

Homework 1
Computer Science
B351 Spring 2017
Prof. M.M. Dalkilic

Mange Chen

January 23, 2017

All the work herein is mine.

Introduction

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

- A *.pdf with the written answers called `h1.pdf`
- A Python script called `p1.py` for the drone scanning the floor plan
- A Python script called `p2.py` for rock-paper-scissors

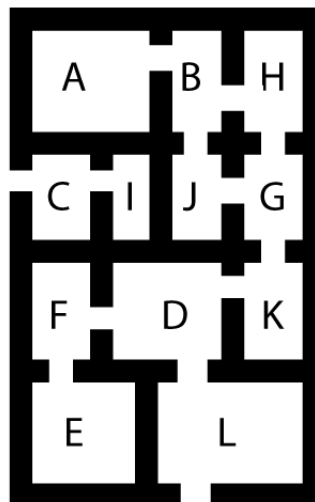
. If you've attempted extra credit, add the comment `#ExtraCredit` to the programs and `ExtraCredit` to the homework near the top so it's visible and obvious. I am providing this L^AT_EX document for you to freely use as well. Please enjoy this homework and ask yourself what interests you and then how can you add that interest to it! Finally, each homework question is worth 100 points.

Homework Questions

1. Problem 3.2 in the text.

- (a) Initial Start At(0, 0), Facing(0, 1).
 Turn(North), At(x), Facing(0, 1)
 Turn(East), At(x), Facing(1, 0)
 Turn(South) At(x), Facing(0, -1)
 Turn(West) At(x), Facing(-1, 0)
 Move(n blocks), At($x + y \min(n, D_{\max}(x, y))$), Facing(y)
 $D_{\max}(x, y)$ is the maximum distance that moving the robot in direction y from point x but not hitting a wall.
 Suppose there are S blocks, the state space is $4S$.
- (b) Move(n blocks), At($x + y \min(n, D_{\max}(x, y))$), Facing(y)
 $4I + 2(S - I)$
- (c) Move(North), At($x + (0, 1) D_{\min}(x, (0, 1))$)
 Move(South), At($x + (0, -1) D_{\min}(x, (0, -1))$)
 Move(East), At($x + (1, 0) D_{\min}(x, (1, 0))$)
 Move(West), At($x + (-1, 0) D_{\min}(x, (-1, 0))$)
 D_{\min} is the minimum distance in direction y from x. We do not need to keep track of the robot's orientation. The number of states is I.
- (d) Simplification:
 - i. The robot can feel the wall.
 - ii. The robot has and only has four direction.
 - iii. The robot knows how far it moves after moving a certain distance.

2. Define *Artificial Intelligence* as rational behavior. What is a problem that AI is *not* suited for?
 Rational behavior within *Artificial Intelligence* typically means one that maximizes its expected utility by giving certain current knowledge.
 Emotion problem is not suited for AI.
3. Assume you're programming an office security drone who's mission is to move through a floor, checking for any movement. You've been given the floor plan shown below. There are two entrances to the floor: L, C.



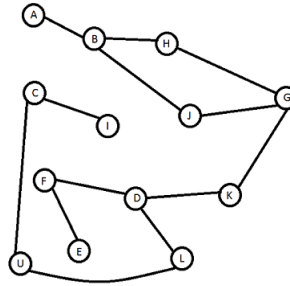
- (a) Using a graph $G = (V, E)$, model this plan so that the drone can navigate the floor. (*hint*) You'll need to have an additional vertex that represents the outside—call this vertex U. G is an *undirected* graph (edges have no direction). An edge is now either $\{u, v\}$ or $\{(u, v), (v, u)\}$. This impacts the implementation. For an adjacency list, you'll have to decide whether to have one edge or both.

A, B, H, C, I, J, G, F, D, K, E, L

(A, B), (B, H), (B, J), (H, G), (J, G), (I, C), (C, U), (G, K), (K, D), (D, F), (F, E), (D, L), (L, U)

- (b) Using an adjacency list, model this floor plan.

See maze2.png



- (c) Trace a DFS on G starting at U.

Single routes:

$U \rightarrow C$

$U \rightarrow L$

$C \rightarrow I$

$L \rightarrow D$

$D \rightarrow K$

$D \rightarrow F$

$F \rightarrow E$

$K \rightarrow G$

$G \rightarrow J$

$G \rightarrow H$

$B \rightarrow J$

$H \rightarrow B$

$B \rightarrow A$

Routes hit the end will be:

$U \rightarrow C \rightarrow I$

$U \rightarrow L \rightarrow D \rightarrow F \rightarrow E$

$U \rightarrow L \rightarrow D \rightarrow K \rightarrow G \rightarrow H \rightarrow B \rightarrow A$

$U \rightarrow L \rightarrow D \rightarrow K \rightarrow G \rightarrow J \rightarrow B \rightarrow A$

- (d) The drone takes 4 minutes to scan a room and 2 minutes to move from one room to another. The drone needs 7 minutes to move from L to C (or *vice versa*). What is the approximate total time it takes for the drone to check the floor? How would you annotate the graph to reflect this cost? (*You're not asked to redo the graph—just explain what you'd do*)

Scan each room: $4 \times 12 = 48$

Shortest route: $2 + 7 + 2 + 2 + 2 + (2 + 2) + 2 + 2 + 2 + 2 + 2 + (2) + 2 + 2 = 35$

The number with parentheses means deuplicate routes.

