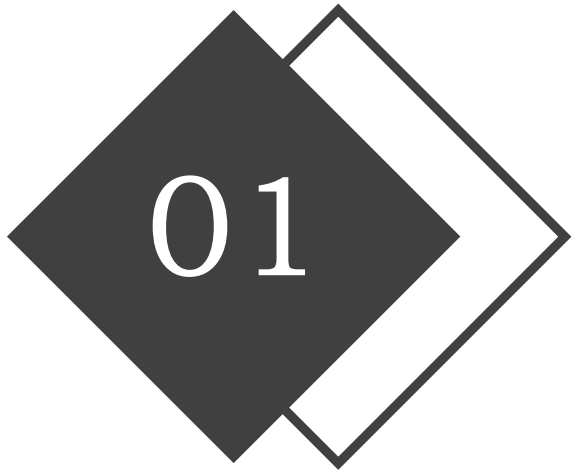
盲目搜索

目录

- 理论课回顾
  - 形式化一个搜索问题
    - 问题的定义
    - 问题的解的定义
    - 为什么要搜索算法
  - 问题求解算法的性能
  - 盲目搜索策略
    - 宽度优先搜索(BFS)
    - 深度优先搜索(DFS)
    - 深度受限搜索
    - 迭代加深搜索



搜索问题定义

形式化一个搜索问题

1.1 问题的定义 [1]

用5个组件形式化定义一个search problem：

1）initial state 初始状态

2）Actions 行动

3）transition model 转移模型

4）goal test 目标测试

5）path cost 路径耗散

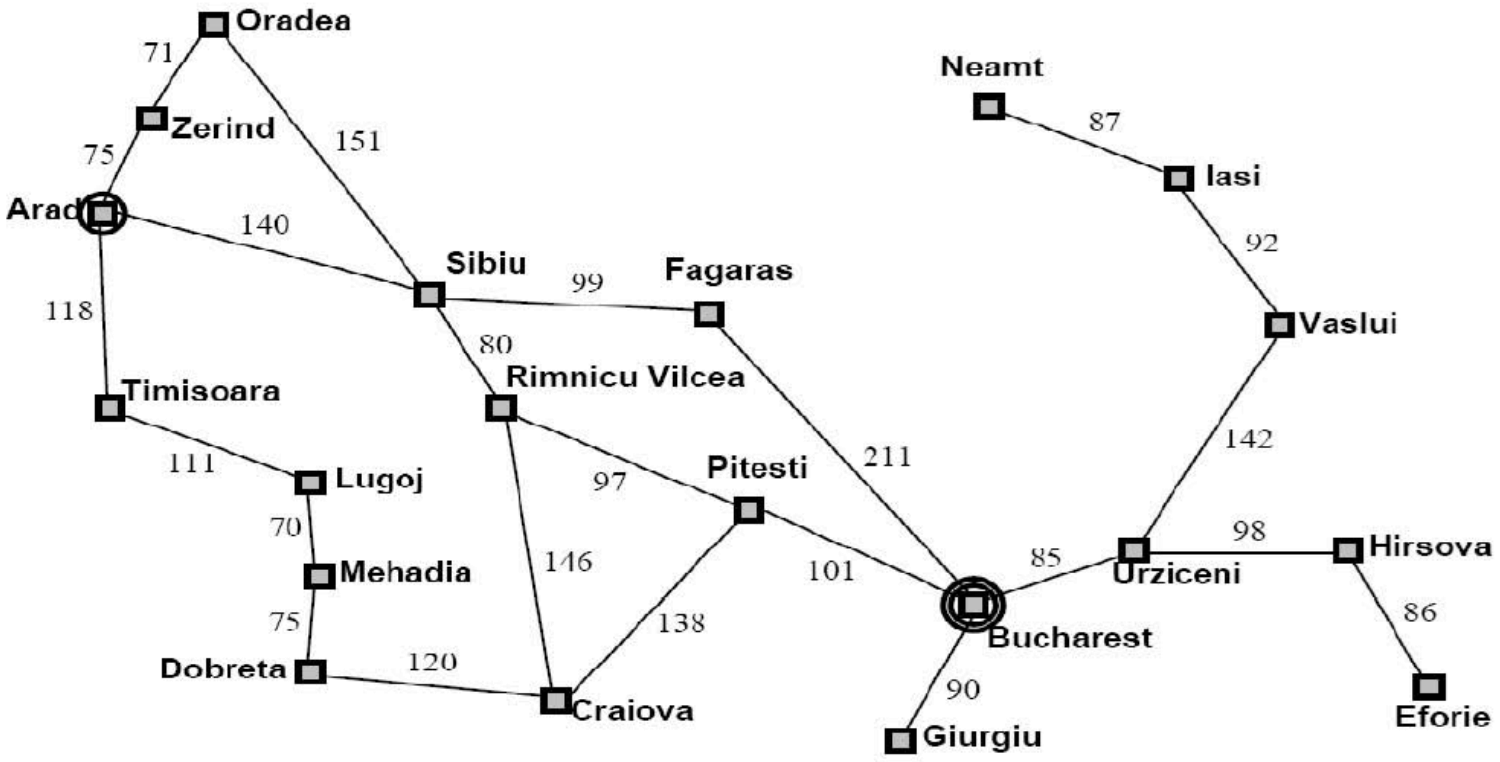
1.2 问题的解（solution）的定义

A solution to a problem is an action sequence that leads from the initial state to a goal state.

