Homework 1

Fall 2022: CS 260 Design and Analysis of Algorithms

Instructions: This homework is due on **Monday, October 10, 2022** at the beginning of the class. You can work in groups and should submit hardcopies of the homework handwritten on A4-size papers. Write names of group members clearly on top of the first page as well as the total number of pages. You may have queries related to this homework which should be directed to the TA.

Zainab Alsuwaykit [zainab.alsuwaykit@kaust.edu.sa]
Office hours: Sunday 11:00-12:30
Office location: Building 1, B1-0203-WS20 (Level 0)

1. **(8 points)** Arrange the following functions according to their growth rates, i.e., a function f stays before a function g if $f = O(g)$. All logs are of base 2.

$10^{-6}n^7$	$\sqrt{n}(\log n)$	$n/\log n$	n^n	$2^{n^{2/3}}$	$(\sqrt{n})^n$	$\log\log n$	0.0003	$2^{n/\log n}$	$1/n^3$

.

2. **(5 points)** Sort the given sequence of integers $\boxed{92,\ -14,\ 4,\ -2,\ 12,\ 2,\ -78,\ 21}$ using MERGESORT.

3. (a) **(3 points)** Construct (as described in the lecture slides) a 2-3 tree for the given sequence of integers $\boxed{7,\ 0,\ 8,\ 28,\ 16,\ 5,\ 56}$, inserted from the left to the right. Show all steps.

   (b) **(2 points)** Now delete the node with the value $\boxed{7}$. Show all steps.

4. **(9 points)** Let f and g be two asymptotically positive functions such that $f(n) = O(g(n))$. For each of the following statements, decide whether it is **true** or **false** and give a proof or counterexample.

   (a) $\log_a f(n) = O(\log_a g(n))$ for some $a > 1$,

   (b) $2^{f(n)} = O(2^{g(n)})$,

   (c) $\sqrt{f(n)} = O(\sqrt{g(n)})$.

5. **(8 points)** Design an $O(n \log n)$-time algorithm that, for given integers x and $a_1, a_2, \ldots, a_n$, determines whether or not there exist a_i and a_j whose difference is exactly x. Justify the time complexity of the algorithm.

6. We are given an array $A[1..n]$, which stores a sequence of 0's and then followed by a sequence of 1's.

   (a) **(3 points)** Design an $O(\log n)$-time algorithm to find the location of the last 0, i.e, find k such that $A[k] = 0$ and $A[k+1] = 1$. Justify the time complexity of the algorithm.

   (b) **(5 points)** Suppose that k is much smaller than n. Is it possible to improve the running time of our algorithm to $O(\log k)$ instead of $O(\log n)$? Justify your answer.

7. **(5 points)** Use Karatsuba's algorithm to perform the following integer multiplication:

$$176 \times 3018 .$$

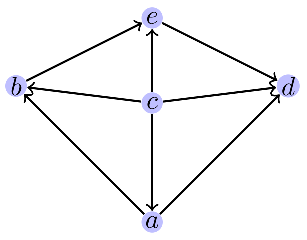
   You continue splitting of numbers until you have one-digit numbers.

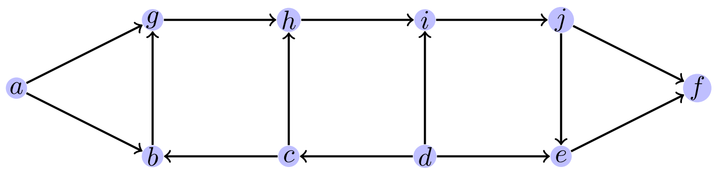
8. **(5 points)** Use Strassen's algorithm to perform the following matrix multiplication AB. Count the total number of scalar multiplications required to compute the product.

$$A = \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix}, B = \begin{pmatrix} 4 & 1 \\ 2 & 0 \end{pmatrix}$$

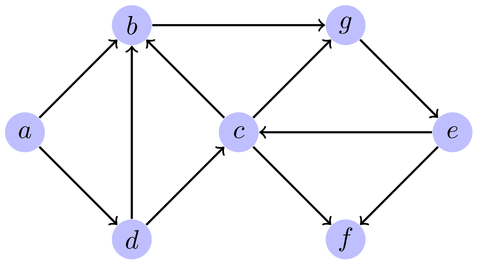
9. **(8 points)** Design a linear time algorithm for the following problem: for a given directed graph G, check if, for every two nodes u and v, there is a directed path from u to v or from v to u. Justify the time complexity of the algorithm.

10. **(6 points)** Find topological ordering for the following two graphs:

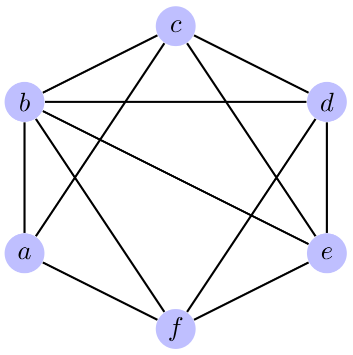




11. **(5 points)** For the following directed graph, use *breadth-first* search algorithm to find all strongly connected components.



12. **(5 points)** Is the following graph planar? Explain the answer.



13. **(5 points)** Consider four matrices of the following respective sizes:

$$M_1 : 1 \times 3, \quad M_2 : 3 \times 2, \quad M_3 : 2 \times 4, \quad M_4 : 4 \times 6.$$

Using dynamic programming algorithm find the minimum number of scalar multiplications required to compute the product $\prod_{j=1}^{4} M_j$ as well as an optimal parenthesization.

14. **(5 points)** Given an array of five integers, $1, 3, -3, 1, 2$, first find the set of all contiguous subarrays with maximum sum then find all the contiguous subarrays with maximum length in this set using extentions of dynamic programming approach (see 4. Dynamic Programming, Slides 32–47).

15. **(8 points)** Design an $O(n^2)$-time algorithm based on dynamic programming to find the maximum length of monotonically increasing subsequence of a sequence of n pairwise different integers. For example, $(1, 3, 9)$ is a monotonically increasing subsequence of maximum length of the sequence $(1, 7, 3, 2, 9)$. Justify the time complexity of the algorithm.

16. **(5 points)** Use dynamic programming algorithm to find edit distance between the pair of strings `STATIONARY` and `TERTIARY`.