Andrew Forste | David Jefts

1) See .gif files depthFirstSearch.gif, greedyBestFirstSearch.gif, and AStar.gif for answers.
2)

How is a state within the algorithm represented?

- Each state is comprised of one or more nodes. Each node contains its f(n) value and its parent node. It also contains children nodes that it has attempted to visit in a local list. Nodes that have been visited are in a list called “closed”. Nodes that are available to visit but have not yet been visited are stored in a priority queue called “open”.

What is the successor function?

- First, check to make sure that the current node is not the start node. If it is, check whether all nodes have been visited on the current path. If they have been visited, then end the program successfully
- Once a node is at the top of the priority queue, it gets sent to the closed list and its children added to the open list (if the child is not already on the closed list, or if the child is the start/stop node).
- The priority is based on the heuristic (see below), plus the distance traveled up to that point
- Continue taking off nodes and adding children. If the children are all on the closed list, traverse back 1 parent and go to a different child. If still no valid options, continue moving up parents in a while list.
    - After moving up, remove child from global closed list but add it to the parent's closed list

What is an admissible heuristic?

- Straight line distance from the current node back to the start/stop location (will never go over, even if two cities are in a straight line).

How would you define the path cost?

- The distance between each pair of cities as defined in the problem.

What are the goal criteria?

- Visit all cities exactly once, except for the start city which should be visited twice
- Minimum distance traveled

3)

How is state represented?

- Priority queue of all items (ie nodes) that have not been attempted to put into the knapsack (called the open list)
    - Prioritized by price/weight
- List of items that have been attempted (and failed) to be put into the knapsack (called the closed list)
- “Knapsack” object with a list of items in the knapsack, total value, and available weight remaining.

What is the successor function?

1. Start with the item with the highest heuristic value and place it in the knapsack (if weight allows)
2. Pick the item with the next highest price/weight ratio

3. If that item will *not* put the knapsack over maximum weight, add it and go back to step 2
4. Otherwise, add the item to the closed list and go back to step 2
5. Once the queue is empty, end the program and return the items in the knapsack

What is an admissible heuristic?

- (Price / weight) with higher numbers preferred

What is the value / reward that you are seeking to optimize?

- We are seeking to maximize the value of the bag while staying within our weight constraint

Is this a maximization problem or a minimization problem?

- Maximization problem