CSE 101: Midterm 1 Solution

April 29, 2005—Day 10.5

---

1. *7 pts.* Mergesort is in the following time bounds:

   (a) $O(n \log n)$

   (b) $\Omega(n \log n)$

   (c) $\Theta(n \log n)$

   (d) All of the above

   (e) None of the above

   d). Mergesort is in $\Theta(n \log n)$ and so is also in $O(n \log n)$ and $\Omega(n \log n)$

2. *7 pts.* A greedy algorithm for making change will work with:

   (a) 1-cent, 2-cent, 4-cent, 8-cent, and 32-cent coins

   (b) 1-cent, 5-cent, 25-cent and 75 -cent coins

   (c) 1-cent, 2-cent, 5-cent, 8-cent, and 32-cent coins

   (d) a and b

   (e) a and c

   (f) c and b

   (g) a, b, and c

   (h) None of the above

   g). A greedy algorithm will work with all of these denominations.

3. *7 pts.* Which of the following statements is correct about minimum-cost comparison-based sorting complexity?

   (a) No algorithm of any kind can sort any kind of input in less than $\Omega(n \log n)$.

   (b) Algorithms that do only comparisons between elements require at least $\Omega(n \log n)$ comparisons for all inputs

(c) All paths in a comparison-based decision tree have length at least $n \log_2 n$

(d) No paths in a comparison-based decision tree have length greater than $n \log_2 n$

(e) All of the above

(f) None of the above

f) None of the above. a) isn't true because we know that counting sort can sort integers from 1..n in linear time. b) isn't true because some inputs can be sorted in less than $n \log n$ time. c) is false for the same reason as b).d) is false because our analysis didn't come up with an upper-bound on the number of comparisons.

4. *7 pts.* We want to prove that any comparison-based sorting algorithm requires at least $n-1$ comparisons for all inputs of the form $A[1..n]$. Which of the following is the idea of a correct proof:

   (a) Make each $A[i]$ a node in a graph, and record each comparison as an edge. If we have fewer than $n-1$ edges, the graph is not connected. Elements in one of the connected components could be greater to or less than those of another of the connected components.

   (b) If we compare $A[1]$ to $A[2]$, and, in general, $A[i-1]$ to $A[i]$, we need at least $n-1$ comparisons. Otherwise one of the elements won't be compared, and could be less than all others, or greater than all others

   (c) The best case input is if it is already sorted. All that is needed is to verify elements are monotonically non-decreasing. That requires $n-1$ comparisons.

   (d) Insertion sort requires at least $n-1$ comparisons for all inputs.

   (e) None of the above

   a) By looking at the groups of elements that haven't been compared against each other, we can argue that any two groups that haven't been compared can cause two different sorted permutations that yield the same output. All the other answers assume how a particular algorithm is working.

5. *7 pts.* Given an algorithm operating on a graph with n nodes and m edges that sorts the edges, and then does constant-time operations on each node and each edge, which is the best bound on its runtime:

   (a) $O(n^2 \log m)$

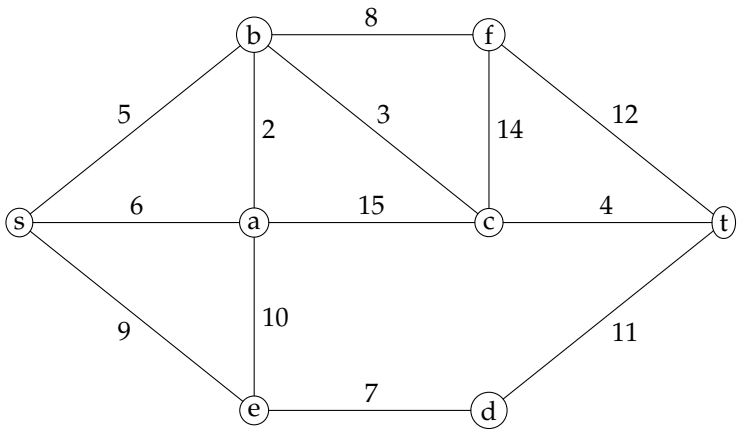
   (b) $O(m \log m)$

   (c) $O((m+n) \log m)$

   (d) $O(min(m,n) \log n)$

   c) Sorting the edges takes $O(m \log m)$ time. Constant operations on each node and each edge takes time $O(m+n)$. The tightest bound, then is $(n+m \log m)$, but that isn't one of the choices. a) is too big. b) could be too small (if $n < m \log m$). d) could be too small (if $n < m \log n$).

6. *7 pts.* Assume you are using Dijkstra's algorithm to find the minimum path from s to t in the following graph. If we say that s is added *first*, which node is added *fourth*?



