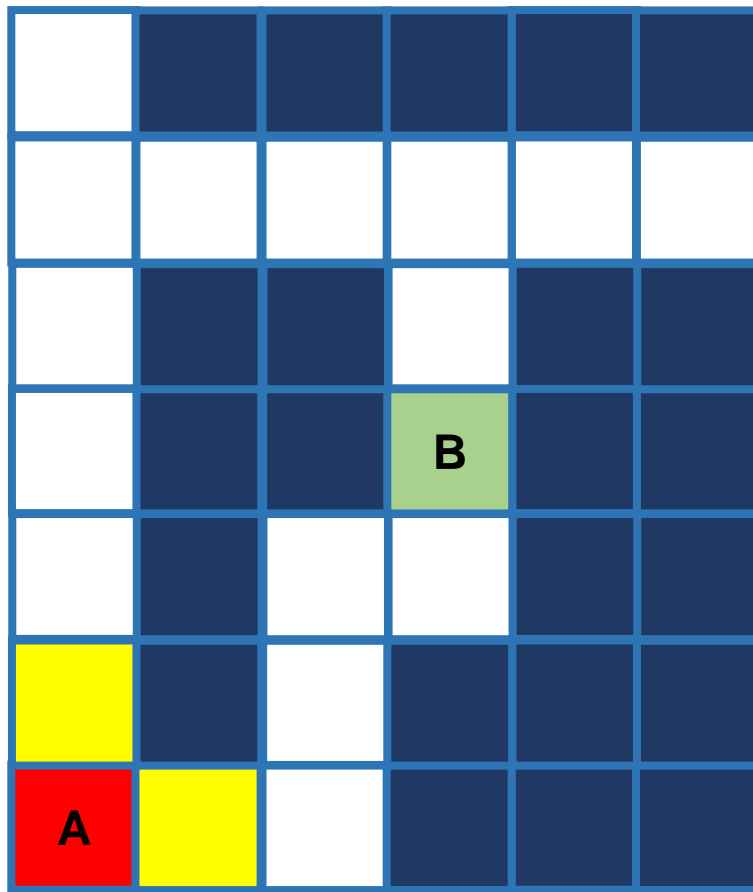
DFS会更好吗?

- 广度优先搜索



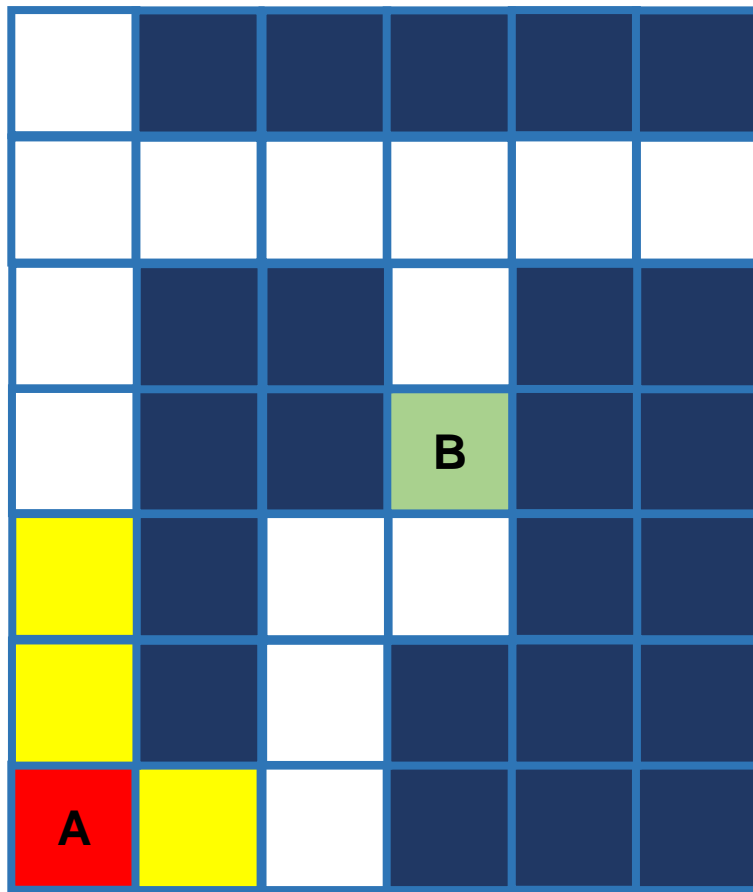
DFS会更好吗?

- 广度优先搜索



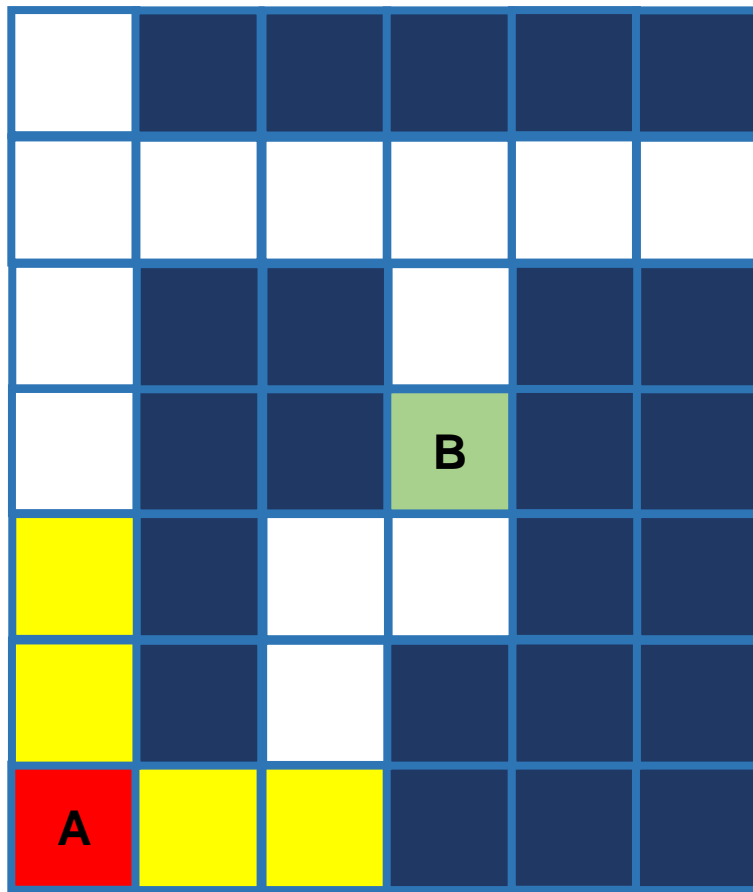
DFS会更好吗?

- 广度优先搜索



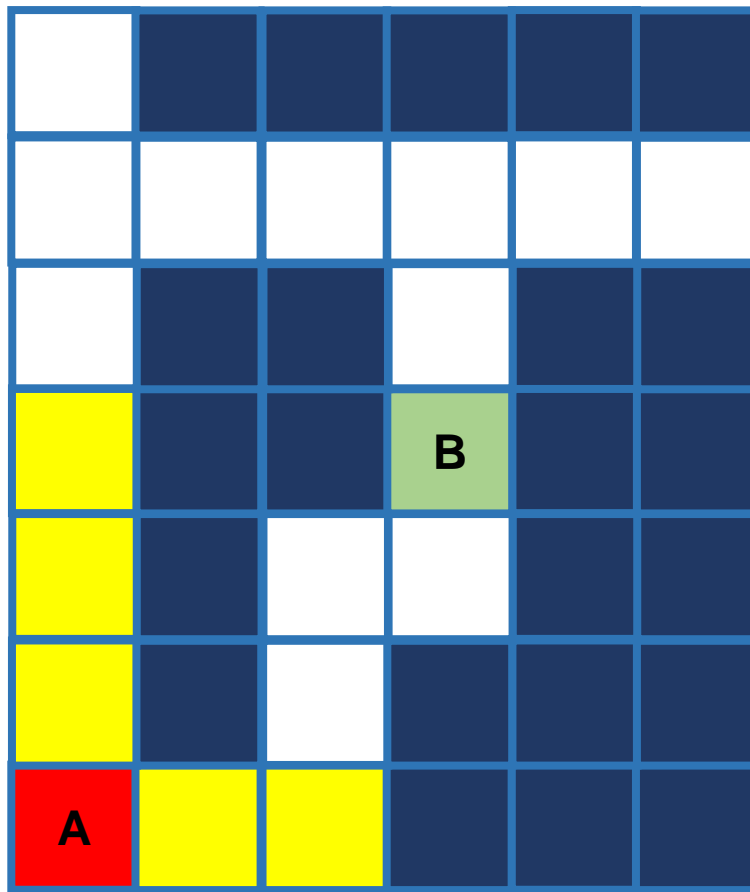
DFS会更好吗?

- 广度优先搜索



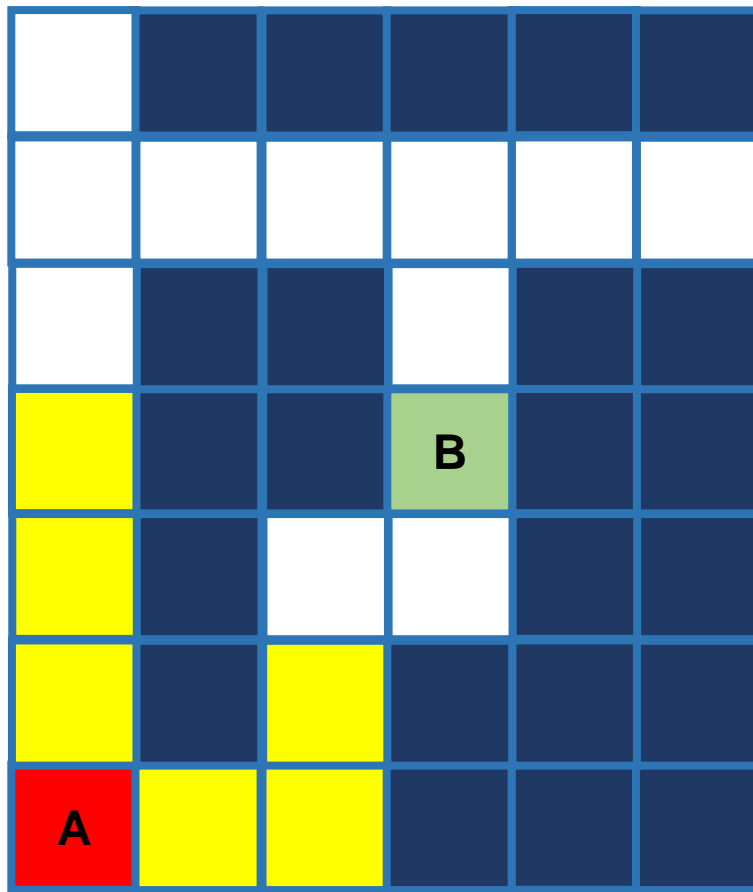
DFS会更好吗?

