**PART 1**

**PART 2**

1) 

2) 

3) 

4) 

**PART 3**

**PART 4**

**1) Is breadth-first generally a better search strategy than depth-first for these inputs? Explain why or why not.**

In general, the time complexity of breadth-first is equal to O(bd), and depth-first is O(bm) (where d=depth of solution, m=max depth of tree). For this type of problem, the goal node is always located at a certain fixed depth (depending on the sentence specification), therefore the depth of solution and max depth of the tree are the same (i.e. d=m). So the time complexity of breadth and depth are both equivalent.

However, the space complexity of depth-first (O(bm)) is better than breadth-first (O(bd)), therefore depth-first is the superior of the two strategies.

**2) Describe the heuristic you chose for the directed search.**

Heuristic function:

Our heuristic function calculates the product of its children heuristic value. This is done recursively, until it reaches the depth of the solution (i.e. the goal nodes).

**3) Does your heuristic guarantee that the highest-probability sentence will always be found? Explain why or why not.**

No, because the more nodes you’re multiplying together, the lower the probability will be. So for the more descendants (neighbors, neighbors of its neighbors, etc.) that a node has, the lower the heuristic value will be.

**4) Is it possible for your heuristic search to have worse run-time performance than a depth-first or breadth-first search? If worse performance is possible, explain what would cause a worse performance to occur. If not, explain why worse performance would never occur.**

Yes, it is possible for our heuristic to have worse run-time performance than depth and breadth first. For one of the examples (2) that we tried heuristic considered 8258 nodes, while depth and breadth both only considered 379 nodes.

One of the reasons for this is that heuristic causes a lot of backtracking, considering more nodes. Another reason is because the heuristic function is recursive, causing the runtime complexity to be larger.