   (a) a

   (b) b

   (c) c

   (d) d

   (e) e

   (f) f

c). The second node added is b. The third is a. The fourth is c.

7. *12 pts.* If $T(n) = 5T(n/5) + O(n^2)$, what is the asymptotic complexity (O, Θ, or Ω notation)? Justify your answer.

   Using the master method, a is t, b is 5, and f is $O(n^2)$. Since f(n) is polynomially bigger than $n^{\log_b a} = n^1 = n$, the runtime is $O(n^2)$. Note that the answer isn't $\Theta(n^2)$ because f(n) can be less than $O(n^2)$ (for example, it could be $n^{1.5}$.

8. *13 pts.* True or false? (If true, prove, if false, provide a counter-example.) Given a graph, G, partition G into two parts and construct a minimal spanning tree for each part. Then, connect the two parts by the shortest edge connecting the two parts. The resulting tree is a minimal spanning tree for G.

   False. Consider a graph with four nodes: a, b, c, d. Let the weight of edge (a,b) be 100, weight of edge (b, c) = 1, weight of edge (a, c) = 2, weight of edge (c, d) = 101. Partition G into vertices a, b, and vertices, c, d. The MST of the first is (a, b). the MST of the second is (c, d). The cheapest edge between them is (b, c) with a weight of 1. Total weight is 202. But, the MST of G is the edges (b, c), (a, c), and (a, b) with a total weight of 103.

9. *13 pts.* True or false? (If true, prove, if false, provide a counter-example.) If all edges in a graph have different lengths, then there is a unique shortest path from s to any node.

   False.

   Counter-example: Nodes s, t, u. edge (s, t) with length 3, edge (t, u) with length 2, edge (s, u) with length 5. All edges have different lengths, but there are two paths from s to t: both with length 5

10. *20 pts. total*

    Following is an algorithm that correctly creates a Minimal Spanning Tree by repeatedly deleting the most expensive edge in a cycle from a connected graph G with at least one node.

```
Procedure GreedyMST(E, V)
Precondition: _____________________________________________
```

```
_____________________________________________________
_____________________________________________________
_____________________________________________________
Begin
Sort the edges e from highest cost down to lowest cost
E' = E
Loop
    Loop Invariant: _____________________________________
    _____________________________________________________
    _____________________________________________________
    _____________________________________________________
    _____________________________________________________
    Exit when |E'| = |V| - 1
    e = next edge of E (in reverse sorted order)
    If (V, E' - {e}) is connected
        E' = E' - {e}
    i = i - 1
Endloop
End
Postcondition: (E', V) is a MST of (E, V)
```

(a) *10 pts.* Fill in the Precondition and Loop Invariant necessary to prove the Postcondition.

Precondition: G=(E, V) is a connected graph with $|V| > 0$.

Loop Invariant: (E', V) contains an MST of (E, V)

(b) *10 pts.* Prove the algorithm is true by proving the Loop Invariant and Postcondition.

Loop Invariant:

Initialization: Since (E, V) is connected (by the precondition), (E, V) contains a MST of (E, V). Since E' = E (by the code before the loop), (E', V) contains an MST of (E, V).

Maintenance: Before the loop, (E', V) contains an MST of (E, V). If we remove an edge from E', it belongs to a cycle of (E', V) (since its removal doesn't disconnect (E' - {e})). The edge is the most expensive edge in the cycle (since we iterate through the edges from most

expensive to least expensive. By the cycle property, it is not part of any MST. Thus, after the loop, (E', V) still contains an MST of (E, V).

Termination: E' gets smaller each time through the loop, except for at most $|V| - 1$ edges which might disconnect (E', V). Eventually, $|E'|$ will equal $|V| - 1$ (since, by the loop invariant, (E', V) contains an MST, and therefore (E', V) is connected).

Postcondition: After the loop, (E', V) contains an MST of (E, V) and $|E'| = |V| - 1$, which means that (E', V) is a tree. If (E', V) contains a MST, and is a tree, then E', V) is the MST of (E, V).

11. *Extra credit: 10 points* True or false? (If true, prove, if false, provide a counter-example.) Given an acyclic graph,G with non-negative edge lengths, make a copy G', and modify the edge lengths in G' to be $k - e_{ij}$ for some large constant k (larger than any original edge length). Now, use Dijkstra's algorithm to find the shortest path from s to t in G'. The shortest path from s to t in G' corresponds to the longest path from s to t in G.

    False because paths with more edges will count k more times. Counter-example: G= 3 nodes, s, t, and u with edge weights (s, u) = 2, (u, t) = 3, (s, t) = 4. The longest path from s to t is s-u-t. Construct G' with $k = 5$. Now, (s', u') = 3, (u, t) = 2, and (s', t') = 1. The shortest path is s'-t', which doesn't correspond to the longest path in G.