CMPSC 465 | Data Structures & Algorithms
Spring 2022 | Chunhao Wang and Mingfu Shao

# Assignment 02

## Due: Friday 09:00 am, Jan. 28, 2022

**Instructions:** You may work in groups of up to three people to solve the homework. You must write your own solutions and explicitly acknowledge everyone whom you have worked with or who has given you any significant ideas about your solutions. You may also use books or online resources to help solve homework problems. All consulted references must be acknowledged. The acknowledgements need to be made by answering Problem 1 below.

You are encouraged to solve the problem sets on your own using only the textbook and lecture notes as a reference. This will give you the best chance of doing well on the exams. Relying too much on the help of group members or on online resources will hinder your performance on the exams.

Submissions being late in 2 hours will be accepted with a 20% penalty. Submissions late more than 2 hours will receive 0. There will be no exceptions to this policy, as we post the solutions soon after the deadline.

For the full policy on assignments, please consult the syllabus.

**Formatting:** Start a new page for each problem.

**1. (0 pts.) Acknowledgements.** The assignment will receive a 0 if this question is not answered.

1. If you worked in a group, list the members of the group. Otherwise, write "I did not work in a group."
2. If you received significant ideas about your solutions from anyone not in your group, list their names here. Otherwise, write "I did not consult anyone except my group members".
3. List any resources besides the course material that you consulted in order to solve the material. If you did not consult anything, write "I did not consult any non-class materials."

**2. (10 pts.)** Solve each of the following recursions using master's theorem. Give the closed form of $T(n)$ in Big-Theta notation; you don't need to show your middle steps.

1. $T(n) = 8 \cdot T(n/2) + 100 \cdot n^3$.
2. $T(n) = 8 \cdot T(n/2) + 1000 \cdot n^{3.5}$.
3. $T(n) = 16 \cdot T(n/2) + n \cdot \log(n)$
4. $T(n) = 2 \cdot T(n/2) + n \cdot \log(n)$.
5. $T(n) = 8 \cdot T(n/2) + n^{3.5} \cdot \log^2(n)$.

**3. (10 pts.)** Suppose you have $m$ sorted arrays, each with $n$ elements, and you want to combine them into a single sorted array with $mn$ elements.

1. If you do this by merging the first two arrays, next with the third, then with the fourth, until in the end with the last one. What is the time complexity of this algorithm, in terms of $m$ and $n$?
2. Give a more efficient solution to this problem, using divide-and-conquer. What is the running time?

**4. (10 pts.)** If we add one more partition step in find-pivot function in selecting problem, and use 3 as the size of each subarray, the algorithm would become:

function find-pivot ($A$, $k$)
  Partition $A$ into $n/3$ subarrays;
  Let $M$ be the list of medians of these $n/3$ subarrays;
  Partition $M$ into $n/9$ subarrays;
  Let $M'$ be the list of medians of these $n/9$ subarrays;
  return selection($M'$, $|M'|/2$)
end function;

Analyze the running time of the new selection algorithm with the find-pivot function described above.

**5. (10 pts.)** Suppose you are given an array $A[1 \cdots n]$ of integers which can be positive, negative or 0. A sub-array is a contiguous sequence of elements from $A$. In particular, for any two indexes $i$ and $j$ with $1 \leq i \leq j \leq n$, $A[i \cdots j]$ is the sub-array that starts at index $i$ and ends at $j$. The sum of the sub-array $A[i \cdots j]$ is the sum of all the numbers it contains: $A[i] + A[i+1] + ..... + A[j]$. For example, if $A = [5, 1, -3, -2, 4, 0]$, the sum of $A[0 \cdots 3]$ is 1 and the sum of $A[1 \cdots 4]$ is 0. Given $A$, design a divide-and-conquer algorithm to find the sub-array of minimum sum. For example, in the array $A = [5, 1, -3, -2, 4, 0]$, the sub-array of minimum sum is $A[2, 3]$ with sum $-5$. Your algorithm should run in $O(n \log n)$ time.

**6. (0 pts.)** *(NOTE: you don't need to submit your solution for this problem.)* Consider recurrence relation $T(n) = \Theta(n) + T(a \cdot n) + T(b \cdot n)$, $T(1) = 1$, $0 < a < 1$, $0 < b < 1$. Prove the following:

1. $T(n) = \Theta(n)$, if $a + b < 1$.
2. $T(n) = \Theta(n \cdot \log n)$, if $a + b = 1$.

