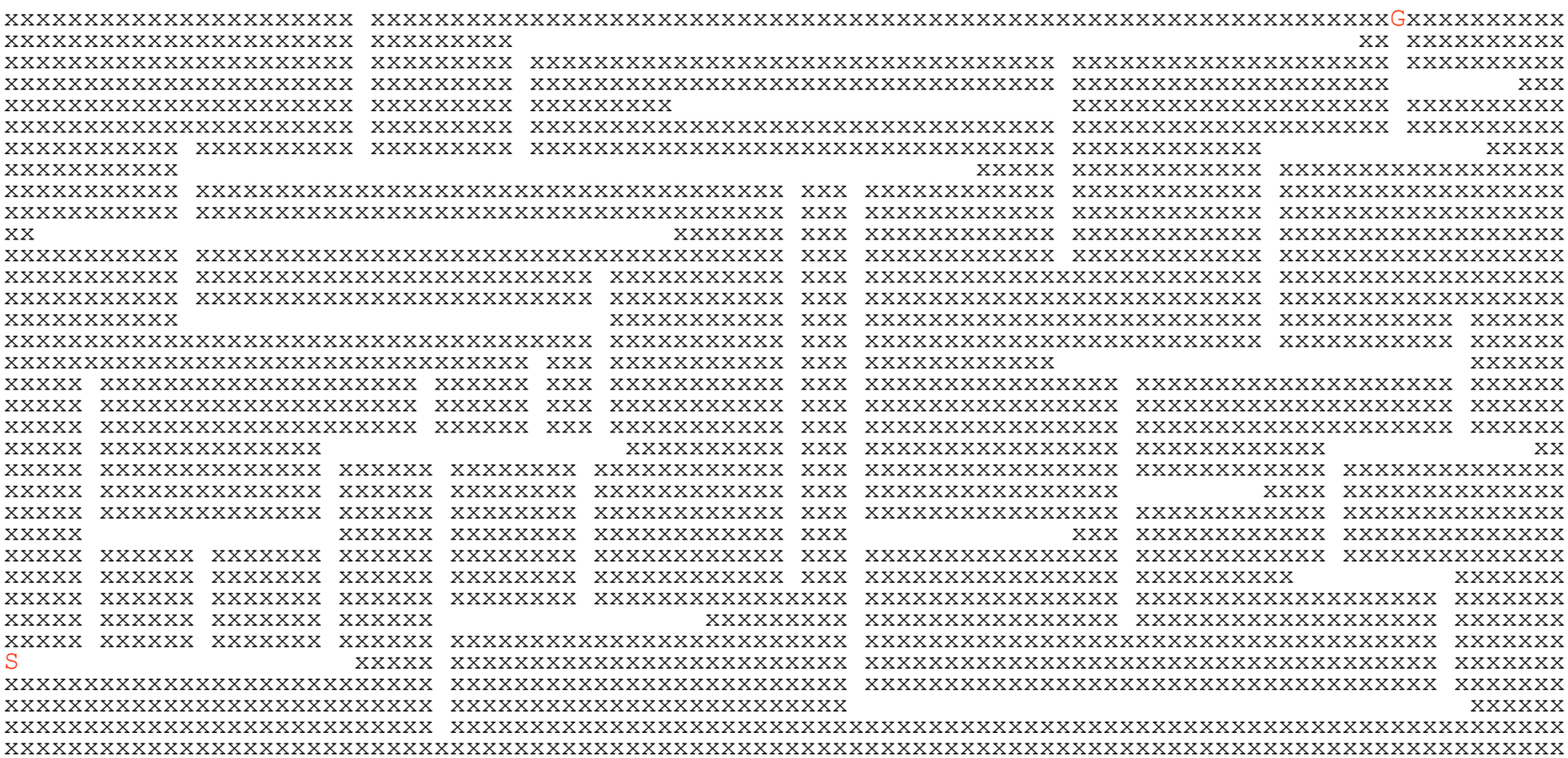
人工智能实验 1 ——迷宫寻路

UESTC · 2022 fall

