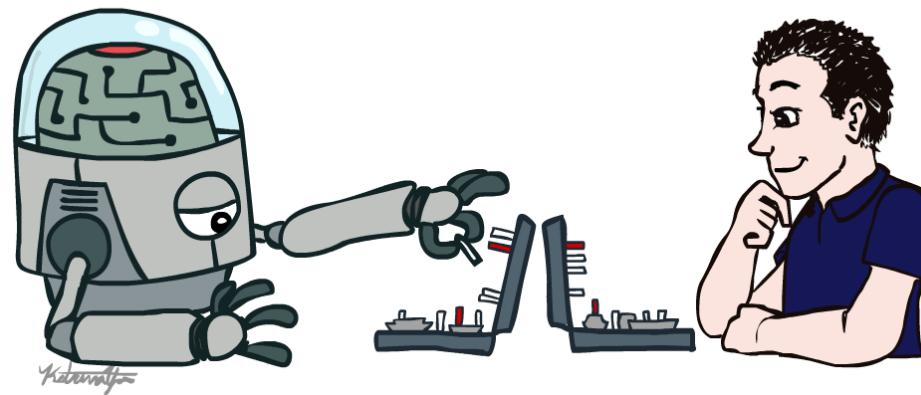


人工智能导论

简介，继续

讲师：齐琦



人工智能历史发展的启示

- 想人一样的思考、行动 → 合理的思考、行动
- 飞行的启示
 - 像鸟一样的飞行 → 莱特兄弟发明飞机
- 模仿 → 创造



人工智能的未来

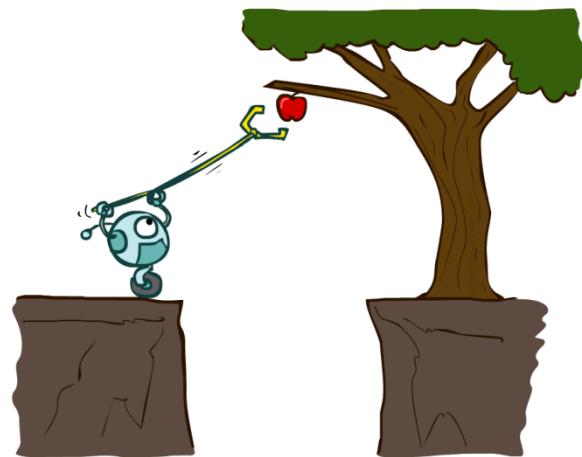
- 人工智能的目的
 - 创造智能系统，越智能越好
 - 同时去更好的理解人的智能
 - 扩大智能带给我们的益处
- 如果我们真能造出超级智能机器，那么
 - 可能帮助人类避免战争，生态灾难，成就不朽，和在宇宙中扩展空间
 - 这也将会是人类历史上最大的事件
 - 『也有可能是最最后一次』

人工智能的未来—超级人工智能

- 超越人的人工智能有可能实现
 - 量子计算机可大幅提高计算能力
 - 但并不是一个直接的目标
- 既有益处，也有忧虑
 - 所有人类文明都来自于智能
 - 有可能超越人类理解和控制（电影“Transcendence”提示的）
- 超级人工智能在当今难以实现
 - 计算能力的提升和让一个机器有意识之间没有关系
 - 研究人大脑机制还处在初级阶段
- 引发人工智能科学家预先思考
 - 类似思考存在于其他领域，例如核物理，基因工程。
 - 人工智能发展的道路？
 - 最好的情况？最坏的情况？
 - 我们可能影响人工智能的未来？
 - 发挥有利的一面，避免不利的一面。
 - 是我们这一时代的任务。

人工智能： 智能体和环境

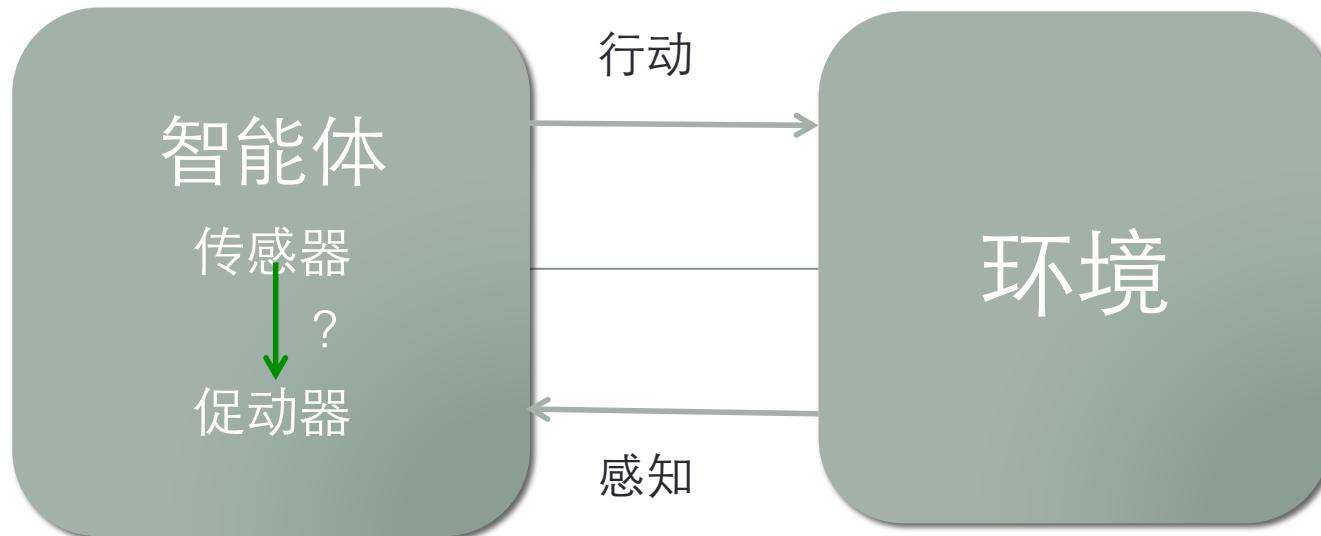
讲师：齐琦



提纲

- 智能体和环境
- 合理性
- PEAS (性能指标, 环境, 促动器, 传感器)
- 环境类型
- 智能体类型

智能体和环境



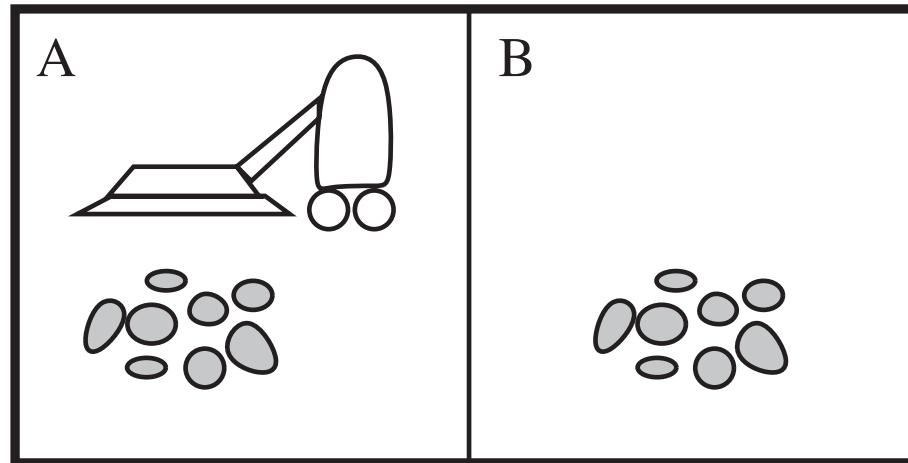
智能体和环境

- 一个智能体通过自身的传感器感知环境，并通过自身的促动器去相应的行动。
- 人是智能体
 - 传感器 = 视觉，听觉，触摸，嗅觉，味觉，主体感觉
 - 促动器（激励器） = 肌肉，分泌物质，大脑状态的改变
- 口袋计算器
 - 传感器 = 按键状态传感器
 - 促动器 = 数字显示
- 人工智能主要研究对象
 - 有自己的计算资源和能力的智能体
 - 和所处环境要求相对复杂的决策过程

智能体函数和智能体程序

- 智能体函数：感知历史（序列）到行动的映射
 - $f: \mathcal{P}^* \rightarrow \mathcal{A}$
- 智能体程序 \mathcal{I} ，运行在某个计算机 M ，以实现 f ：
 - $f = \text{Agent}(\mathcal{I}, M)$
 - 输入是当前的感知，忽略感知历史
 - 运行在智能体内
 - M ，有限的运算和存储空间
 - 程序运行可能比较慢；也可能被新的感知所打断（或忽略它们）。
 - 不是每一个智能体函数都能被实现
 - M 的限制

举例：吸尘器

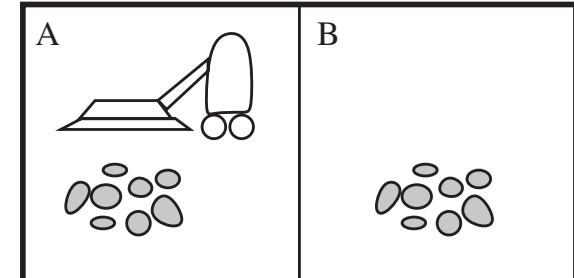


- 感知对象：[地点，洁净状态] ([A, 有灰尘])
- 行动：左，右，吸尘，无动作

吸尘器智能体

智能体函数

感知序列	行动
【A, 干净】	向右移动
【A, 灰尘】	吸尘
【B, 干净】	向左移动
【B, 灰尘】	吸尘
【A, 干净】 , 【B, 干净】	向左移动
【A, 干净】 , 【B, 灰尘】	吸尘
...	...



智能体程序

Function 简单吸尘器([位置,状态]) **returns** 一个行动
If 状态 = 灰尘 **then return** 吸尘
else if 位置 = A **then return** 向右移动
else if 位置 = B **then return** 向左移动

合理性 (Rationality)

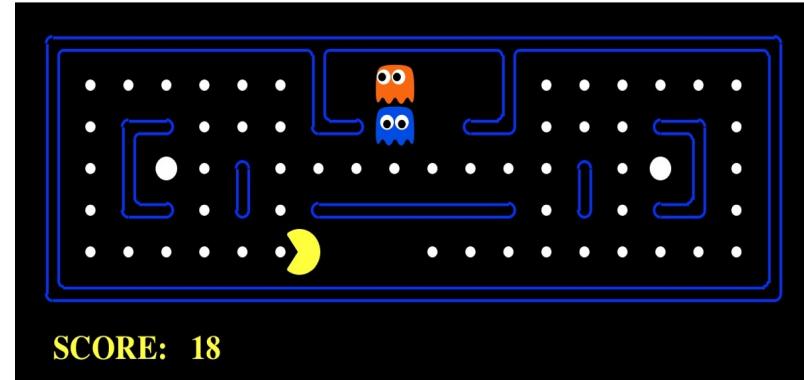
- 固定的性能指标 (performance measure) 衡量环境状况
 - 评估智能体的行为所导致环境变化后的状态
 - 比如，每小时每一平米干净的地面加一分
 - 关注的是环境状态，而不是智能体的态度、状态
 - 同样的性能指标，可能被不同行为的智能体所成就；所带来的思考
- 合理性决定于4个因素
 - 性能指标，智能体的预先知识，智能体能够采取的行动集合，和智能体的感知序列
- 一个合理的智能体 (rational agent)
 - 给定最新的感知序列，和关于环境的预先知识，选择为了最大化性能指标期望值的行动。
 - 问题：之前定义的吸尘器智能体实现的是一个合理的智能体函数吗？

合理性，继续

- 合理的智能体对环境是无所不知的吗?
 - 不是。受限于传感器及其感知。
- 合理的智能体具有超常的洞察力吗?
 - 不是。可能缺少对环境动态变化的掌握。
- 那么，合理的智能体在环境中探索和学习吗?
 - 是的。只有具备这些，才能在未知的环境中行动。
- 合理的智能体不尽然都会表现的成功，但是，
 - 它们都是有自主性的（不是简单规则化的选择行为动作）

任务环境描述- PEAS

- 性能指标(Performance measure)
 - -1 每走一步; +10 一个豆子;
+500 赢; -500 输...
- 环境 (Environment)
 - 整个游戏环境
- 促动器(Actuators)
 - 左、右、上、下
- 传感器 (Sensors)
 - 整个游戏环境可见



环境描述：自动驾驶出租车

- 性能指标
- 环境
- 促动器
- 传感器



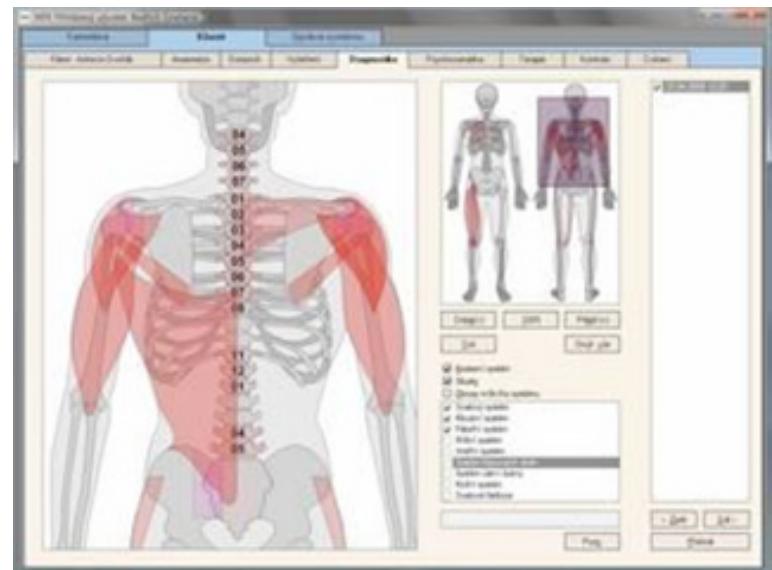
环境描述：自动驾驶出租车

- 性能指标
 - 收入，乘客满意度，油耗，罚单，保险费
- 环境
 - 街道，乘客，行人，其他车辆
- 促动器
 - 方向盘，刹车系统，油门，显示器/扬声器
- 传感器
 - 照相机，雷达，引擎传感器，麦克风，速度仪表



环境描述：医疗诊断系统

- 性能指标
 - 患者康复，费用，诊断准确性
- 环境
 - 患者，医护人员，保险公司，法院
- 促动器
 - 屏幕显示，电子邮件
- 传感器
 - 键盘，鼠标



环境类型（属性）

- 全部可观察 vs 部分可观察(fully observable vs partially observable)
 - 每时每刻观察到全盘环境
- 单一智能体 vs 多智能体 (single vs multi-agents)
 - B的行为是最大化一个性能指标，其数值由A的行为决定
 - 竞争； 合作
- 变化确定性的 vs 随机变化（变化不确定）的(deterministic vs stochastic)
 - 当前状态和智能体行为决定环境下一个状态
 - 忽略其他智能体行为所引发的不确定性
 - 不确定(uncertain)环境: 在部分可观察，或随机变化的环境

环境类型（属性），继续

- 由片段组成的 vs 关联序列的 (episodic vs sequential)
 - 作业环境由独立的片段组成；每片段中，智能体观察一次，做出一个动作；每片段中得行为相互独立
 - 序列环境中，智能体行为有相关性
- 静态 vs 动态的 (static vs dynamic)
 - 智能体在思考时，环境是否在改变
 - 半动态的，要求智能体能做出快速反应
 - 出租车自动驾驶；填字拼成语；计时象棋比赛
- 离散性 vs 连续性 (discrete vs continuous)
 - 决定于环境的状态，时间的处理，智能体的感知和行动
 - 象棋，出租车自动驾驶

环境类型

环境属性	Pacman	医疗诊断	出租车	计时象棋	质检机器人
可观察性					
智能体					
环境变化确定性					
环境变化的关联性					
环境变化的动态性					
状态连续性					

环境类型

环境属性	Pacman	医疗诊断	出租车	计时象棋	质检机器人
可观察性	全部	部分	部分	全部	部分
智能体	单个	单个	多个	多个	单个
环境变化确定性	不确定	不确定	不确定	确定	不确定
环境变化的关联性	联续的	联续的	联续的	联续的	片段的
环境变化的动态性	动态	动态	动态	半动态	动态
状态连续性	连续的	连续的	连续的	离散的	连续的

