Write a program by defining an user defined function that is used to produce the rounded matrix as described in the above example. Find out the time complexity of your algorithm/function.

Write a recursive algorithm to compute the maximum element in an array of integers. You may assume the existence of a function "max(a, b)" that returns the maximum of two integers a and b.

Sol

Function FIND-ARRAY-MAX(A, n)
1: if (n = 1) then
2: return(A[1])
3: else
4: return(max(A[n], FIND-ARRAY-MAX (A, n − 1)))
5: end if

N.B

a) **Lab. Excerise (LE):** These are the **assignments** that ask each student to independently create small programs **during the lab time**.
b) **Home Exercise (HE):** These are the assignments to be done during lab time by each student if lab. assignments are completed before lab. time or may be assigned as **post-lab homework** and submitted in the next lab class
c) **Round Exercise (RE):** These are the **group assignments/small projects** to be done by each group round the time and to be submitted one copy per group at any time if asked to submit after one week of RE given or at the end of the course completion.

The approach of the Lab proceedings: LE-HE-RE

Solution Hints: Finding a feasible rounding can be reduced to finding a max flow in a flow network with nodes corresponding to rows and columns in A and capacities corresponding to row and sum columns. The total running time depends on the time to find a max flow. With a Ford-Fulkerson method, this is O(n4).

Solution:

Use a singly linked list to store those integers. To implement INSERT(x, S), we append the new integer to the end of the linked list. This takes $\Theta(1)$ time. To implement REMOVE-BOTTOM-HALF(S), we use the median finding algorithm taught in class to find the median number, and then go through the list again to delete all the numbers smaller or equal than the median. This takes $\Theta(n)$ time.

Suppose the runtime of REMOVE-BOTTOM-HALF(S) is bounded by cn for some constant c. For amortized analysis, use $\Phi = 2cn$ as our potential function. Therefore, the amortized cost of an insertion is $1 + \Delta\Phi = 1 + 2c = \Theta(1)$. The amortized cost of REMOVE-BOTTOM-HALF(S) is $cn + \Delta\Phi = cn + (-2c \times \frac{n}{2}) = 0$.

LAB - 4
Divide and Conquer Method

(Binary Search, Merge Sort, Quick Sort, Randomized Quick Sort)

PROGRAM EXERCISE

Lab. Exercise (LE)

4.1 Write a program to search an element x in an array of n integers using **binary search** algorithm that uses divide and conquer technique. Find out the best case, worst case and average case time complexities for different values of n and plot a graph of the time taken versus n. The n integers can be generated randomly and x can be choosen randomly, or any element of the array or middle or last element of the array depending on type of time complexity analysis.

4.2 Write a program to **sort a list** of n elements using the **merge sort** method and determine the time required to sort the elements. Repeat the experiment for different values of n and different nature of data (random data, sorted data, reversely sorted data) in the list. n is the user input and n integers can be generated randomly. Finally plot a graph of the time taken versus n.

Home Exercise (HE)

4.3 Write a program to use divide and conquer method to determine the time required to find the maximum and minimum element in a list of n elements. The data for the list can be generated randomly. Compare this time with the time taken by straight forward algorithm or brute force algorithm for finding the maximum and minimum element for the same list of n elements. Show the comparison by plotting a required graph for this problem.

4.4 Write a program that uses a divide-and-conquer algorithm/user defined function for the exponentiation problem of computing a^n where $a > 0$ and n is a positive integer. How does this algorithm compare with the brute-force algorithm in terms of number of multiplications made by both algorithms.