Homework 2

CS6033 Design and Analysis of Algorithms I

Fall 2024

(Sec. B, Prof. Yi-Jen Chiang)

Due: Wed. 10/2 by 1pm
(submit online on NYU Brightspace; one submission per group)
Maximum Score: 100 points

Note: This assignment has 3 pages.

1. (20 points)

Define a sequence of numbers S_n (for integers $n \geq 0$) recursively as follows:

$$S_n = \begin{cases} 1 & \text{if } n = 0, \\ 2 & \text{if } n = 1, \\ 3 & \text{if } n = 2, \\ S_{n-1} + S_{n-3} & \text{if } n \geq 3. \end{cases}$$

(a) Prove **by induction** that $\sum_{i=3}^{n} S_{i-3} = S_n - 3$ for all integers $n \geq 3$. No credit will be given for a different method of proof. **(10 points)**

(b) Prove **by induction** that $S_{n+3} \geq \hat{\phi}^n$ for all integers $n \geq 0$, where $\hat{\phi} \in (1, 2)$ is a root of the equation $x^3 - x^2 - 1 = 0$. No credit will be given for a different method of proof.
(**Remark:** Such $\hat{\phi}$ exists and here is an argument to show why: If we let $f(x) = x^3 - x^2 - 1$, we have $f(1) < 0$ and $f(2) > 0$, and thus the equation $f(x) = 0$ has a solution (root) in the range $(1, 2)$ and we call this root $\hat{\phi}$ — this is just a background information and you do not need to worry about this part.) **(10 points)**

2. (12 points)

What is wrong with the following purported proof that all horses have the same color?
Proof by induction on the number of horses:

Basis Step. There is only one horse. Then clearly all horses have the same color.

Induction Hypothesis. In any group of up to n horses, all horses have the same color.

Induction Step. Consider a group of $n+1$ horses. Discard one horse; by the induction hypothesis, all the remaining horses have the same color. Now put that horse back and discard another; again all the remaining horses have the same color. So all the horses have the same color as the ones that were not discarded either time, and so they all have the same color.

3. (10 points)

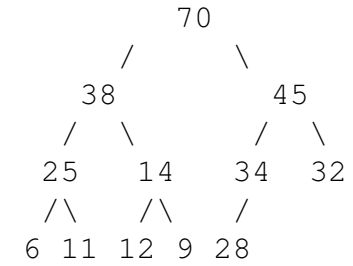
Consider the following sequence of items

22, 30, 11, 46, 35, 14, 25, 62, 40, 83, 55, 86.

Insert these items in the order above, into an initially empty implicit binary min-heap. Show the heap throughout the process but write down the heap as an answer after every 3 INSERT() operations. (Just hand-simulate the behavior of the implicit binary min-heap, and draw the resulting

heap, both throughout the entire process and the ones after every 3 INSERT() operations as answers (note that the former is to show all your work so that partial credits can be given in case you make mistakes, while the latter (answers) are the ones to be graded). Draw them in the form of a **tree** rather than an array.) **(1 + 2 + 3 + 4 = 10 points)**

4. (12 points)
Consider the implicit binary max-heap below. Show the results of performing 4 consecutive EXTRACT_MAX() operations on it. For **each** EXTRACT_MAX() operation, show the item extracted together with the resulting heap.



(Just hand-simulate the behavior of the implicit binary max-heap, and draw the resulting heap after every EXTRACT_MAX() operation. Draw them in the form of a **tree** rather than an array.)
(**Note: 3 points for each result.**)

5. (26 points)
Implicit binary heap implemented by an array is simple and efficient, except that at some point when the array is full and an INSERT() operation comes in, we need to re-allocate a larger array (allocating a new, larger memory space, copying the data items over, and releasing the old memory space), which is expensive. A natural alternative is to implement a binary heap as a binary tree where each node has pointers to its parent, left child and right child. Again, it is a complete binary tree except for the lowest level, which is filled from the left up to some point. Also, as in implicit binary heaps, we maintain a counter, n, for the total number of nodes in the heap. Now the key task here is to be able to find the last node efficiently.

(a) We can represent a path from the root to a node of a binary tree by means of a binary string (e.g., "01101"), where 0 means "go to the left child" and 1 means "go to the right child." Based on this representation, design and analyze an algorithm to find the last node of an n-node binary heap in $O(\log n)$ time.
(**Hint:** Look at **small examples** to derive your algorithm and also to **check** whether your algorithm is correct or not.) **(18 points)**

(b) Discuss how to incorporate the algorithm in **part (a)** into the operations EXTRACT_MIN(Q) (extracting the minimum element from Q) and INSERT(Q, x) (inserting an element x into Q) for a binary min-heap Q. **(3 + 5 = 8 points)**

6. (20 points)
Given an implicit binary max-heap A in an array (where the root $A[1]$ stores the maximum item), a real number x and a positive integer k, design and analyze an algorithm to decide whether the k-th largest item in A is $\geq x$. Your algorithm should run in $O(k)$ worst-case time and use at most $O(1)$ extra space, independent of the size of A.
Note: Your algorithm does **not** need to report the k-th largest item in A; only its relationship with x needs to be decided, to be able to answer "yes" or "no".