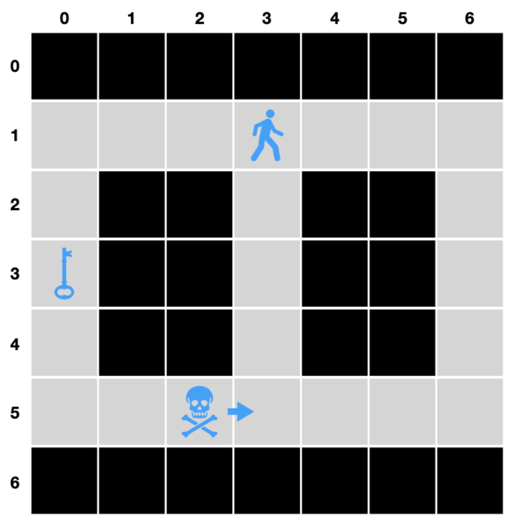
CS410: Artificial Intelligence 2020 Fall
Solutions of Homework 1: Search Algorithms
Due date: 23:59:59 (GMT +08:00), October 3 2020

Consider the following simplified version of the classic Atari video game, *Montezuma's Revenge*: It is played on the board illustrated below. An agent (represented by the person icon in cell (1,3)) wishes to grab the key (in cell (3,0)). A skull starts in cell (5,2) and moves to the right by one cell after each action is executed until it ends up in the rightmost cell, at which point it starts moving to the left, and repeats this pattern back and forth.

The agent can be facing either left or right. There are 10 possible actions for the agent: 2 turning actions (*turn left*,*turn right*) and 8 moving actions (*left*,*right*,*up*,*down*,*left up*,*left down*, *right up*,*right down*). The agent can move up or down while facing either direction, but can move sideways or diagonally only if facing in that direction. For example, if the agent is facing right but tries to move left up, the agent will not move and nothing will happen. Furthermore, if the agent is already facing left and a turn left action is taken, nothing happens.

Lastly, the agent cannot move into a cell **currently occupied** by the skull, or a wall.



1. Answer the following questions for the Montezuma's revenge board above:

(1.a) Let $N$ be the number of possible cell locations that the agent can be in, and let $M$ be the number of possible cell locations that the skull can be in. Recall that for "pacman pathing", the representation of the state was $(x, y)$ where $x$ was the row and $y$ was the column of pacman's position. Describe a representation of a state in the state space for this game and give an expression for the size of the state space.

Representation of the state space:
States: ($x_{\text{agent}}$, $y_{\text{agent}}$, facing_direction of the agent, $x_{\text{skull}}$, $y_{\text{skull}}$ )
Actions: move up, down, left, right, right up, right down, left up, left down; turn left, turn right
Successor: update location ($x_{\text{agent}}$, $y_{\text{agent}}$) and facing direction
Goal test: is ($x_{\text{agent}}$, $y_{\text{agent}}$)=key location(3,0)?

Size of the state space:
$2NM$

Explanation of each term in the size of the state space:
Agent positions $N$: 23 ;
Skull positions $M$: 7;
Agent facing direction: 2

(1.b) Please fill in the following pseudocode for the **getSuccessor** function for this game:

**Algorithm 1** getSuccessor(state)

```
1:  successors ← empty list
2:  for action ∈ left, left_up, left_down, right, right_up, right_down, up, down, turn_left, turn_right do
3:      if action ∈ {left,left_up,left_down} and state.facing_direction == right then
4:          continue
5:      end if
6:      if action ∈ {right,right_up,right_down} and state.facing_direction == left then
7:          continue
8:      end if
9:      if action ∈ {turn_right} and state.facing_direction == right then
10:         continue
11:     end if
12:     if action ∈ {turn_left} and state.facing_direction == left then
13:         continue
14:     end if
15:     next_state ← state.apply_action(action)
16:     if next_state not out of bound and next_state is not occupied by the wall and next_state is not occupied by the skull then
17:         successors.append(next_state)
18:     end if
19: end for
20: return successors
```

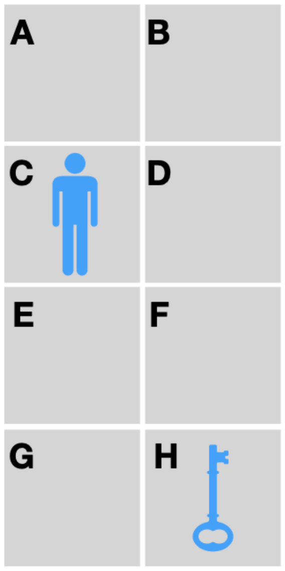
(1.c) What is the goal test?
Is $(x_{\text{agent}}, y_{\text{agent}})$=key location(3,0)?