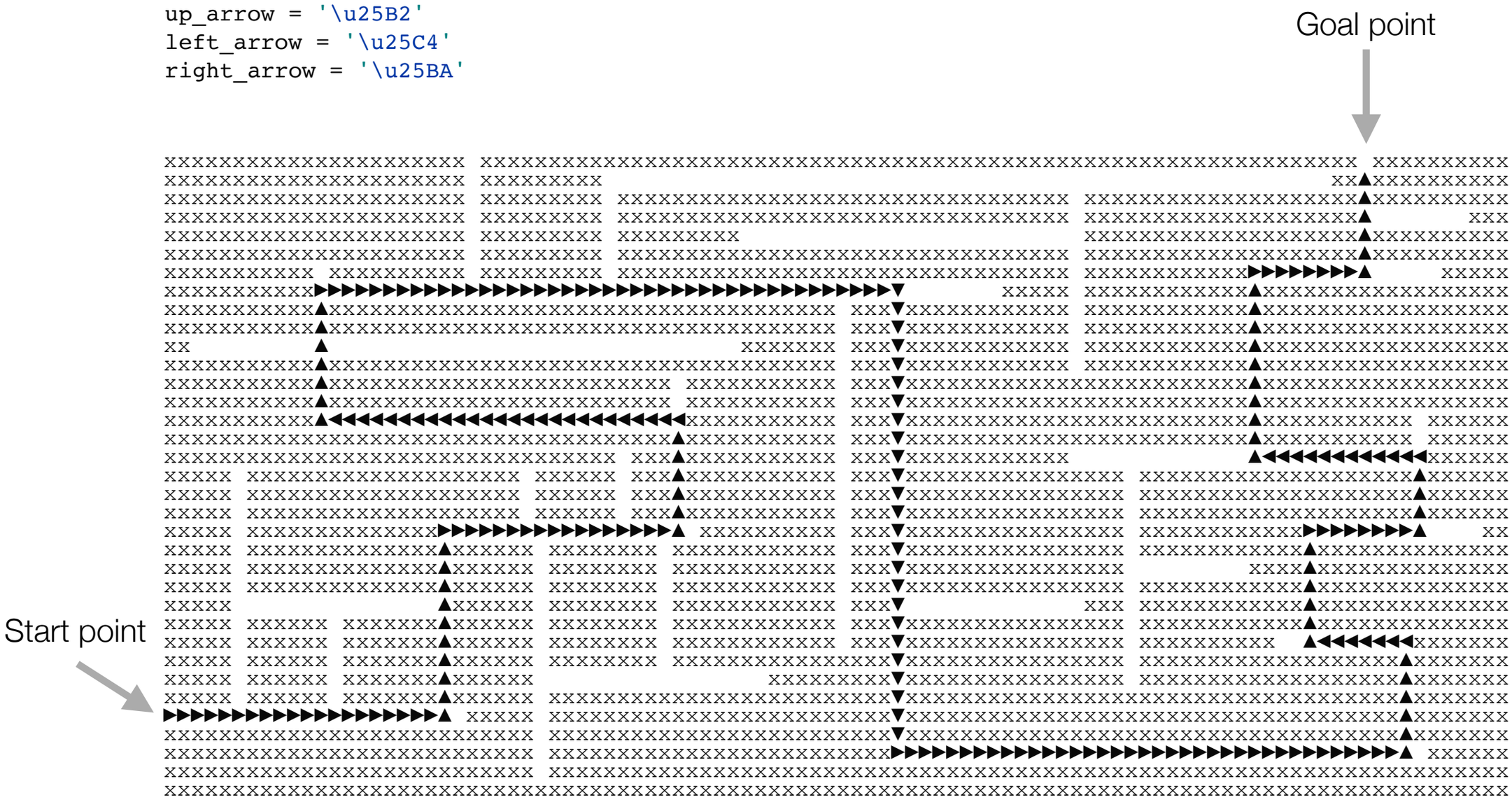
```
# 迷宫文件中，x表示墙，空格表示路
blocked_tile = 'x'
free_tile = ' '
```

