Homework 2
Computer Science
B551 Spring 2018
Hasan Kurban

Jinju Jiang(Hellen)

February 23, 2018

Introduction

The aim of this homework is to get you acquianted with problem solving and the steps (Real World → Concept → Logic → Implementation). You will turn-in four files

- A *pdf with the written answers called `hw2.pdf`
- A Python script called `rv1.py`
- A Python script called `rv2.py`
- A Python script called `rpsg.py` for rock-paper-scissors.

I am providing this LaTeX document for you to freely use as well. Please enjoy this homework and ask yourself what interests you and then how can you add that interest to it! Finally, questions 4 and 5 are worth 50 points each whereas questions 1,2 and 3 are 20 worth points each.

Homework Questions

1. Problem 3.10 (p. 115) in the text.
   Answer to problem 1:
   A state is a situation/configuration which a process can take.For example,In games, a state is any one set of all possible configurations within the game.
   a state space is the set of values which a process can take. For example,In games, the state space is the set of all possible configurations within the game. Notated the state space by graph, then it is a graph whose nodes are the set of all states,linked with actions that transform one state into another.
   A search tree is a tree structure which is a hierarchy of linked nodes where each node represents a particular state. The link can be directed link or undirected link. The link between nodes is the action which can be taken to reach to another node from one node.Normally the root node is the start state or initial state. **A search node** is a node in the search tree.
   A goal is a state that the agent/process is trying to reach.
   An action is something that the agent/process can choose to do.For example, the available actions for a robot is moving forward, turning to East,South,West or North. **A successor function** is a description of possible actions, a set of operators. It is a transformation function on a state representation, which convert it into another state.The successor function defines a relation of accessibility among states.
   The branching factor in a search tree is the number of childrens to each parent node. It represents how many actions the process/agent can do at each node.

2. Problem 3.18 (p. 117) in the text.
   Answer to problem 2:
   The worst case for iterative deepen search would be there is only on child for each node as the following figure: the total cost is $1 + 2 + 3 + ... + n = n(n+1)/2 = O(n^2)$

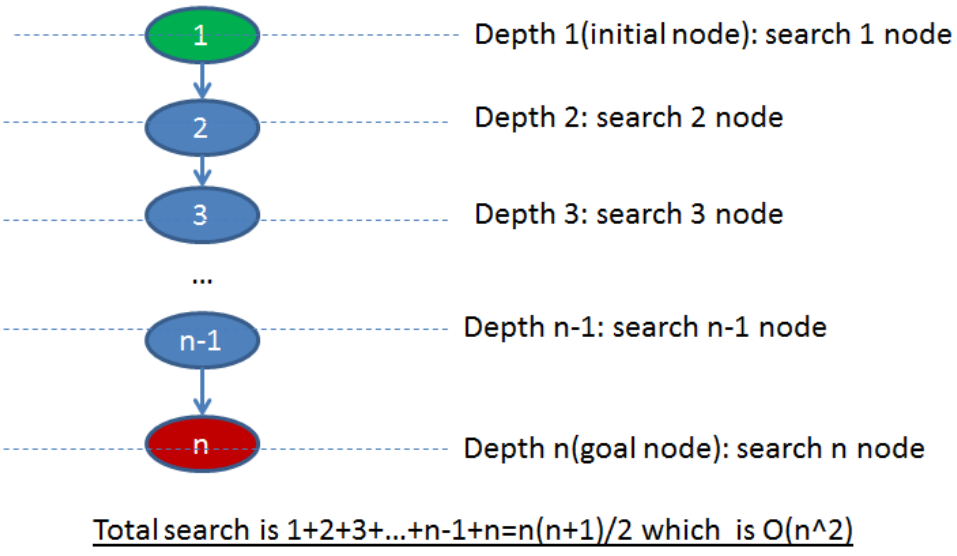


Figure 1: Illustrate the worst case for iterative deepen search

3. The text (page 95) describes consistency as:

$$h(n) \quad \leq c(n, a, n') + h(n')$$

for state n, its successor n' and action a. For $G = (\{A, B, C\}, \{(A, B), (A, C), (B, C)\})$, $Cost =$