部分答案也依赖于环境是如何定义的。

智能体的设计

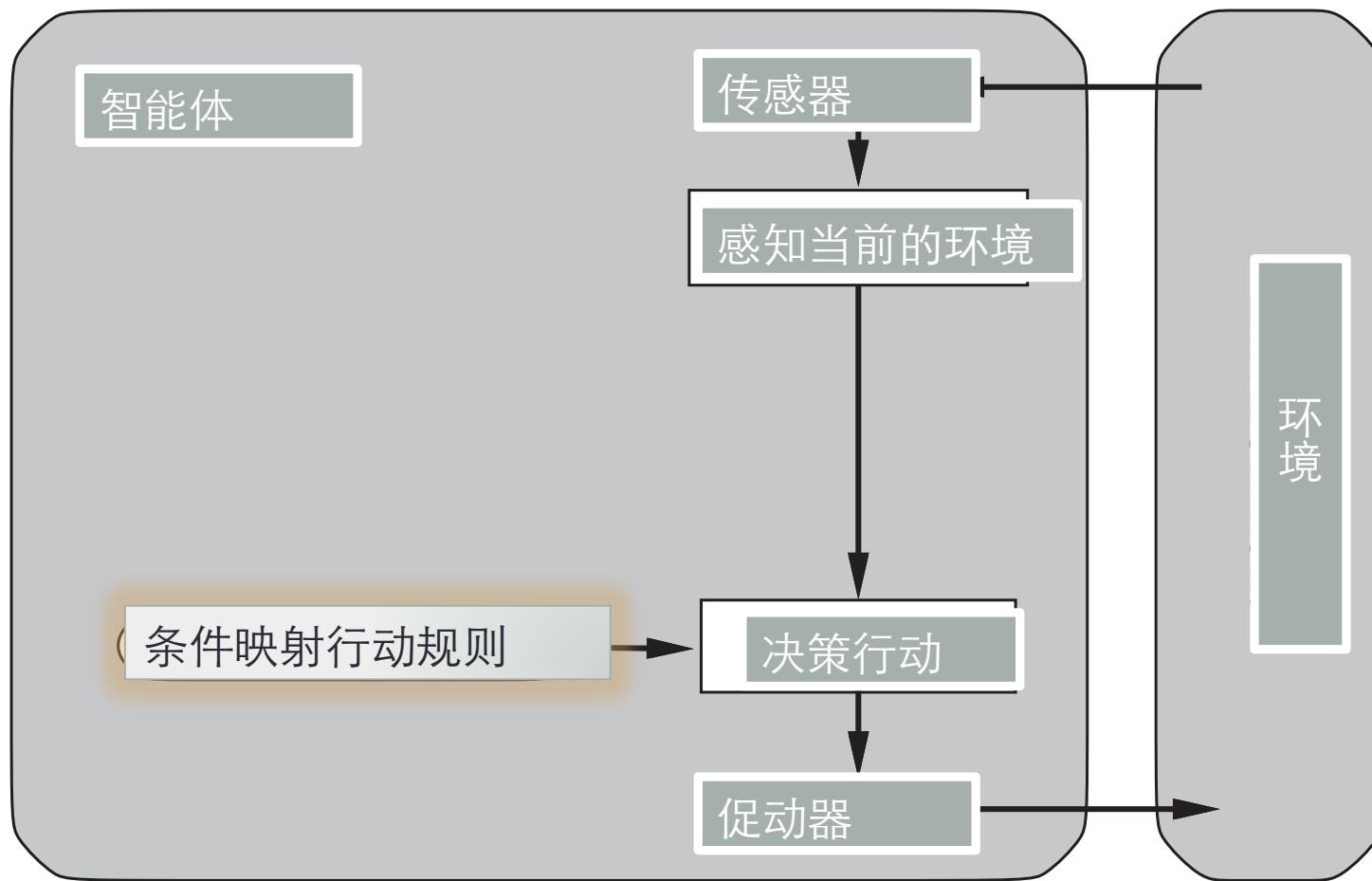
- 环境类型决定智能体的设计

- 这里不讨论其物理框架（包括计算机器，传感器，促动器）；只关于智能体程序的设计
- 部分可观察的 => 智能体需要存储内存
- 环境变化不确定的 => 需要准备偶发事件
- 多个智能体 => 可能需要一些随机的行为
- 静态的 => 有时间去计算一个合理决策
- 连续时间的 => 需要连续运行的控制器

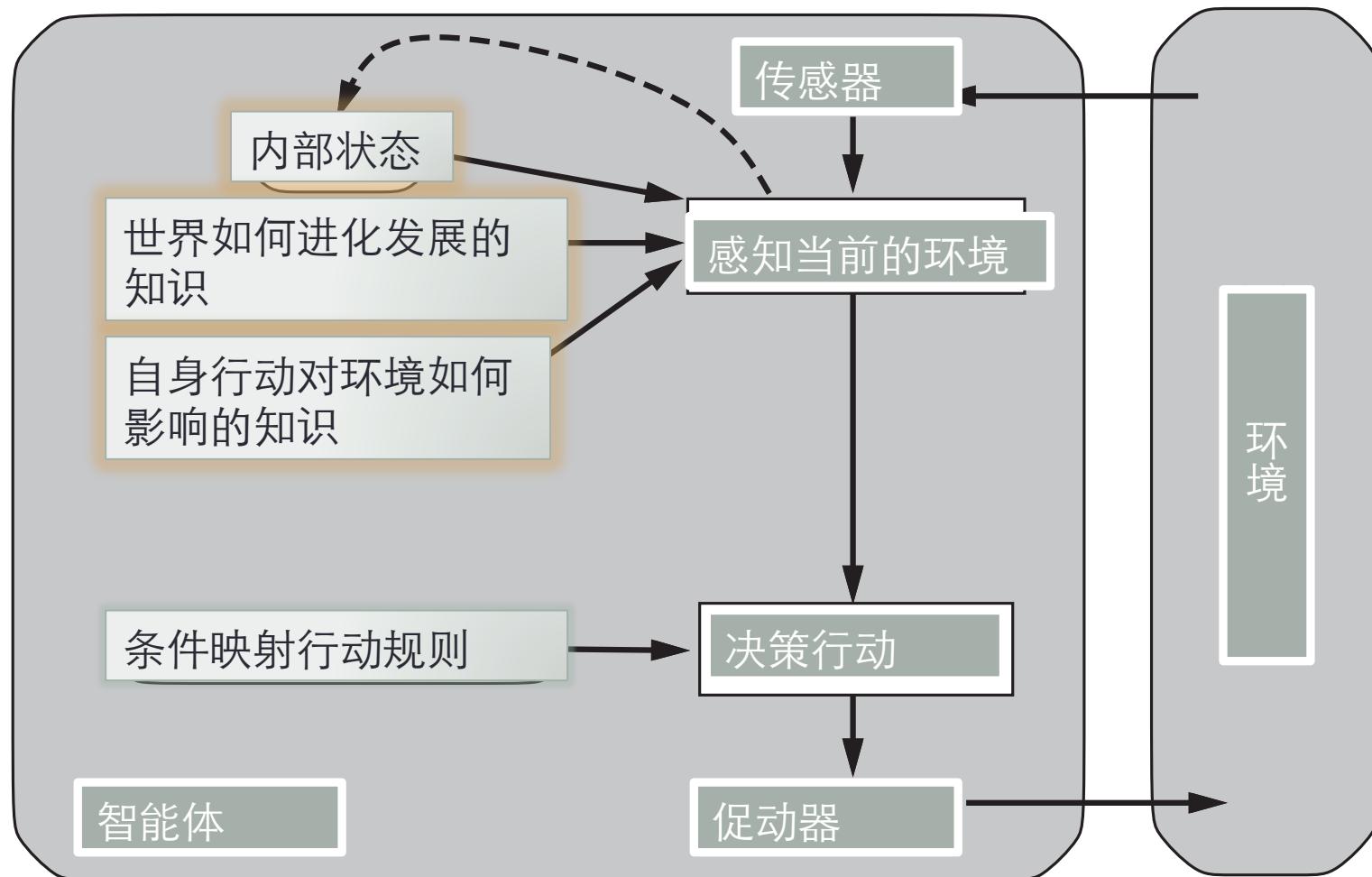
智能体类型

- 从简单的到复杂的
 - 简单反射型智能体
 - 记录状态的反射智能体
 - 基于目标的智能体
 - 基于功效的智能体

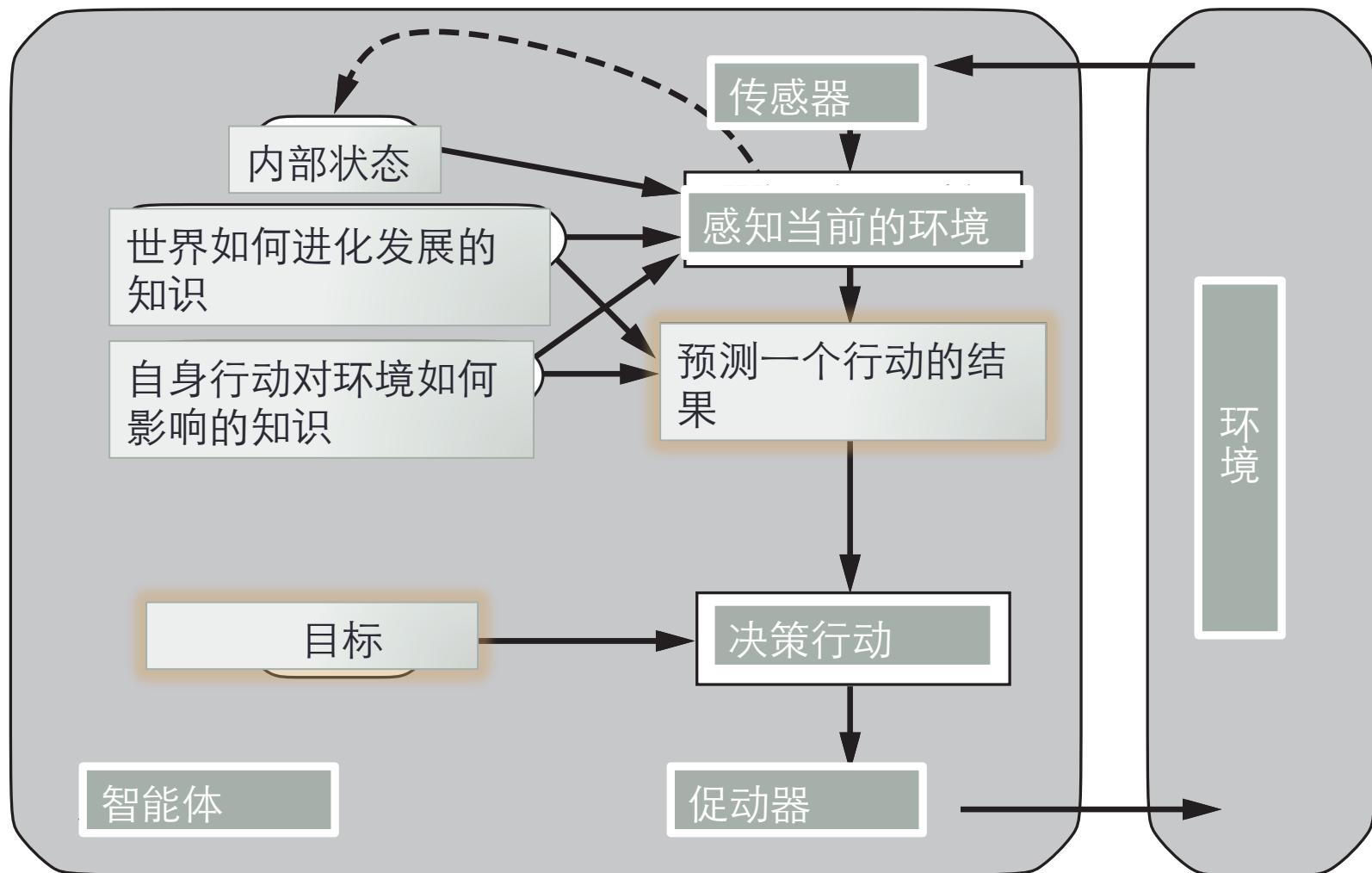
简单反射智能体



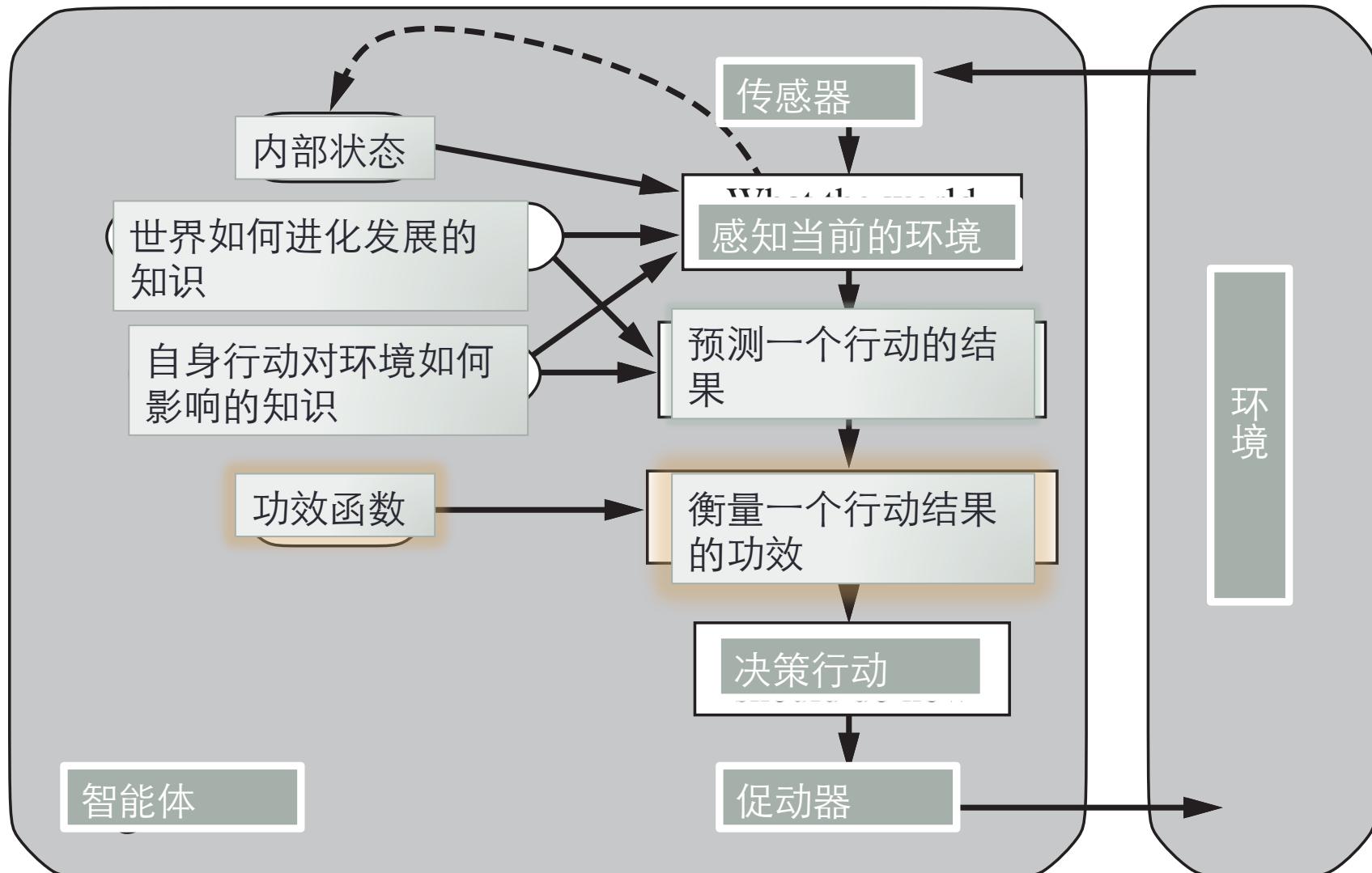
记录状态的反射智能体



基于目标的智能体



基于功效的智能体

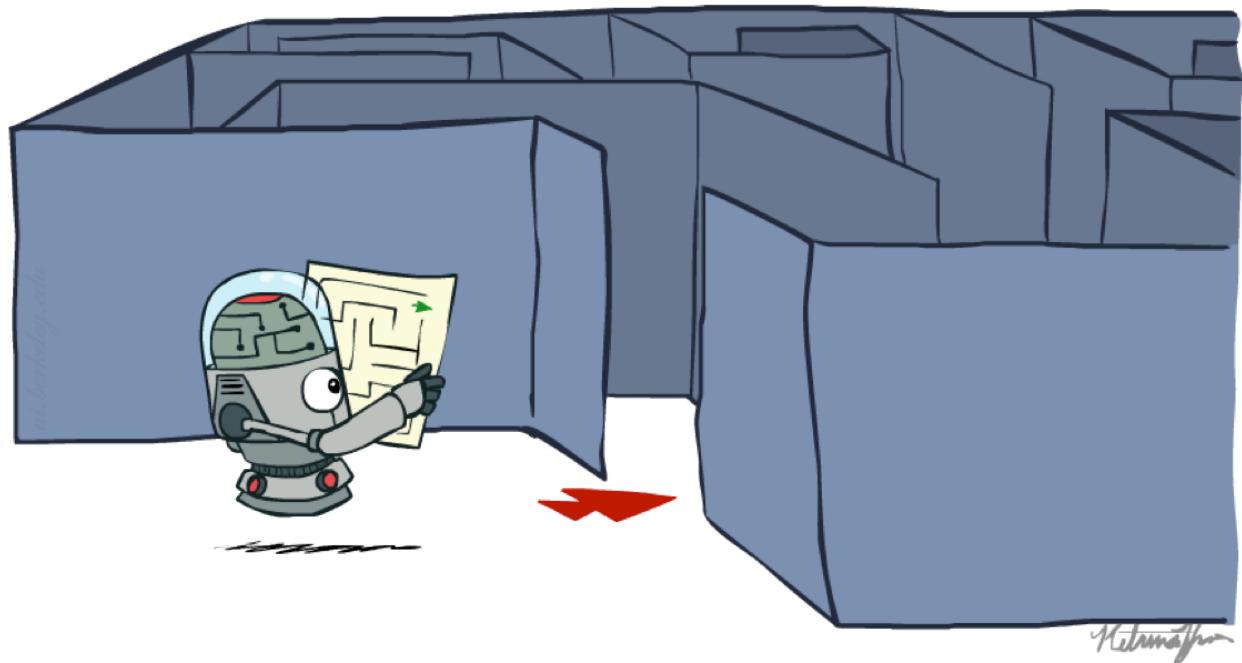


总结

- 一个**智能体**通过自身的**传感器**和**促动器**，与**环境**作互动。
- **智能体函数**（描述智能体在不同情况下采取的行动），由**智能体程序**在一台**机器**上来实现。
- PEAS 描述用来定义任务环境；准确的定义对智能体的设计很关键。
- 越困难的环境要求越复杂的智能体设计和复杂的结构。

