Practice problems

Analysis of iterative algorithms

• Give a tight bound on the number of Hello’s produced as a function of $n$

\textbf{Algorithm} LOOP2 ($n : \text{integer}$)
  \begin{algorithmic}
    \State $j \leftarrow 1$ to $n^2 \log n$ do
    \For {$i \leftarrow 1$ to $j$ do}
      \For {$i \leftarrow 1$ to $j$ do}
        \State print “Hello”
  \end{algorithmic}
Analysis of iterative algorithms

• Give a tight bound on the number of Hello’s produced as a function of \( n \)

\[
\text{Algorithm LOOP2 (} n : \text{integer) }
\]
\[
\text{for } j \leftarrow 1 \text{ to } n^2 \log n \text{ do}
\]
\[
\text{for } i \leftarrow 1 \text{ to } j \text{ do}
\]
\[
\text{print “Hello”}
\]

• Answer: \( \Theta(n^4 \log^2 n) \)

Analysis of iterative algorithms

• Give a tight bound on the number of Hi’s produced as a function of \( n \)

\[
\text{Algorithm LOOP1 (} n : \text{integer) }
\]
\[
\text{for } i \leftarrow 1 \text{ to } n \log^2 n \text{ do}
\]
\[
\text{for } j \leftarrow i \text{ to } j \text{ do}
\]
\[
\text{print “Hi”}
\]
\[
\text{while } j \leq n \text{ do}
\]
\[
\text{j } \leftarrow j + 1
\]
Analysis of iterative algorithms

• Give a tight bound on the number of Hi’s produced as a function of $n$

Algorithm $\text{LOOP1} \ (n : \text{integer})$

\[
\begin{align*}
\text{for } & i \leftarrow 1 \text{ to } n \log^2 n \text{ do} \\
\text{ } & j \leftarrow i \\
\text{while } & j \leq n \text{ do} \\
\text{ } & \text{print} \ “\text{Hi}” \\
\text{ } & j \leftarrow j + 1
\end{align*}
\]

• Answer: $\Theta(n^2)$

Analysis (recurrence relation)

Solve using the Master Theorem

\[
T(n) = \begin{cases} 
1 & n = 1 \\
7T\left(\frac{n}{2}\right) + n^2 & n > 1 
\end{cases}
\]

Solution: The first case applies.

$T(n) \in \Theta\left(n^{\log_2 7}\right)$
Analysis (recurrence relation)

Solve using the Master Theorem

\[
T(n) = \begin{cases} 
1 & n = 1 \\
9T\left(\frac{n}{3}\right) + n^2 \log n & n > 1 
\end{cases}
\]

Solution: The second case applies \((k = 1)\).

\(T(n) \in \Theta\left(n^2 \log^2 n\right)\)

---

Analysis (recurrence relation)

Solve using the Master Theorem

\[
T(n) = \begin{cases} 
1 & n = 1 \\
T\left(\frac{n}{2}\right) + n \log n & n > 1 
\end{cases}
\]

Solution: The third case applies.

\(T(n) \in \Theta\left(n \log n\right)\)
Lower bounds

The authors of a paper that you are asked to review claim to have designed a new data structure for priority queues that supports both the operations INSERT and EXTRACT.MINIMUM in $O(1)$ worst-case time. Should you reject the paper? Why? You cannot make any assumption on the functionality of the data structure.

Amortized analysis

We want to use an unsorted array to support the following two operations for a set $S$ of integers:

- **INSERT($S$, $x$)** inserts integer $x$ into set $S$

- **DELETE-LARGER-HALF($S$)** deletes the largest $\lfloor |S|/2 \rfloor$ integers from $S$ (where $|S|$ is the current size of $S$)

Explain how to implement these operations on an unsorted array so that any sequence of $m$ operations runs in $O(m)$ time (or $O(1)$-time amortized, per operation). For simplicity, you can assume that the array is large enough so that we do not have to deal with reallocation when it gets full.
Divide & Conquer (design)

Suppose you are given an array $A = \{a_1, a_2, \ldots, a_n\}$ of $n$ distinct integers. You are told that the sequence of values $a_1, a_2, \ldots, a_n$ is unimodal, that is for some index $p \in [1, n]$, the values in the array increase up to position $p$ in $A$, and then decrease the remainder of the way until position $n$. Give an algorithm to find the position $p$ in $O(\log n)$ time. You can assume $n$ to be a power of 2.

Divide & Conquer ($\log n$ design)

The median of a set of numbers $\{a_1, a_2, \ldots, a_n\}$ is the element $a_i$ such that there are $\lceil n/2 \rceil$ elements smaller than or equal to $a_i$, and there are $\lfloor n/2 \rfloor$ greater than or equal to $a_i$. In other words, the median is the element in the middle when the elements are sorted. For example, the median of $\{7, 3, 4, 1, 9, 2, 13\}$ is 4.

You are given two sorted arrays $A$ and $B$ of size $n$ each (for simplicity, you can assume $n$ to be some power of 2 and that the numbers are distinct). Give an algorithm to find the median of all $2n$ numbers in $O(\log n)$ time. Remember to analyze the complexity of your algorithm.
Divide & Conquer ("black box")

Given an unsorted array $A$ of $n$ distinct floating point numbers we want to print the smallest $\lceil \sqrt{n} \rceil$ numbers of $A$ in sorted order. For instance given $A = \{3.1, 4.2, 1.013, 2.12, 5.50, 6.12, 0.15, 8.2, 9.1\}$ containing 9 numbers, the algorithm is supposed to print 0.15, 1.013, 2.12 in sorted order. Give a $O(n)$-time algorithm for this problem. Hint: Use linear-time SELECT (as a black box) to solve this problem.

Divide & Conquer ("black box")

Give a linear-time algorithm that determines whether an unsorted sequence of $n$ real numbers contains a number that occurs at least $n/4$ times in the sequence. You algorithm should report "no" if no such number exist; otherwise, it should report all numbers occurring at least $n/4$ times. Argue about the correctness of your algorithm. Assume that SELECT works also when the elements are not distinct.