CPSC 365 S20 Homework #3 Self-Assessment

Spring 2020

Mark the bubble corresponding to your answer. When the options are YES/NO, YES=A and NO=B. Otherwise, the corresponding bubbles are as indicated. You can use bubble E for N/A.

Problem 1 1. Did you falsely claim that the concatenation of two or more best paths must also be a best path?

YES [A] NO [B]

If yes then STOP. Grade = 0/2.

2. Did you account for both ways P could be better than P_2: a) lower dirt distance

YES [A] NO [B]

If no, subtract 0.2 points and skip the next item.

3. Did you do the math to show that P_1PP_3 then has shorter dirt distance than $P_1P_2P_3$ and is hence better?

YES [A] NO [B]

If no, subtract 0.1 points.

4. same dirt distance and lower total distance?

YES [A] NO [B]

If no, subtract 0.2 points.

5. Did you do the math to show that P_1PP_3 then has the same dirt distance as $P_1P_2P_3$ and shorter total distance and is hence better?

YES [A] NO [B]

If no, subtract 0.1 points.

Algorithm

6. Did you use a modified Dijkstra's algorithm?

YES [A] NO [B]

If no then your solution does not fit this rubric and will require further examination.

7. Did you modify the algorithm by replacing the priority/d' comparison with an implementation of "better"?

YES [A] NO [B]

If no then subtract 0.2 points.

8. Did you describe the same modification for the comparisons done by the priority queue?

YES [A] NO [B]

If no then subtract 0.1 points.

9. Did you modify the algorithm by replacing the new priority computation with component-wise addition of $(dirt, total)$ tuples?

YES [A] NO [B]

If no then subtract 0.2 points.

Running Time

10. Did you end up with a running time of $O(n \log n)$ for sparse graphs and $O(n^2)$ for dense graphs?

YES [A] NO [B]

If yes then skip the next item.

11. Did you end up with a running time of $O(m \log n)$

YES [A] NO [B]

If yes then subtract 0.1 points; if no then subtract 0.25 points.

12. Did you specify a binary heap (for $O(m \log n)$) or Fibonacci heap or d-ary heap with $d = \frac{m}{n}$ for the priority queue?

YES [A] NO [B]

If no then subtract 0.1 points.

Problem 2 26. Does your solution use a minimum spanning tree algorithm or something else aside from an algorithm for shortest paths?

YES [A] NO [B]

If yes then STOP. Grade = 0/2.

27. Does your solution solve the shortest paths problem for each pair of vertices where one is in S and one in $V - S$?

YES [A] NO [B]

If yes then STOP. The difference between the worst case for Dijkstra's algorithm when you want the distance to a single destination vs. the distance to all destinations is the same, so you could just Dijkstra's algorithm once from each vertex in $V - S$. Grade = 0.75/2.

28. Does your solution solve the shortest paths problem for each vertex in $V - S$?

YES [A] NO [B]

If yes then Skip to the running time section, starting with 1.25 points. This will result in a worst-case of $\Theta(n^2 \log n)$. You can do better.

29. Does your solution solve the shortest paths problem for each vertex in S?

YES [A] NO [B]

If yes then skip to the running time section, starting with 1.5 points. This is better when S is small, but in the worst case, S could have $\Theta(n)$ vertices and so this will still result in a worst-case of $\Theta(n^2 logn)$. You can still do better.

30. Does your solution a) run Dijkstra's algorithm after initializing $d'[v]$ for all vertices in S to 0 [A]; b) run Dijkstra's on a modified graph with a new source and edges of weight 0 to each vertex in S [B]. NO [C]

If yes then skip to the population count section, starting with 2 points. This will result in a $O(n \log n)$ running time with the appropriate choice of data structures.

31. Does your solution make some other modification to Dijkstra's algorithm that is equivalent to the above?

YES [A] NO [B]

This will require further investigation, but start with 2 points.

Population Count

32. Did you either a) keep a running total of the population or dequeued vertices, stopping when the total is at least b [A]; or b) sort the vertices by distance and compute the cumulative sums, finding the index where the sub exceeds b [B]; or c) something equivalent [C]? NO [D]

If no, subtract 0.25 points.

33. Did you account for vertices at the same distance as the one you stopped at in the previous step?

YES [A] NO [B]

If no, subtract 0.1 points.

Running Time

34. If you used a sort, did you choose an $O(n \log n)$ sort and state which one you used?

YES [A] NO [B]

If no, subtract 0.1 points.

35. Do you claim the correct running time for your approach?

YES [A] NO [B]

If no then subtract 0.5 points and STOP.

36. Did you state that you're using a binary heap, d-ary heap with $d = \frac{m}{n}$, or Fibonacci heap?

YES [A] NO [B]

If no then subtract 0.1 points.

37. Did you state that you're using an adjacency list for the graph?

YES [A] NO [B]

If no then subtract 0.1 points. An adjacency matrix will take $O(n^2)$ time to iterate over all the edges.

Problem 3 51 .Did you attempt an edge exchange argument?

YES [A] NO [B]

If yes then skip to the proof details section.

Proof Details

52. Did you use the Cut Property or the Cycle Property from the book without modifying the graph to ensure that all edge weights are unique?