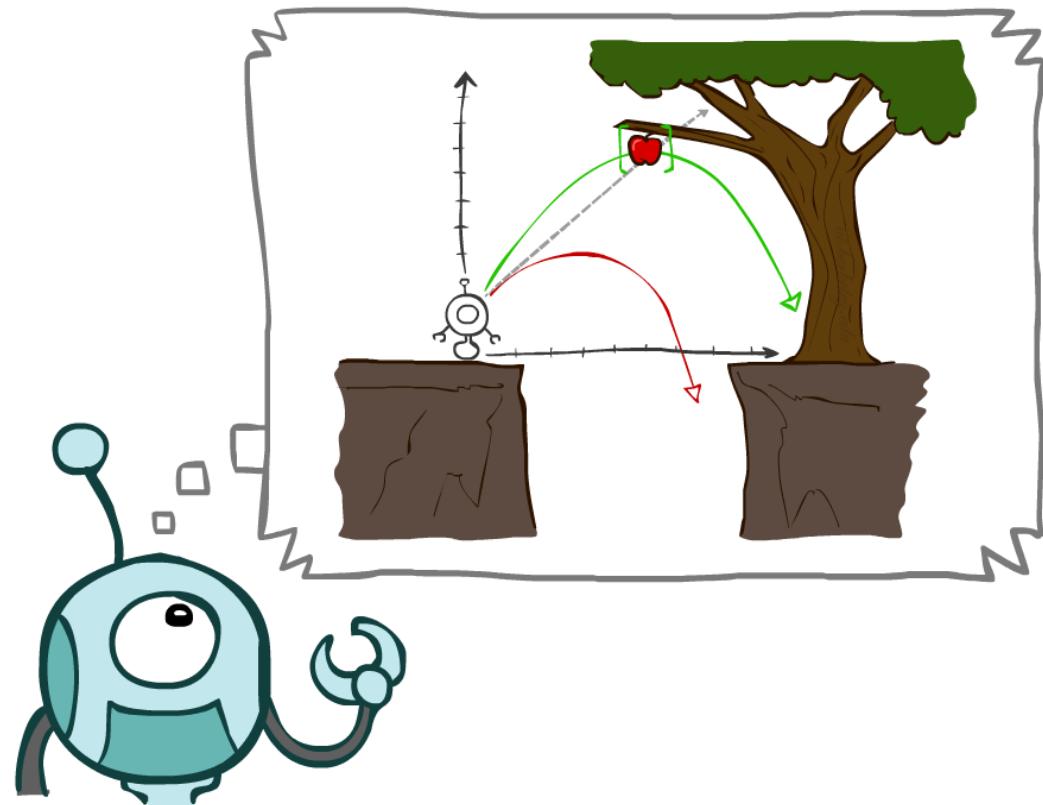
人工智能导论： 搜索

齐琦



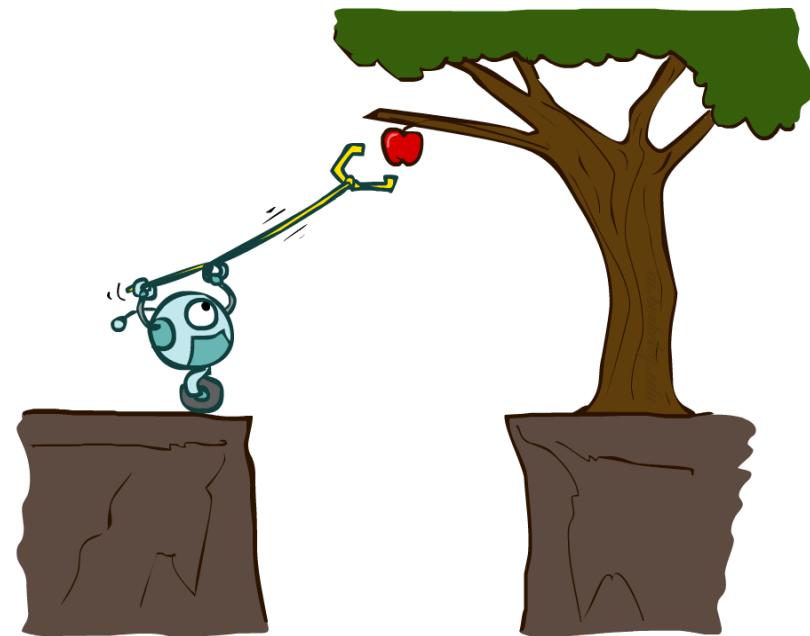
提纲

- 提前计划的智能体
- 搜索问题
- 基础搜索法
 - 深度优先搜索
 - 广度优先搜索
 - 基于成本的统一搜索

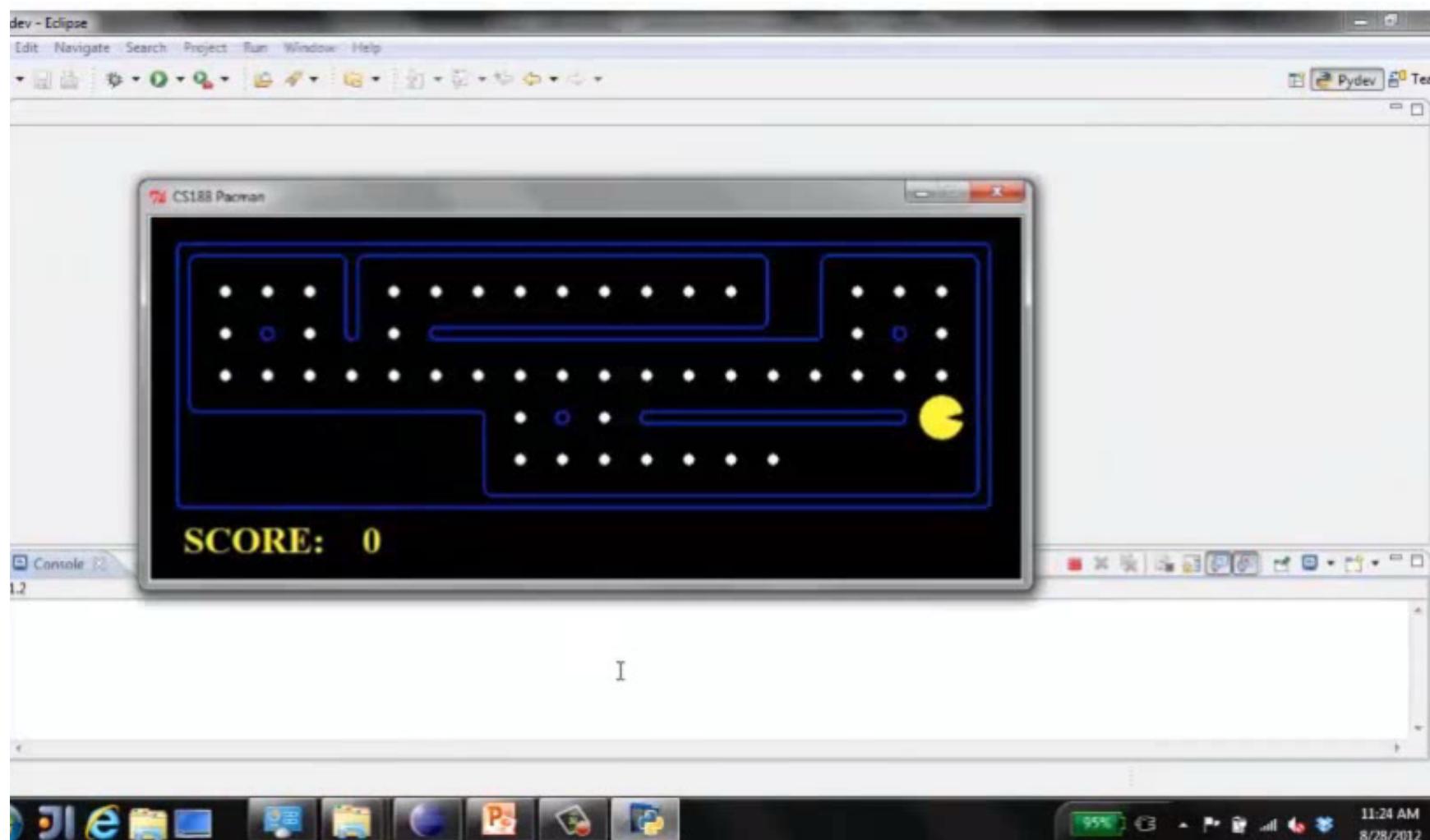


提前计划的智能体

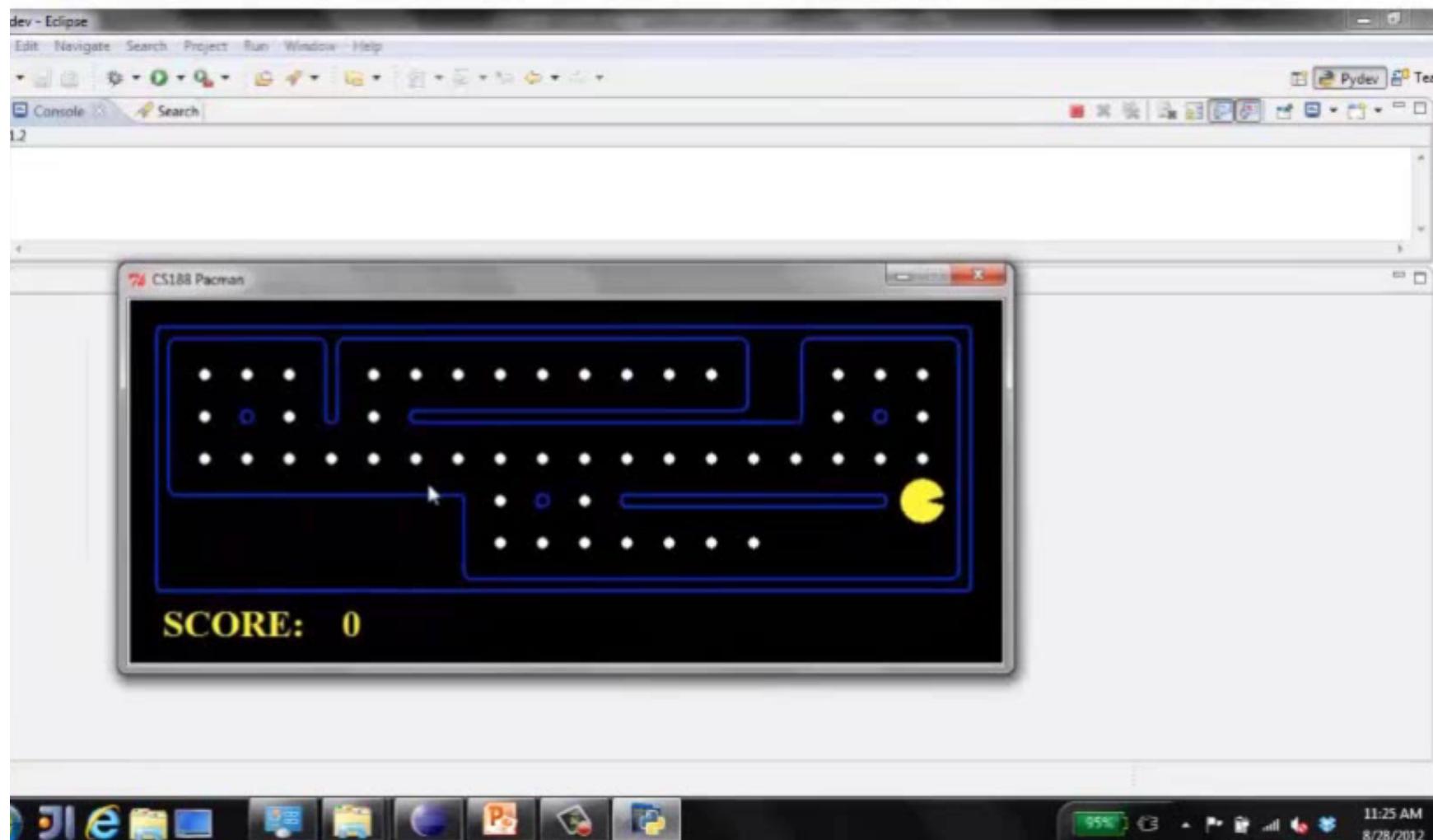
- 做计划的智能体
 - 决策基于对行动的预测结果
 - 一定有一个转换模型：根据不同的行动，环境是如何演变的
 - 必须有一个目标
- 可能的策略
 - 在开始之前制定一个完整、优化的计划，然后再执行。
 - 先制定一个简单、贪心(greedy)的计划，开始执行；当走错的时候再重新计划。



行动后改变计划一演示视频



完整计划后再执行一演示视频

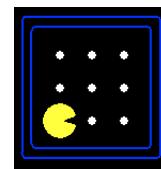
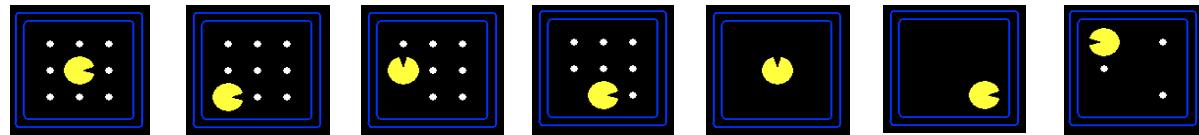


搜索问题

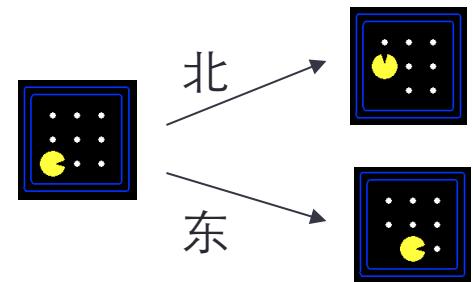


搜索问题

- 一个**搜索问题**包括：
 - 一个状态空间
 - 在每一个状态里，一个可允许的动作集合
 - 一个转换模型 $\text{Results}(s, a)$
 - 一个步骤成本函数 $\text{cost}(s, a, s')$
 - 一个开始状态，和一个目标到达测试
- 一个**解决方案**是一序列动作（一个规划），从开始状态到一个目标状态