Goal point



Start point

```
# 要打印到.txt文件中的Unicode字符（箭头）
down_arrow = '\u25BC'
up_arrow = '\u25B2'
left_arrow = '\u25C4'
right_arrow = '\u25BA'
```



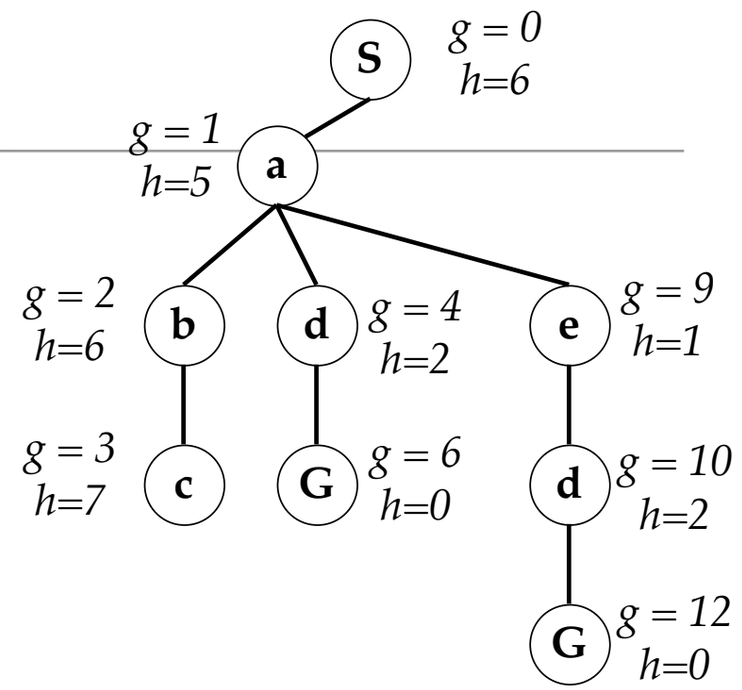
搜索问题

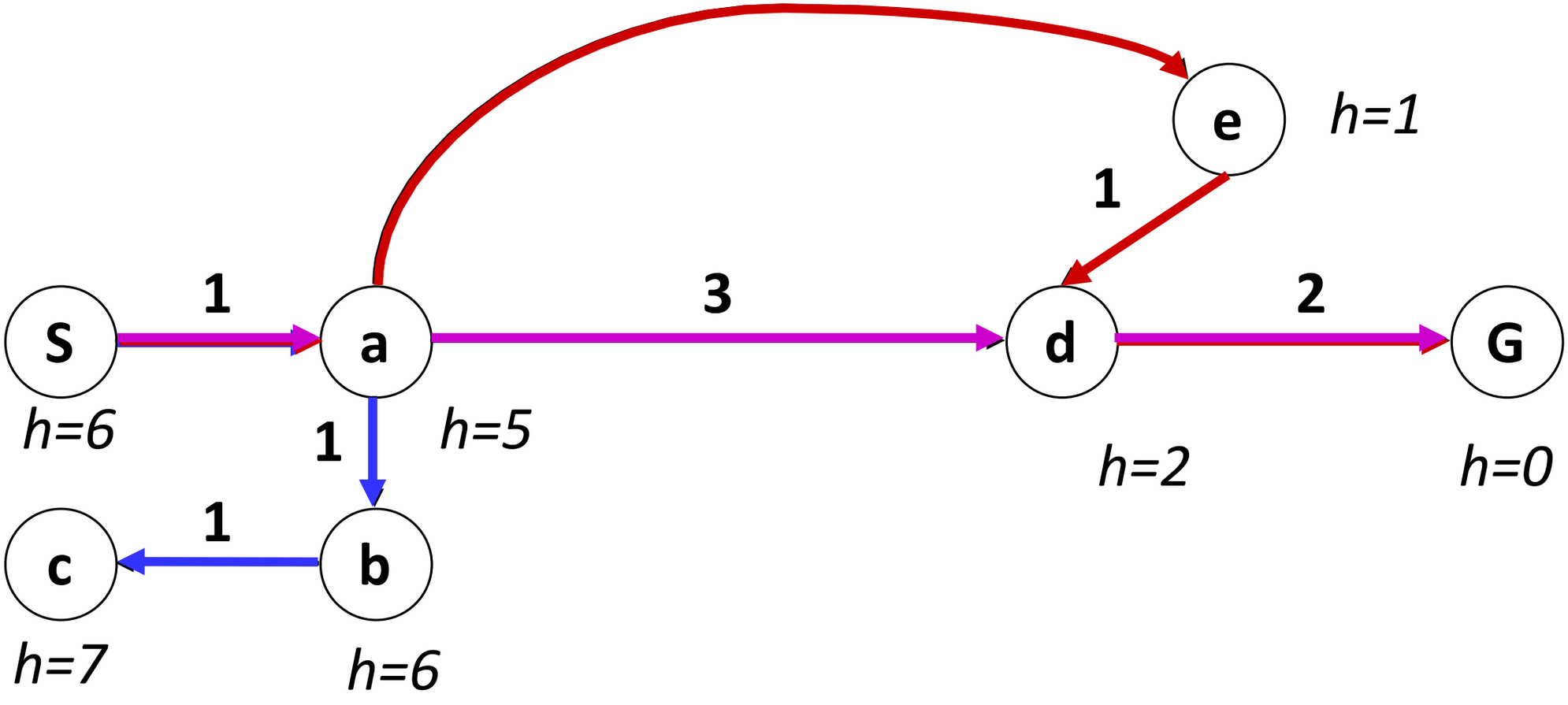
- 一个搜索问题包含：
  - 一个状态空间：当前所处的位置
  - 在每一个状态里，一个可允许的动作集合：NSEW
  - 一个转换模型：移动到下一个位置
  - 一个步骤成本函数：1
  - 一个开始状态，和一个目标到达测试：S，G

- 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

结合统一搜索和贪婪搜索

- UCS: 把路径成本排序, 即来程的成本 g(n)
- Greedy: 排序按照与目标的临近性, 即前程成本 h(n)
- A* 搜索: f(n) = g(n) + h(n)



