CSE 373: 23 Spring Practice Final Solutions

Name:	UW Email: @uw.edu

1 Sorting

What is the best-case runtime for this algorithm in terms of the size of the array, `n`? Provide a simplified and tight Big-Oh bound, then describe or provide an input array that would result in best-case performance.

The best case runtime is O($nlog(n)$). An input array that could result in this performance is [5, 3, 7, 1, 4, 6, 8], so that the binary tree remains complete after each insertion. Then each insertion will take O($log(n)$) time, resulting in O($nlog(n)$) runtime overall.

What is the worst-case runtime for this algorithm in terms of `n`? Provide a simplified and tight Big-Oh bound, then describe or provide an input array that would result in worst-case performance.

The worst case runtime is O(n^2). An input array that could result in this performance is [1, 2, 3, 4, 5, 6]. This will build a degenerate tree, so each insertion will take O(n) time. This results in O(n^2) time overall.

Is the sort stable? No justification required.

Yes. For a BST with duplicates, the elements that are to the right of a given node *r* are >= to r. So if *a* was inserted before *b*, then *b* would be to the right of *a* after insertion. Thus, when we do an inorder traversal, the relative order of *a* and *b* will be maintained.

Is the sort in-place? If not, provide a simplified and tight Big-Oh bound for the extra space complexity.

No. To insert n elements into the BST, we need O(n) extra space.

Finally, what is one improvement you could make to the algorithm so that the worst case runtime matches that of the best case? For this part, assume no duplicates.

Instead of using a vanilla BST, we can use a self-balancing BST like an AVL or red/black tree. This would make the best and worst case runtime O(nlog(n)). Since we didn't talk about how self-balancing trees handle duplicates, we had you assume no duplicates for this part. But there is a way to adapt AVL trees to handle duplicates: construct an `AVLTreeMap<T, List<T>>`. Each object maps to a list of its duplicates. This would ensure the sort is still stable.

2.1 Final Grading

In one word, what should each vertex represent in the graph?

Building.

In one short sentence, what should each edge represent in the graph?

There is an edge between two buildings if and only if there is a segment of pavement connecting them.

Should the edges in the graph be directed or undirected?

Undirected. Connections are bidirectional, so there's no reason to have a directed graph.

Should the edges in the graph be weighted or unweighted?

Weighted.

If you answered "Weighted" to the previous question, in one phrase, what should each edge weight represent? If you answered "Unweighted" to the previous question, leave it blank.

Average walking time or the length of each path. This is data we could get online.

Explain how you can efficiently determine which TA arrives first. You **do not** need to explain how the algorithm is implemented. For example, if you want to use breadth-first search in your answer, simply say "use BFS to achieve…". For your algorithm to be considered "efficient", it must call Djikstra's algorithm **at most once**.

Solution 1:
Add an additional "dummy node" to the graph. Create an edge from the dummy node to each starting building with weight 0. Run Djiskstra's algorithm using the dummy as the start node. Extract the shortest path from the dummy to CSE 3. Take the building immediately following the dummy along this path, and return the corresponding TA name using the input map.

This works because we want to compute the shortest path from each starting point to CSE 3, and take the shortest one among all these paths. The shortest path from the dummy node to CSE 3 must contain this path. So we can efficiently compute the answer by calling Djistra's algorithm exactly once from the dummy node.

Solution 2:
Instead of creating a dummy node, we can make the "starting" point of our Dijkstra's algorithm the secret CSE building (target location). Afterwards, we take our shortest path tree to each building, and take the nearest TA.

2.2 Robots

In one short sentence, what does each edge represent in the graph?

There is an edge from one pair of locations to another if one robot stays at the same location but the other moves to a neighboring node.

Are the edges in the graph directed or undirected?

Undirected. If you can reach a pair of locations p1 from another pair p2, then you can also reach p2 from p1. So relationships are bidirectional.

Are the edges in the graph weighted or unweighted?

Unweighted.

Explain how you can determine whether such a schedule exists. If you plan to use one of the graph algorithms you've learned in class, you do not need to explain how the algorithm is implemented. For example, if you want to use breadth-first search in your answer, simply say "use BFS to achieve…".

First construct a map from each pair of nodes in *G* to the shortest distance between them. This can be done by running BFS on every node in *G*. Then traverse every node in the graph *H* and remove any node where the distance between the two locations is <= r. If the vertex (a, b) no longer exists, return false. Otherwise, run BFS starting from (a, b). If (c, d) is contained in the resulting SPT (in other words (c, d) is reachable from (a, b)), return true. Otherwise, return false.