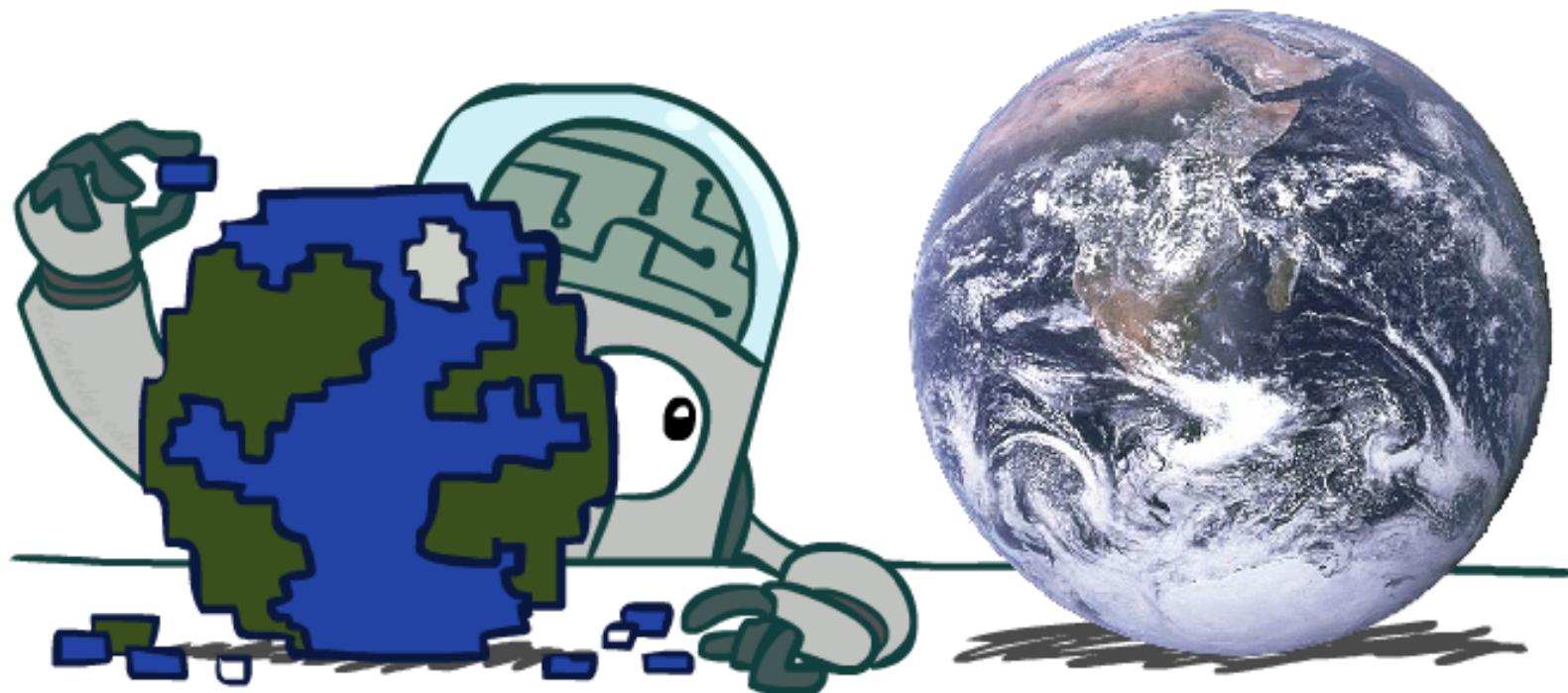


{北, 东}

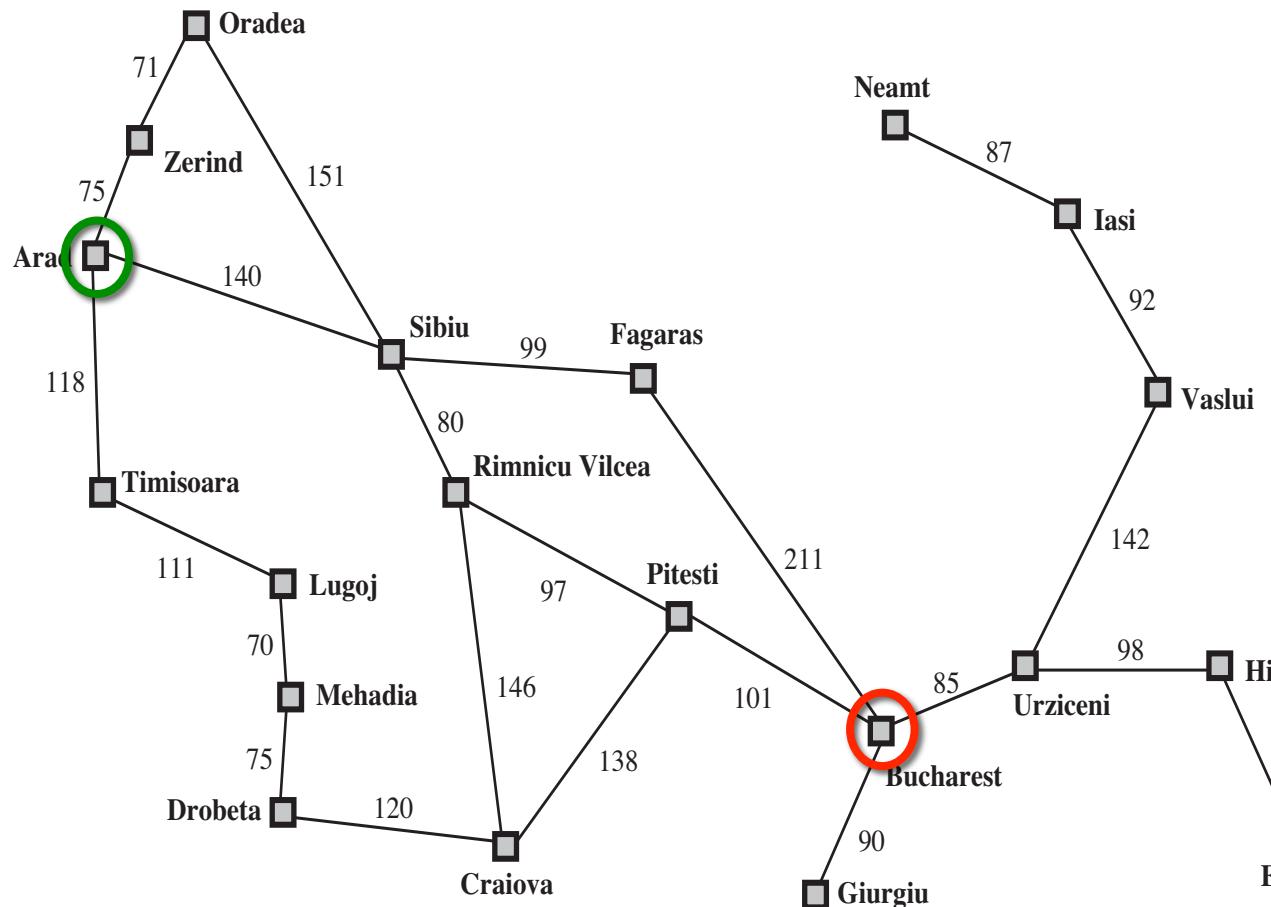


搜索问题

- 定义搜索问题是一个建模的过程
 - 抽象的数学描述



举例：在罗马尼亚旅行



- 状态空间
 - 城市
- 动作
 - 去一个邻居城市
- 转换模型
 - $\text{Result}(A, \text{Go}(B)) = B$
- 步骤成本
 - 距离
- 开始状态
 - Arad
- 目标测试
 - 当前城市
==Bucharest?
- 解决路径 ?

状态空间可能会很大

- 真实世界里的状态包括许多细节
- 搜索问题定义的状态是一种抽象，去掉不必要的细节

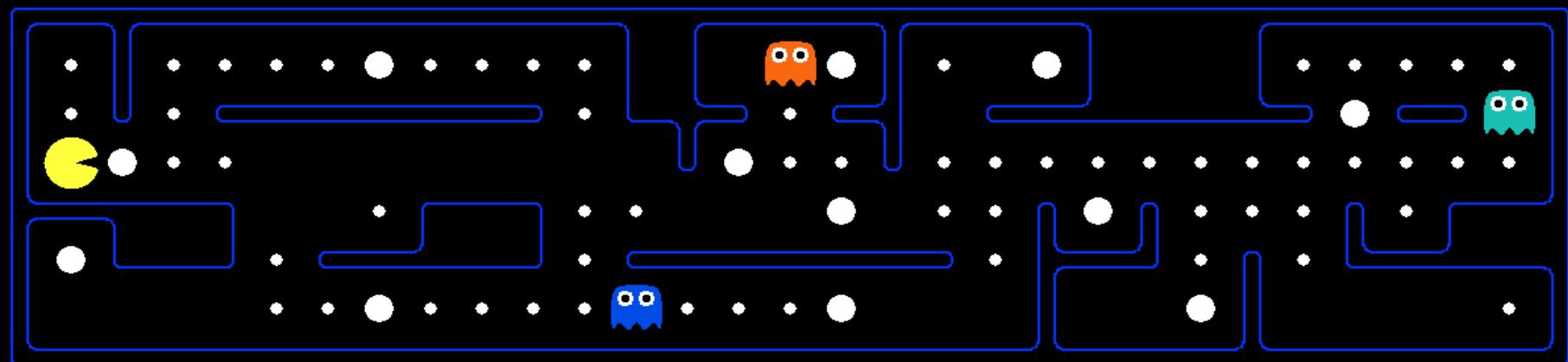
- 路径搜索问题
 - 状态: (x, y) 位置
 - 行动: 北南东西
 - 转换模型: 更新位置
 - 目标测试: $(x, y) ==$ 终点

MN 个状态 ($|x|=M, |y|=N$)

Pacman 吃豆子问题

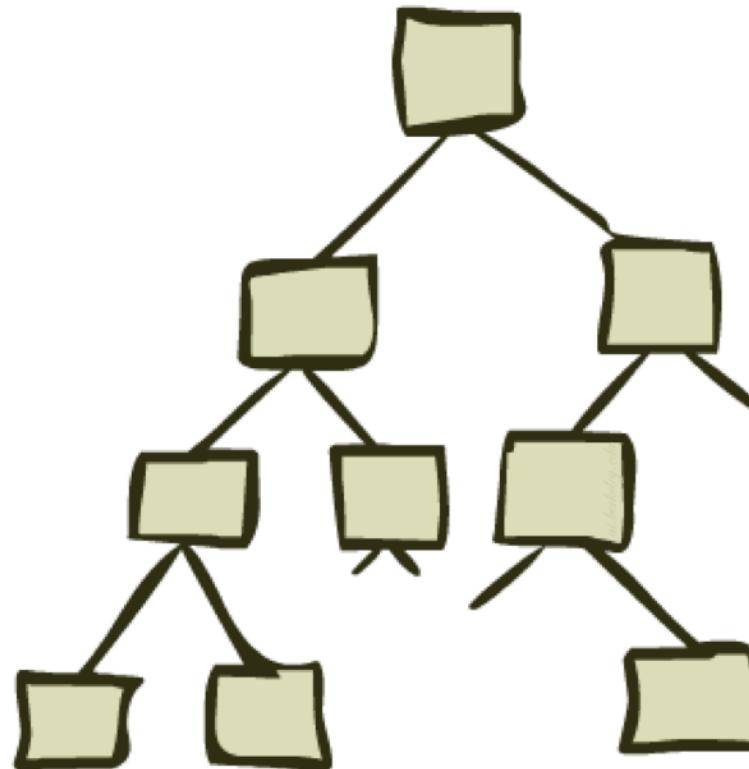
- 状态: $((x, y), \text{豆子布尔})$
 - 行动: 北南东西
 - 转换模型: 更新位置, 和豆子布尔
 - 目标测试: 豆子布尔全都是 `false`
- $MN2^{MN}$ 个状态