- 广度优先搜索



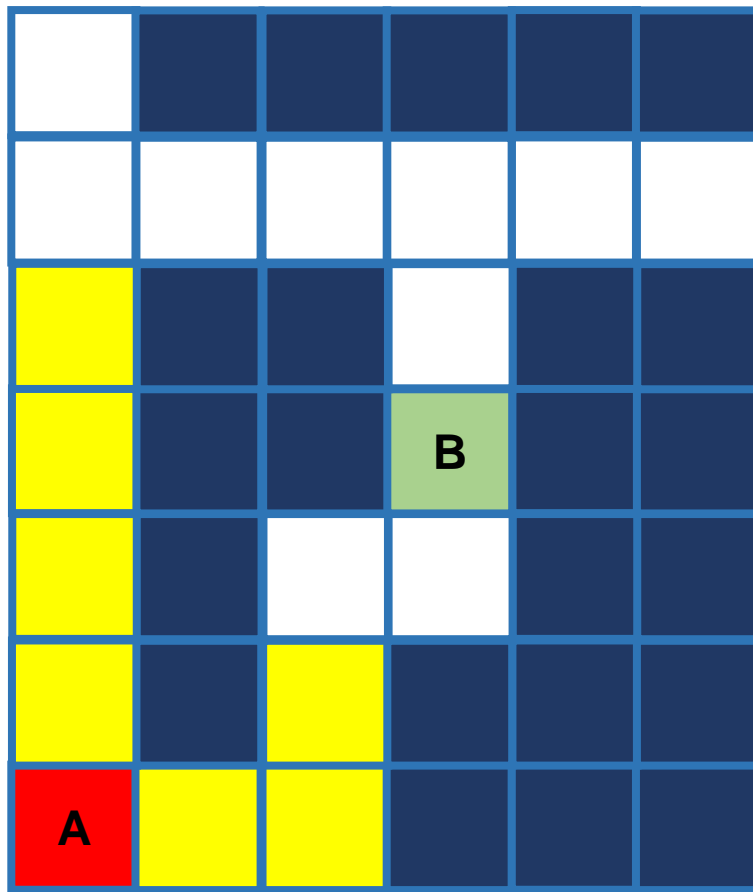
DFS会更好吗?

- 广度优先搜索



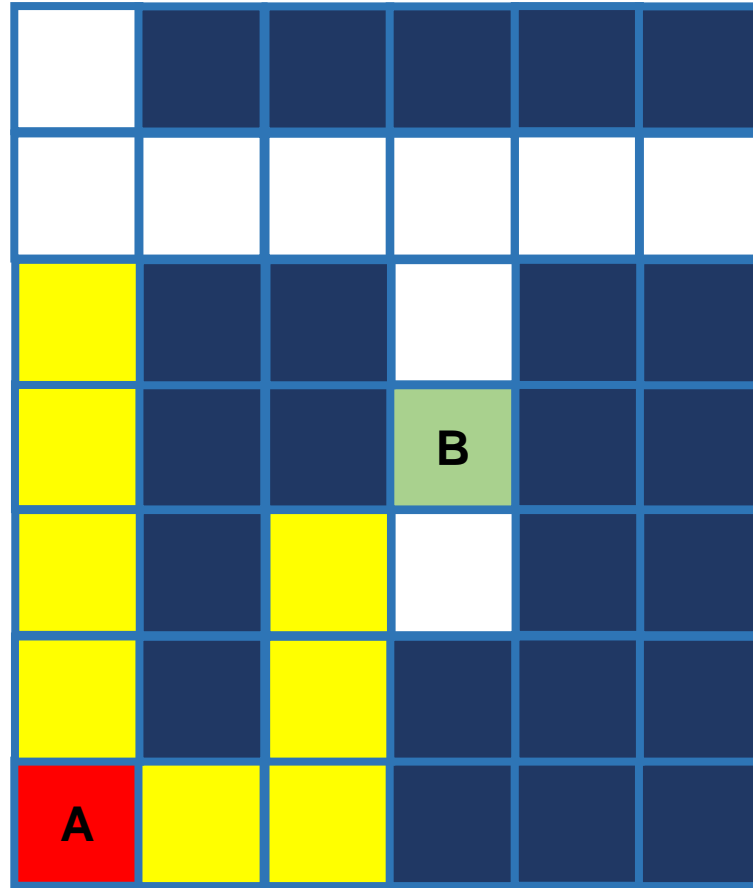
DFS会更好吗?

- 广度优先搜索



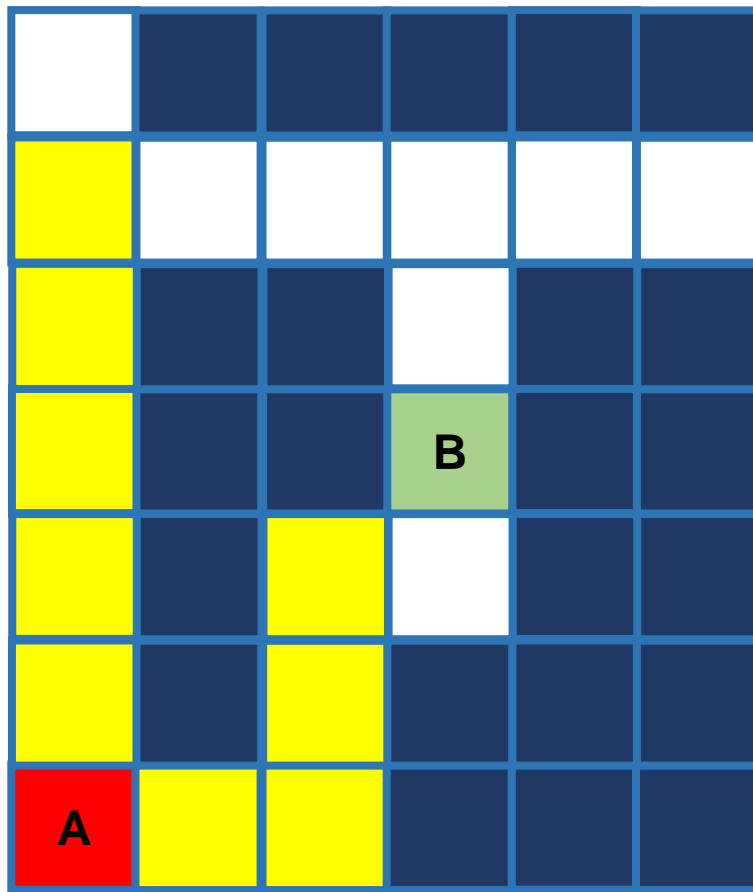
DFS会更好吗？

- 广度优先搜索



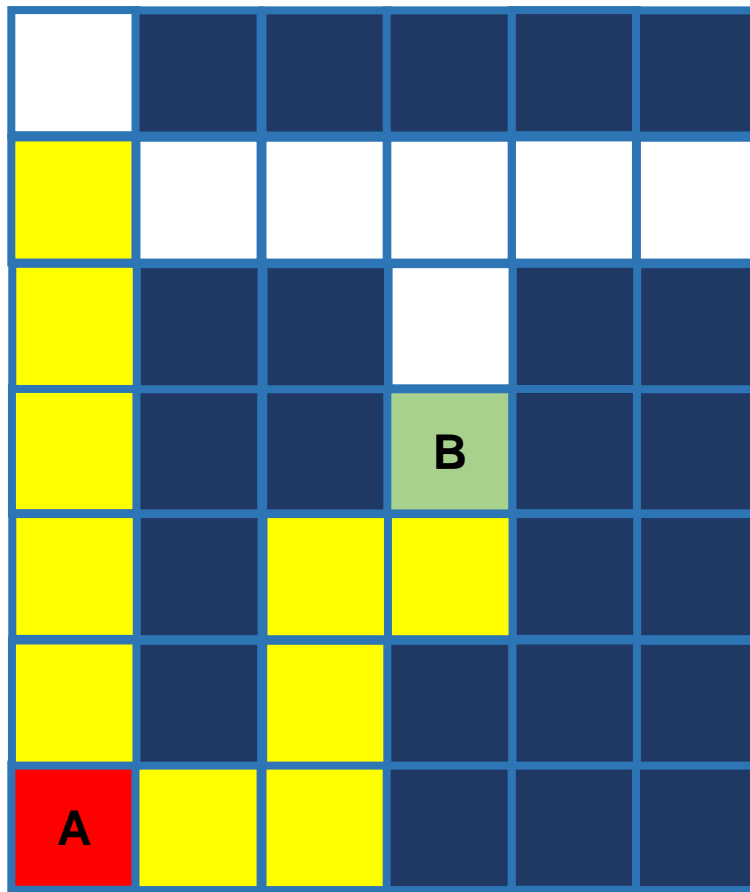
DFS会更好吗?

- 广度优先搜索



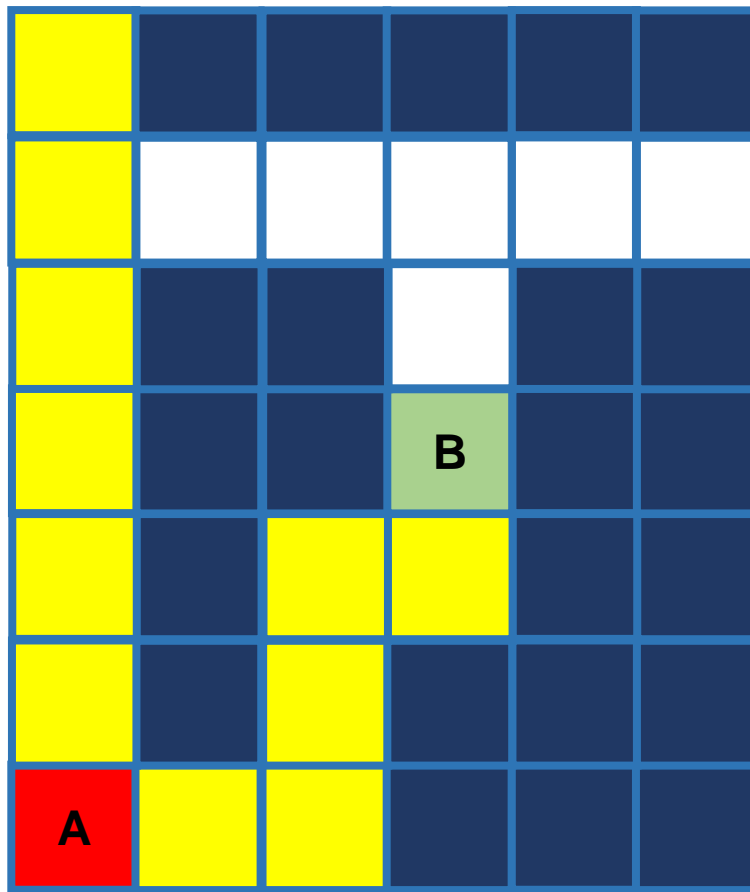
DFS会更好吗?