**Solution.** Assume that the term $\Theta(n)$ in the recursion admits a lower bound of $d_1 n$ and upper bound of $d_2 n$, for large enough $n$. That is $T(n) \geq d_1 n + T(an) + T(bn)$ and $T(n) \leq d_2 n + T(an) + T(bn)$.

1. We first prove that $T(n) = O(n)$, by induction. Assume that $T(n) \leq c_2 n$ for large enough $n$, where $c_2 = d_2/(1 - a - b)$. Now we prove that $T(n+1) \leq c_2(n+1)$. We can write

$$T(n+1) \leq d_2(n+1) + T(a(n+1)) + T(b(n+1)).$$

We have $a(n+1) \leq n$ and $b(n+1) \leq n$ for large enough $n$ as $a < 1$ and $b < 1$. By the inductive assumption we have

$$\begin{aligned} T(n+1) &\leq d_2(n+1) + c_2 a(n+1) + c_2 b(n+1) \\ &\leq d_2(n+1) + c_2(a+b)(n+1) \\ &= (d_2 + c_2(a+b))(n+1) \\ &= c_2(n+1). \end{aligned}$$

It's easy to verify the last equation, i.e., $d_2 + c_2(a+b) = c_2$ with $c_2 = d_2/(1 - a - b)$. In fact this is where we determine the value of $c_2$. You can also see why $a + b < 1$ is required here. We then prove that $T(n) = \Omega(n)$, by induction. Assume that $T(n) \geq c_1 n$ for large enough $n$, where

$c_1 = d_1/(1-a-b)$. Now we prove that $T(n+1) \geq c_1(n+1)$. We can write

$$\begin{aligned} T(n+1) &\geq d_1(n+1) + T(a(n+1)) + T(b(n+1)) \\ &\geq d_1(n+1) + c_1 a(n+1) + c_1 b(n+1) \\ &\geq d_1(n+1) + c_1(a+b)(n+1) \\ &= (d_1 + c_1(a+b))(n+1) \\ &= c_1(n+1). \end{aligned}$$

The last equation holds as $d_1 + c_1(a+b) = c_1$ with $c_1 = d_1/(1-a-b)$.

2. We first prove that $T(n) = O(n \log n)$, by induction. Assume that $T(n) \leq c_2 n \log n$ for large enough $n$, where $c_2 = -d_2/(a \log a + b \log b)$. Now we prove that $T(n+1) \leq c_2(n+1)\log(n+1)$. We can write

$$\begin{aligned} T(n+1) &\leq d_2(n+1) + T(a(n+1)) + T(b(n+1)) \\ &\leq d_2(n+1) + c_2 a(n+1)\log(a(n+1)) + c_2 b(n+1)\log(b(n+1)) \\ &\leq d_2(n+1) + c_2(a\log a + b \log b)(n+1) + c_2(a+b)(n+1)\log(n+1) \\ &= (d_2 + c_2(a\log a + b\log b))(n+1) + c_2(n+1)\log(n+1) \\ &= c_2(n+1)\log(n+1). \end{aligned}$$

The last equation holds as $d_2 + c_2(a\log a + b \log b) = 0$ with $c_2 = -d_2/(a\log a + b\log b)$. We then prove that $T(n) = \Omega(n\log n)$, by induction. Assume that $T(n) \geq c_1 n \log n$ for large enough $n$, where $c_1 = -d_1/(a\log a + b\log b)$. Now we prove that $T(n+1) \geq c_1(n+1)\log(n+1)$. We can write

$$\begin{aligned} T(n+1) &\geq d_1(n+1) + T(a(n+1)) + T(b(n+1)) \\ &\geq d_1(n+1) + c_1 a(n+1)\log(a(n+1)) + c_1 b(n+1)\log(b(n+1)) \\ &\geq d_1(n+1) + c_1(a\log a + b \log b)(n+1) + c_1(a+b)(n+1)\log(n+1) \\ &= (d_1 + c_1(a\log a + b\log b))(n+1) + c_1(n+1)\log(n+1) \\ &= c_1(n+1)\log(n+1). \end{aligned}$$

The last equation holds as $d_1 + c_1(a\log a + b \log b) = 0$ with $c_1 = -d_1/(a\log a + b\log b)$.