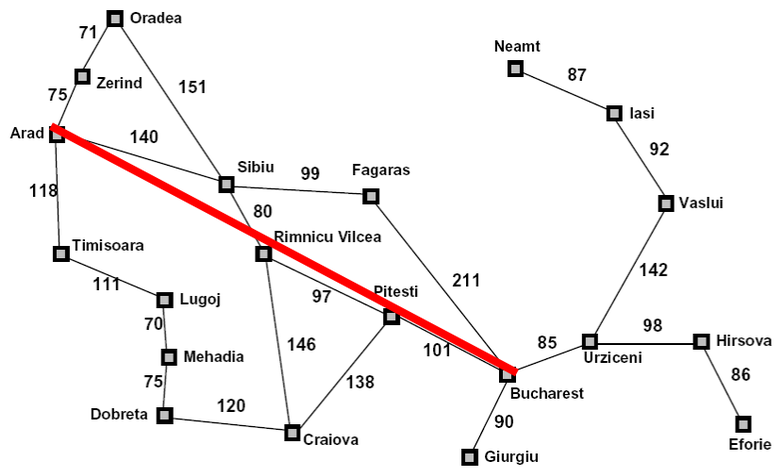


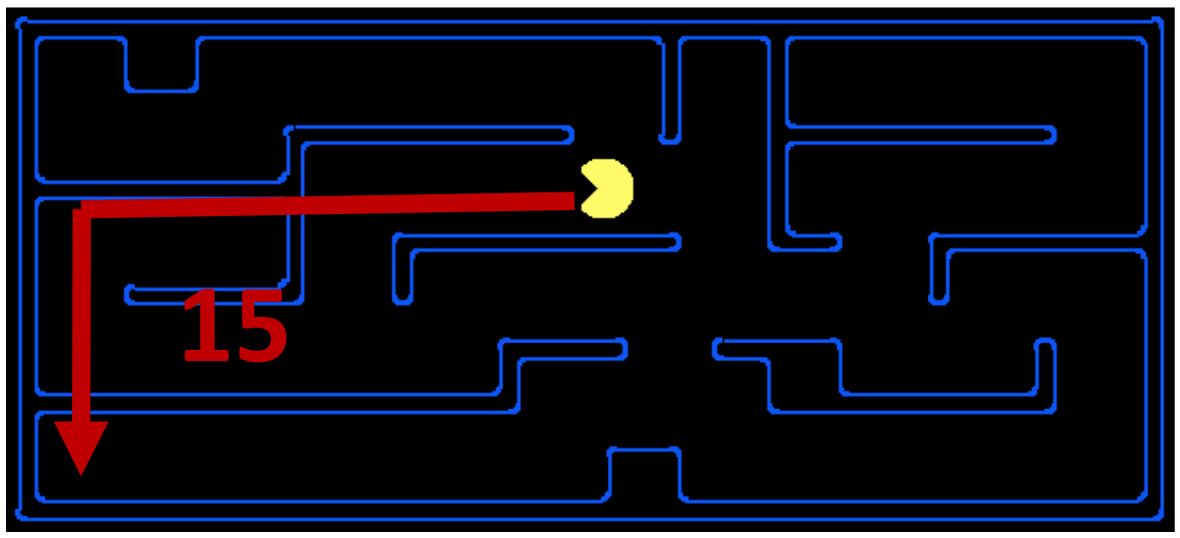
创建可接纳的启发式函数

- 在求解很难的搜索问题时，大部分的工作是找到可接纳的启发式函 数。

- 可接纳的启发式函数信息，通常是对应的松弛问题(relaxed problems)的解, 解除对行动的限制。