即：一个问题的解是：一个**行动**序列

举例

Currently in Arad, need to get to Bucharest

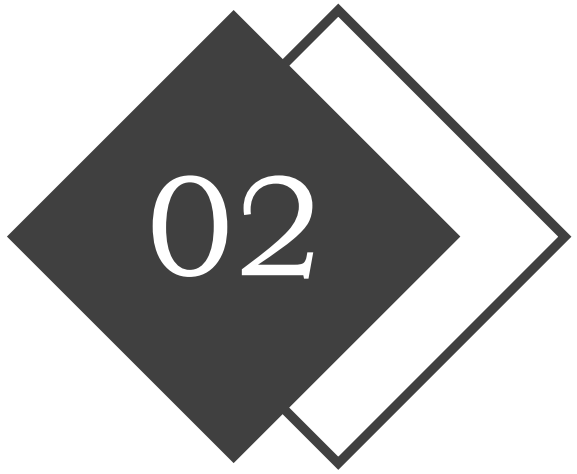


- **States:** the various cities you could be located in.
- **Actions:** drive between neighboring cities.
- **Initial state:** in Arad
- **Goal:** in Bucharest
- **Solution:** the route, the sequence of cities to travel through to get to Bucharest.

求解算法的性能

四个方面：

- 完备性：当问题有解时，这个算法能否保证找到解。
- 最优性：搜索策略能否找到最优解。
- 时间复杂度：找到解所需要的时间，也叫搜索代价
- 空间复杂度：执行搜索过程中需要多少内存空间

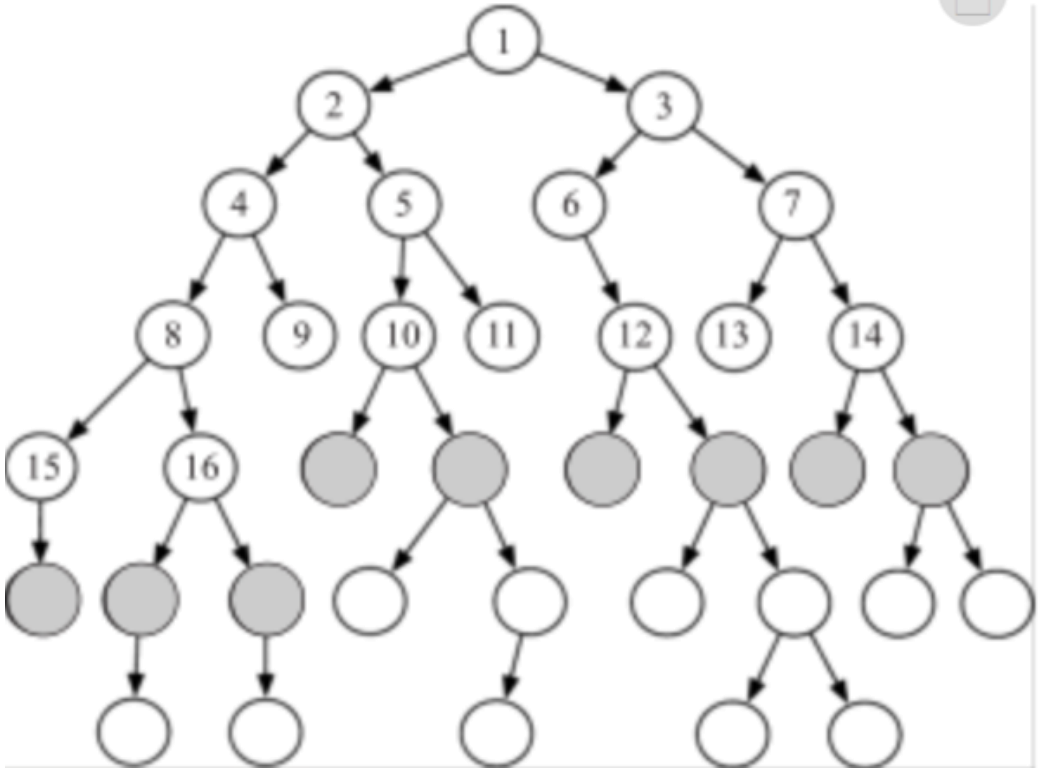


盲目搜索策略

盲 目 搜 索 策 略

- 宽度优先搜索
- 深度优先搜索
- 深度受限搜索
- 迭代加深搜索

宽度优先搜索(BFS)