- 广度优先搜索



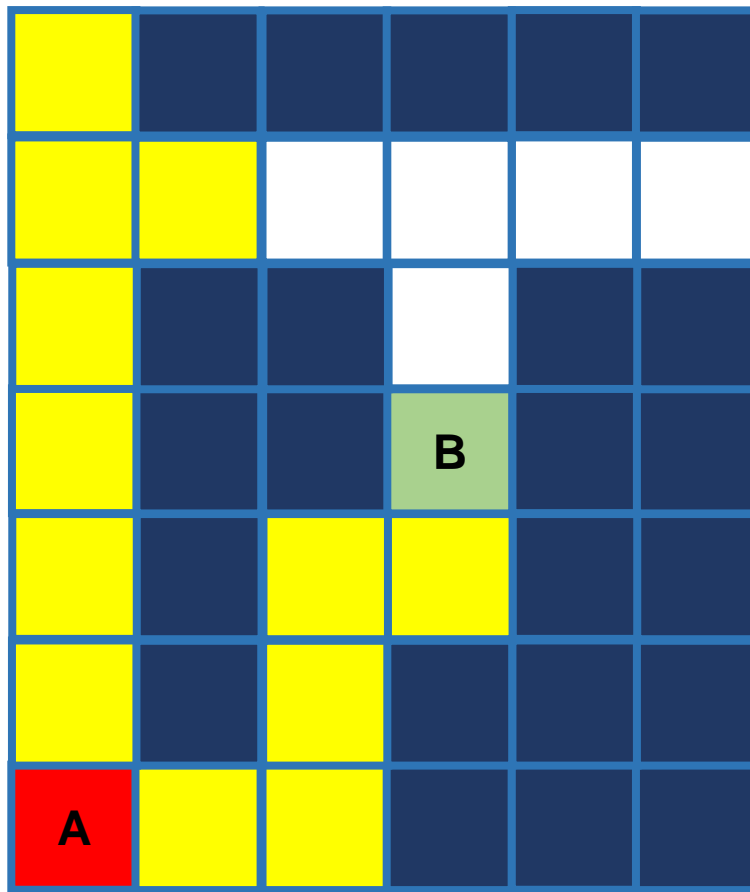
DFS会更好吗?

- 广度优先搜索



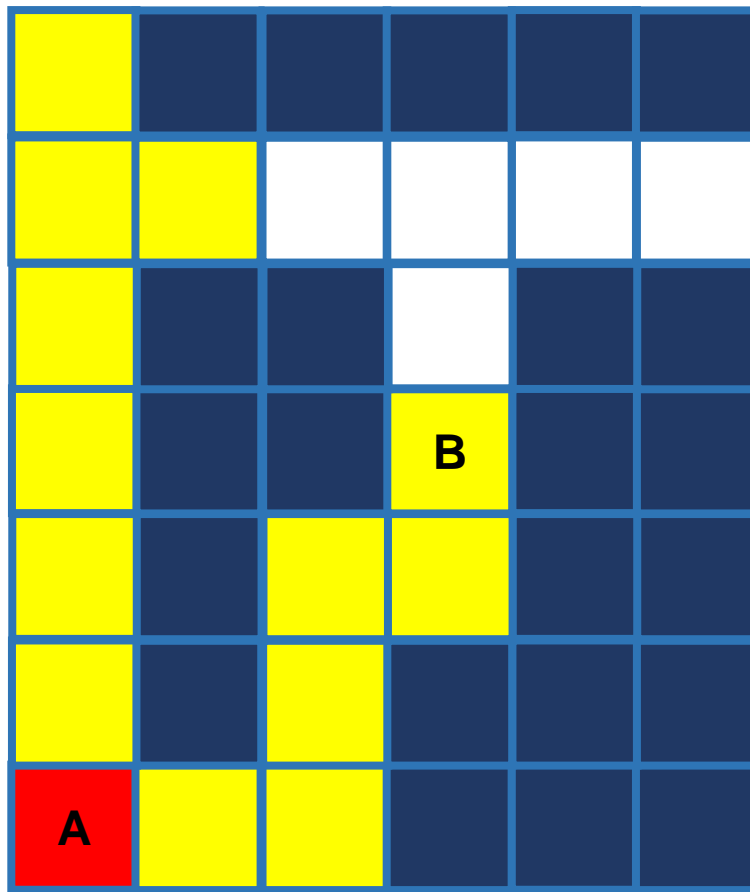
DFS会更好吗?

- 广度优先搜索



DFS会更好吗?

- 广度优先搜索



有问题吗？

- 请随时举手提问。



BUSS 3620.人工智能导论

#3. 代码示例：迷宫

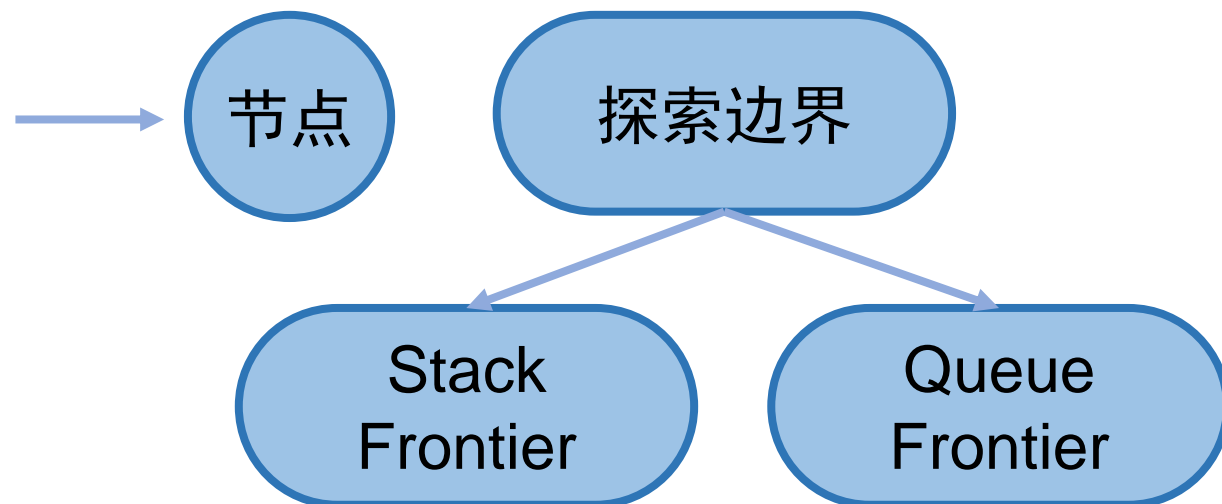
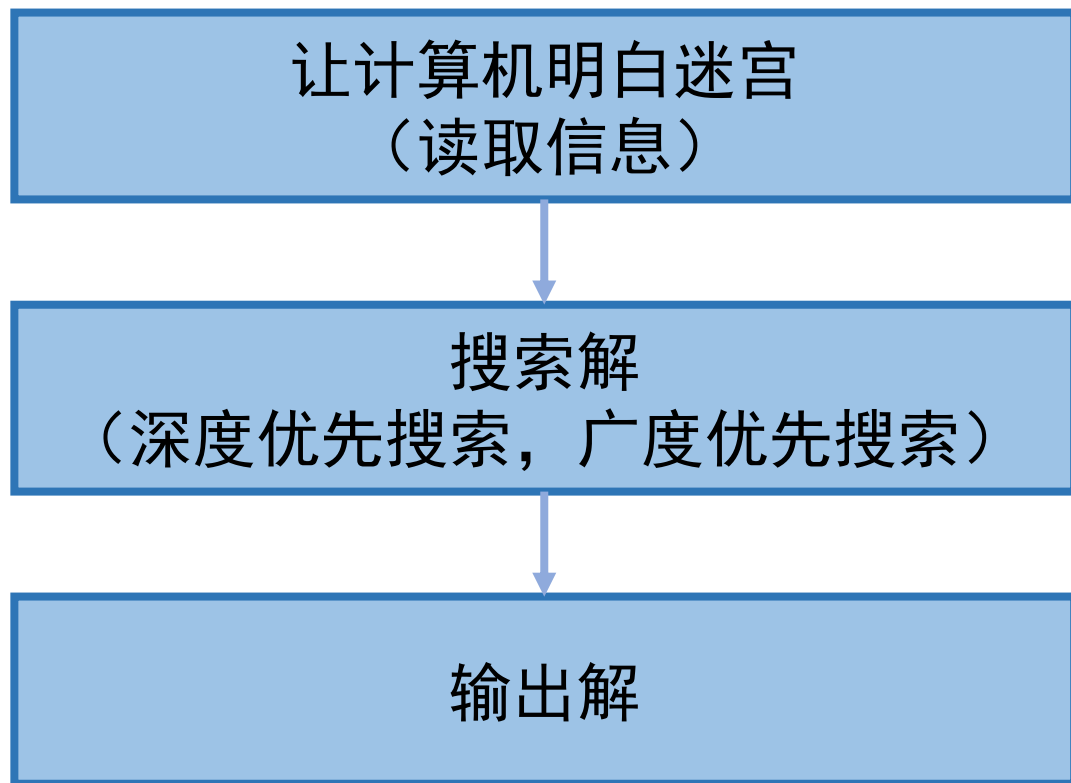
刘佳璐

安泰经济与管理学院

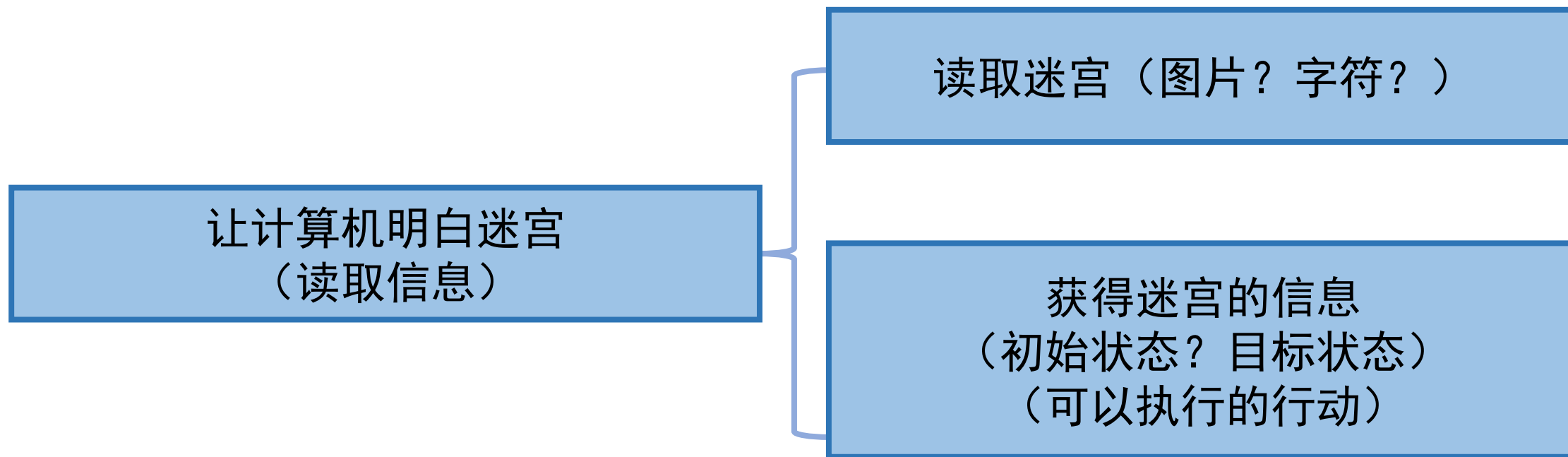
上海交通大学

代码框架

- 步骤:

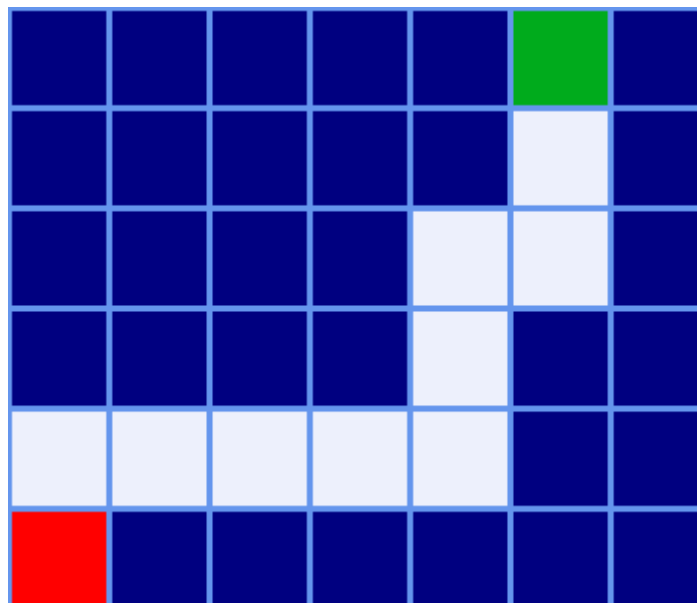


解迷宫 – 让计算机明白迷宫



解迷宫 – 让计算机明白迷宫

- 在txt文件中表示迷宫（易于阅读）
 - # 表示墙壁
 - 空格表示可以同行的路径
 - A 代表起始状态
 - B 代表目标状态



1	#####B#
2	##### #
3	##### #
4	##### ##
5	#####
6	A#####

解迷宫 – 让计算机明白迷宫

- 初始化迷宫
 - 读取迷宫txt文件
 - 验证是否含有初始状态和目标状态
 - 找到可以执行的行动
 - 确定迷宫的高度和宽度
 - 找到墙壁的位置(由(i, j)表示)
 - 找到起始状态和目标状态的位置
 - 初始化迷宫的解(空的)

```
42 class Maze():
43
44     def __init__(self, filename):
45
46         # Read file
47         with open(filename) as f:
48             contents = f.read()
49
50         # Validate start and goal
51         if contents.count("A") != 1:
52             raise Exception("maze must have exactly one start point")
53         if contents.count("B") != 1:
54             raise Exception("maze must have exactly one goal")
55
56         # Determine height and width of maze
57         contents = contents.splitlines()
58         self.height = len(contents)
59         self.width = max(len(line) for line in contents)
```


解迷宫 – 让计算机明白迷宫

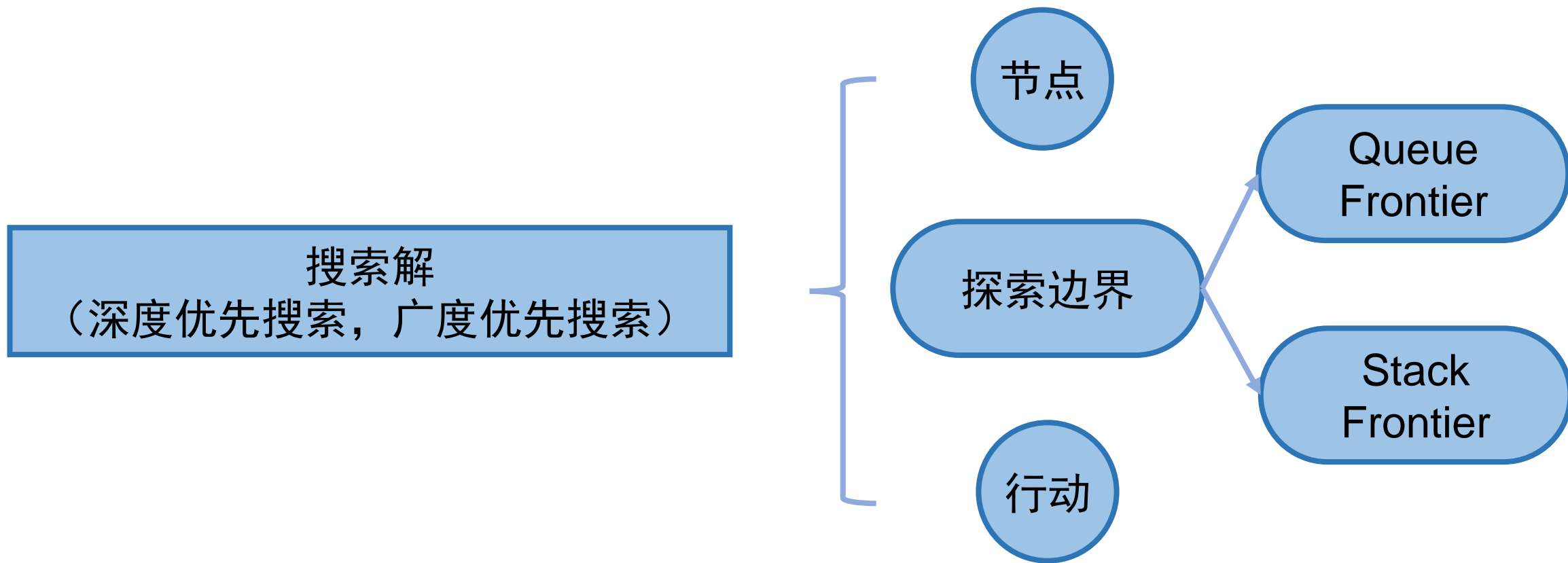
- 初始化迷宫
 - 读取迷宫txt文件
 - 验证是否含有初始状态和目标状态
 - 找到可以执行的行动
 - 确定迷宫的高度和宽度
 - 找到墙壁的位置(由(i, j)表示)
 - 找到起始状态和目标状态的位置
 - 初始化迷宫的解(空的)

```
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
```

```
# Keep track of walls
self.walls = []
for i in range(self.height):
    row = []
    for j in range(self.width):
        try:
            if contents[i][j] == "A":
                self.start = (i, j)
                row.append(False)
            elif contents[i][j] == "B":
                self.goal = (i, j)
                row.append(False)
            elif contents[i][j] == " ":
                row.append(False)
            else:
                row.append(True)
        except IndexError:
            row.append(False)
    self.walls.append(row)

self.solution = None
```

解迷宫 – 搜索解



解迷宫 – 搜索解

- 节点 Node
 - 一个状态 A **state**
 - 父节点 A **parent** (产生当前节点/状态的那个节点/状态)
 - 一个行动 An **action** (从父节点/状态到当前节点/状态所执行的那个动作)
- 练习#4: 定义一个类Node, 包含上述属性

解迷宫 – 搜索解

- 探索边界 Frontier
 - 添加节点
 - 查看节点是否已经在探索边界里
 - 判断探索边界是否是空的
 - 选择并移除一个节点
 - 后进先出 Last-in first-out (stack)
 - 先进先出 First-in first-out (queue)

练习 #5

- 定义一个类StackFrontier
 - 有一个属性frontier
 - 初始时为一个空的列表
 - 有四个方法
 - add
 - contains_state
 - empty
 - remove

练习 #5

- add函数
 - 功能：将一个node添加到frontier中
- contains_state函数
 - 功能：查看某个状态是否已经在frontier中了
 - 输入：一个状态state
 - 输出：True or False

- empty函数
 - 功能：判断frontier是否是空的
 - 输出：True or False
- remove函数
 - 功能：如果frontier是空的，报错“empty frontier”，否则移除并选择frontier中的**最后一个**节点
 - 输出：选中的node

练习 #6

- 定义一个类 QueueFrontier
 - 继承自父类 StackFrontier
 - 更改remove函数
 - 功能：如果frontier是空的，报错“empty frontier”，否则移除并选择frontier中的第一个节点
 - 输出：选中的node

解迷宫 – 搜索解

- actions(state): 找到某个状态可以执行的所有合法行动
 - 行动: Up, down, left, right
 - 合法
 - 不在迷宫外
 - 不会碰到墙

```
103 def neighbors(self, state):
104     row, col = state
105     candidates = [
106         ("up", (row - 1, col)),
107         ("down", (row + 1, col)),
108         ("left", (row, col - 1)),
109         ("right", (row, col + 1))
110     ]
111
112     result = []
113     for action, (r, c) in candidates:
114         if 0 <= r < self.height and 0 <= c < self.width and not self.walls[r][c]:
115             result.append((action, (r, c)))
116     return result
```


解迷宫 – 搜索解

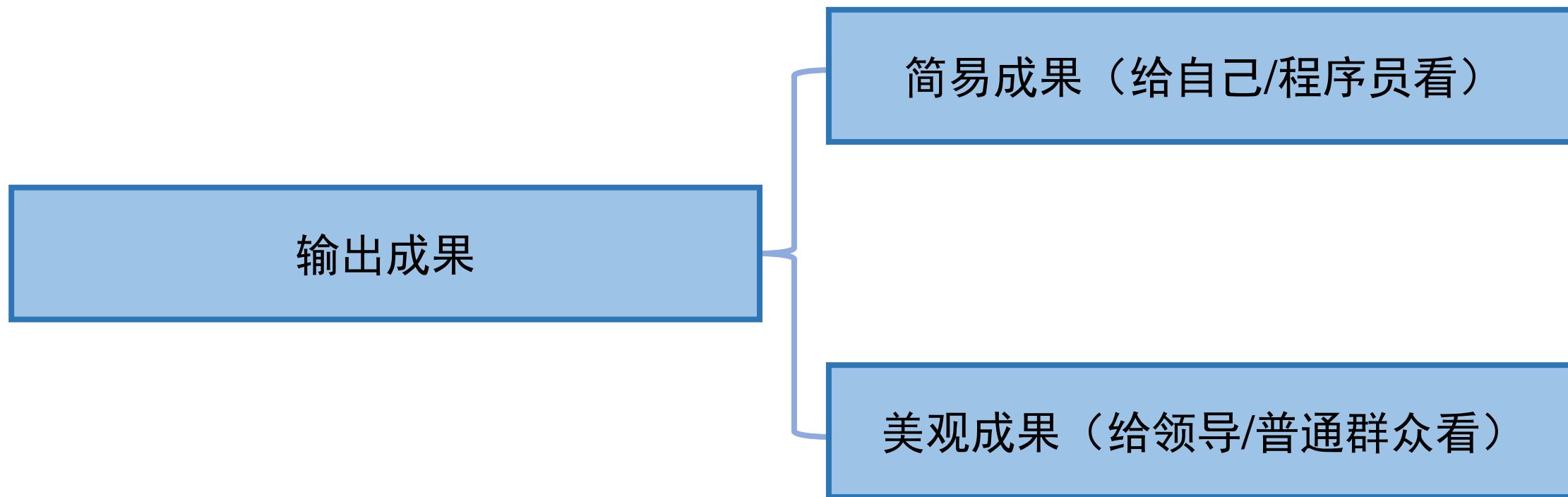
- 从一个包含初始状态(initial state)的探索边界(frontier)开始
- 初始化一个空的**已探索集合(empty explore set)**
- 重复:
 - 如果探索边界(frontier)是空的，那么问题就无解
 - 否则：从探索边界(frontier)中选择并移除一个节点
 - 如果这个节点就是目标状态，那么返回问题的解
 - 否则：
 - **将这个节点加入到已探索集合**
 - 扩展这个节点，如果生成的节点不在已探索集合，那么添加这个节点到探索边界(frontier)中