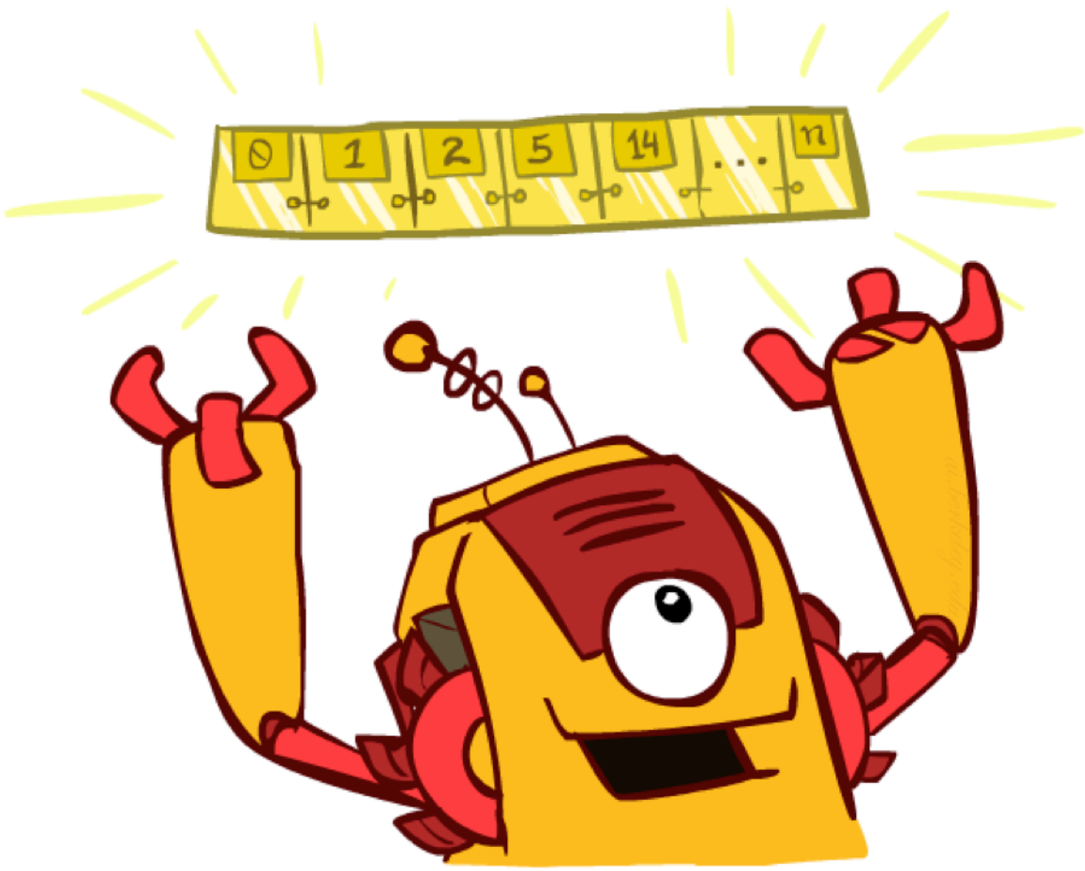
366





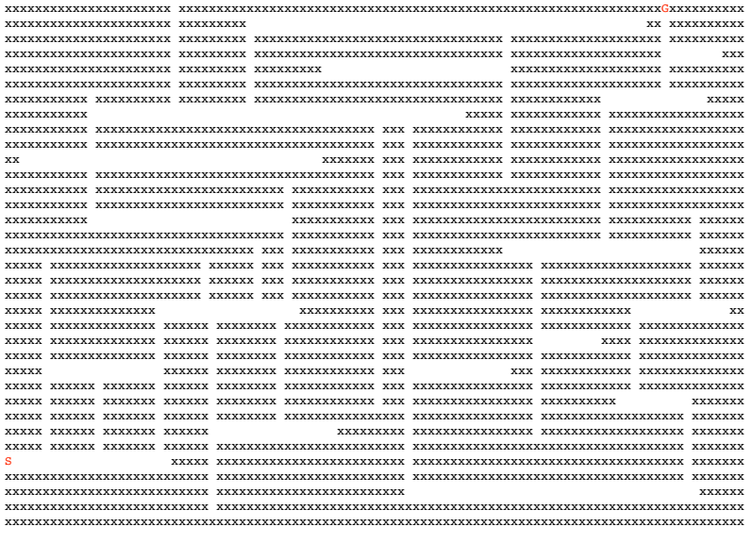
都是一个队列

- 除了边缘策略，所有这些搜索算法是相同的
  - 从概念上讲，所有的边缘都是优先队列(即集合与附加的节点优先级)
  - 实际上,DFS和BFS可以用栈和队列，避免log (n)的排序开销
  - 甚至可以同一套代码来实现，只需要一个参数来确定出队列的方式

