ASSIGNMENT NO. 02/2024

ASHESI UNIVERSITY

CS456 – ALGORITHM DESIGN & ANALYSIS

Coverage : Brute Force / Analysis / Proof Techniques/Recurrence Relation

Total Points:	50 pts
Assigned :	Monday, 29th January 2024
Due :	Friday, 9th February 2024, 11:59pm.
Assignment Type:	Individual
Submit :	Latex for written work, Source files for code

Ashesi University Berekuso, E/R Ghana

Q1. Hypothesize a likely efficiency class of an algorithm based on the following empirical **observations of its basic operation's count:** Count is the time taken by the algorithm to finish/complete its execution, and size is input size.

size	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
count	11,966	24,303	39,992	53,010	67,272	78,692	91,274	113,063	129,799	140,538

[5 pts]

Q2. Consider the following recursive algorithm.

a. What does this algorithm compute? Or what does it do?

b. Set up a recurrence relation for the algorithm's basic operation count and solve it(find a closed form format for the recurrence relation).

ALGORITHM *Min1*$(A[0..n-1])$
//Input: An array $A[0..n-1]$ of real numbers
if $n = 1$ **return** $A[0]$
else $temp \leftarrow Min1(A[0..n-2])$
  if $temp \leq A[n-1]$ **return** $temp$
  else return $A[n-1]$

[3, 3-recur, 4-solve – 10pts]

Q3. Consider another algorithm for solving the problem of Q2 above, which recursively divides an array into two halves: call *Min2(A[O .. n* - 1]) where *Min2* is shown below.

ALGORITHM $Min2(A[l..r])$

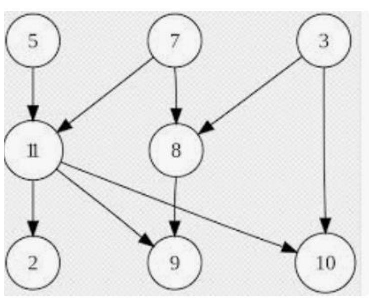
  if $l = r$ **return** $A[l]$
  else $temp1 \leftarrow Min2(A[l..\lfloor(l+r)/2\rfloor])$
    $temp2 \leftarrow Min2(A[\lfloor(l+r)/2\rfloor+1..r])$
    if $temp1 \leq temp2$ **return** $temp1$
    else return $temp2$

a. Set up a recurrence relation for the algorithm's basic operation count and **solve** it.

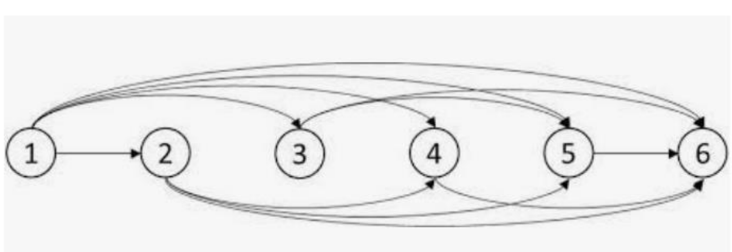
b. Which of the algorithms *Mini* or *Min2* is faster? Can you suggest an algorithm for the problem they solve that would be more efficient than both of them?

Q4. Topological Sort [a-3 -recur, a-4-solv 3=b – 10 pts]

(a) Apply the DFS-based algorithm to solve the topological sorting problem for the following directed acyclic graph:



(b) Apply the source-removal algorithm to solve the topological sorting problem for the following directed acyclic graph:

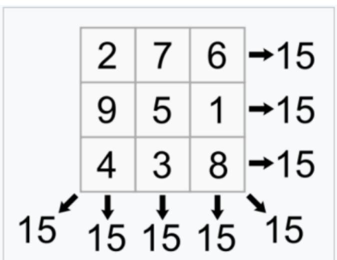


[5, 5 – 10 pts]

Q5.

<u>Programming Part</u>

Magic squares A magic square of order *n* is an arrangement of the numbers from 1 to *n2* in an *n-by-n* matrix, with each number occurring exactly once, so that each row, each column, and each main diagonal has the same sum. For example, the magic square of size 3 is shown below.



a. Prove that if a magic square of order *n* exists, the sum in question must be equal to *n(n2* + 1)/2.

[3]

b. Design an exhaustive-search algorithm for generating all magic squares of order *n.*

[3]

c. Go to the Internet or your library and find a better algorithm for generating. magic squares.

[1]

d. Implement the two algorithms-*the exhaustive search* and the *one you have found*- and run an experiment to determine the largest value of *n* for which each of the algorithms is able to find a magic square of order *n* in less than one minute of your computer's time.

[4, 4]

[15 points]