Chris Fenton
Charlie Fornaca
Steven Meaney
Gavin Plesko

AI Midterm Revisions

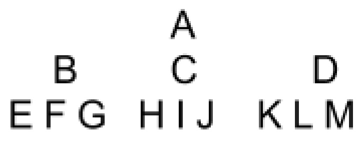
Monday October 31st, 2016

---

1. The goal of this question is to compare space complexities for BFS and DFS. Assuming that the root of a search tree has depth 0, and the optimal goal is at depth 2, compare BFS and DFS with concrete examples. Number the nodes and indicate the Goal.

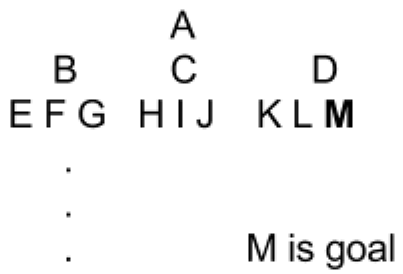
a) Draw a search tree with goal at depth 2 and branching factor 3 that would use the most memory for BFS. Assume that the children are ordered left to right. List the order in which nodes are visited.

BFS Path: [A,B,C,D,E,F,G,H,I,J,K,L,M (Goal)]



b) Draw a search tree with goal at depth 3 and branching factor 3 that would use the most memory for DFS. Assume that the children are ordered left to right. List the order in which nodes are visited. How does the number of nodes visited change with the depth of the tree, still assuming that the goal is at depth 2?

[a,b,e,... **X,** c, h, … **Y,** d,k, … **M**] Where X and Y are arbitrarily large numbers that represent the end of our data.



c) [optional] If the shallowest goal is at depth d, what are the worst case space and time complexities for BFS and IDFS?

IDFS: Time complexity O(b^d), Space complexity O(d)
BFS: Time complexity O(b^d), Space complexity O(b^d)

2. A* search: what is the condition for a heuristic function to be *admissible*?

For a heuristic to be admissible, it never overestimates the cost of reaching the goal.

3. Give an example of an admissible heuristic function for the 8-puzzle:

An example could be found by using a heuristic such as the Manhattan Distance (estimation based on applying the same concept as city blocks). This would be an admissible heuristic because it estimates the movement of each sliding tile without overestimation.

4. Given the following board position for 8-queens, representing this board configuration as an array of integers. The current state is [1,6,3,5,7,2,4,1]

▨ = Queen

```
|▢|▢|▢|▢|▢|▢|▢|

|▨|▢|▢|▢|▢|▢|▨|

|▢|▢|▢|▢|▨|▢|▢|

|▢|▢|▨|▢|▢|▢|▢|

|▢|▢|▢|▢|▢|▨|▢|

|▢|▢|▢|▨|▢|▢|▢|

|▢|▨|▢|▢|▢|▢|▢|

|▢|▢|▢|▢|▨|▢|▢|
```

a) Explain how local search might work:

> Calculate a score to optimize. In this case, we can count the number of queens that collide, and we'll try to minimize this score (0 is the goal). For each queen, we can move it one up or down 1 space and keep the state that produces the lowest score. We can an array where the indexes are the columns and the values are the rows.

b) Defining a reasonable cost function (computing it for this state):

> Number of conflicts between a queen and another queen (4 in this state).

c) Defining when two states are adjacent:

> Two states are adjacent if they can be made equal by a single move.

d) Draw a picture of an adjacent state with lower cost:

```
|▨|▢|▢|▢|▢|▢|▢|

|▢|▢|▢|▢|▢|▢|▨|

|▢|▢|▢|▢|▨|▢|▢|

|▢|▢|▨|▢|▢|▢|▢|

|▢|▢|▢|▢|▢|▨|▢|

|▢|▢|▢|▨|▢|▢|▢|

|▢|▨|▢|▢|▢|▢|▢|

|▢|▢|▢|▢|▨|▢|▢|
```

5. Using your state representation for 8-queens from the previous problem, write pseudo-code for a genetic algorithm. Highlight the 3 most important characteristics/components of a genetic algorithm.

> Randomly generate a set of *k* population states.

- Each successor state is a String.
- Each state is rated numerically against a **fitness function**.

    - A higher value returned would mean it is better.

> Successor states are generated by **reproducing** two parent states.

    - States represented in a string are split at a randomly chosen comparison point.
    - Halves are recombined once switched.
    - Compared to fitness function.

> A random **mutation** is introduced to well-scoring offspring states.

Fitness function - A fitness function would be an ideal numeric value that the population is striving to move towards. The fitness function is objective.

Mutation - During the mutation phase, a randomly selected spot in the string is randomly changed in attempt to work towards a higher fitness score.

Reproduction - Much like actual genetic reproduction, the “genes” of the Strings are unzipped at a randomly selected spot and switched with each other. When they are re-combined, they will both be compared to the fitness function.