解迷宫 – 搜索解

```
119 def solve(self):
120     """Finds a solution to maze, if one exists."""
121
122     # Keep track of number of states explored
123     self.num_explored = 0
124
125     # Start with a frontier that contains the initial state
126     start = Node(state=self.start, parent=None, action=None)
127     frontier = StackFrontier()
128     frontier.add(start)
129
130     # Start with an empty explore set
131     self.explored = set()
132
133     # Keep looping until solution found
134     while True:
135
136         # If the frontier is empty, then there is no solution
137         if frontier.empty():
138             raise Exception("no solution")
139
140         # remove (select) a node from the frontier
141         node = frontier.remove()
142         self.num_explored += 1
```

```
134 while True:
135
136     # If the frontier is empty, then there is no solution
137     if frontier.empty():
138         raise Exception("no solution")
139
140     # remove (select) a node from the frontier
141     node = frontier.remove()
142     self.num_explored += 1
143
144     # If the node contains goal state, return the solution
145     if node.state == self.goal:
146         actions = []
147         cells = []
148         while node.parent is not None:
149             actions.append(node.action)
150             cells.append(node.state)
151             node = node.parent
152         actions.reverse()
153         cells.reverse()
154         self.solution = (actions, cells)
155         return
156
157     # Else:
158     # Add the node to the explore set
159     self.explored.add(node.state)
160
161     # Expand node, add resulting nodes to the frontier if they aren't already in the frontier or the explored set
162     for action, state in self.neighbors(node.state):
163         if not frontier.contains_state(state) and state not in self.explored:
164             child = Node(state=state, parent=node, action=action)
165             frontier.add(child)
```

解迷宫 – 输出成果



解迷宫 – 输出成果

- 简易成果（在终端中输出成果）

- ■ 代表墙壁
- 空格表示可以同行的路径
- * 表示找到的解
- A 代表起始状态
- B 代表目标状态

```
■■■■■B
■■■■■*
■■■■■**
■■■■■*
*****
A■■■■■
```

解迷宫 – 输出成果

- 简易成果（在终端中输出成果）

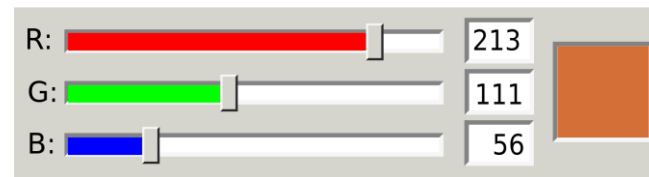
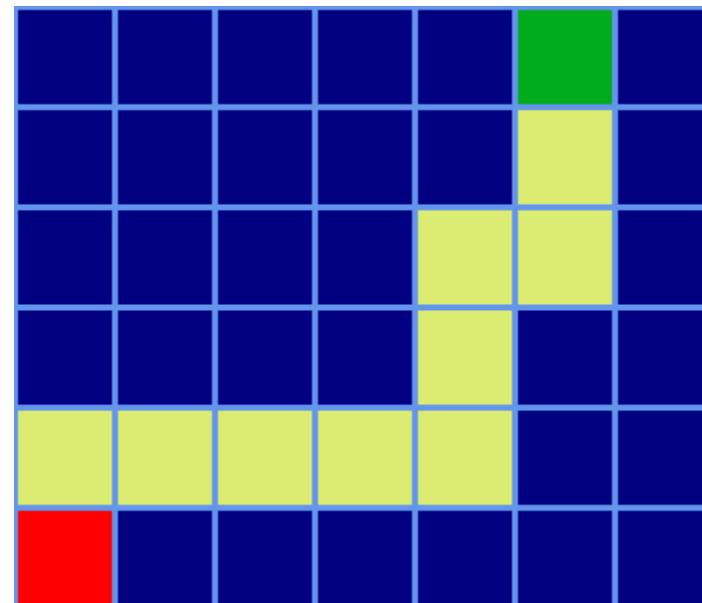
- ■ 代表墙壁
- 空格表示可以同行的路径
- * 表示找到的解
- A 代表起始状态
- B 代表目标状态

```
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

def print(self):
    solution = self.solution[1] if self.solution is not None else None
    print()
    for i, row in enumerate(self.walls):
        for j, col in enumerate(row):
            if col:
                print("■", end="")
            elif (i, j) == self.start:
                print("A", end="")
            elif (i, j) == self.goal:
                print("B", end="")
            elif solution is not None and (i, j) in solution:
                print("*", end="")
            else:
                print(" ", end="")
        print()
    print()
```

解迷宫 – 输出成果

- 美观成果（给领导/普通群众看）
 - 墙壁：(0, 0, 128)深蓝色
 - 可通行的路径：(237, 240, 252)白色
 - 找到的解：(220, 235, 113)黄色
 - 起始状态：(255, 0, 0)红色
 - 目标状态：(0, 171, 28)绿色
 - 已探索过：(212, 97, 85)棕色



解迷宫 - 输出成果

```
168 def output_image(self, filename, show_solution=True, show_explored=False):
169     from PIL import Image, ImageDraw
170     cell_size = 50
171     cell_border = 2
172
173     # Create a blank canvas
174     img = Image.new(
175         "RGBA",
176         (self.width * cell_size, self.height * cell_size),
177         (100,149,237) # light blue
178     )
179     draw = ImageDraw.Draw(img)
180
181     solution = self.solution[1] if self.solution is not None else None
182     for i, row in enumerate(self.walls):
183         for j, col in enumerate(row):
184
185             # Walls
186             if col:
187                 # dark blue
188                 fill = (0, 0, 128)
189
190             # Start
191             elif (i, j) == self.start:
192                 # red
193                 fill = (255, 0, 0)
194
195             # Goal
196             elif (i, j) == self.goal:
197                 # green
198                 fill = (0, 171, 28)
```

```
199     # Solution
200     elif solution is not None and show_solution and (i, j) in solution:
201         # yellow
202         fill = (220, 235, 113)
203
204     # Explored
205     elif solution is not None and show_explored and (i, j) in self.explored:
206         # brown
207         fill = (212, 97, 85)
208
209     # Empty cell
210     else:
211         # white
212         fill = (237, 240, 252)
213
214     # Draw cell
215     draw.rectangle(
216         ((j * cell_size + cell_border, i * cell_size + cell_border),
217          ((j + 1) * cell_size - cell_border, (i + 1) * cell_size - cell_border))),
218         fill=fill
219     )
220
221     img.save(filename)
```

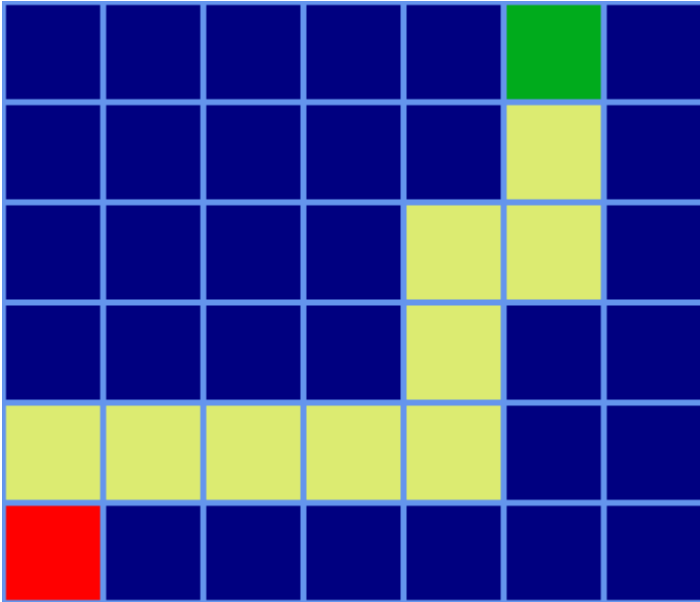
截图(Alt + A)

main()

- 确认终端输入的参数格式是正确的
- 读取迷宫
- 解迷宫
- 输出结果

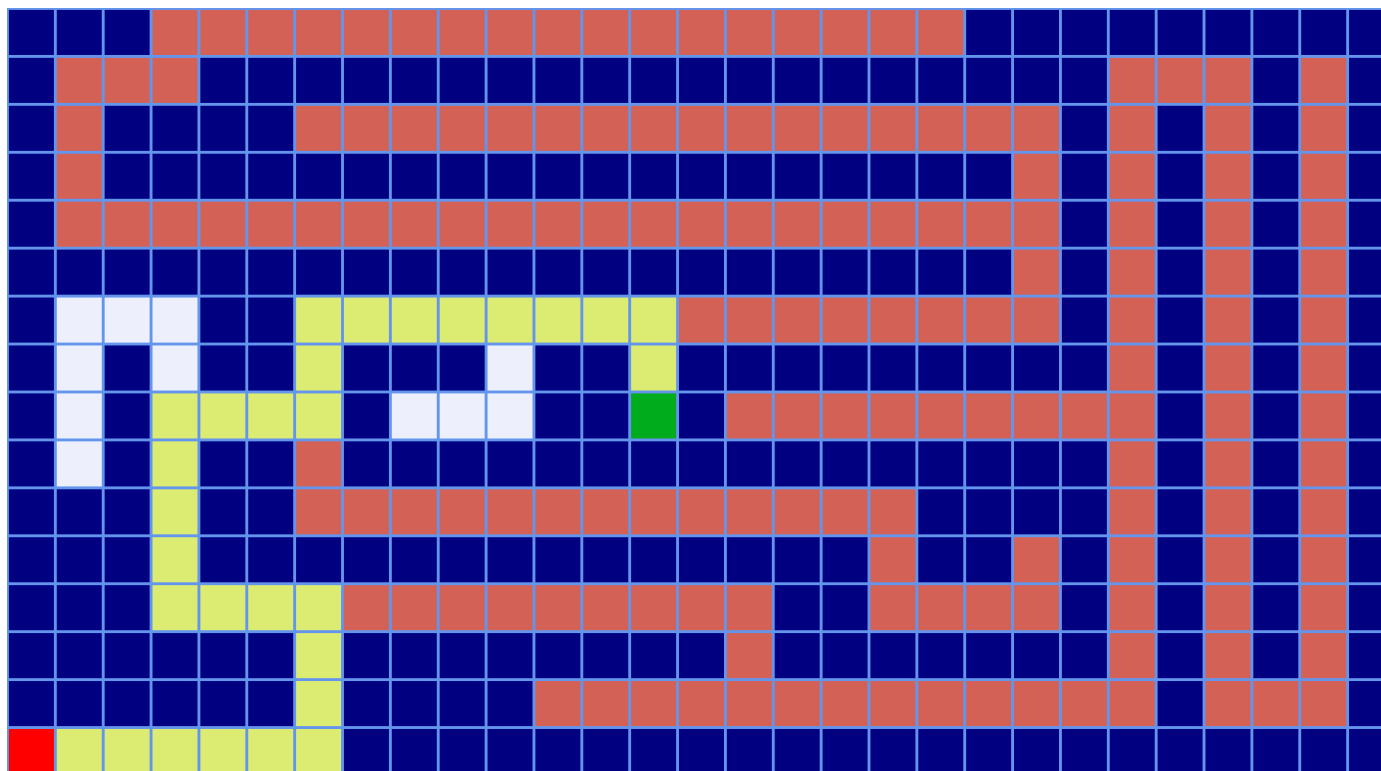
```
225 ▶ if __name__=="__main__":  
226     # make sure the terminal argument is in the correct format  
227     if len(sys.argv) != 2:  
228         sys.exit("Usage: python maze.py maze.txt")  
229  
230     # Initialize the maze  
231     m = Maze(sys.argv[1])  
232     print("Maze:")  
233     m.print()  
234  
235     # Solve the result  
236     print("Solving...")  
237     m.solve()  
238  
239     # Output the result in the terminal  
240     print("States Explored:", m.num_explored)  
241     print("Solution:")  
242     m.print()  
243  
244     # Output the result to general audience  
245     m.output_image("maze.png", show_explored=True)
```