YES [A] NO [B]

If yes, then subtract 1.25 points and skip to "Edge Exchange". The results of those theorems only hold when the edge weights are distinct. Ours are not.

53. Did you make the false claim that any edge in an MST is the light weight edge across any cut it crosses?

YES [A] NO [B]

If yes, subtract 1.5 points and skip to "Edge Exchange".

54. Did you use the fact that if there are different MSTs then there must be a cut with a non-unique minimum weight edge crossing it?

YES [A] NO [B]

If yes, then subtract 0.2 points and skip to "Edge Exchange". You would have to prove this using an edge exchange argument.

55. To show that the resulting tree is a *minimum* spanning tree, did you use the fact that any edge in a minimum spanning tree is a light weight edge across some cut?

YES [A] NO [B]

If yes, subtract 0.3 points and skip to "Edge Exchange". This is true, but you would have to prove it – using an edge exchange argument. You also probably need to specify a cut for this to be useful.

56. Did you use the fact that any edge in a minimum spanning tree is a light weight edge across the cut defined by vertices reachable from one endpoint without using the edge in question?

YES [A] NO [B]

If yes, subtract 0.2 points and skip to "Edge Exchange". This is true, but you would have to prove it – using an edge exchange argument.

Edge Exchange

57. Did you attempt to swap an edge from T with an edge not in T but in some other minimum spanning tree T'?

YES [A] NO [B]

If no then subtract 1.25 points and STOP.

58. Did you select an edge from T' that crosses between the connected components that are created when removing the edge from T?

YES [A] NO [B]

If no then subtract 1.0 points and stop.

59. Did you explain why the resulting tree is a spanning tree?

YES [A] NO [B]

If no then subtract 0.05 points.

60. Did you make the opposite swap in T'?

YES [A] NO [B]

If no, subtract 0.25 points and STOP.

61. Did you explain why the resulting tree is a spanning tree?

YES [A] NO [B]

If no then subtract 0.05 points.

62. Did you compare both resulting trees to the original MSTs to conclude that the two swapped edges have the same weight?

YES [A] NO [B]

If no, subtract 0.1 points.

Problem 4 76. Did you iterate over all edges, attempting to find one that can be swapped into T to create another minimum spanning tree?

YES [A] NO [B]

If yes, then skip to algorithm details.

77. Did you apply a minimum spanning tree algorithm with each edge in T removed in turn, checking for whether the resulting tree has the same total cost as the original?

YES [A] NO [B]

If yes, then score 1.0 points and stop. The repetition over all edges in T will result in an algorithm that is not as efficient as it could be.

78. Did you apply Kruskal's algorithm [A] or Prim's algorithm [B] with the edge comparisons altered to break ties in cost in favor of edges not in T over edges in T? NO [C]

If Kruskal or Prim, skip to the running time section If no, then stop – your algorithm does not fit this rubric and will require further examination.

Algorithm Details

79. For each edge to add, did you attempt to find the edge in T that it would replace?

YES [A] NO [B]

If no, then stop – your algorithm does not fit this rubric and will require further examination. If incorrect, the score would be 0.75 points.

80. Did you select the edge that (u, v) would replace by finding the maximum weight edge on the path in T from u to v?

YES [A] NO [B]

If no then subtract 0.5 points and stop.

81. Did you compare the weight of that edge to the weight of (u, v) and return "NOT UNIQUE" (or something equivalent) if they are equal?

YES [A] NO [B]

If no then subtract 0.25 points and stop.

82. Did you return "UNIQUE" (or something equivalent) only after all iterating over all edges not in T?

YES [A] NO [B]

If no then subtract 0.1 points and stop.

Running time

83. Did you explain how to tell if an edge is in the tree or not?

YES [A] NO [B]

If no then subtract 0.25 points and skip the next item.

84. Did you explain how to tell if an edge is in the tree or not in $O(1)$ time after some preprocessing that assumes an nxn matrix can be zeroed in $\Theta(1)$ time?

YES [A] NO [B]

If yes then skip the next step, but realize that zeroing any array takes time proportional to the number of entries in the array, which will be $\Theta(n^2)$ for the nxn matrix – thinks like `calloc` or `memset` or Java's array initializer just hide the loop from you. Even if there is a hardware "zero a memory block" capability, that hardware will work in units of some fixed size block, so zeroing an arbitrary range of addresses will take multiple operations.

85. Did you explain how to tell if an edge is in the tree or not in $O(1)$ time after possibly $O(m \log n)$ preprocessing?

YES [A] NO [B]

If yes then add 0.1 points.

86. Did you end up with a running time of $O(m \log n)$?

YES [A] NO [B]

If no then subtract 0.25 points and stop.

87. Did you specify an adjacency list for the graph?

YES [A] NO [B]

If no then subtract 0.1 points. Iterating through all the edges will require $O(n^2)$ time when using an adjacency matrix, which is worse than $O(m \log n)$ for sparse graphs.

88. If you used Kruskal's or Prim's algorithm, did you specify a union/find data structure or a priority queue data structure that will result in an $O(m \log n)$ running time?

YES [A] NO [B]

If no then subtract 0.1 points.