更复杂的情况



- 问题：吃掉所有豆子，同时保持所有怪物都在被威慑状态
- 状态空间如何定义？
 - 智能体位置，豆子布尔值，能量丸布尔值，剩余威慑时间

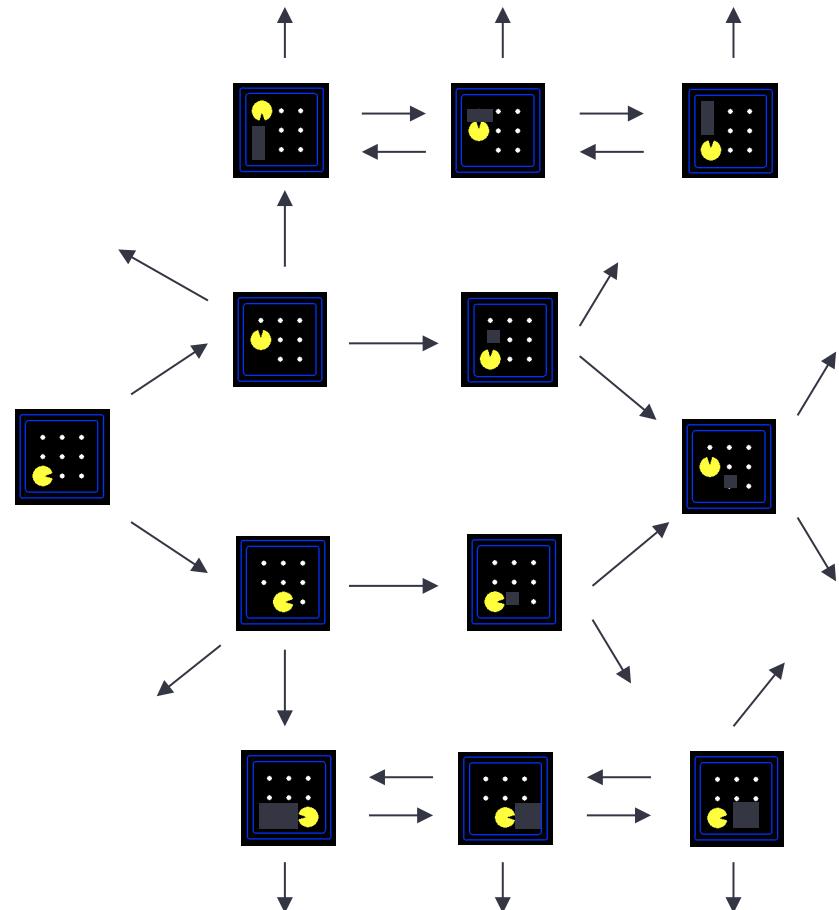
状态空间图 和 搜索树



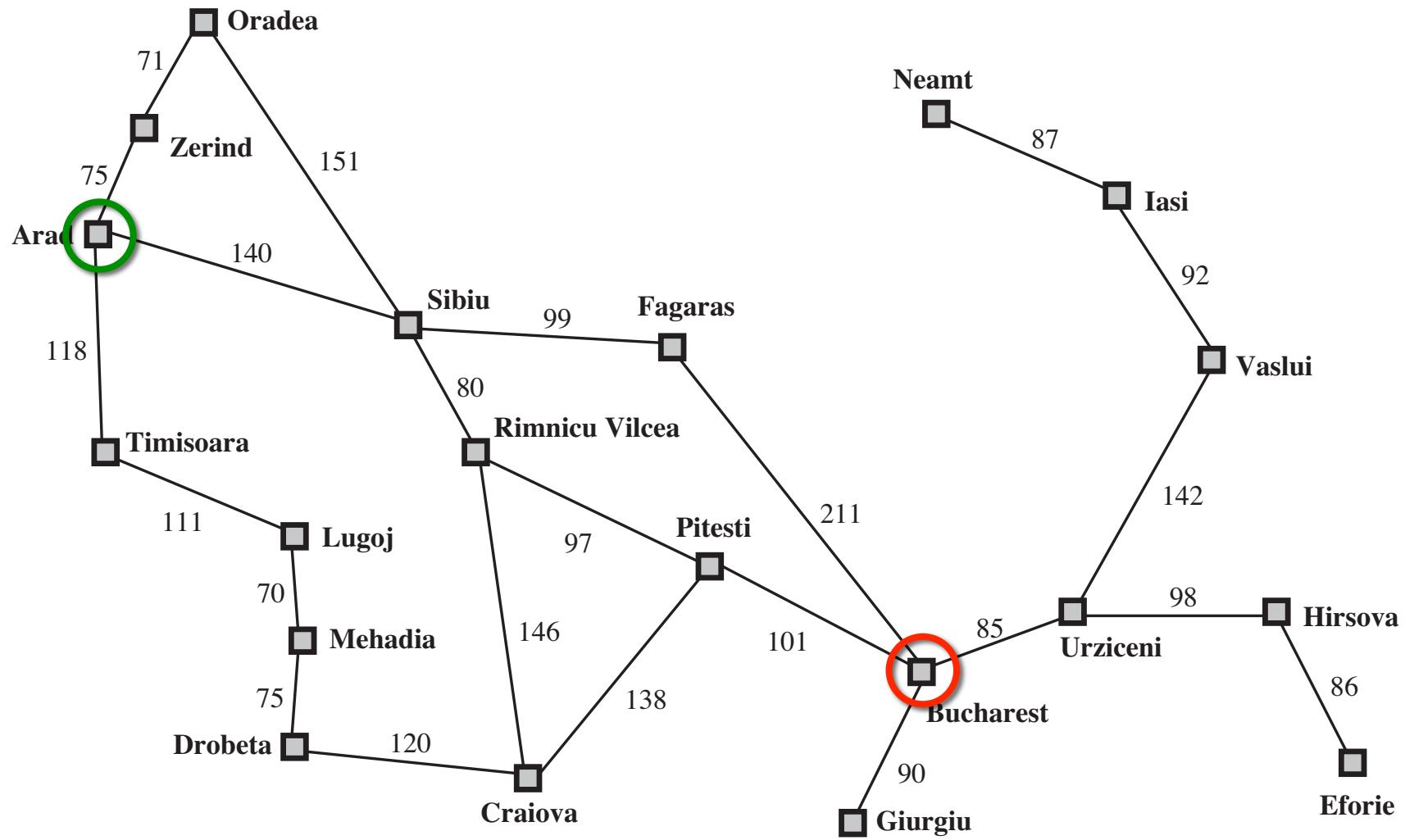
区别在哪里？

状态空间图

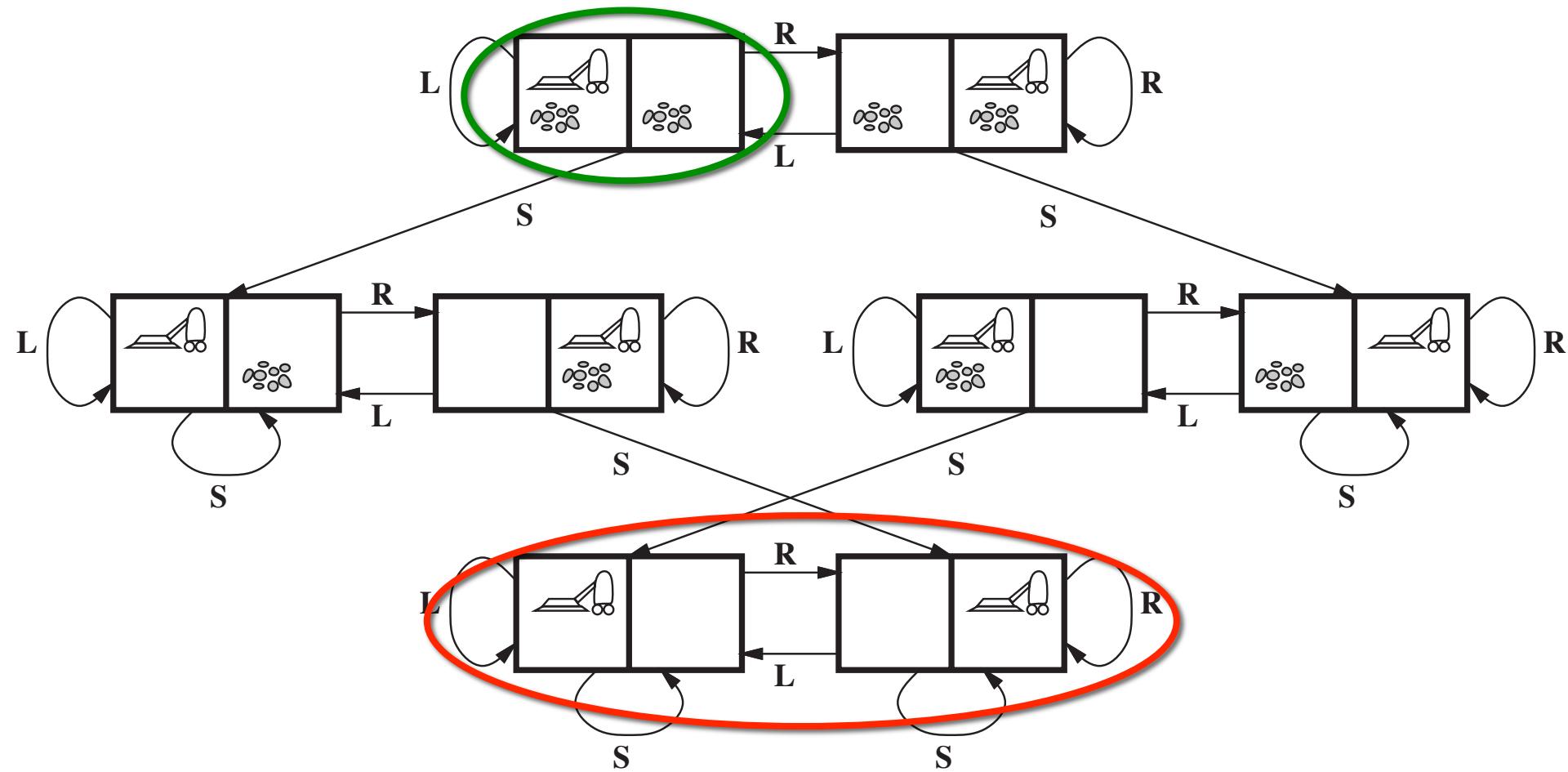
- 对搜索问题的数学表达
 - 节点是搜索问题中定义的状态
 - 边代表动作所导致的转换
 - 目标测试是当前节点是否在目标节点中
- 每个状态只出现一次！
- 状态图有时很大难以完全构建



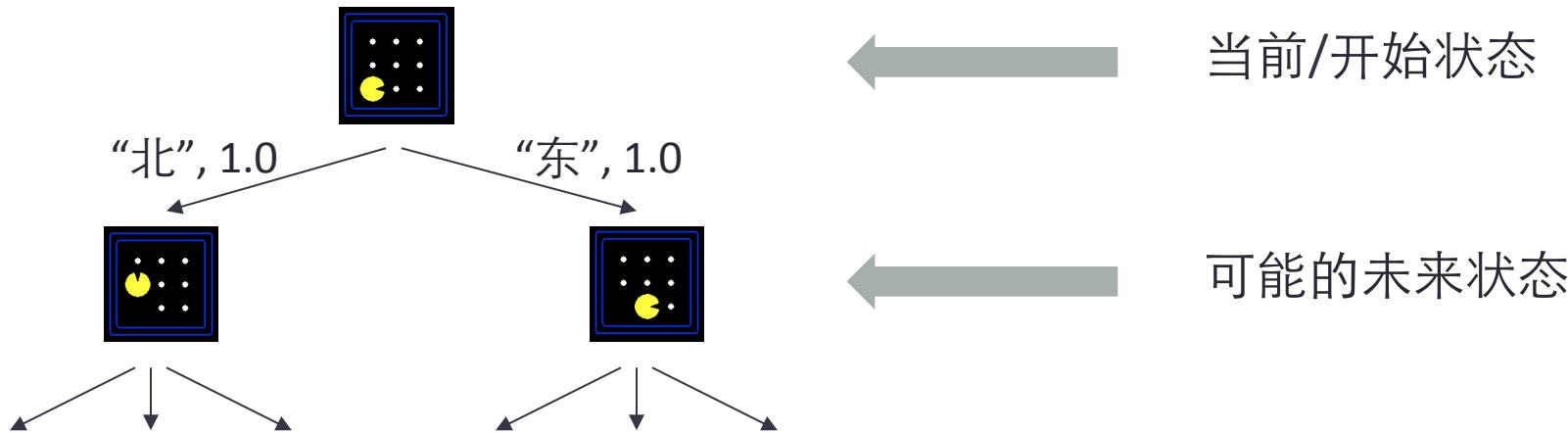
状态图举例



状态图举例

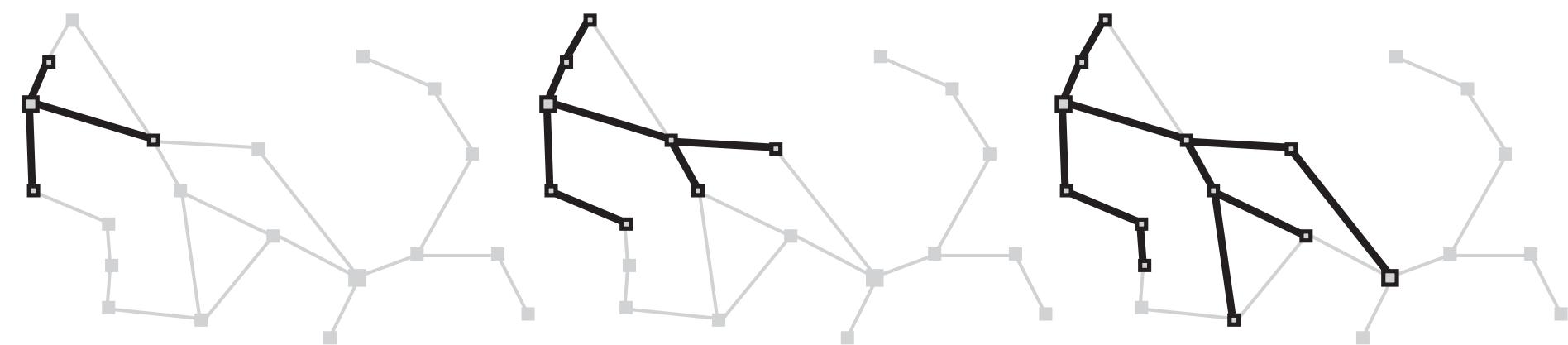


搜索树



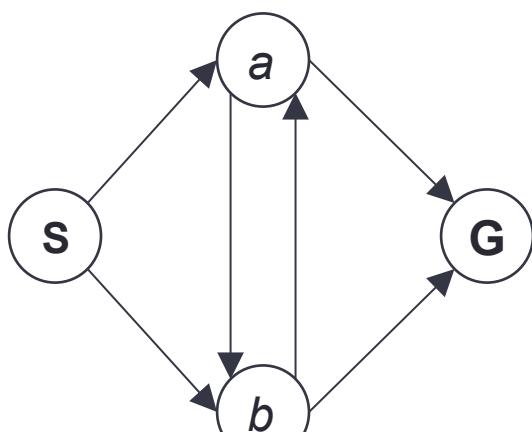
- 树上反映的是尝试的行动规划及其结果。
- 开始状态是根节点。
- 子节点反映的是可能的动作结果。
- 一个节点是一个状态，但一个状态可能出现多次，反映的时不同行动规划的结果。
- 对于大多数问题，我们实际上很难建立整个搜索树。