算法 宽度优先搜索 BFS

输入: 初始状态 s_0, 动作集合, 转移模型 T, 目标检测函数

```
初始化队列 queue ← [s_0]
while queue 非空 do
    s ← queue.pop()
    for s 的所有可行动作 a do
        获取下一状态 s' = T(s, a)
        if s' 是目标 then
            return 搜索路径
        else
            queue.append(s')
        end if
    end for
end while
return 问题无解
```

- 节点扩展顺序与目标节点的位置无关;
- 用一个先进先出（FIFO）队列实现;

深度优先搜索 (DFS)

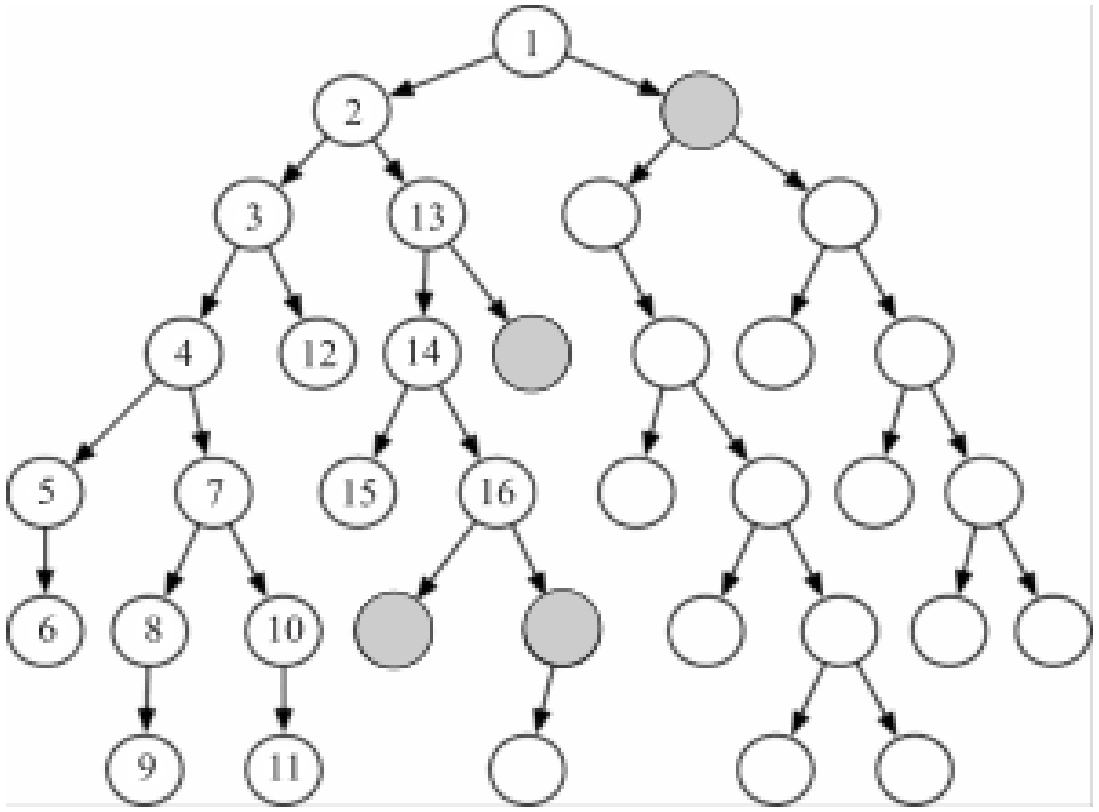


图 3-5　深度优先搜索中节点的扩展顺序

算法 深度优先搜索 DFS

输入: 初始状态 s_0, 动作集合, 转移模型 T, 目标检测函数

```
if s0 下无可执行动作 then
    return 当前无解
else if s0 是目标 then
    return 搜索路径
else
    for all s0 的可行动作 a do
        获取下一状态 s' = T(s0, a)
        若 s' 未访问, 则以 s' 为初始状态递归执行 DFS
    end for
end if
```

有界深度/深度受限搜索 (Depth-limited Search)

算法 有界深度搜索 DLS

输入: 初始状态 s_0, 动作集合, 转移模型 T, 目标检测函数, 深度限制 d

```
1: if d = 0 then
2:     return 当前无解
3: else if s0 是目标 then
4:     return 搜索路径
5: else
6:     for all s0 的可行动作 a do
7:         获取下一状态 s' = T(s0, a)
8:         若 s' 未访问, 则以 s' 为初始状态, d − 1 为深度限制, 递归执行 DLS
9:     end for
10: end if
```

迭代加深搜索（Iterative Deepening Search ）

算法 迭代加深搜索 IDS

输入: 初始状态 s_0, 动作集合, 转移模型 T, 目标检测函数

```
for d = 1 to +∞ do
    以 s0 为初始状态, d 为深度限制执行 DLS
    if 本次搜索找到解 then
        return 搜索路径
    else if 本次搜索所有节点深度小于 d then
        return 搜索失败
    end if
end for
```