**Department of Information and Computer Science**

**ICS 381: Principles of Artificial Intelligence**  
**First Semester 2015/2016 (151)**

**Written Homework No. 1**  
**[Search Algorithms]**  
**Posted: Sunday 26th February 2017**  
**Due: Sunday 05th March 2017 @ 11:59**

**Submission Procedure:**

1. Each team must submit one submission with a cover page indication the team member’s IDs, names and contribution in the homework solutions as follows:

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Contribution** |
| 201330470 | Ali Slais | 50% in Q1 |
| 201224780 | Ibrahim Al-Beladi | 50% in Q1 |
| 201351850 | Mustafa Al-Turki | 50% in Q2 |
| 201381710 | Maged Saeed Al-shaibani | 50% in Q2 |

### Question 1 [All Roads Leads to Rome] [20 Points]

This map, illustrated in Figure 1, shows approximate mean driving times (in hours) between pairs of cities. For each of the following graph search strategies, work out the order in which states are expanded, as well as the path returned by graph search. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first. The start and goal states are Warsaw and Rome, respectively. Remember that in **graph search**, a state is expanded only once. The book and the slides present two correct, yet slightly different, versions of the algorithm. Please use the algorithm in the slides.



**Figure 1: Central Europe Map with Mean Driving Times (in hours).**

1. Fill the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Order** | **Path** | **Grade** |
| Depth-first search | **W**  **W-B**  **W-B-M**  **W-B-M-R** | **W-B-M-R** | **3 pt** |
| Breadth-first search | **W**  **W-B**  **W-M**  **W-O**  **W-B-M**  **W-M-B**  **W-M-R** | **W-M-R** | **3 pt** |
| Uniform cost search | **W**  **W-O**  **W-B**  **W-M**  **W-M-V**  **W-O-V**  **W-B-M**  **W-O-V-M**  **W-B-M-V**  **W-M-V-R** | **W-M-V-R** | **3 pt** |
| Greedy search | **W**  **W-M**  **W-M-R** | **W-M-R** | **3 pt** |
| A\* search | **W**  **W-M**  **W-B**  **W-M-V**  **W-M-V-R** | **W-M-V-R** | **3 pt** |

Use the following heuristic:

**h(Odesa) = 20 hrs, h(Budapest) = 12 hrs, h(Munich) = 3 hrs**

**h(Venice) = 3 hrs, h(Rome) = 0 hrs, h(Warsaw) = 30 hrs.**

1. Is the heuristic admissible? Consistent? Briefly justify your answers. [**5 pts**]

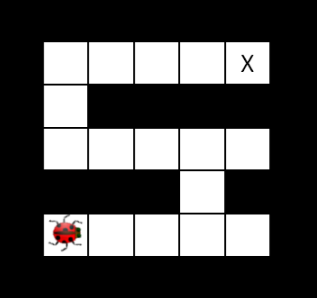
Not admissible because there are estimated costs that are greater than the real costs, the estimated cost for Warsaw = 30, the minimum cost from Warsaw to the goal = 24

Not consistent because the cost from Munich to Venice (3) + estimated cost from Venice to the goal (3) is not less than or equal to the estimated from Munich to the goal (3)

### Question 2 [Hive Minds] [40 Points]

You control one or more insects in a rectangular maze-like environment. At each time step, an insect can move into an adjacent square if that square is currently free, or the insect may stay in its current lo cation. In the case of multiple insects’ adjacent insects cannot swap lo cations. Squares may be blocked by walls. There are N non-wall squares. Optimality is always in terms of time steps; all actions have cost 1 regardless of the number of insects moving or where they move. For each of the following scenarios, precisely but compactly define the state space and give its size. Then, give a non-trivial admissible heuristic for the problem and the maximum branching factor. Your answers should follow the format in the example case below. Full credit requires a minimal state space (i.e. do not include extra information) and a reasonable non-trivial heuristic. The heuristic must be admissible, but you are not required to prove admissibility. The illustrations are given as examples; your answers should not assume a specific maze.

**Example [Lonely Bug]:** You control a single insect as indicated in the maze shown in Figure 2, which must reach a designated target location X. There are no other insects moving around.

  
 **Figure 2: Lonely Bug Maze.**

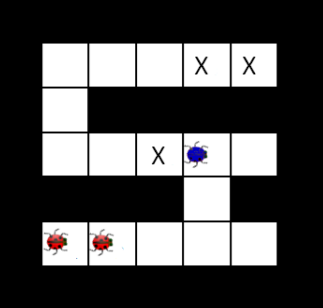
**State space description:** A tuple (x, y) encoding the x and y coordinates of the insect.

**State space size:** N.

**Heuristic:** The Manhattan distance from the insect’s location to the target.

**Maximum branching factor:** 5

a) **Swarm Movement:** You control K insects, one blue and K − 1 red. There are K target locations. In each time step all insects move simultaneously (or some may stay in place). Each insect can go to any target location. An example is shown in Figure 3. Fill the missing information below.