迷宫 #1



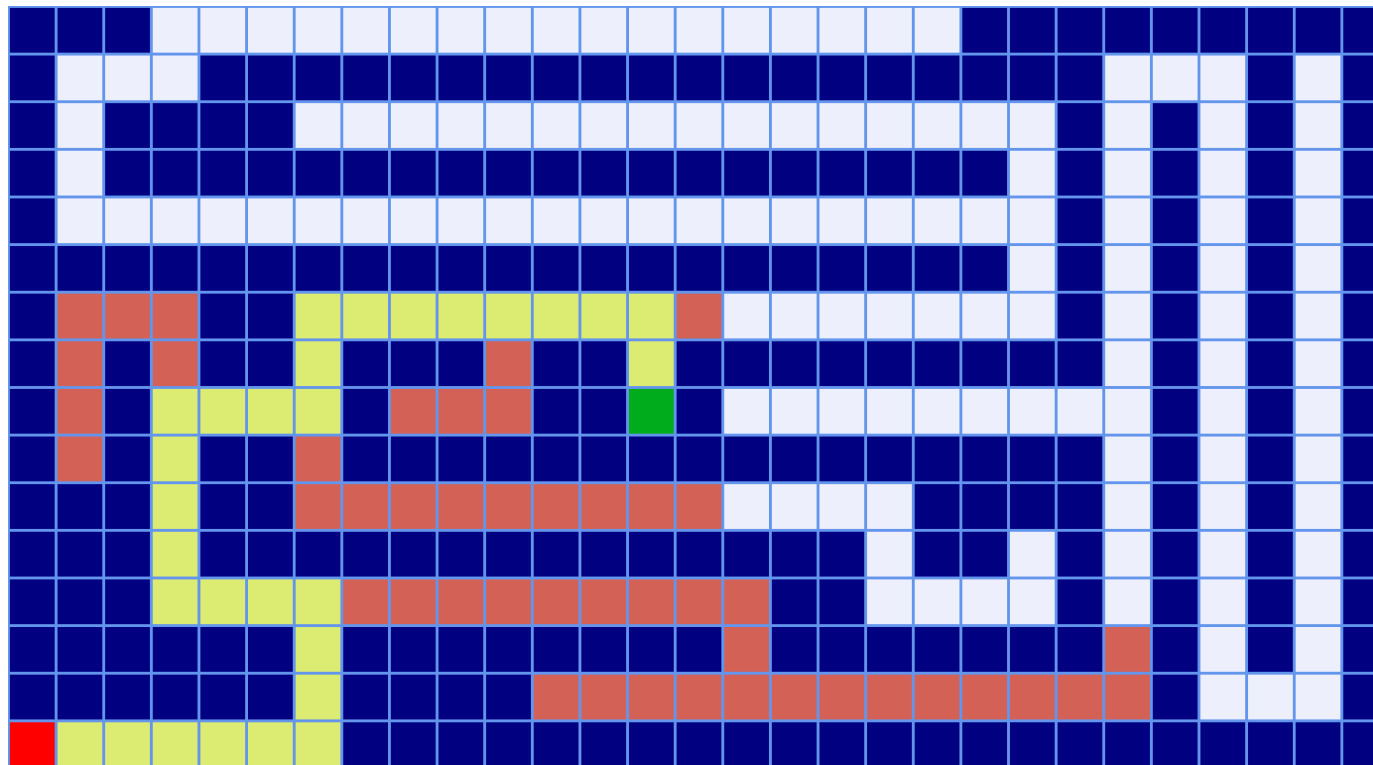
迷宫 #2

- 深度优先搜索有可能在找到解之前探索所有可能的路径



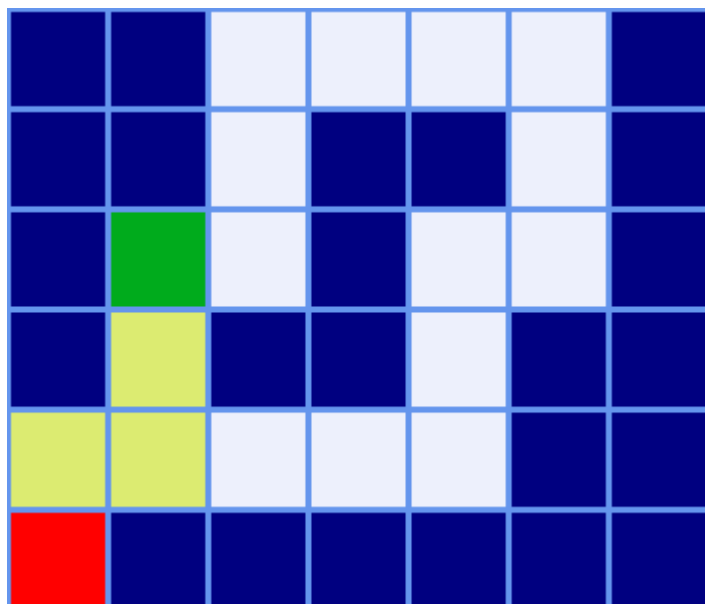
迷宫 #2

- 如果目标状态靠近初始状态，则广度优先搜索更有效

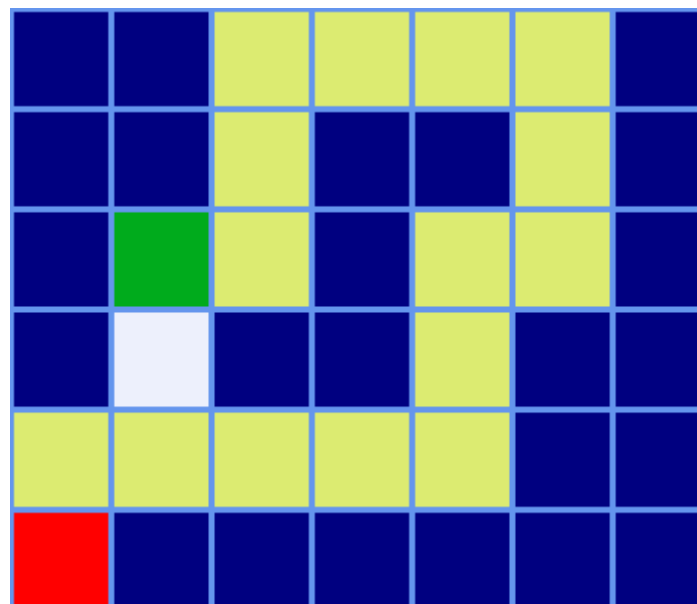


迷宫 #3

广度优先搜索 BFS



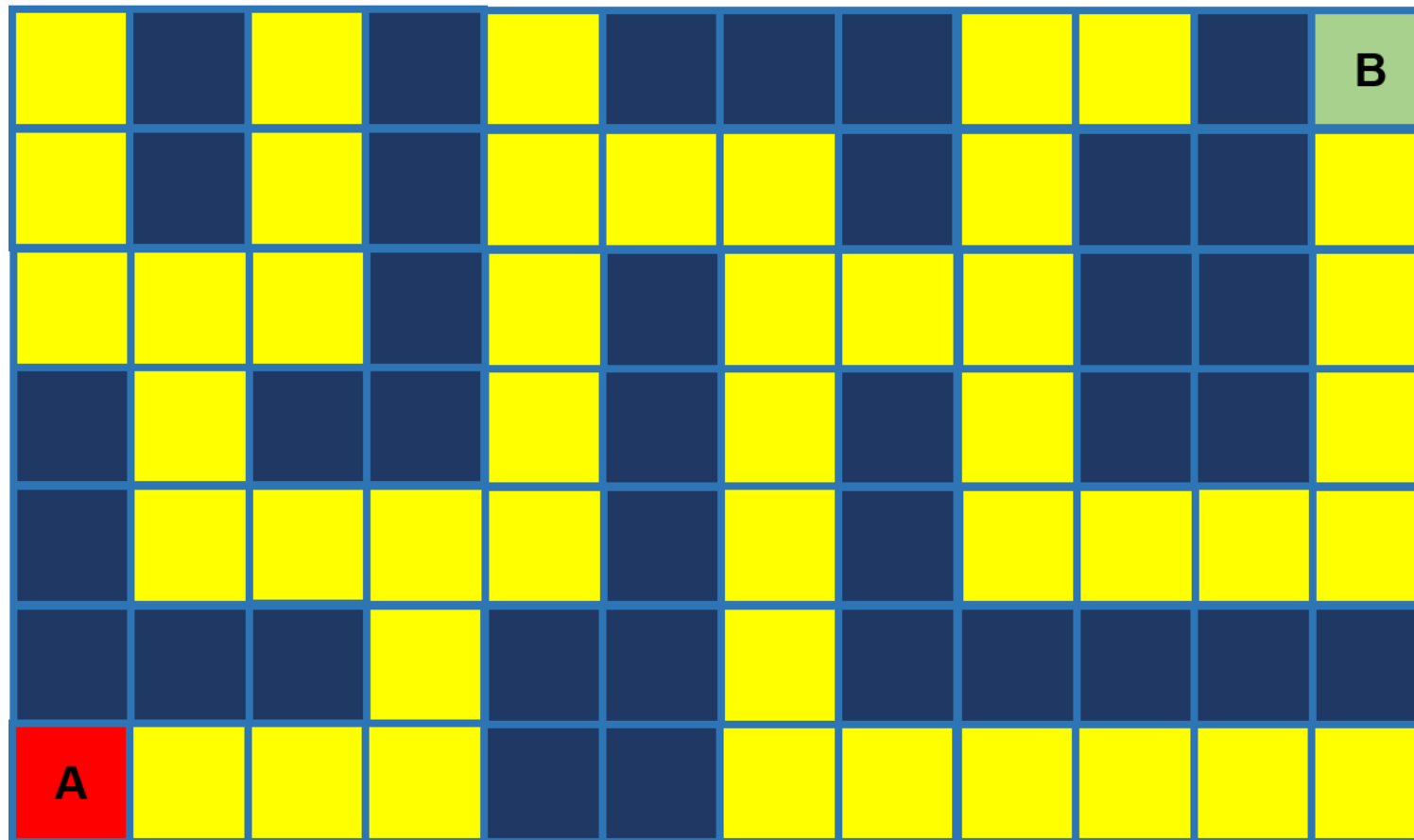
深度优先搜索 DFS



深度优先搜索可能并不总是找到最佳解决方案

迷宫

- 如果目标状态与初始状态相距甚远，
则广度优先搜索可能需要探索很多步骤



有问题吗？

- 请随时举手提问。



BUSS 3620.人工智能导论

#4. 提示性搜索