状态图 vs 搜索树

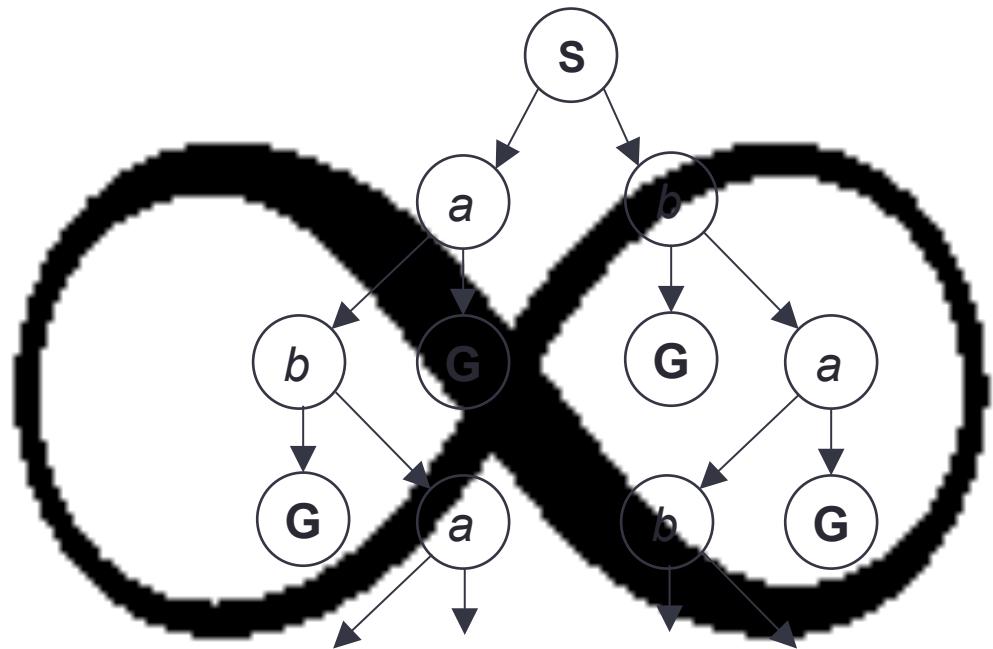


问题：状态图 vs 搜索树

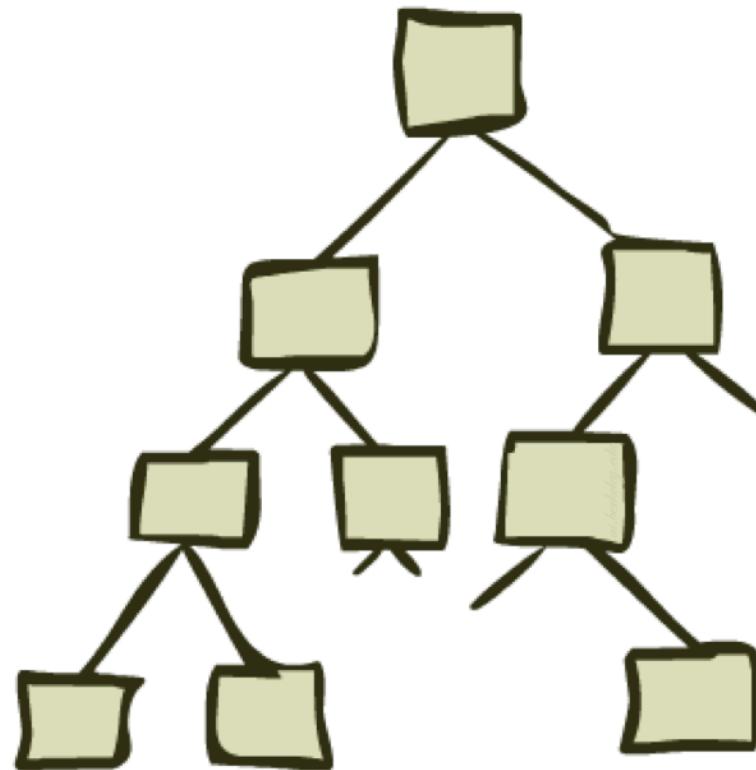
有一个4节点状态图：



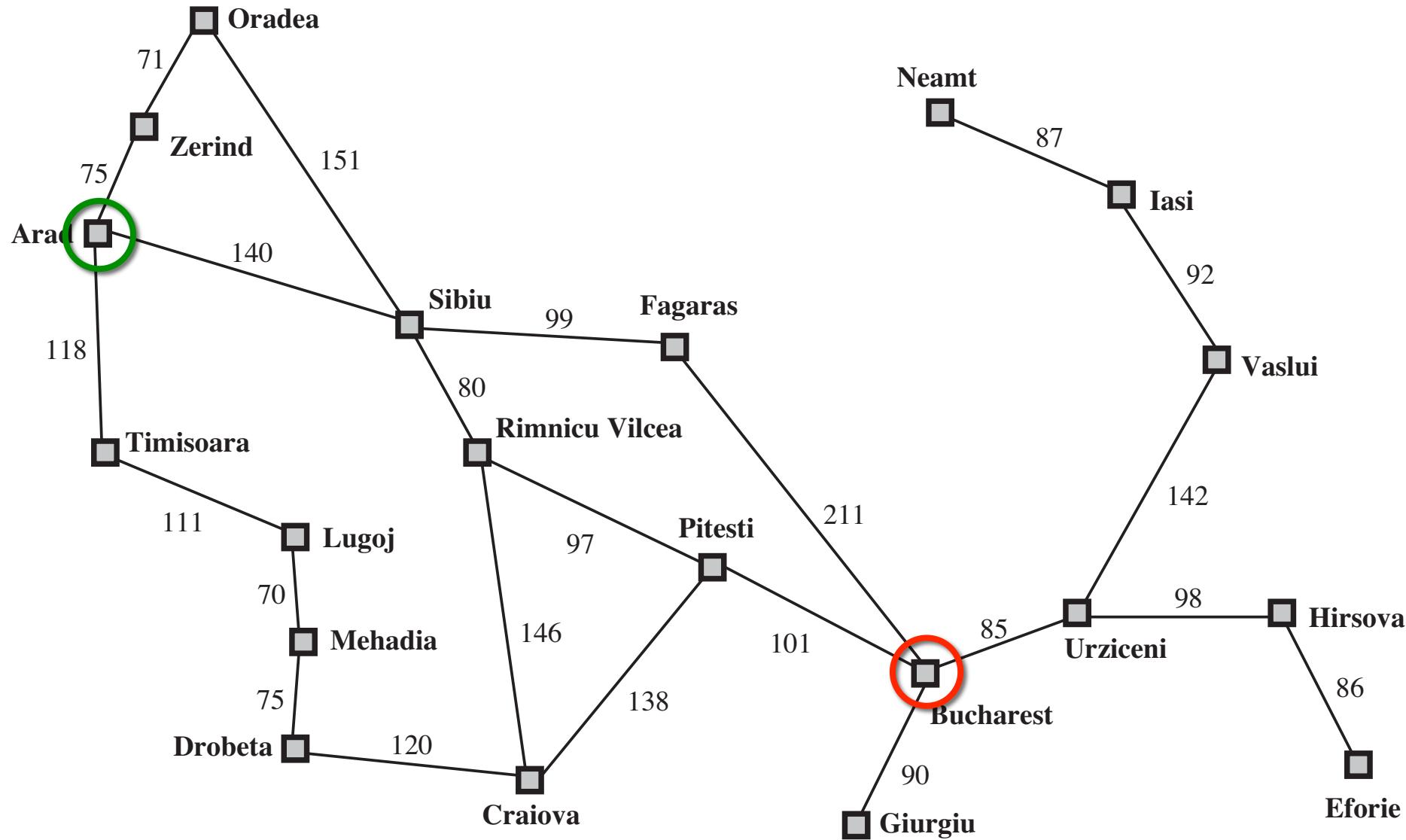
它的搜索树有多大（从S状态开始）？



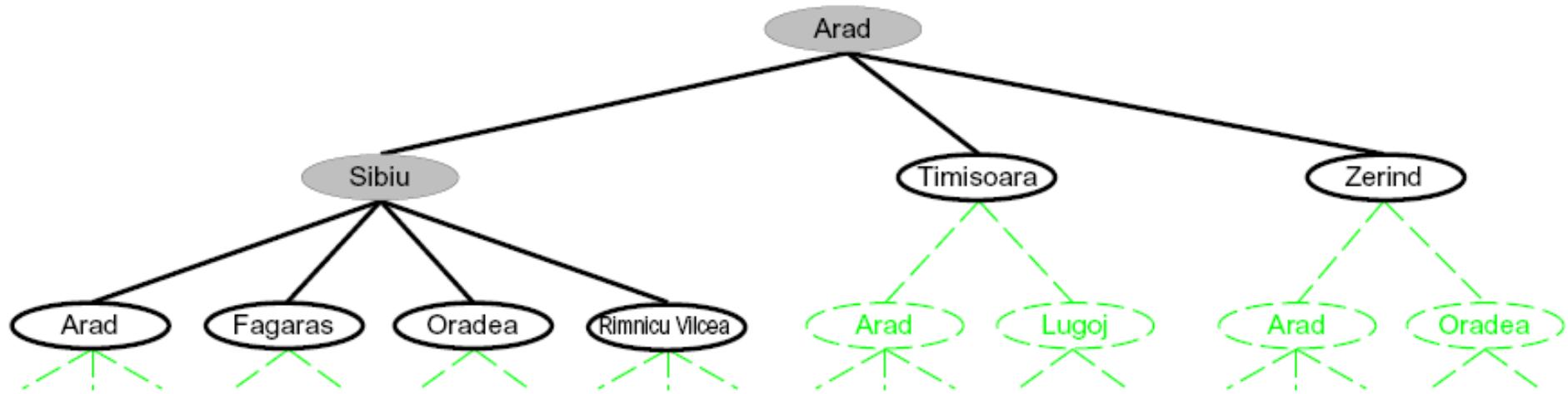
树搜索



搜索问题：罗马尼亚旅行



搜索树搜索



- 扩展树节点（潜在的行动规划）
- 从搜索前沿（当前的所有叶节点）中考虑扩展
- 树节点扩展的越少越好
- 探索策略

树搜索方法框架

function 树搜索(问题) **returns** 一个 解, 或 失败

把问题的初始状态放入 搜索前沿

loop do

if 搜索前沿 为空 **then return** 失败

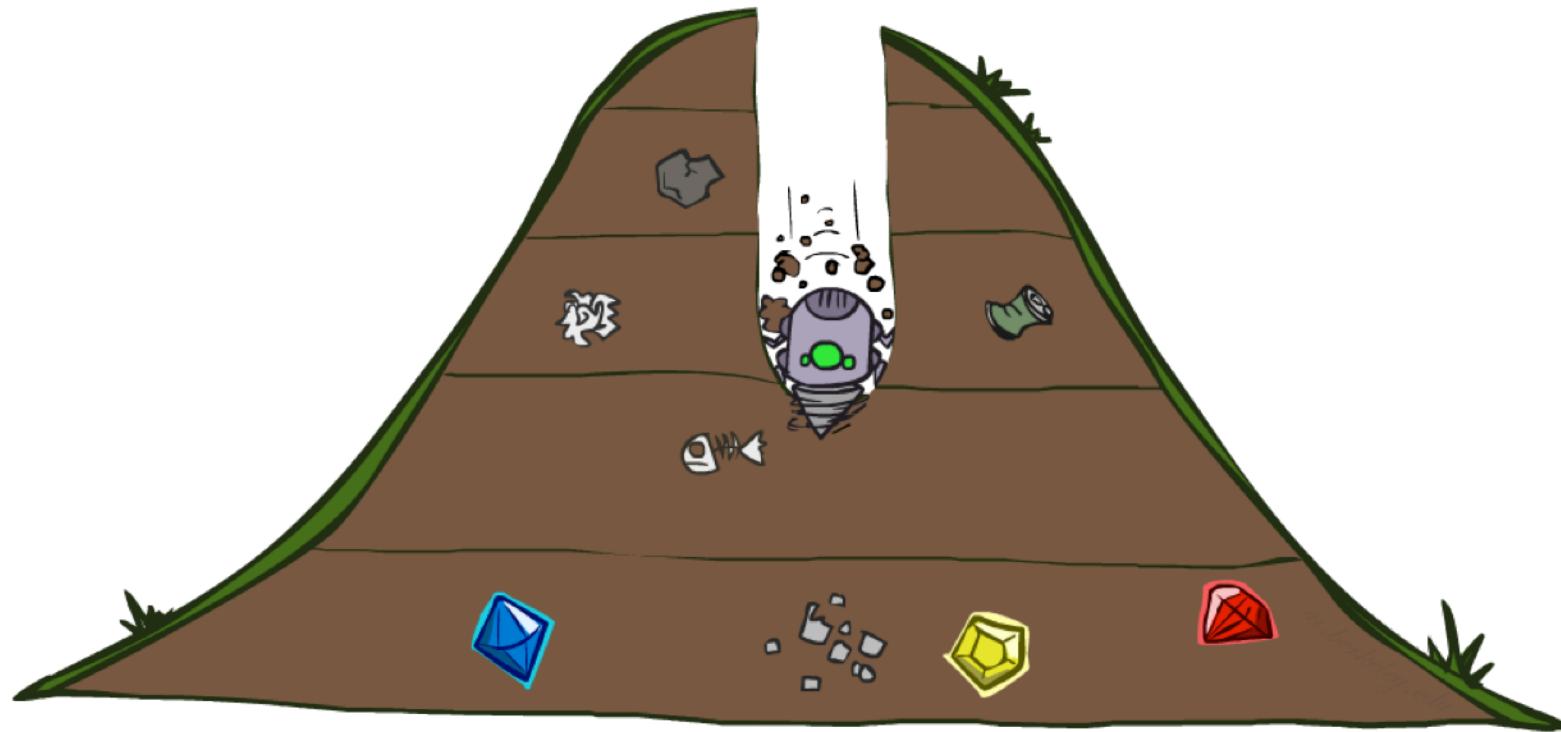
 选择一个叶 节点 并把它从 搜索前沿 中移除

if 这个 节点 包含一个目标状态 **then return** 相应的解 (路径)

 扩展这个选择的 节点, 把扩展结果的 节点 加入 搜索前沿

- 关键元素：
 - 搜索前沿
 - 扩展
 - 探索(选择)策略
- 主要问题： 从哪些前沿叶节点开始探索扩展？

深度优先搜索

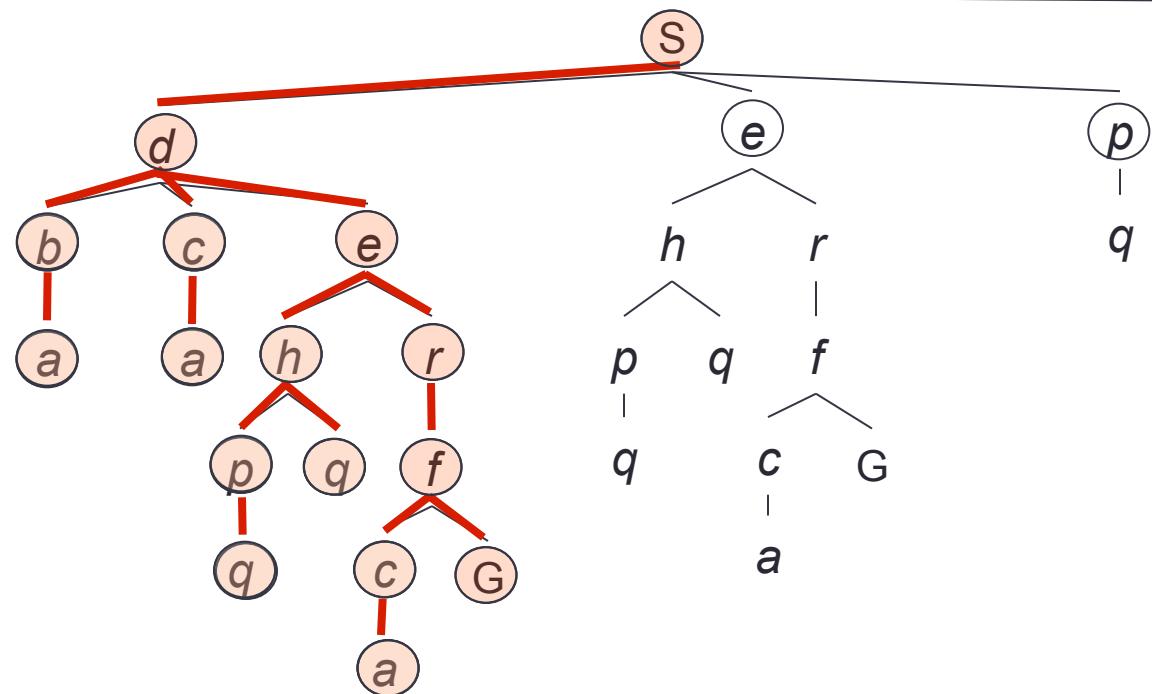
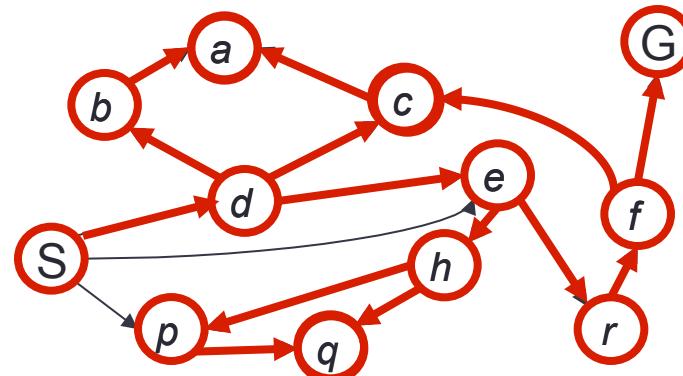


深度优先搜索

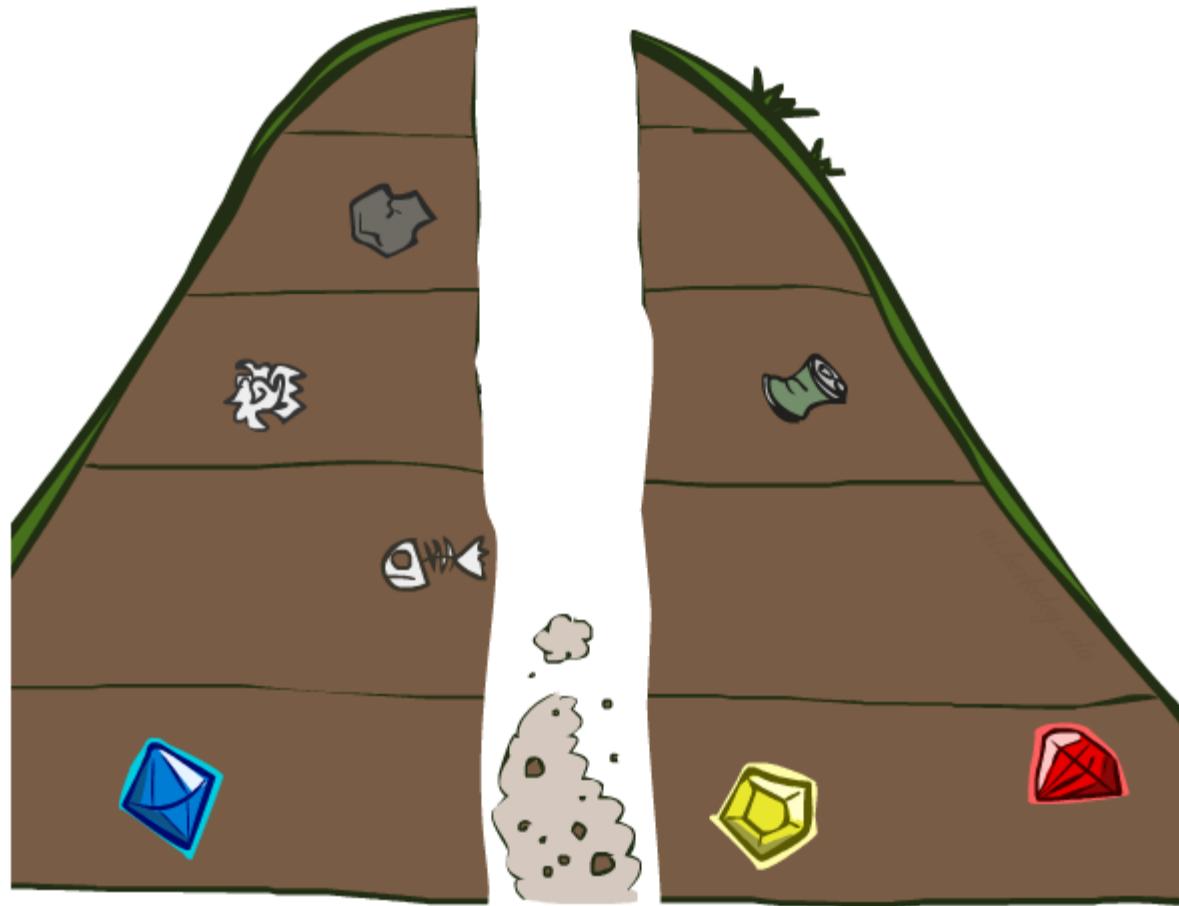
策略：总是最先扩展一个最深的节点

实现：

前沿是一个后进先出的栈

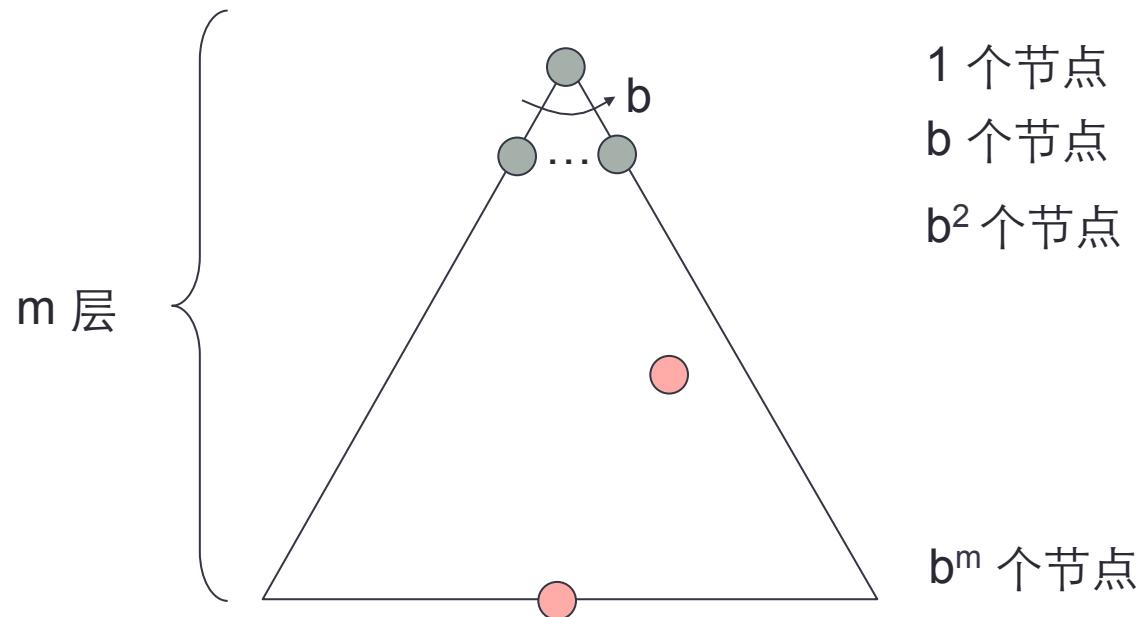


搜索算法的属性



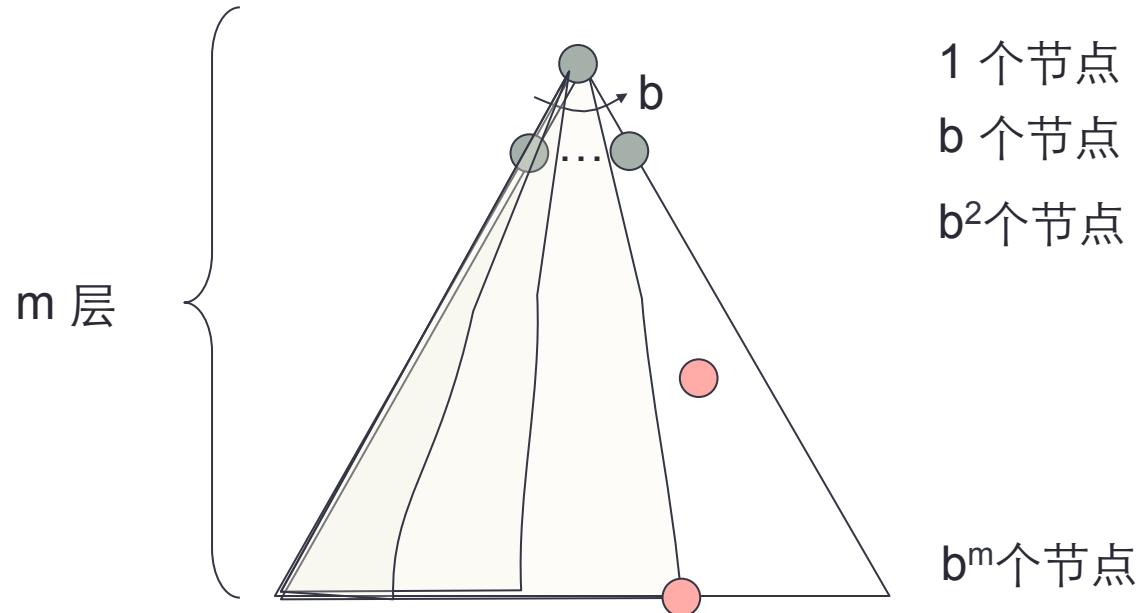
搜索算法的属性

- 完全性：保证能找到一个存在的解？
- 最优性：保证能找到最小路径成本的解？
- 时间复杂性？
- 空间复杂性？
- 搜索树：
 - b ：最大分支数
 - m ：最大深度
 - 解可能存在于不同深度
- 整个树的节点总数？
 - $1 + b + b^2 + \dots + b^m = O(b^m)$

