CSE 551 Homework 2
Solutions

27^{th} September, 2020

Submission Instructions: Deadline is **11:59pm on 10/04/2020**. Late submissions will be penalized, therefore please ensure that you submit (file upload is completed) before the deadline. Additionally, you can download the submitted file to verify if the file was uploaded correctly. **Please TYPE UP YOUR SOLUTIONS and submit a PDF** electronically, via *Canvas*.
Furthermore, please note that the graders will grade 2 out of the 4 questions randomly. Therefore, if the grader decides to check questions 1 and 4,and you haven't answered question 4, you'll lose points for question 4. Hence, please answer all the questions.

1. Design an algorithm to compute the 2nd smallest number in an unordered (unsorted) sequence of numbers $\{a_1, a_2, ..., a_n\}$ in $n + \lceil log_2(n) \rceil - 2$ comparisons in the worst case. If you think such an algorithm can be designed, then show how it can be done. If your answer is no, then explain why it cannot be done. **[25 points]**

   Solution:

   Let us consider a tournament. Assume that you have a list of n players (denoted by unique integers). One tournament rule could be the following: the smallest number of two numbers, wins. Now, we can use the FindMinMax algorithm taught in class to find out the smallest number in n - 1 comparisons. Realize that, in such a tournament the smallest number (best player) is guaranteed to play the second smallest number (second best player), if we follow our rule. Therefore, once we have discovered the best player, we need to examine *the subtree from where the best player originated*. This is because, the best player could have played the second best player at any level in the tournament (and not just the final round). This analysis takes $\lceil log_2(n) \rceil - 1$ comparisons. Therefore, in total, we have at most $n + \lceil log_2(n) \rceil - 2$ comparisons to find the 2nd second best player in the tournament.

   Another similar way to solve this to construct a min heap. Construction of the min heap would take $O(n)$. This would entail that the minimum value

in the sequence is present in the root. Remove the minimum and replace it with the rightmost element in the tree, following level order traversal. Heapify the new tree and you will have the second smallest element as the root now. Heapify takes $O(logn)$ time. Hence the total time to find the second smallest element is less than $n + \lceil log_2(n) \rceil - 2$.

2. Consider the product of matrices of size $n \times n$, where n is a power of 3. Using divide-and-conquer the problem can be reduced to the multiplication of 3×3 matrices. The conventional method requires 27 multiplications. In how many multiplications must one be able to multiply 3×3 matrices so that the resultant computing time is $O(n^{2.81})$? Do the same for the case when n is a power of 4 and the problem is reduced to the multiplication of 4×4 matrices. **Show all your work**. **[25 points]**

   Solution:

   The recurrence relation for following Strassen's method is of the form: $T(n) = 7T(n/2) + O(n^2)$

   where 7 is the number of multiplications required, $n/2$ is the size of the sub-problem and $O(n^2)$ is the recombination cost [Refer to the algorithms textbook]. We know that solving this, we end with with $T(n) = O(n^{2.81})$. Now, for computing the number of multiplications necessary for multiplying 3×3 matrices following the divide and conquer approach we have:

   $T(n) = XT(n/3) + O(n^2)$, where we need to compute the value of X.

   We know that the overall time complexity is $O(n^{2.81})$, which implies that $XT(n/3)$ is the main contributor to this time complexity as $O(n^{2.81} > O(n^2)$. Therefore, we have,

   $XT(n/3) = n^{2.81}$

   Using Master's Theorem, we can rewrite the Left Hand Side as,

   $n^{log_3 X} = n^{2.81}$, where $a = X$ and $b = 3$

   $log_3 X = 2.81$,

   $X = 21.9$,

   $X = 21$.

Similarly for the multiplication of 4×4 matrices, we have

$n^{log_4 X} = n^{2.81}$,

$log_4 X = 2.81$,

$X = 49$

3. If k is a non-negative constant, then prove that the recurrence: **[25 points]**

$$T(n) = k \text{ , for } n = 1 \text{ and}$$
$$T(n) = 3T(\tfrac{n}{2}) + kn, \text{ for } n > 1$$

has the following solution (for n a power of 2):

$$T(n) = 3kn^{log_2 3} - 2kn$$

Solution:

The last row of the recursion tree has $3^{log_2 n}k = n^{log_2 3}k$ elements:

Now, we get:

$T(n) = kn + (\frac{3}{2})kn + (\frac{3}{2})^2 kn + ... + (\frac{3}{2})^{log_2(n-1)}kn + n^{log_2 3}k$

$=> T(n) = kn\frac{(\frac{3}{2})^{log_2 n}-1}{\frac{3}{2}-1} + kn^{log_2 3}$ (Geometric Progression)

$=> T(n) = kn\frac{n^{log_2(\frac{3}{2})}-1}{\frac{3}{2}-1} + kn^{log_2 3}$

$=> T(n) = 2kn(n^{log_2(\frac{3}{2})} - 1) + kn^{log_2 3}$

$=> T(n) = 2kn(n^{(log_2 3 - log_2 2)} - 1) + kn^{log_2 3}$

$=> T(n) = 2kn^{(log_2 3 - 1 + 1)} - 2kn + kn^{log_2 3}$

$=> T(n) = 3kn^{log_2 3} - 2kn$ **(proved)**

4. Let $n = 2p$, $V = (v_1, ..., v_n), W = (w_1, ..., w_n)$. Then we can compute the vector product VW by the formula:

$\sum_{1 \le i \le p}(v_{2i-1} + w_{2i}) \times (v_{2i} + w_{2i-1}) - \sum_{1 \le i \le p} v_{2i-1} \times v_{2i} - \sum_{1 \le i \le p} w_{2i-1} \times w_{2i}$

which requires $3n/2$ multiplications. Show how to use this formula for the multiplication of two $n \times n$ matrices giving a method which requires $n^3/2 + n^2$ multiplications rather than the usual n^3 multiplications. **[25 points]**

Solution:

Multiplications of two $n \times n$ matrices to compute the resultant $n \times n$ matrix C involves product of n^2 vector multiplications. The matrix A may be viewed as made up of n row vectors $V_1, V_2, ..., V_n$ and similarly B may be viewed as made up of n column vectors $W_1, W_2, ..., W_n$. The element C_{11} of C is V_1W_1, C_{12} is V_1W_2 and so on.

In order to compute C_{11} we will require 3p multiplications. In order to compute $C_{12}, ..., C_{1n}$ will require only 2p multiplications, as the second term in the formula involving V_1 is already computed while computing C_{11}.

Similarly, in order to compute $C_{21}, ..., C_{n1}$ we will require only 2p multiplications, as the third term in the formula involving W_1 is already computed while computing C_{11}.

In order to compute C_{22} to C_{2n} and C_{32} to C_{3n}, ..., C_{n2} to C_{nn}, $((n-1)^2)$ entries, only p multiplications will be needed as the second and the third terms of the formula need not be computed again. Thus the total number of multiplications that will be needed is:

$3p + (n-1) \times 2p + (n-1) \times 2p + (n-1)^2 \times p$

$3n/2 + (n-1) \times n + (n-1) \times n + (n-1)^2 \times n/2$

$n^3/2 + n^2$