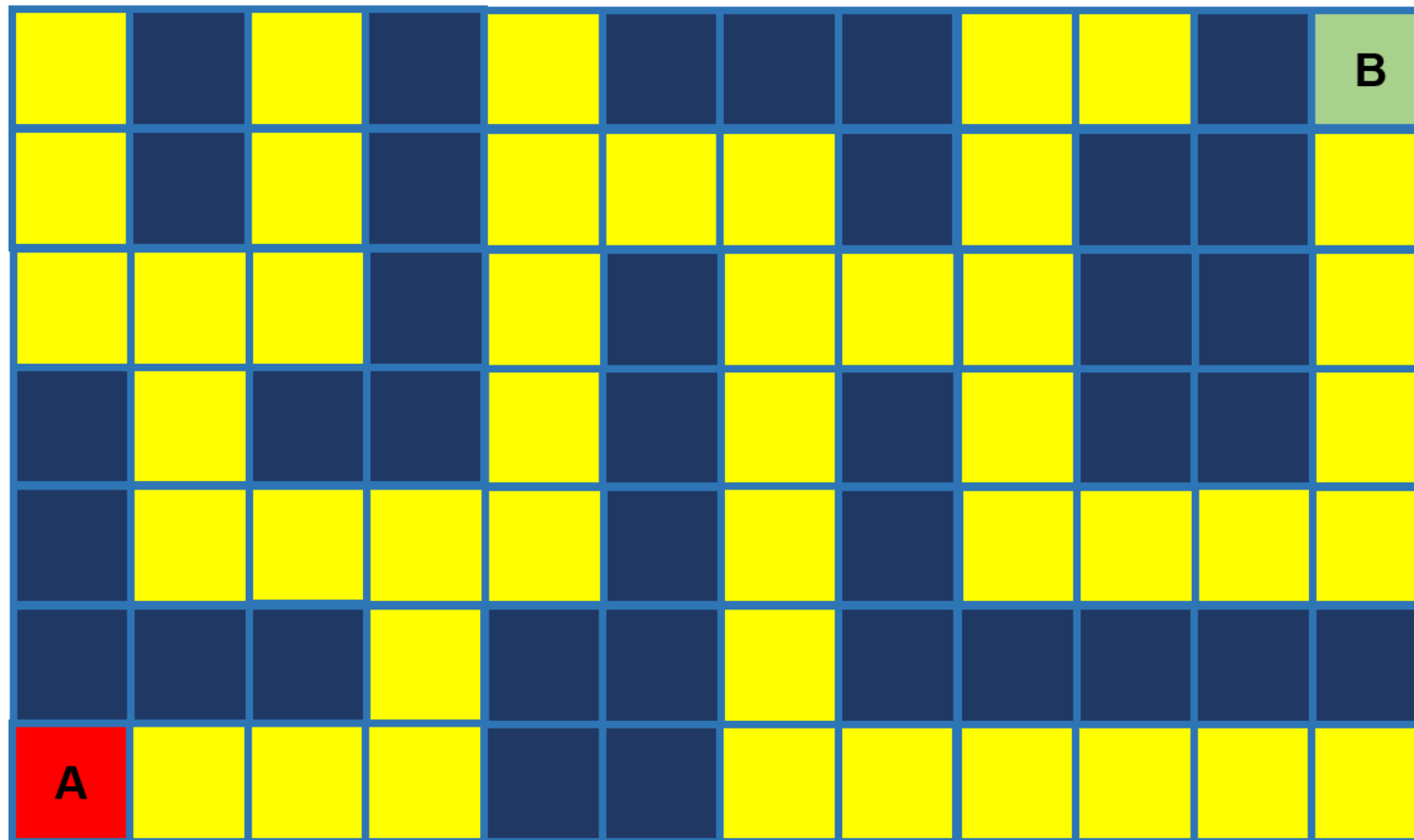
刘佳璐

安泰经济与管理学院

上海交通大学

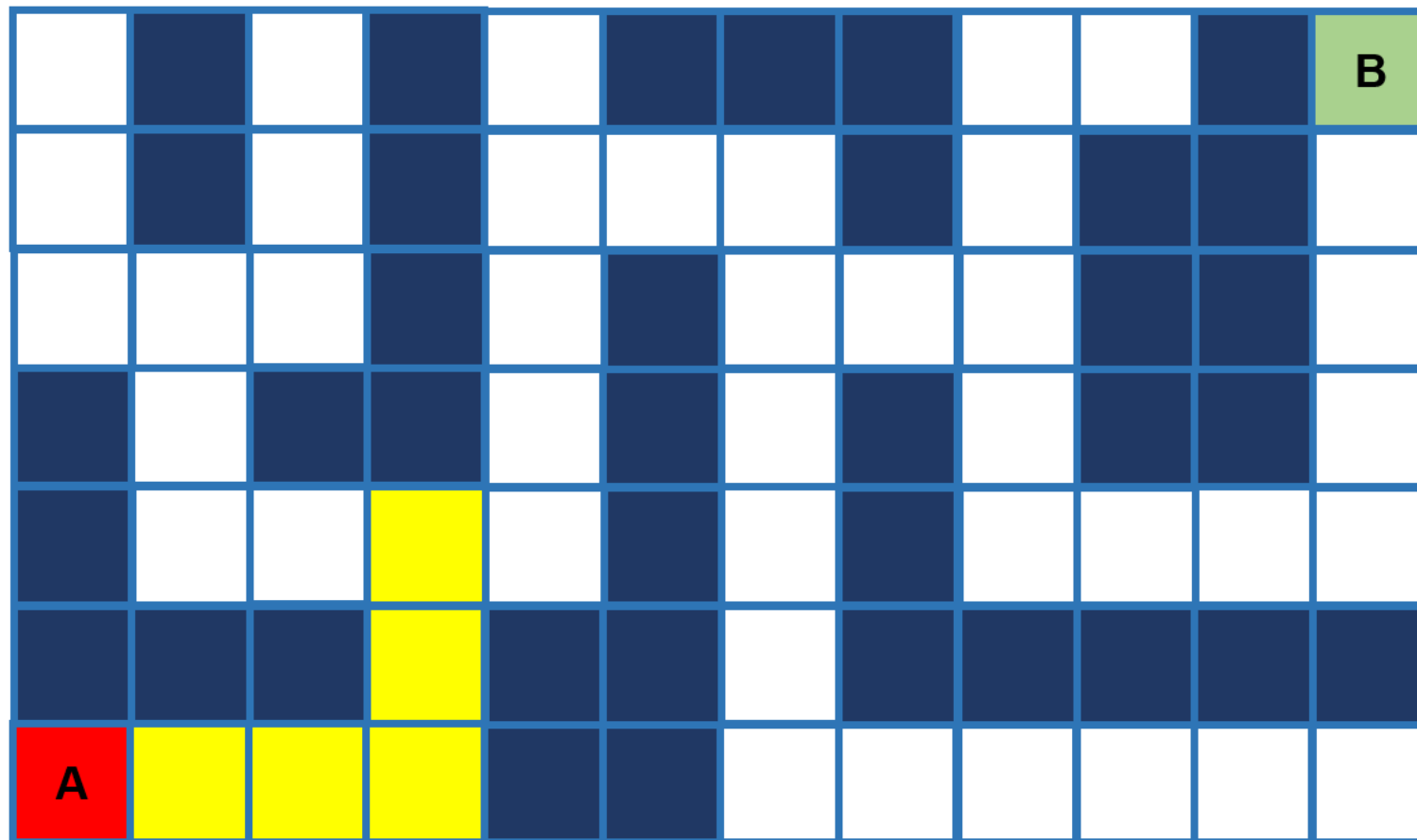
迷宫

- 如果目标状态与初始状态相距甚远，则广度优先搜索可能需要探索很多步骤



你会怎么做？

- 可能向右
 - 假设:
 - 知道坐标
 - 知道距离



搜索算法

- 无信息搜索 Uninformed search
 - **不使用与问题相关的信息**来指导搜索的策略
 - 例如: Depth-first search, Breadth-first search
 - 迷宫的结构不影响搜索策略
- 提示性搜索 Informed search
 - **使用与问题相关的信息**来指导搜索的策略,可以让搜索更高效
 - 例如: 迷宫:选择地理位置**更接近**目标的路线