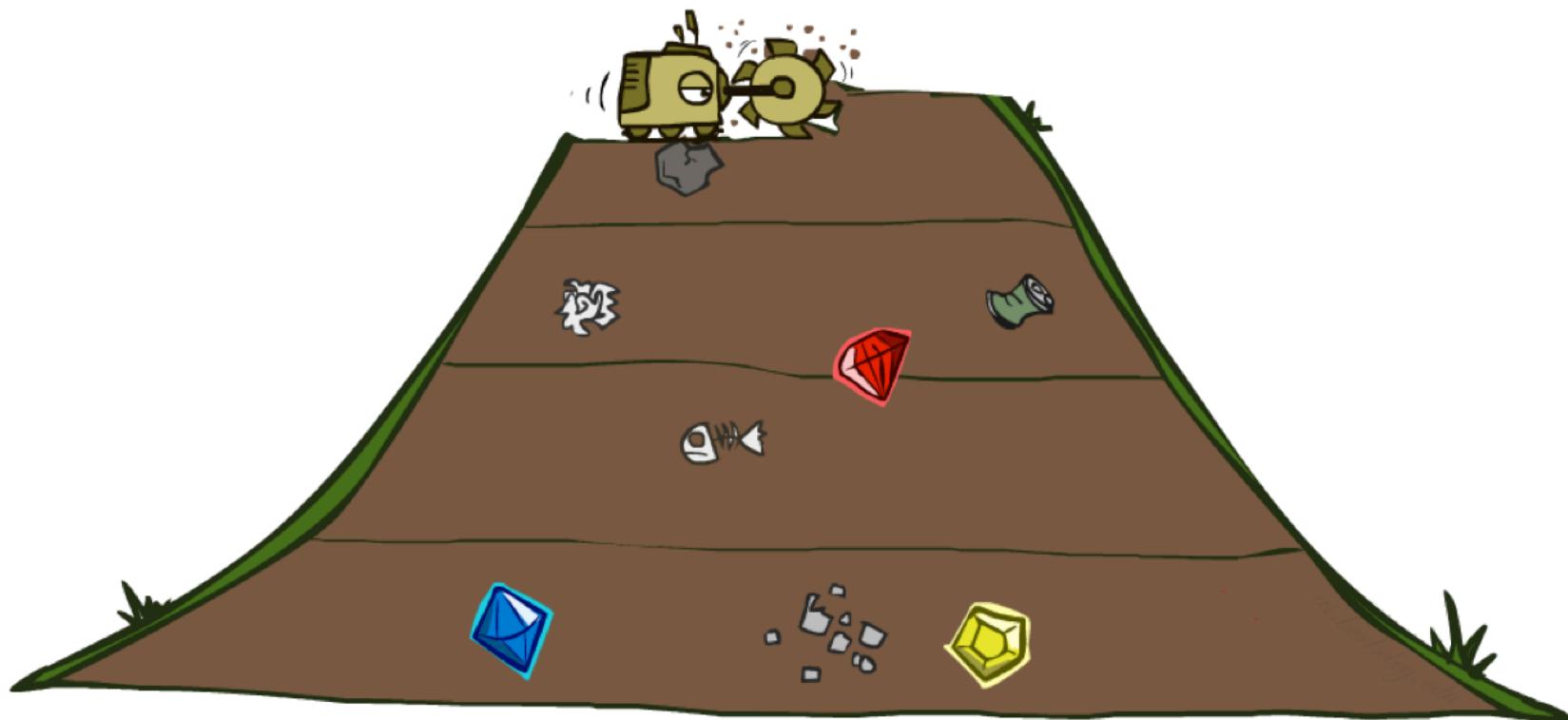


深度优先搜索 (DFS) 属性

- 哪些节点被扩展?
 - 某些左边的节点
 - 可以产生整个树
 - 如果 m 是有限的, 时间复杂度 $O(b^m)$
- 存储前沿需多少空间?
 - 只存储一条路径从根到一个叶节点及沿路相关兄弟节点, 所以 $O(bm)$
- 完全性?
 - m 可能是无穷。除非可以避免循环搜索
- 优化性?
 - 不优化。发现的可能是树最左边的一个次优解



广度优先搜索 (BFS)

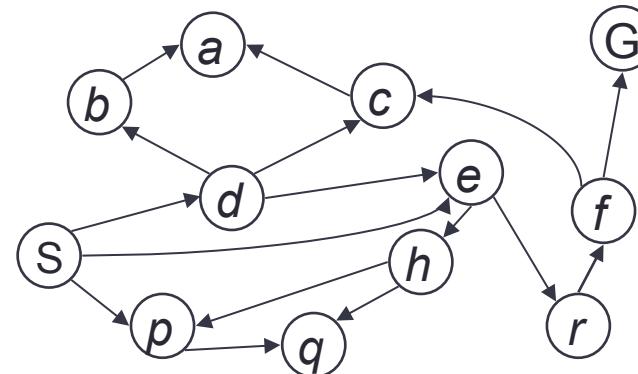


广度优先 (BFS) 搜索

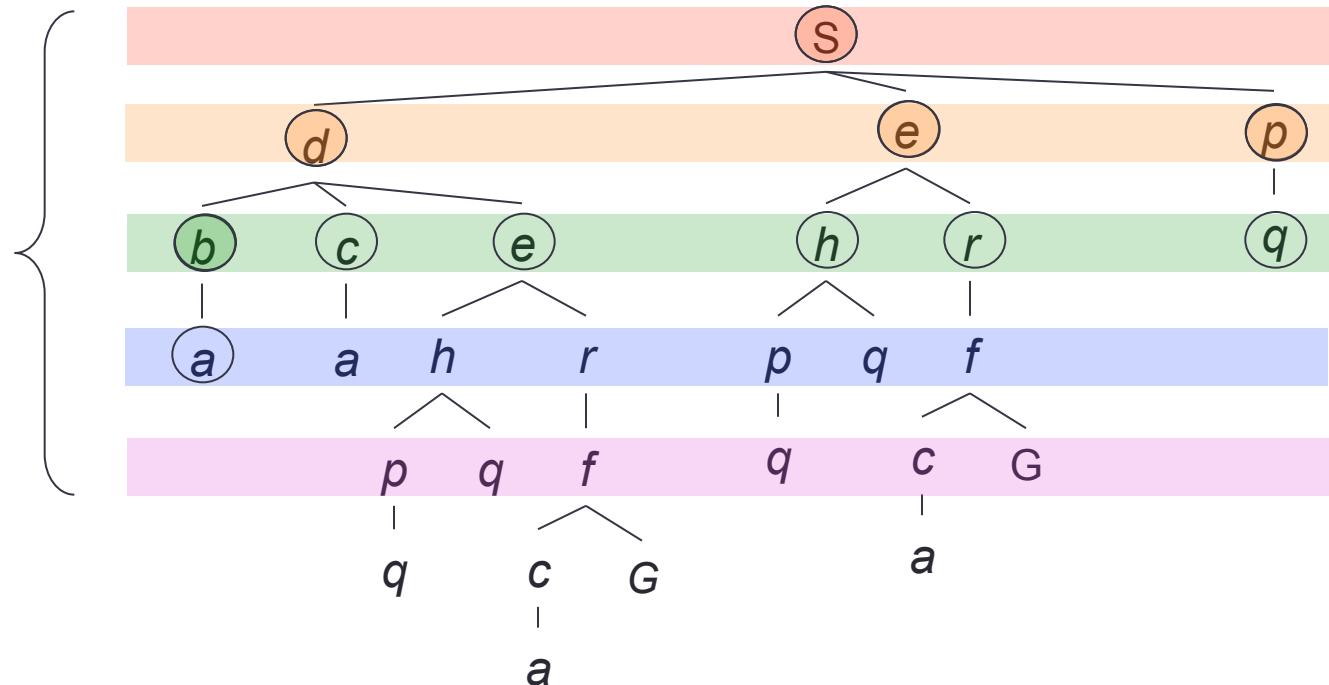
策略：优先扩展最浅的节点

实现：

先进先出的队列

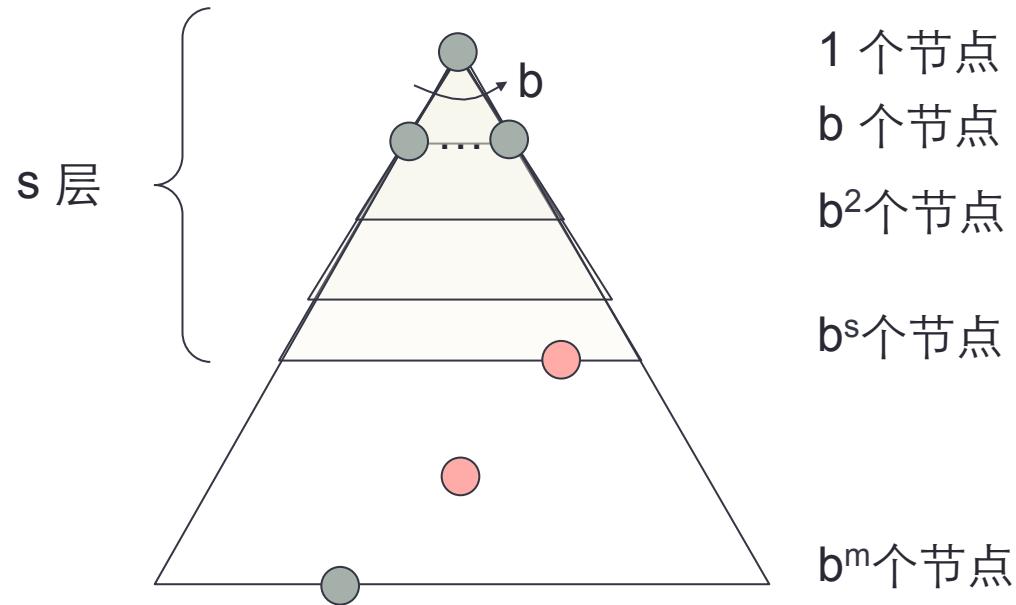


搜索
层次

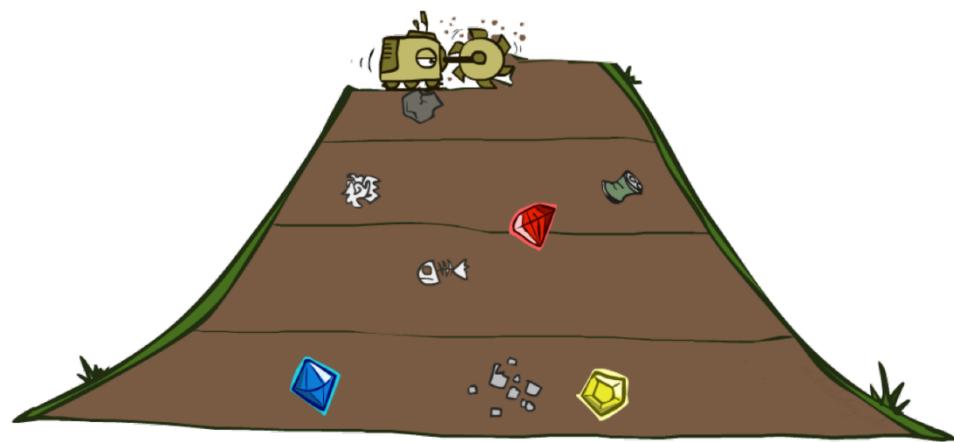
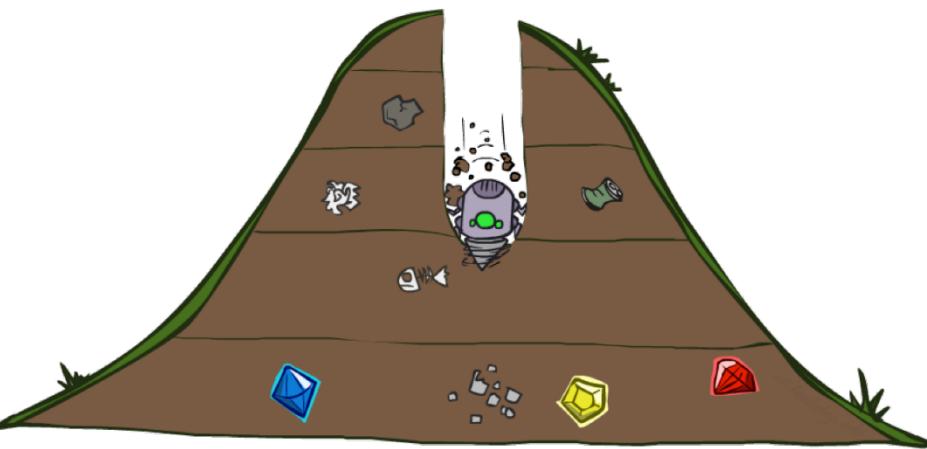


广度优先搜索属性

- 哪些节点被扩展?
 - 处理所有最浅层解以上的所有节点
 - 如果最浅解的深度为 s
 - 搜索时间 $O(b^s)$
- 前沿探索需要的存储空间
 - 大概是最后一层, 所以 $O(b^s)$
- 完全性?
 - 如果一个解存在, s 一定是有有限的, 所以是完全的!
- 优化性?
 - 是, 如果所有步骤成本都是1时



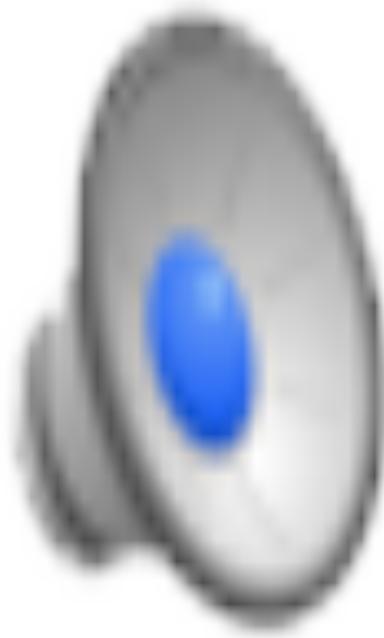
问题：DFS vs BFS



问题：深度优先搜索 vs 广度优先搜索

- 什么时候？广度优先搜索 优于 深度优先搜索
- 什么时候？深度优先搜索 优于 广度优先搜索

演示视屏：迷宫 DFS/BFS (1)

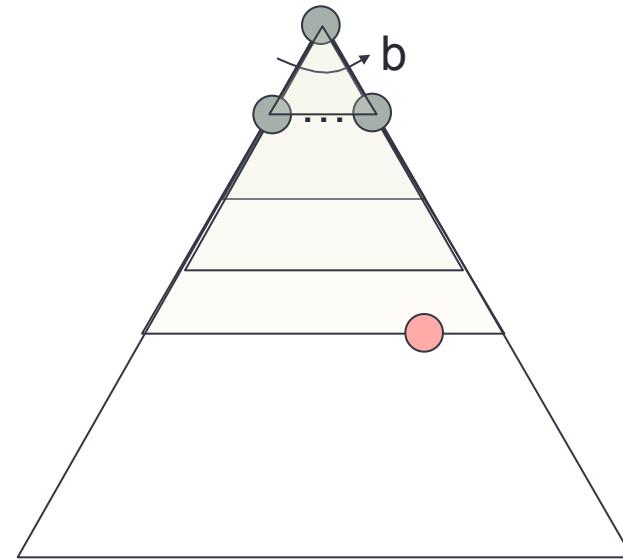


演示视屏：迷宫 DFS/BFS (2)

