CSC 225 Final Exam
Fall 1992

1. [10] For the following algorithms, express the best and worst case time complexities using $\Theta$ notation. Assume the data is stored in an array of size $n$.

| Algorithm | Best Case Time Complexity | Worst Case Time Complexity |
|---|---|---|
| (a) Linear Search | | |
| (b) Binary Search | | |
| (c) MaxSort | | |
| (d) Heapsort | | |
| (e) Quicksort | | |

2. [20] Circle true or false for each question and justify your answer.

(a) Under the decision tree model, any algorithm for sorting 4 numbers requires at least 5 comparisons in the worst case.

**True** **False**

(b) An algorithm which takes $O(n^2)$ time in the worst case should always be preferred over an algorithm which is $O(n^3)$ in the worst case.

**True** **False**

(c) The lower bound for sorting is $\Omega(n \log n)$ so it is impossible to find a sorting algorithm which is $O(n)$ in the worst case.

**True** **False**

(d) In order to find a minimum weight spanning tree, we might have to completely order the edges of the graph by edge weight. Thus under the comparison model, any algorithm for minimum spanning tree is $\Omega(m \log m)$ in the worst case, $m$ is the number of edges.

**True** **False**

3. [20] Define

(a) the MST problem,

(b) the average case complexity of an algorithm,

(c) a lower bound for a problem, and

(d) amortized cost of an algorithm.

4. Let $f(k) = \sum_{i=1}^{k} i\, 2^i$.

(a) [5] Prove that $k\ 2^k$ is a lower bound for $f(k)$.

(b) [5] Prove that $k\,2^{k+1}$ is an upper bound for $f(k)$.

(c) [5] State a definition of the set of functions $\Theta(f(n))$. Do not use limits in your definition. (You can define it in terms of $O$ and/or $\Omega$ as long as you define them without using limits).

(d) [5] Use parts (a), (b), and (c) to prove $f(k)$ is in $\Theta(k\,2^k)$. Do not use limits.

5. You work as a programmer for the phone company. You are given a long list of $n$ telephone bills and a list of $p$ checks from good customers. Consider the following two algorithms for matching customers with bills so we can determine who has not paid.

**Approach 1:** Do not sort the telephone bills. Do linear search for each of the $p$ bill payments.

**Approach 2:** Sort the bills and the bill payments by telephone number and do a merge to match customers with bills.

(a) [10] What are the worst case time complexities of the two approaches?

(b) [5] When is Approach 1 faster asymptotically? (Use Big O type notation to express your answer).

(c) [5] When is Approach 2 faster asymptotically? (Use Big O type notation to express your answer).

6. [20] Use Dijkstra's Shortest Path algorithm to compute the length of a shortest path from vertex one to each of the other vertices of the following graph. The exam had a graph here. Make up your own example. Mark the tree edges as follows:

**Solution:**

| **Vertex** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| **Length of Shortest Path** | | | | | | | |

7.(a) [5] Draw the picture (directed graph) associated with the following UNION/FIND array.

| Vertex | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Parent | 2 | 2 | 3 | 1 | 5 | 2 | 5 | 2 |

(b) [5] Give pseudo-code for a non-collapsing FIND using the data data structures as given in part (a).

(c) [10] Give Pseudo code for a collapsing FIND using the data data structures as given in part (a).

8. Apply the Heapsort algorithm discussed in class (with a max-heap) to the following array showing the data stored in the array after each of the indicated steps. As well, draw a binary tree representation of the heap at each step.

(a) [10] Building the heap.

| | Indices: | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|
| Step | Initial data: | 2 | 3 | 5 | 6 | 7 | 9 | 1 |
| 1 | Filter_up(List[1]): | | | | | | | |
| 2 | Filter_up(List[2]): | | | | | | | |
| 3 | Filter_up(List[3]): | | | | | | | |
| 4 | Filter_up(List[4]): | | | | | | | |
| 5 | Filter_up(List[5]): | | | | | | | |
| 6 | Filter_up(List[6]): | | | | | | | |

**Binary Tree picture of the heap after each step:**

(b) [10] Sorting by repetitively deleting the maximum element.

| Step | Indices: | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|
| 6 | (from previous page): | | | | | | | |
| 7 | Delete_Max: | | | | | | | |
| 8 | Delete_Max: | | | | | | | |
| 9 | Delete_Max: | | | | | | | |
| 10 | Delete_Max: | | | | | | | |
| 11 | Delete_Max: | | | | | | | |
| 12 | Delete_Max: | | | | | | | |

**Binary Tree picture of the heap after each step:**