  
**Figure 3: Swarm Movement Using K Bugs.**

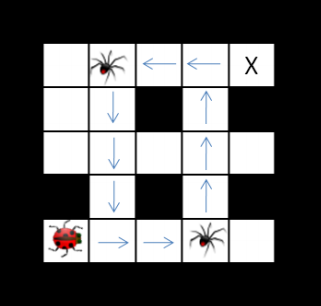
**State space description:** a tuple of (xr1,yr1), (xr2,yr2)..(xrk-1,yrk-1) and (xb, yb)to indicate the position of insects on the maze.

**State space size:** N(N-1)(N-2)..(N-k+1)

**Heuristic:** Manhattan distance from the position of the insect to the goal

**Maximum branching factor:** 5^k

b) **Patrolling Guards:** You again control a single insect, but there are G spiders patrolling known paths as shown in Figure 4. Specifically, at time t each guard g will be at position (xg(t), yg(t)) (in general, guard movements need not be a function of a guard’s current location, but you may assume that the tuple of guard positions follows a known pattern that repeats with period T and two guards cannot be in the same cell at the same time). Similarly to question a), your insect cannot take an action which moves it into either a guard’s current location or the location a guard is about to occupy.

  
 **Figure 4: Patrolling Guards Maze.**

**State space description:** (xg,yg) tuples to indicate the position of the guards and (x,y) tuples to indicate the position of the insect

**State space size:** NT(T-1) (T-2) ..(T-G+1)

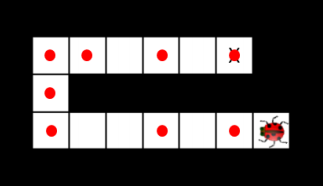
**Heuristic:** dis (to the goal) - (dis(to the first guard) = dis(to the second guard))

**Maximum branching factor:**

5\*2^(G) if spiders can stay in their place

5 if they only move in the pattern

c) **Step On It:** Your single insect is alone in the maze again. This time, it can speed up as long as it does not change direction. Specifically, after a move of ***v*** squares in some direction, it can move up to ***v + 1*** squares in that same direction on the next time step. It can move fewer than ***v + 1*** squares in that direction, as well, and it can move one square in any other direction (or stand still). Moving v squares requires that all intermediate squares passed over, as well as the ***v***-th square, currently be empty. The cost of a multi-square move is still 1 time unit. Let ***L*** be the size of the longest straight corridor in the maze. In the example below (see Figure 5) , ***L = 7*** and the dots in the maze below indicate where the insect will be after each time step in the optimal (fewest time step) plan:

**  
 Figure 5: Step on It Maze (L = 7).**

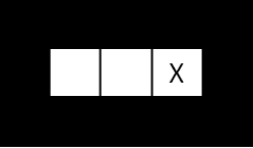
**State space description:** tuples of (x,y) to indicate the insect’s position and a function to determine the positions of the red dots in the maze.

**State space size:** N

**Heuristic:** Manhattan dis(to the goal) + red-dots function

**Maximum branching factor:** 5

d) **Lost at Night**: It is night and you control a single insect. You know the maze, but you do not know what square the insect will start in. Let ***M*** be the set of non-wall squares in the maze, i.e. ***|M| = N***. You must pose a search problem whose solution is an all-purpose sequence of actions such that, after executing those actions, the insect will be on the exit square, regardless of initial position. The insect executes the actions mindlessly and does not know whether its moves succeed: if it uses an action which would move it in a blocked direction, it will stay where it is. For example, in the maze shown in Figure 6, moving left twice and then right twice guarantees that the insect will be at the exit regardless of its starting position.

  
 **Figure 6: Lost At Night Maze.**

**State space description:** A tuple (direction, #step) eg. (South, 1) encoding the number of steps done and in which direction, an integer Z encoding the number of consecutive steps moved in the current direction with no intermediate change in direction

**State space size: 2^N**

**Heuristic:** Number of consecutive steps in the direction of the goal

**Maximum branching factor:** 5N

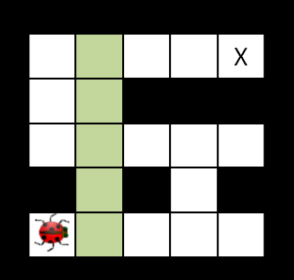
**State space description:**

**State space size:**

**Heuristic:**

**Maximum branching factor:**

e) **Escape:** Your insect is again alone in a maze, but certain squares are filled with pesticide, shown below in green (see Figure 7). Those squares are safe to travel through provided the insect does not accumulate more than ***L*** time steps in the pesticide, at which point it would die. You must find a solution where no more than ***L*** pesticide squares are used.

  
 **Figure 7: Escape! Maze.**

**State space description:** A tuple (x, y) encoding the x and y coordinates of the insects, a tuple (x, y) encoding the x and y coordinates of the squares filled with pesticide, an integer F encoding the number of steps done on the square with pesticide.

**State space size:** N \* L

**Heuristic:** The Manhattan distance from the insect’s location to the target + F.

**Maximum branching factor:** 5L

**Note:** Each question part is worth 8 points.