Name:

Advanced Programming, II Semester, 2014–2015

Quiz 6, 6 April 2015

Answer all questions in the space provided. Use the reverse for rough work, if any.

---

1. Consider the following strategy to solve the single source shortest path problem with positive integer edge weights from source s.

   - Replace each edge with weight w by w edges of weight 1 connected by new intermediate nodes.
   - Run BFS(s) on the modified graph to find the shortest path to each of the original vertices in the graph.

   Which of the following statements is correct?

   (a) This strategy will not solve the problem correctly.

   (b) This strategy will solve the problem correctly and is as efficient as Dijkstra's algorithm.

   (c) This strategy will solve the problem correctly but is not as efficient as Dijkstra's algorithm.

   (d) This strategy will only work if the graph is connected.

   (5 marks)

   Answer: (c) The size of the graph blows up according to the edge weights, but the strategy is otherwise correct.

2. Suppose we want to extend the Union-Find data structure to support the operation *Reset(c)*, which takes as input the name of a component c and then breaks up c into singleton components. For instance if $c = 3$ and c currently consists of $\{1, 3, 7\}$, then *Reset(c)* will produce three components called 1, 3 and 7 consisting of $\{1\}$, $\{3\}$ and $\{7\}$, respectively.

   Which of the following is correct about the cost of adding *Reset(c)* to the array+list and tree implementations of Union-Find?

   (a) Array+list representation: $O(n)$, Tree representation: $O(n)$

   (b) Array+list representation: $O(size(c))$, Tree representation: $O(n)$

   (c) Array+list representation: $O(n)$, Tree representation: $O(size(c))$

   (d) Array+list representation: $O(size(c))$, Tree representation: $O(size(c))$

   (5 marks)

   Answer: (b) In the array+list representation we have the list of members of c which allows us to update the contents of c in time $O(size(c))$. In the tree representation there is no easy way to identify all elements that belong to component c without scanning the entire set, so it takes time $O(n)$.