2. Now, consider a simplified version of the board below, which has **no skull** and **no facing-direction for the agent** (i.e., the agent can move in any of the 8 directions as long as it remains in the board). For the four following graph search algorithms, perform the search procedure yourself and provide answers to the questions below regarding the nodes expanded during the search as well as the final path found by the algorithm.

On this board, assume that a diagonal move has a cost of 3, whereas moving left, right, up, or down has a cost of 1. Do notice the difference in costs, and recall which algorithms use this cost information and which algorithms do not.

Remember that the search procedure should begin at the agent's starting position (C). To break ties when adding nodes of equal cost to the fringe, follow alphabetical order.

Finally, when listing the order/number of nodes expanded, do not include nodes which are taken off the fringe but discarded immediately due to already having been visited.



(2.a) **Breadth first graph search** Recall that BFS computes the smallest number of steps, $b(v)$, taken to get to a node $v$ from the start node.

Order of nodes expanded:
C ABDEF GH

Number of nodes expanded:
8

Path returned:
CEH

Length of path:
2

Cost of path:
4

Please fill in the form below.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| $b(s)$ | 1 | 1 | 0 | 1 | 1 | 1 | 2 | 2 |
| length from $C$ to $s$ | 1 | 1 | 0 | 1 | 1 | 1 | 2 | 2 |
| cost from $C$ to $s$ | 1 | 3 | 0 | 1 | 1 | 3 | 2 | 4 |

(2.b) **Depth first graph search**

Order of nodes expanded:
CABDEFGH

Number of nodes expanded:
8

Path returned:
CABDEFGH

Length of path:
7

Cost of path:
11

Please fill in the form below.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| length from $C$ to $s$ | 1 | 2 | 0 | 3 | 4 | 5 | 6 | 7 |
| cost from $C$ to $s$ | 1 | 2 | 0 | 3 | 6 | 7 | 10 | 11 |

(2.c) **Uniform cost graph search**
Recall that UCS keeps track of the lowest cost, $c(v)$, to get from the start node to the node $v$.
Order of nodes expanded:
CADEBFGH

Number of nodes expanded:
8

Path returned:
CDFH

Length of path:
3

Cost of path:
3

What is $c(A), c(B), \ldots c(H)$?

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| $c(s)$ | 1 | 2 | 0 | 1 | 1 | 2 | 2 | 3 |

(2.d) **A* graph search (with Manhattan distance to the goal as the heuristic)** Recall that A* computes $f(v)$ for the nodes $v$ that it expands, with $f(v) = c(v) + h(v)$ where $c(v)$ is the lowest cost to reach $v$ from the start node and $h(v)$ is an estimate of the distance from $v$ to the goal.
Order of nodes expanded during the search:
CDEFGH

Number of nodes expanded during the search:
6

Path returned by the search:
CDFH

Length of path returned by the search:
3

Cost of path returned by the search:
3

What is $f(A), f(B), \ldots f(H)$? Note that here, we are asking for the true $f(v)$ values as dictated by the definition, which is the value populated by the search algorithm only if it were to expand every node. This particular search problem doesn't end up expanding all nodes, so the $f(v)$ estimate maintained by the algorithm on the queue is not the true $f(v)$ value that we're asking for. Hint: you can fill out these values directly by looking at the board.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| $f(s)$ | 5 | 5 | 3 | 3 | 3 | 3 | 3 | 3 |

3. Given your answers above, what are the qualitative differences between the results achieved by BFS, DFS, UCS, and A*? Which one finds the shortest path (in number of steps)? Which one finds the optimal path (in cost)?