$\{((A,B),2),((A,C),5),((B,C),1)\}$, and $h(A)=1, h(B)=4, h(C)=3$. Is this consistent?
Answer to problem 3: Yes, it is consistent, please see below figure for proof:

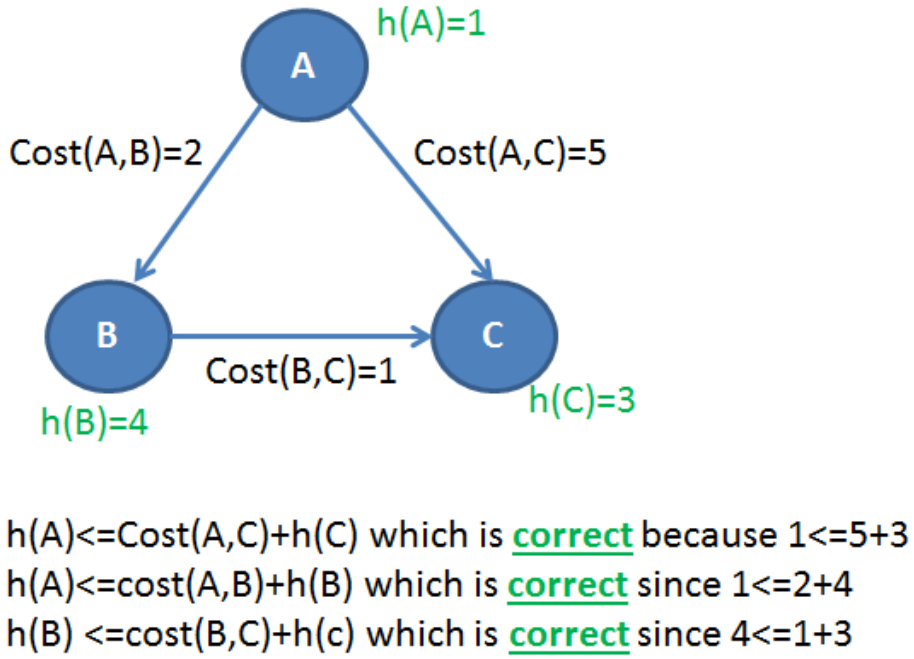
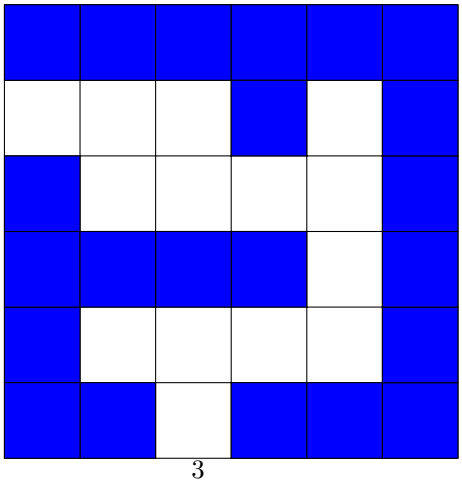


Figure 2: Illustrate the consistent heuristic function

4. Assume you're programming a robot named R to navigate a 2D surface. The robot can only move forward a single step to an adjacent square (not diagonally), but can also rotate ± 90 degrees. R has a single sensor on its front that determines if there is an obstruction, perhaps a wall, is in its path. Your task is to read in a 2D plan and starting at location from the southmost (bottom) side, navigate to another side. The plan below has an opening at (3,1). *One* path is: (3,1), (3,2), ..., (3,5), (2,5), (1,5). If R is at (4,2) facing north, then its sensor would return 1. If R is at (4,2) and facing east, its sensor would return 0. If R is at (2,2) facing west, to move to (3,2), rotate(90), rotate(90), step. You can *start* R on any available open square on the bottom – you'll have to decide what direction R is facing. The plan is encoded as an array of ones and zeros. The plan below:



would be encoded as:

```
111111
000100
100001
111101
100001
110111
```

(a) Given a floor plan `f.txt` (read in the file), return True and the series of instructions needed to navigate R if there is a path and False otherwise. Name this program `rv1.py`.
Answer to question 4(a):
Please check *hw2_rv1.py* for full code. I am using DFS search to get the path, and then using customized function *navigation_path* to print out the instructions as the following:

```
Here it is the shortest path by DFS search:
move to: (2, 0) facing to:  N
move to: (2, 1) facing to:  N
rotate+90
move to: (3, 1) facing to:  E
move to: (4, 1) facing to:  E
rotate-90
move to: (4, 2) facing to:  N
move to: (4, 3) facing to:  N
rotate-90
move to: (3, 3) facing to:  W
move to: (2, 3) facing to:  W
rotate+90
move to: (2, 4) facing to:  N
rotate-90
move to: (1, 4) facing to:  W
move to: (0, 4) facing to:  W
```

(b) Improve R's programming by returning the *shortest* path if it exists. Name this program `rv2.py`.
Answer to question 4(b):
Using A start search(optimized by heuristic function) and print the navigation for robot. Please check the full code in *hw2_rv2.py*.And here it is the running results:

```
Here it is the shortest path by A* search:
step to: (2, 0) facing to:  N
step to: (2, 1) facing to:  N
rotate+90
step to: (3, 1) facing to:  E
step to: (4, 1) facing to:  E
rotate-90
step to: (4, 2) facing to:  N
step to: (4, 3) facing to:  N
rotate-90
step to: (3, 3) facing to:  W
step to: (2, 3) facing to:  W
rotate+90
step to: (2, 4) facing to:  N
rotate-90
step to: (1, 4) facing to:  W
move to: (0, 4) facing to:  W
```

(c) Discuss your search techniques in both solutions. State explicitly your $\hat{h}, \hat{g}, \hat{f}$.
Answer to question 4(c):
My heuristic function is manhattan distance from the state to goal state. the actual cost is the movement cost: for 1 step is 10, for rotation, the actual cost is 0. the total cost is the actual cost plus heuristic cost. so $\hat{f} = \hat{h} + \hat{g}$.
$\hat{h} = |x_0 - x| + |y_0 - y|$, (x_0, y_0) is goal cell coordination, (x, y) is the current cell
$\hat{g_1} = \hat{g_0} + 10$, $\hat{g_0}$ is previous actual cost, $\hat{g_1}$ is the actual cost after one step.
the rotation cost(+90 or -90) is zero.

5. Extend Rock/Paper/Scissors from the last assignment that has the computer playing a human. You'll additionally have $100 dollars worth of $1 chips. *Before* you show your selection, you must place a wager (at least $1). Keep the computer's strategy uniform and independent for both how it plays and how it bets. The maximum amount of chips that can be wagered is $\min\{c, h\}$ where c, h are the counts of computer and human chips respectively. Compare this R/P/S with your earlier version and discuss. Name this program `rpsg.py`.
Answer to question 5:

- for each game round, Robby and computer will play 50 times or lose all chips(100 chips),then that round game over
- Robby and computer played 50 the above game rounds
- Robby and computer will show chips randomly
- Robby will record the historic game records, he always show the opposite of the most common cards(R,P,S) which computer already did.
- The game results is exported to csv file(out.csv)
- After analyzing game results, we found that Robby has 5039 chips, which is more than computers chips(4961).
- By comparing chips for each round, Robby won 25 round games. But if compared with games quantity, Robby won 24 round games
- Win games quantity does not mean winning chips.
- the full code is in *hw2_rpsg.py*, and the output csv file is in out.csv

Here it is another strategy:

- Robby uses very conservative strategy to show chips: he only show 1 chip every round.
- Other strategy is the same as before. By using this strategy.
- obby also win during the the test (I ran the program 2 times(every time 50 rimes 50 rounds game).
- The output files are in out1.csv and out2.csv
- Full code is in *hw2_rpsg_v2.py*,