**Project 1: Subway Search  
Grading Rubric**

**Question 1 – Subway Navigation Problem definition**

|  |  |
| --- | --- |
| **3** | SubwayNavigationProblem class is free of all but minor errors. |
| **2** | Solution has one of the following issues:   * pathCost() function is not overridden * Assumes all links between two stations have equal cost, or misses that stations can be adjacent in multiple ways. (Could appear in pathCost() or successors() functions) * Minor errors in determining path cost or successor states * Initial state or goal state not accurately tracked |
| **1** | Multiple issues from above list, or major errors in determining path cost or successor states. |
| **0** | Solution is blank/skeleton or does not compile. Solution does not override successors() function. |

**Question 2 – Depth-First Search**

|  |  |
| --- | --- |
| **4** | Algorithm is implemented correctly. |
| **3** | One of the following issues:   * Searches frontier/fringe or explored set for duplicate nodes, rather than duplicate states. * Search starts with successors of initial state, rather than initial state itself * Minor errors in path cost calculation * Fails to output one of the required elements (final path, total cost, # of search nodes visited) |
| **2** | Multiple issues from the list above, or:   * Fails to check frontier/fringe or explored set for repeated states at all (implements Tree-Search rather than Graph-Search) |
| **1** | Code compiles but crashes from an exception. |
| **0** | Not attempted or skeleton code provided. Code does not compile. |

**Question 3 – Breadth-First Search**

|  |  |
| --- | --- |
| **4** | Algorithm is implemented correctly |
| **3** | One of the following issues:   * Performs goal test when node is visited, rather than when node is expanded * Searches frontier/fringe or explored set for duplicate nodes, rather than duplicate states * Search starts with successors of initial state, rather than initial state itself * Minor errors in path cost calculation * Fails to output one of the required elements (final path, total cost, # of search nodes visited) |
| **2** | Multiple issues from the list above, or:   * Fails to check frontier/fringe or explored set for repeated states at all (implements Tree-Search rather than Graph-Search) |
| **1** | Code compiles but crashes from an exception. |
| **0** | Not attempted or skeleton code provided. Code does not compile. |

**Question 4 – A\* Search**

|  |  |
| --- | --- |
| **4** | Algorithm is implemented correctly. |
| **3** | One of the following issues:   * Searches frontier/fringe or explored set for duplicate nodes, rather than duplicate states. * When duplicate states with lower values are found, the priority queue is not correctly updated. (Or incorrectly includes states with higher values) * Search starts with successors of initial state, rather than initial state itself * Minor errors in path cost calculation * Minor errors in heuristic calculation * Fails to output one of the required elements (final path, total cost, # of search nodes visited) |
| **2** | Multiple issues from the list above, or:   * Fails to check frontier/fringe or explored set for repeated states at all (implements Tree-Search rather than Graph-Search) * Priority Queue is ordered by path-cost or heuristic, rather than their sum * Major errors in heuristic calculation, or wrong heuristic used |
| **1** | Code compiles but crashes from an exception. |
| **0** | Not attempted or skeleton code provided. Code does not compile. |

**Question 5 – Revised Problem Definition**

|  |  |
| --- | --- |
| **3** | New subclass of Problem (or SubwayNavigationProblem) is free of all but minor errors. |
| **2** | Solution has one of the following issues:   * Straight-line distance calculated incorrectly between goal station and potential nearby station * Any issues from Question 1 which are new (rather than repeated due to inheritance). |
| **1** | Multiple issues from above list, or:   * Work is done to check for nearby stations, but goalTest() only compares to the goal station originally provided |
| **0** | * Solution is blank/skeleton or does not compile. * Solution does not override goalTest() function and no evidence of checking for nearby stations is provided. |

**Question 6 – Eight Puzzle Problem**

|  |  |
| --- | --- |
| **4** | EightPuzzleProblem class is free of all but minor errors. |
| **3** | One of the following issues:   * Minor errors in heuristic calculation * Heuristic calculation includes the Manhattan distance for the blank space (should be excluded) * Goal state is incorrectly represented * Initial state incorrectly arranged from input * Minor errors in determining successor states (e.g., one of the directions is calculated improperly) |
| **2** | Multiple issues from above list, or one of the following:   * Uses a heuristic other than Manhattan distance * Major errors in determining successor states, such as:   + Fails to make sufficiently deep copies of puzzle board when determining successors   + Indexing errors result from successor functions |
| **1** | Multiple issues from lists for 2 and 3 above, and/or:   * Does not override h() function (no heuristic provided) |
| **0** | Solution is blank/skeleton or does not compile. |

**Question 7(a) – Admissible Heuristics for Subway Navigation Problem**

|  |  |
| --- | --- |
| **Full** | Any admissible heuristic, such as:   * Assume all stations are equidistant (so long as the assumed distance is no greater than the distance between the two closest stations) |
| **Partial** | Any of the following:   * Heuristics that would overestimate under some circumstances * Heuristics that might be admissible in other problems but fail here, such as:   + Manhattan distance (subways can go on diagonals). (If using a different system, this might be admissible.) * Heuristics that mix units (path cost is distance, but heuristic is time) * Heuristics that effectively require that we’ve already solved the problem |
| **No** | No answer provided, or the answer misunderstands what a heuristic is or what admissibility means. |

**Question 7(b) – Consequence of Data Issues**

|  |  |
| --- | --- |
| **Full** | Answer should recognize that the flaws in the data causes straight-line-distance to sometimes overestimate the true distance, meaning that the heuristic is not admissible and solutions found by A\* may therefore be sub-optimal. |
| **Partial** | Answer hints at sub-optimality or inefficiency (A\* will visit additional nodes), but misses admissibility argument. |
| **No** | No answer provided, or the answer does not mention admissibility, optimality, or efficiency concerns. |

**Question 7(c) – Use of Iterative Deepening on Eight Puzzle**

|  |  |
| --- | --- |
| **Full** | Answer should recognize that fast BFS result and slow DFS result implies that the goal is relatively shallow but on the far side of the tree. This suggests that iterative deepening would improve the run time since we wouldn’t search as deeply before checking the other side of the tree for shallow solutions. |
| **Partial** | Incorrectly asserts that iterative deepening would be costly because it regenerates nodes, or makes claims based on space rather than time. |
| **No** | No answer provided, or the answer shows misunderstanding of iterative deepening or the differences between BFS and DFS |