W1D3

QUESTION 1

Divide And Conquer (DAC)

You are given an n x n integer 2d-array M[0 .. n – 1, 0 .. n – 1] with the following properties:

(1) The number of rows is equal to the number of columns.

(2) Each row is in nondecreasing order.

(3) Each column is in nondecreasing order.

See the examples (M1, M2 and M3) below.

5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24
25	26	27	28	29

5	7	8	14	15
6	9	13	16	23
10	12	17	22	24
11	18	21	25	28
19	20	26	27	29

5	10	15	20	25
6	11	16	21	26
7	12	17	22	27
8	13	18	23	28
9	14	19	24	29

Let us call such arrays "Sorted Square of order n" for the purpose of this question.

(a) Write three different algorithm to generate Sorted Squares (of order n for n > 0) of the type M1, M2 and M3.

(b) Write a search algorithm *without DAC* “searchSS(M, key)” such that if key is present in the Sorted Square M, your algorithm will print both row and column numbers as a pair. If the key is not present in the Sorted Square M, it will print “Not Found”.

Example: For the Sorted Square *M2, if the key is 23, it will print (1, 4). However, if the key is 34, your algorithm will print “Not Found”.*

(b1) What is the time complexity of searchSS? Please explain your claim.

(b2) What is the space complexity of searchSS? Please explain your claim.

(c) Write a search algorithm *with DAC* “DACsearchSS(M, key)” such that if key is present in the Sorted Square M, your algorithm will print both row and column numbers as a pair. If the key is not present in the Sorted Square M, it will print “Not Found”.

(c1) What is the time complexity of DACsearchSS? Please explain your claim.

(c2) What is the space complexity of DACsearchSS? Please explain your claim.

(d1) Compare the algorithms searchSS and DACsearchSS using mathematical tools.

(d2) Compare the algorithms searchSS and DACsearchSS using empirical tools.

Write an extensive lab report outlining important concepts you learned through this example. In particular, comment of the appropriateness of DAC in this case.

QUESTION 2

Order them based on their complexity.

2^n , 2^(2n), 2^(n + 1), 2^(2^n) (Note: ^ stands for exponent operation. Example: 2^n = 2^n)