CS4725/CS6705 - Fall 2017
Lab Activity # 1

Name: SAMPLE SOLUTIONS

Student ID: _____

Note: Your answers to these questions are to be handed in by 4:00 pm on **Wednesday, September 27, 2017**, but you are encouraged to complete these pages as soon as possible.

In this lab activity, we looked at the steps required to solve search problems using some provided Java code based on the algorithms in our textbook. The questions below are designed to get you to think about the different search methods that you are using.

The search problem being studied in this activity is the "Reverse" problem described during the lab session.

Here is a reminder of the steps that we followed in order for you to copy the provided code into your own account and to run the code:

- Create a new directory for yourself, change to that directory, create the required subdirectories under that directory and change to the `reverse` sub-sub-subdirectory.

```
mkdir AILabs
cd AILabs
mkdir -p aima/search/reverse
cd aima/search/reverse
```

- Copy the provided code into your `reverse` directory.

`cp /fcs/courses/cs4725/F17/lab1/* .` (Note the space and period after the *)

- Move back up to your `AILabs` directory, tell the system where to find a lot of the classes that have been provided to us, and then run the code.

```
cd ../../..
setenv CLASSPATH .:/fcs/courses/cs4725/aima-java/build
java aima/search/reverse/ReverseDemo
```

1. Test results from "Reverse" test cases

Test case 1: 4 5 6 7 3 2 1 8

Search method	Successful? (Y/N)	Nodes expanded	Solution found (sequence of actions)	# of actions in solution
BFS	Y	27	4,7	2
DLS, limit = 6	Y	8	2,2,2,2,4,7	6
DLS, limit = 5	Y	25	2,2,4,7	4
DLS, limit = 4	Y	6	2,2,4,7	4
DLS, limit = 3	Y	23	4,7	2
DLS, limit = 2	Y	4	4,7	2
IDS	Y	5	4,7	2

Test case 2: 6 7 8 3 5 4 2 1

Search method	Successful? (Y/N)	Nodes expanded	Solution found (sequence of actions)	# of actions in solution
BFS	Y	1309	4,6,5,8	4
DLS, limit = 8	Y	157	2,2,2,2,4,6,5,8	8
DLS, limit = 6	Y	155	2,2,4,6,5,8	6
DLS, limit = 4	Y	153	4,6,5,8	4
DLS, limit = 3	N	57		
IDS	Y	219	4,6,5,8	4

Test case 3: 8 7 1 5 4 3 6 2

Search method	Successful? (Y/N)	Nodes expanded	Solution found (sequence of actions)	# of actions in solution
BFS	Y	18303	8,5,3,4,6	5
DLS, limit = 7	Y	2588	2,2,8,5,3,4,6	7
DLS, limit = 6	Y	17002	8,2,5,4,5,6	6
DLS, limit = 5	Y	2586	8,5,3,4,6	5
DLS, limit = 4	N	400		
IDS	Y	3052	8,5,3,4,6	5

Test case 4: 3 7 1 4 6 2 5 8

Search method	Successful? (Y/N)	Nodes expanded	Solution found (sequence of actions)	# of actions in solution
BFS	Y	315523	3,5,6,3,7,2,6	7
DLS, limit = 8	Y	51137	2,4,6,3,7,5,2,3	8
DLS, limit = 7	Y	29715	3,5,6,3,7,2,6	7
DLS, limit = 6	N	19608		
IDS	Y	52590	3,5,6,3,7,2,6	7

2. Discussion questions

(a) In class, we talked about the **optimality** of different search methods. Comment briefly on how your results on these specific test cases demonstrate what we learned.

For example, if one of the techniques is not optimal, point to specific test cases that demonstrate this for our example problem. If a technique **is** optimal, you don't have to prove it, but you should argue why your test results seem to support this.

Breadth-first search and iterative deepening always found an optimal (shortest) solution to the problem. Depth-limited search is **not** optimal if we set the limit too high. Note, for example, the solutions of length 6 and 4 that were found in test case #1 when a length-2 solution existed.

(b) In class, we talked about the **completeness** of different search methods. Comment briefly on how your results on the specific test cases demonstrate what we learned.

As discussed in class, breadth-first search and iterative deepening search are complete, as long as certain conditions hold (which they do in this example if we assume that all actions have the same cost). We observed that these two algorithms found a solution every time. Depth-limited search is complete if we set the limit high enough, but in certain cases (test case 2, limit 3; test case 3, limit 4; test case 4, limit 6), it was unable to find a solution because there existed no solutions before the depth limit l was reached.

(c) In class, we talked about the **time complexity** (number of nodes expanded) of different search methods. Comment briefly on the number of nodes expanded by the different techniques, relative to each other, in our test cases.

The number of nodes expanded by breadth-first search is always relatively high because it expands every node at every depth until it finds a solution of some depth d.

For depth-limited search, the number of nodes expanded tends to be quite low, in general. For example, in test case #2, DLS found its solutions after 153 to 157 nodes, while BFS required 1309. As discussed in class, DLS is a good choice if we know the exact depth at which solutions will be found.

Iterative deepening expanded fewer nodes than BFS in all cases (and uses much less memory as well). It will always expand **more** nodes than DLS with the limit set to exactly the right depth. (See test case #4, for example, where IDS expanded 52590 nodes, while DLS-7 expanded only 29715 nodes.) However, if we don't know in advance what the solution depth will be, IDS is generally a better choice since it is both optimal and complete, and the extra number of expanded nodes is not that high in the grand scheme of things.

(d) If we were using tree search (without avoiding repeated nodes) and using the way in which we have ordered the successors of each node, standard **depth-first search** would be doomed to fail on this particular problem. Provide an example to demonstrate why this is the case.

On this particular problem, with the order in which we have decided to attempt different actions, depth-first search will just keep trying the sequence of moves $2, 2, 2, 2, 2, 2, 2, \cdots$ forever. For example, in test case #1, it will try

$$45673218 \rightarrow 54673218 \rightarrow 45673218 \rightarrow 54673218 \rightarrow 45673218 \rightarrow \cdots$$

and will never terminate unless a depth limit is set.