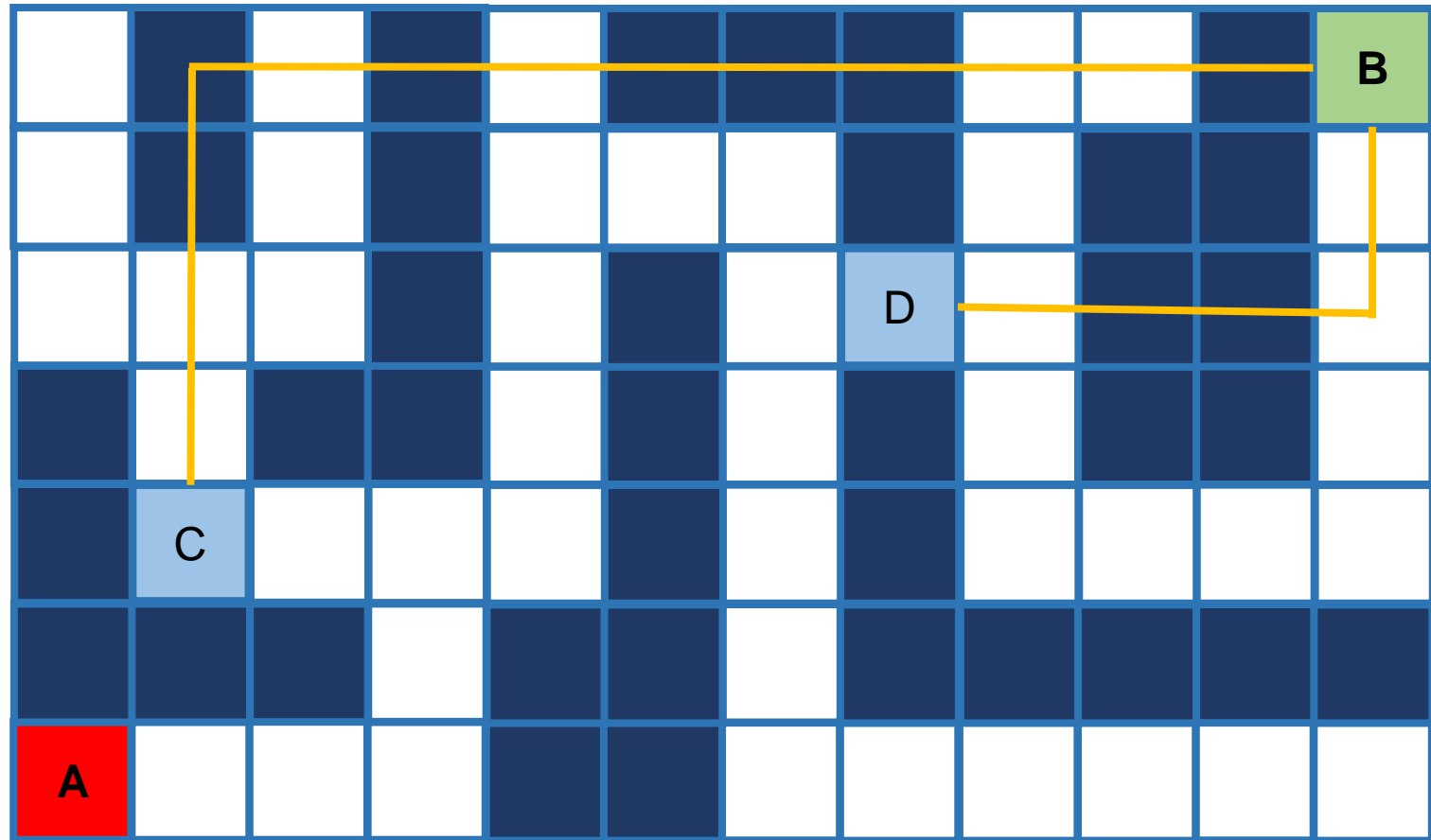
启发式搜索 Informed search

- 贪婪最佳优先搜索 Greedy best-first search
 - 扩展**最接近目标**的节点的搜索算法
 - 由启发式函数**heuristic function $h(n)$** 来估计距离终点的接近程度

启发式函数 Heuristic function

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$



贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 曼哈顿距离 Manhattan distance

- $|x_1 - x_2| + |y_1 - y_2|$

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索 Greedy best-first search

- 启发式估计可能是**不准确的**

- 参考距离，而非真实

- 探索**更少**的状态

- 重点是**好的**启发式估计

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

贪婪最佳优先搜索能够找到最优解吗？

- 练习 #7

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

贪婪最佳优先搜索能够找到最优解吗？

- 最优：20步

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

如何改进GBFS?

- 最优: 20步
- GBFS
 - 33步
 - 局部最优

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

如何改进GBFS?

- A* 搜索
 - 扩展 $g(n) + h(n)$ 值最小的节点的搜索算法
 - $g(n)$ = 起始节点到当前节点已经产生的代价/成本
 - $h(n)$ = 当前节点到目标节点的预估代价/成本

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	1+16 =17	15	14		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	1+16 =17	2+15 =17	14		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	9	8	7	6	5	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	8	7	6	5	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	7	6	5	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	6	5	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	5	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	4		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	13		6+11 =17						5		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	13		6+11 =17						14+5 =19		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	13		6+11 =17						14+5 =19		3
	14	6+13 =19	5+12 =17		10	9	8	7	6		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	13		6+11 =17						14+5 =19		3
	14	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	13		6+11 =17						14+5 =19		3
	14	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	13		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	13		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	10	9	8	7	6	5	4	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	9	8	7	6	5	4	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	8	7	6	5	4	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	7	6	5	4	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	6	5	4	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	15+6 =21	5	4	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	15+6 =21	16+5 =21	4	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	15+6 =21	16+5 =21	17+4 =21	3	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	15+6 =21	16+5 =21	17+4 =21	18+3 =21	2	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	15+6 =21	16+5 =21	17+4 =21	18+3 =21	19+2 =21	1	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	15+6 =21	16+5 =21	17+4 =21	18+3 =21	19+2 =21	20+1 =21	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- $g(n) + h(n)$
- 能找到**最优解**

	11+10 =21	12+9 =21	13+8 =21	14+7 =21	15+6 =21	16+5 =21	17+4 =21	18+3 =21	19+2 =21	20+1 =21	B
	10+11 =21										1
	9+12 =21		7+10 =17	8+9 =17	9+8 =17	10+7 =17	11+6 =17	12+5 =17	13+4 =17		2
	8+13 =21		6+11 =17						14+5 =19		3
	7+14 =21	6+13 =19	5+12 =17		10	9	8	7	15+6 =21		4
			4+13 =17		11						5
A	1+16 =17	2+15 =17	3+14 =17		12	11	10	9	8	7	6

A* 搜索

- 满足以下两个条件时，A*搜索的解是最优的：

① 启发式函数 $h(n)$ 是可接受的(admissible)

- 从不高估到目标代价。预估代价要么是准确的，要么比实际代价更小。

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

A* 搜索

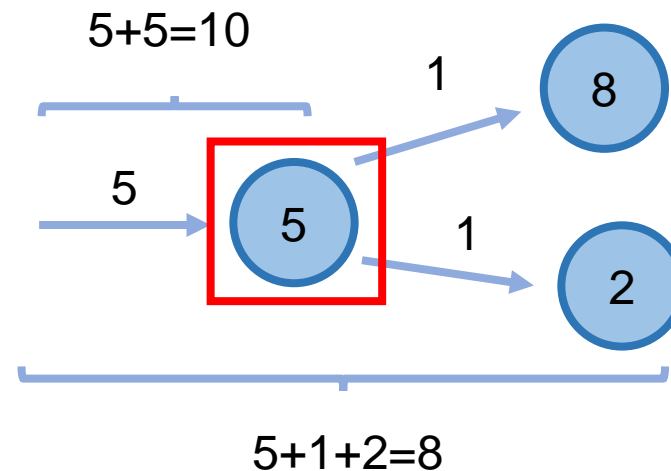
- 满足以下两个条件时，A*搜索的解是最优的：

- ① 启发式函数 $h(n)$ 是可接受的(admissible)

- 从不高估到目标代价。预估代价要么是准确的，要么比实际代价更小。

- ② 启发式函数 $h(n)$ 是一致的(consistent)

- 对于任意节点 n ，以及需要步骤成本(step cost)为 c 的后续节点 n' ， $h(n) \leq h(n') + c$

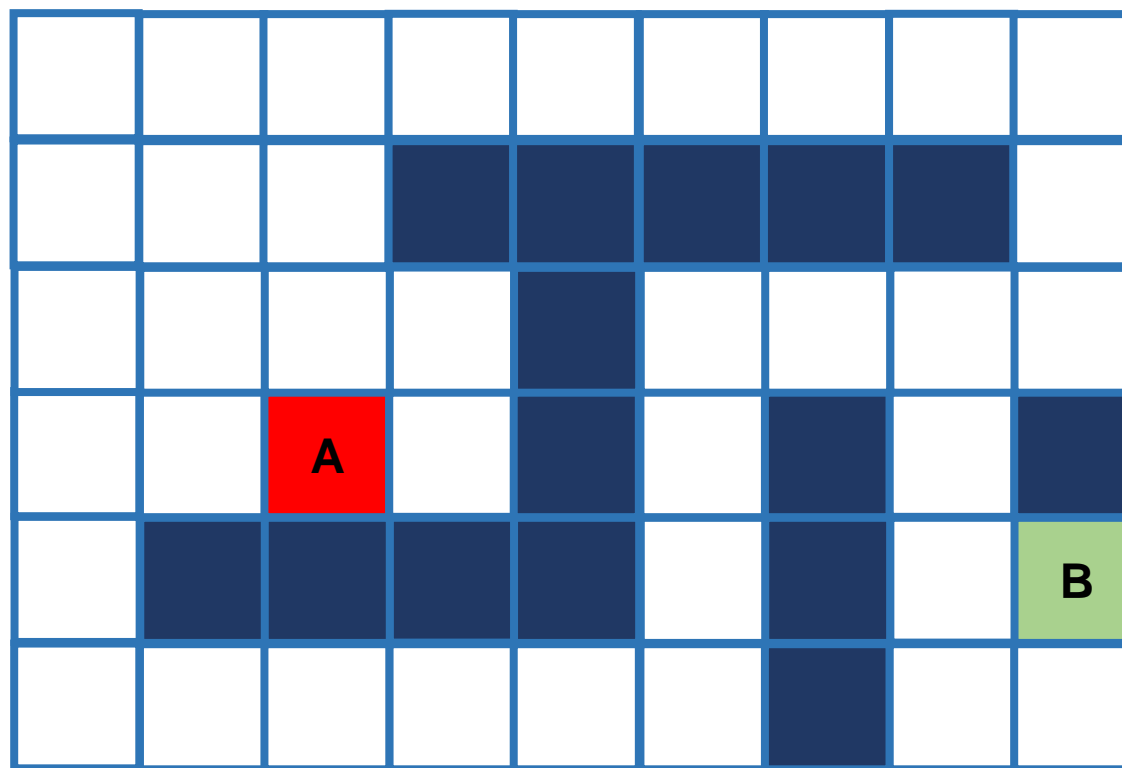


A* 搜索

- 满足以下两个条件时，A*搜索的解是最优的：
 - ① 启发式函数 $h(n)$ 是可接受的(admissible)
 - 从不高估到目标代价。预估代价要么是准确的，要么比实际代价更小。
 - ② 启发式函数 $h(n)$ 是一致的(consistent)
 - 对于任意节点 n ，以及需要步骤成本(step cost)为 c 的后续节点 n' ， $h(n) \leq h(n') + c$
- 最难的是找到一个好的启发式函数

练习 #8

- 使用A*搜索找到从A到B的路径
 - 多个选择
 - 按照右，下，左，上的顺序



A* 搜索的现实应用

- 机器人/自动驾驶
 - 为机器人、汽车找到在环境中导航的最佳路径，避开障碍物/行人到达目的地。
- 游戏
 - 帮助非玩家角色（NPC）在游戏世界中进行操作，实时战略游戏的也帮助寻路。
- 自然语言处理
 - 在给定单词的情况下，搜索最有可能的成为通顺句子的单词序列。
- 物流
 - 为送货卡车找到最佳路线。
- 图像和视频处理
 - 通过将当前帧与上一帧进行比较来检测和跟踪视频流中的对象。

有问题吗?

- 请随时举手提问。