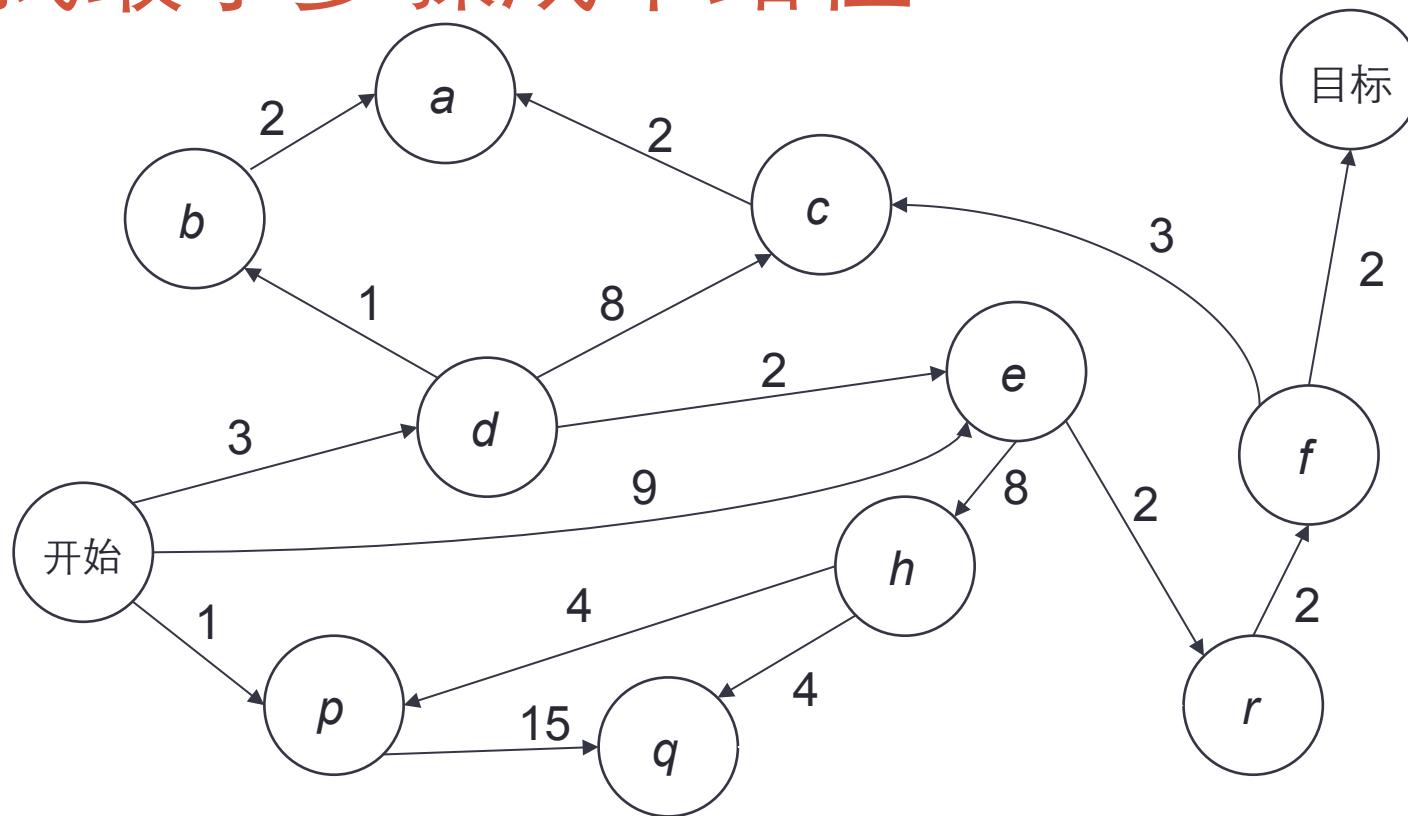


迭代加深

- 想法: 结合DFS的空间优势, 和BFS的时间/搜寻浅解的优势
 - 运行一次深度限制为1的DFS,如果没有解, ...
 - 运行一次深度限制为2的DFS,如果没有解, 继续
 - 运行一次深度限制为3的DFS,如果没有解, ...
- 难道重复搜索不浪费吗?
 - 多数重复搜索集中在浅层, 节点数相对少, 所以还可以。

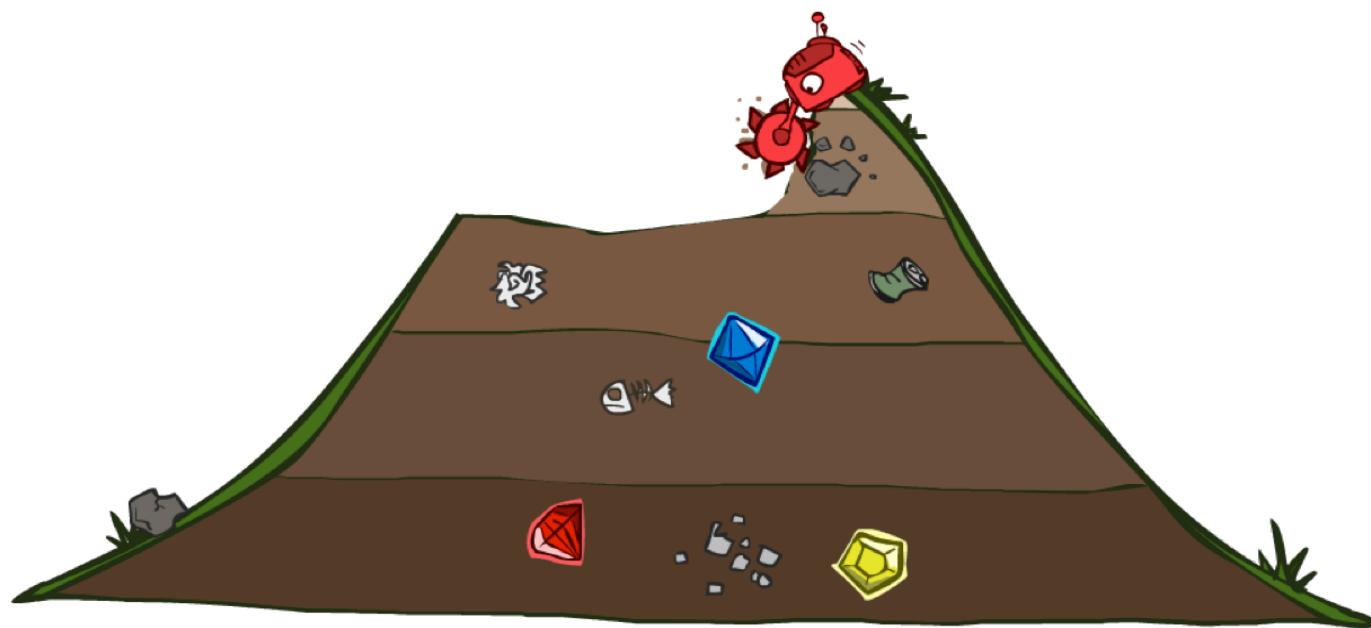


寻找最小步骤成本路径



- BFS找到的是最少行动数量的路径，而不是最小步骤成本路径。

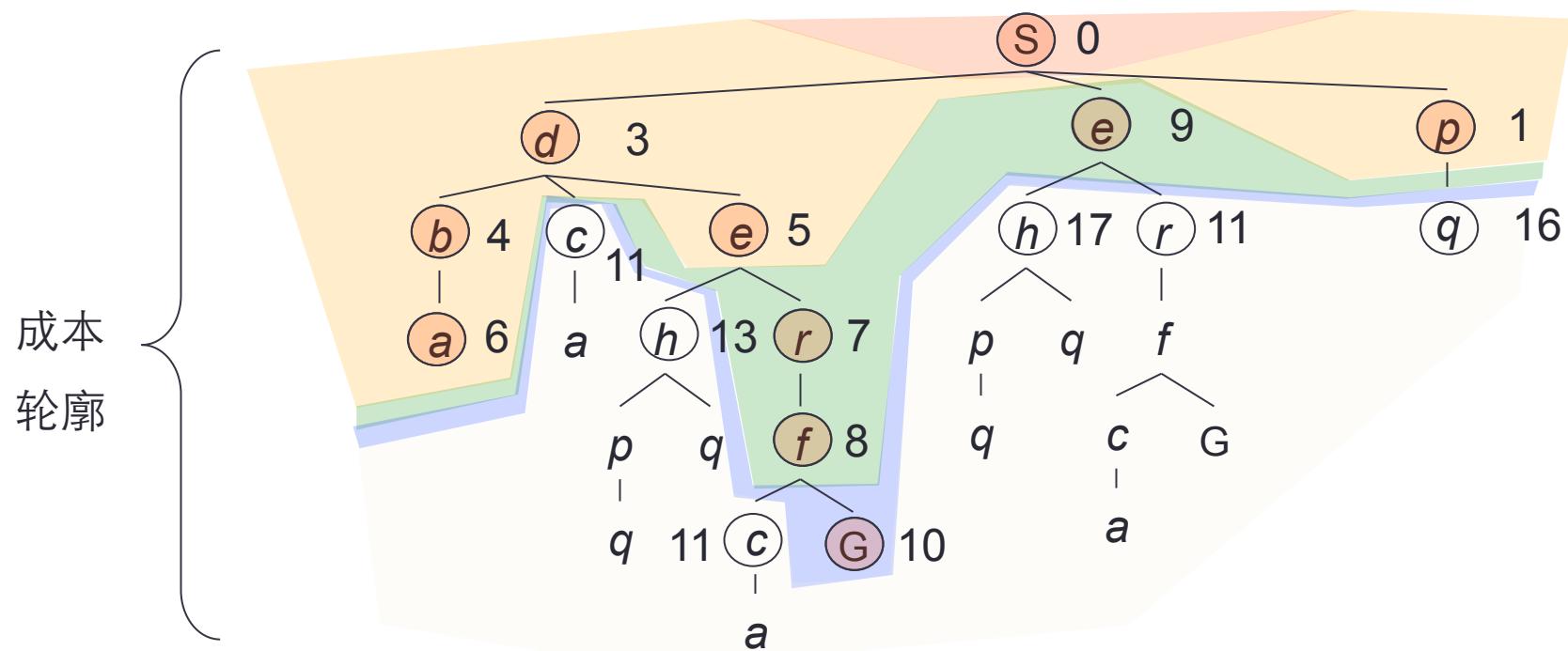
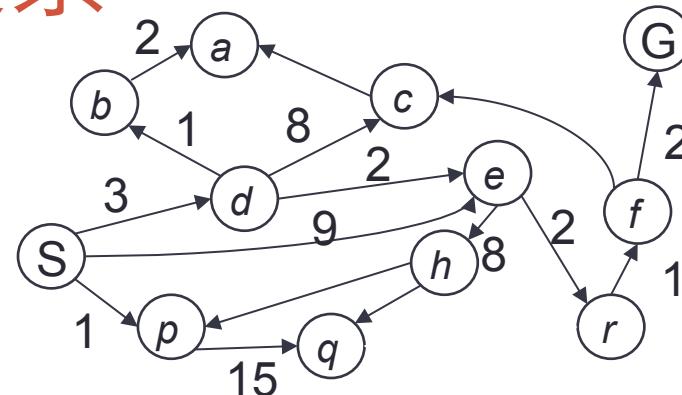
基于成本的统一搜索 (Uniform Cost Search)



基于成本的统一搜索

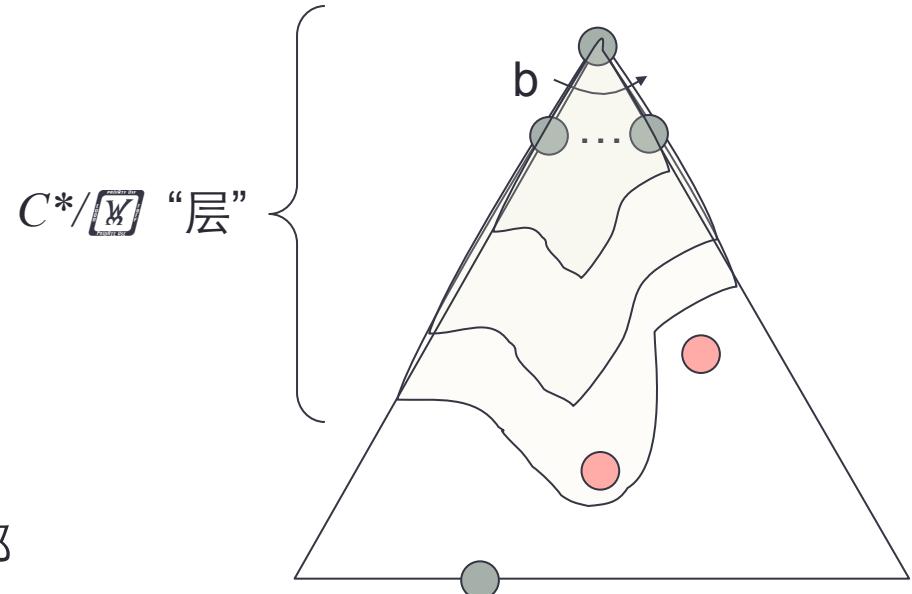
策略：首先扩展一个成本最低的节点

前沿用优先级队列存储
(优先级：累计成本)



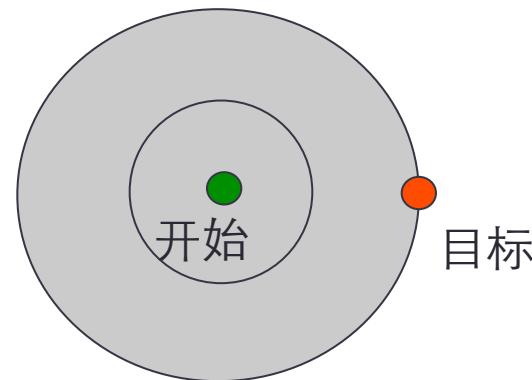
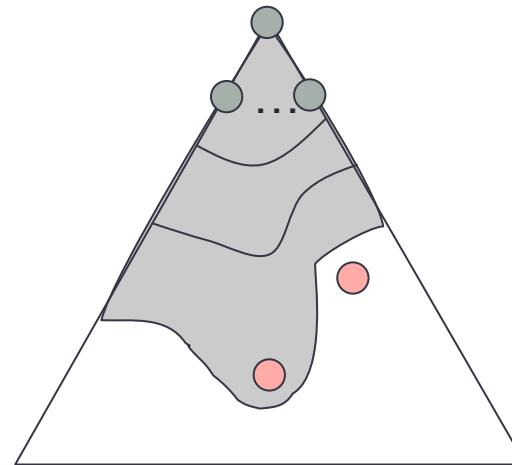
基于成本的统一搜索 (UCS) 属性

- 哪些节点被扩展?
 - 处理所有累计成本小于最低成本解的所有节点
 - 如果那个解的成本是 C^* ，并且步骤成本至少是 $\frac{1}{b}$ ，那么其“有效深度”大概是 $C^*/\frac{1}{b}$
 - 时间花费 $O(b^{C^*/\frac{1}{b}})$ (有效深度指数)
- 前沿节点占用多少空间?
 - 大概总是上一层，所以是 $O(b^{C^*/\frac{1}{b}})$
- 完全性?
 - 假设最优解的成本有限，步骤代价都为正数，则是！
- 优化性?
 - 是。

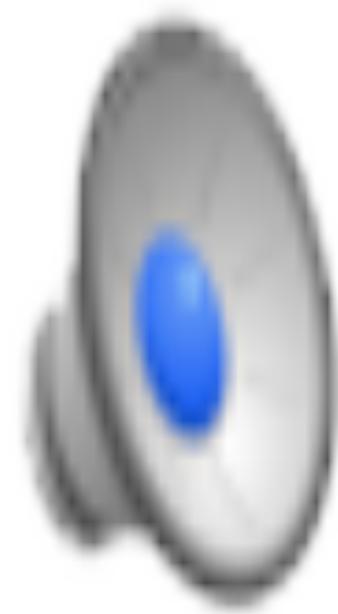


存在不足

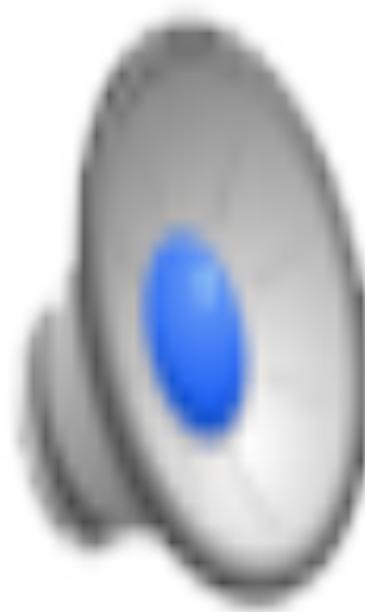
- 探索逐步提高的成本轮廓范围内的节点
- 优势: 完全性和优化性
- 不足:
- 在每个“方向”上都探索
- 没有关于目标位置的信息
- 可以补救 (以后会讲)



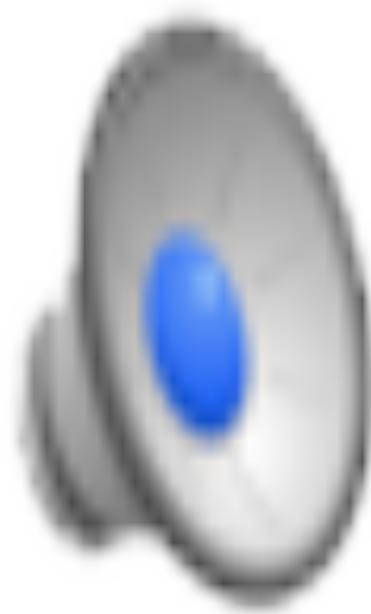
演示视频：空白UCS



演示视频：有深/浅水的迷宫 --- DFS,
BFS, or UCS ? (1)



演示视频：有深/浅水的迷宫 --- DFS,
BFS, or UCS ? (2)



演示视频：有深/浅水的迷宫 --- DFS,
BFS, or UCS ? (3)

