Exercise Sheet 8
COMS10017 Algorithms 2023/2024

Reminder: $\log n$ denotes the binary logarithm, i.e., $\log n = \log_2 n$.

1 Recurrences

Consider the recurrence $T(n) := T(\lfloor \frac{n}{3} \rfloor) + T(\lfloor \frac{2n}{3} \rfloor) + n$, for every $n \geq 2$ and $T(2) = T(1) = 1$.

1. Use the recursion tree method to come up with a guess for an upper bound on the recurrence (in Big-O notation).

   Hint: Ignore the floor operations. Determine the depth of the recursion tree. Determine the "work" that is done in each level of the recursion tree. Sum up the work done in each level to obtain a suitable guess for an upper bound on T.

2. Use the substitution method to prove that the guess obtained in the previous exercise is correct.

   Hint: $0.5 \leq \log(3/2)$

2 Analysis of a Recursive Algorithm

Consider the algorithm ALG listed as Algorithm 1:

Algorithm 1 ALG(n)

Require: Integer array A of length $n \geq 1$, n is a power of two
  $S \leftarrow 0$
  for $i \leftarrow 0 \ldots n-1$ **do**
    $S \leftarrow S + A[i]$
  end for
  if $n \leq 1$ **then**
    return S
  else
    return $S - \text{ALG}(A[0, \frac{n}{2} - 1]) - \text{ALG}(A[\frac{n}{2}, n-1])$
  end if

We assume that the length n of the input array in ALG is always a power of two, i.e., $n \in \{1, 2, 4, 8, 16, \ldots\}$.

1. Let $A = 1, 2, 3, 4$ and let $B = 1, 2, 3, 4, 5, 6, 7, 8$. Draw the recursion trees of the calls ALG(A) and ALG(B). For both trees, annotate each node with the value that is returned by the function call that corresponds to this node.

2. Recall that n is a power of two. Let $T(n)$ be the number of times the function ALG (listed in Algorithm 1) is executed when invoked on an input array of length n (including the initial invocation on the array of length n). Give a recursive definition of $T(n)$.

3. Let $T(n)$ be the function defined in the previous exercise. Use the substitution method to show that $T(n) \in O(n)$.

4. What is the runtime of Alg?

5. Recall that n is a power of two. Describe an algorithm with best-case runtime $\Theta(1)$ and worst-case runtime $\Theta(n)$ that computes the exact same output as Alg.

3 Divide-and-Conquer and the Number of Subproblems

In this exercise, we assume that n is a power of two. We consider a divide-and-conquer algorithm that, on an input of length $n \geq 2$, executes k recursive calls, each on inputs of lengths $n/2$, for some integer $k \geq 1$. The divide and combine phases of the divide-and-conquer algorithm on an instance of size n have a runtime of $O(n)$. We also assume that the runtime on an instance of length 1 is $O(1)$.

1. What is the runtime of the algorithm if $k = 1$?

2. What is the runtime of the algorithm if $k = 2$?

3. What is the runtime of the algorithm if $k = 3$?

4 Optional and Difficult Questions

Exercises in this section are intentionally more difficult and are there to challenge yourself.

4.1 Search in a Sorted Matrix (Difficult!)

We assume in this exercise that n is a power-of-two.

We are given an n-by-n integer matrix A that is sorted both row- and column-wise, i.e., every row is sorted in non-decreasing order from left to right, and every column is sorted in non-decreasing order from top to bottom. Give a divide-and-conquer algorithm that answers the question:

"Given an integer x, does A contain x?"

What is the runtime of your algorithm?