Total number is: $48 + 35 = 83$ minutes

- (e) A single battery for the drone holds about 45 minutes worth of charge. How many batteries does the drone need to scan the floor?

$83/45 = 1.84$

We need two batteries for scanning the floor.

- (f) Using Python, create an adjacency list data structure (I used a dictionary which seems reasonable); then using either a Python package or your own implementation, do a DFS from the *outside* of the floor, *i.e.*, $\text{DFS}(G, U)$. The DFS should yield a sequence of rooms that the drone would scan. See graph.py and I convert capitals to numbers as below.

1 2 3 4 5 6 7 8 9 10 11 12 13
A B C D E F G H I J K L U

- (g) For extra credit, add the cost to the drone and battery life to the search; when the drone has scanned all the rooms, give the percent battery life left.
- (h) For extra extra credit, presume the drone starts and ends at U.

4. Rock/Paper/Scissors is a simple game: numbers 0,1,2 denotes each of these items with an outcome that rock *beats* scissors, scissors *cut* paper, and paper *covers* rock; otherwise it's a tie. Included below (and as file) is a Python script that allows a human to play with a computer.

```
import random as rn

#values of rock, paper, scissors
r,p,s = 0,1,2
#dictionary e.g., rock beats scissors
ws = {r:s, p:r, s:p}

nogames = int(input("Number of games? "))

totgames = 0
compwins = 0
humwins = 0
ties = 0

gamehistory = []

while totgames < nogames:
    human = int(input("r=0,p=1,s=2 "))
    comp = rn.randrange(0,3,1)
    gamehistory.append([human, comp])

    print("Human: {0}, Comp: {1}".format(human, comp))

    if ws[comp] == human:
        compwins += 1
    elif ws[human] == comp:
        humwins += 1
    else:
        ties += 1
    totgames += 1

v = list(map(lambda x: 100*x/totgames, [compwins, humwins, ties]))
print("Stats\ncw% = {0}, hm% = {1}, ties% = {2}".format(*v))
```

- (a) Play 5 times of 10 games (it will go quickly) and record the statistics for your wins, H_1, H_2, \dots, H_5 . *Expectation* gives us a measure of central tendency (sometimes called the mean). We can calculate the expectation as

$$E[H] = \sum_{i=1}^5 H_i/5$$

What is the expectation of a human winning? Are you surprised? Of the computer winning?
The human winning is $hm\% = 40.0$.
I'm not surprised.
The computer winning is $cw\% = 30.0$

- (b) From `python.org` look-up the random function. How is it picking the numbers 0,1,2? Does the computer's choice of numbers depend on its previous choice? How about a yours?
Picking the number is based on previous result (humwins or compwins or ties).
No. I randomly choose the number.
- (c) Remove the human player and add another computer player called Robby. See if you can find a strategy that has Robby's expected winning greater than the computer. You are free to employ whatever you'd like so long as it's yours.
When I run 10 times, the result is $cw\% = 20.0$, $rm\% = 40.0$ (Robby wins), $ties\% = 40.0$
When I run 1000 times, the result is $cw\% = 32.3$, $rm\% = 34.2$ (Robby wins), $ties\% = 33.5$.
This result is surprising. I feel lucky and satisfied of this random result.

Python program:

```
import random as rn

#values of rock, paper, scissors
r,p,s = 0,1,2
#dictionary e.g., rock beats scissors
ws = {r:s, p:r, s:p}

nogames = int(input("Number of games? "))

totgames = 0
compwins = 0
robbywins = 0
ties = 0

gamehistory = []

while totgames < nogames:
    robbys = rn.randrange(0,2)
    comp = rn.randrange(0,3,1)
    gamehistory.append([robbys, comp])

    print("Robby: {0}, Comp: {1}".format(robbys, comp))

    if ws[comp] == robbys:
        compwins += 1
    elif ws[robbys] == comp:
        robbywins += 1
    else:
        ties += 1
    totgames += 1

v = list(map(lambda x: 100*x/totgames, [compwins, robbywins, ties]))
print("Stats\n  cw% = {0}, rm% = {1}, ties% = {2}".format(*v))
```