BFS expands the node layer by layer, it selects the shallowest node for expansion and thus can always find the shortest path to the target node. DFS goes first with depth, which means it will focus on one direction until there is no node to expand in this direction, this may cause longer path to the target. UCS expands the node with the least cost but not consider the information of the path, it could find the optimal path at the end though may with slower speed. A* expands nodes with smallest heuristic target, which considers both the total backward cost used by the UCS and the estimated forward cost used by greedy search, thus it could move towards the most possible/shortest direction to find the optimal path.

In this case, BFS finds the shortest path with smallest steps. A* and UCS finds the optimal path in cost.

4. For the same board and setting as part 2, give an example for each of the following types of heuristics. Please briefly explain why the example you chose satisfies the requested properties. Solutions are open.

(4.a) Admissible and consistent. Note: You can use a heuristic that we have frequently used in this class, or you can just assign any numbers that qualify as an admissible and consistent heuristic.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| Heuristic $h(s)$ | | | | | | | | |

Explanation:

(4.b) Admissible but inconsistent.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| Heuristic $h(s)$ | | | | | | | | |

Explanation:

(4.c) Inadmissible and inconsistent.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| Heuristic $h(s)$ | | | | | | | | |

Explanation:

5. In the previous questions, perhaps you used "relaxed" heuristics; that is, you estimated the true cost of something by evaluating the cost for a simpler version of the problem (which is easier to compute). For example, using euclidean distance would be a "relaxed" heuristic to estimate the length of the shortest path from Arad to Bucharest in Romania.

Formally, we will define a **relaxed heuristic** as a function on a state that returns a value that is always admissible and consistent. In this problem, we will consider two changes ("skull" and "teleportation") to the board/game above, and we will reason about the effect of these changes on the consistency of heuristics.

(5.a) For this new version of the game, your friend Nancy suggests taking the old game setting from part 2. and now adding the ability for the agent to perform a maximum of 1 "teleportation" action during the game. That is, on one of the agent's moves, it can choose to jump from its current state to any other non-goal state on the board.

(5.a.i) How does this new teleportation ability change the state space of the game from part 2. , which was $(x, y)$? Does anything need to be added or removed?
Whether the agent has teleported needs to be recorded.

(5.a.ii) Nancy argues that in this new game, at least one previously consistent heuristic can become inconsistent. Is Nancy right?

- ◯ Yes, and I will give an example below. yes
- ◯ No, and I will provide a proof below.

**Note**: we define heuristics for this problem as being a function of **only** the cell location: They cannot incorporate anything that did not exist in the old version of the game that we are comparing to.

If you believe Nancy is right, give an example of a heuristic that used to be consistent in the old game but is no longer consistent in this new game. Make sure to explain why it is no longer consistent (perhaps with a drawing of a board state and an explanation). The following could be an example which is inconsistent in this case but consistent in previous game.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| $h(s)$ | 4 | 3 | 3 | 2 | 2 | 1 | 1 | 0 |

If you believe Nancy is wrong, provide an argument for why a heuristic that was consistent in the old game must also remain consistent in this new game. Be specific about your reasoning and use mathematical quantities such as heuristic costs of states $h(a)$ and true costs of transitions $c(ab)$.

(5.b) For this new version of the board, your friend Ethan suggests adding the skull back to the old board setting from part 2. , and having the skull move back and forth between the cells E and F.

(5.b.i) How does the presence of this skull change the state space of the game from part 2. , which was $(x, y)$? Does anything need to be added or removed?
The position of skull needs to be recorded.

(5.b.ii) Ethan argues that in this new board, at least one previously consistent heuristic can become inconsistent. Is Ethan right?

◯ Yes, and I will give an example below.

◯ No, and I will provide a proof below. no

**Note**: we define heuristics for this problem as being a function of **only** the cell location: They cannot incorporate the location of the skull, since that did not exist in the old version of the board that we are comparing to.

If you believe Ethan is right, give an example of a heuristic that used to be consistent on the old board but is no longer consistent on this new board. Make sure to explain why it is no longer consistent (perhaps with a drawing of a board state and an explanation.

| State $s$ | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| $h(s)$ | | | | | | | | |

If you believe Ethan is wrong, provide an argument for why a heuristic that was consistent in the old board must also remain consistent in this new board. Be specific about your reasoning and use mathematical quantities such as heuristic costs of states $h(a)$ and true costs of transitions $c(ab)$.

The main idea is that the cost between any two nodes in current version is always larger than that in previous version.