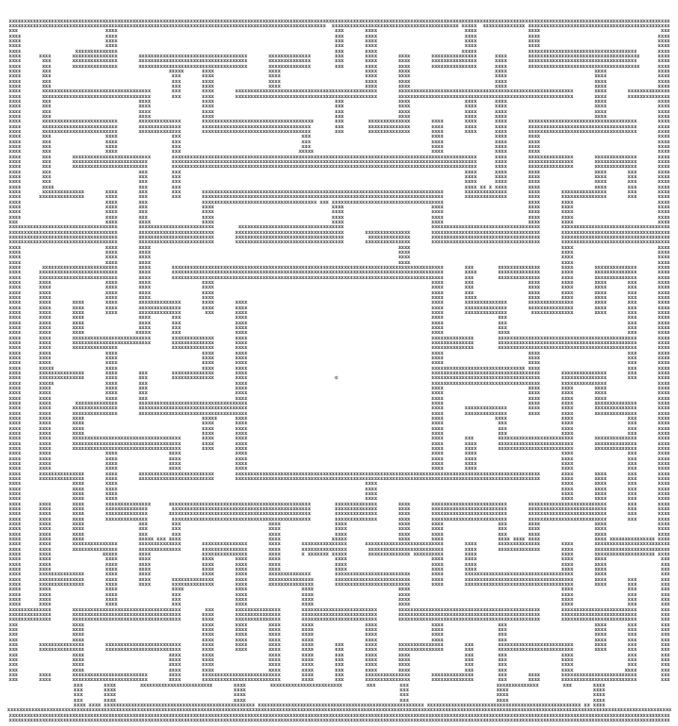


实验要求

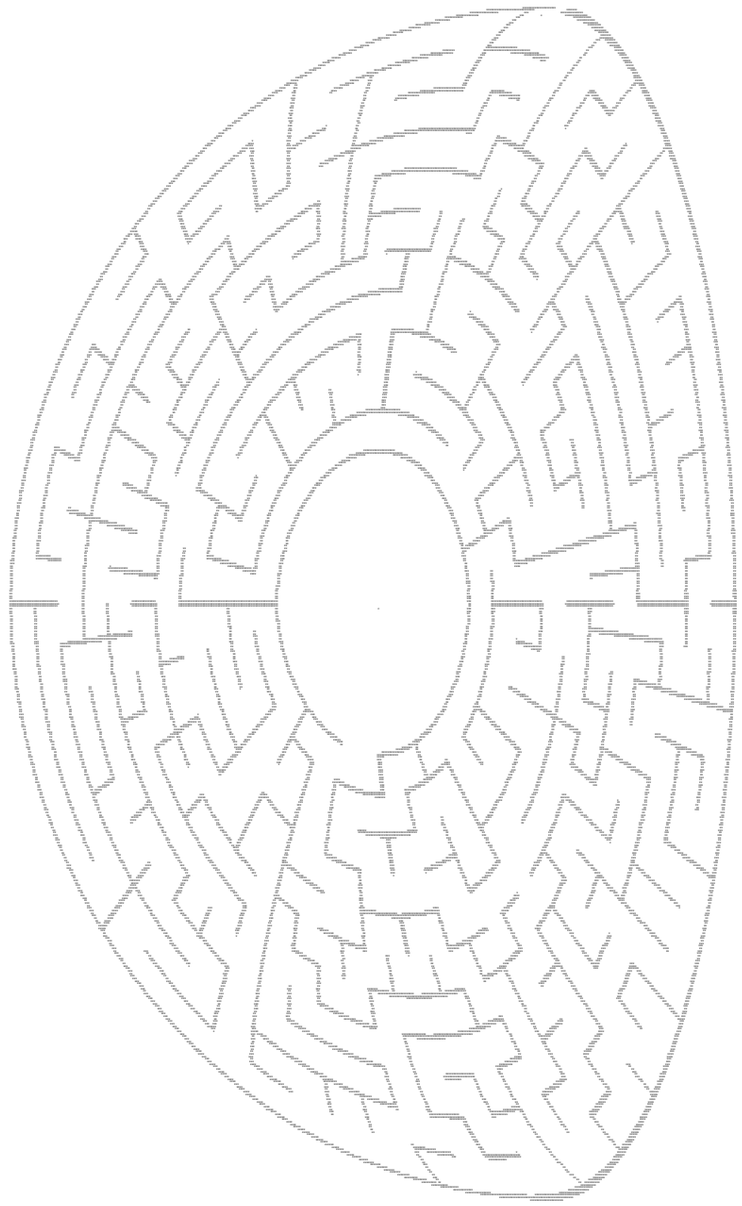
- 用A* 算法实现迷宫的寻路
- 设计合理的启发式函数
- 尽可能快地解决：



Maze



Maze_Hard



Maze_Very_Hard