Assignment 1 Report

November 3, 2023

1 Question 1

(b)

i. The time complexity of the iterative approach is Big Theta(n). As the worst case scenario it will execute the multiplication n times which is Big Theta(n), while in the best case scenario when n=0 it return the result in Big Theta(1).

ii. The time complexity of the divide-and-conquer approach is Big Theta(log(n)). In the best case scenario where n=0 it returns 1 in Big Theta(1), while in the worst case as well as the average case scenarios the function reduces the problem by half in each recursive call as it divides 'n' by 2 (n // 2 or (n - 1) //2) and it continues to divide the problem until it reaches the base case (n=0), which results in logarithmic number of recursive calls and time complexity of Big Theta(log(n)).

iii. The recurrence relation is T(n) = T(n/2) + O(1).
using the Master Theorem : T(n)=aT(n/b)+f(n)
So in our case a=1, b=2, f(n)=O(1), $\log_2(1) = 0$
When we compare f(n) with $n^{\log_b(a)}$ we find that O(1)=n^0 , which means that it is the second case of the master theorem, therefore T(n)=Theta($n^{log_b(a)} log(n)$)= Theta($n^0 log(n)$)= Theta(log(n) .

(c)

i. The graph of running time of the iterative function power(a,n) for running it with n values 1,100,1000,10000,100000 and 1000000.

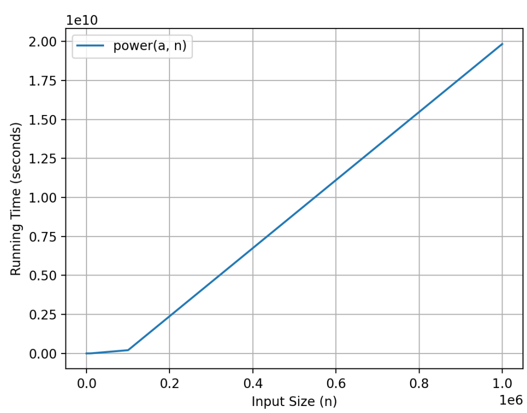

Figure 1: The running time of power(a,n) for different n values

ii. The graph of running time of the divide and conquer function power-div(a,n) for running it with n values1,500,1000,5000,10000,50000,100000,500000 and 1000000.

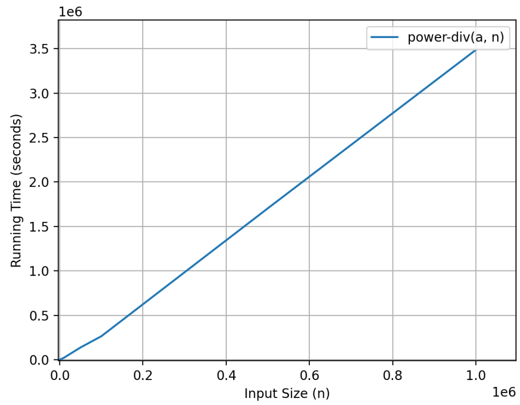

Figure 2: The running time of power-div(a,n) for different n values

(d)

i .The graph of the running time complexity in Figure 1 , represents Theta(n) which confirms the theoretical analysis in (b) i.

ii .The graph of the running time complexity in Figure 2 , represents Theta(log(n)) which confirms the theoretical analysis in (b) ii.
Note: the graphs aren't accurate due to many factors such as the random choice of n values and the running of many processes on the processor.

2 Question 2

(b)

i. The sum-pairs running time complexity can be analyzed into:

1.Sorting the array using merge-sort which has running time complexity of O(nlog(n)) as the sorting is done in two steps :
First: It divides the array into 2 subarrays recursively until each subarray contains 1 element only.(takes O(log(n))
Second: It merges (combines) the 2 subarrays of each recursive call into one sorted array. (takes O(n)) the second step is repeated for each recursive call so the total time complexity of the merge-sort is O(nlog(n))

2. Looping through the sorted array returned by the merge-sort and search for the complement of each element using binary-search. the time complexity of binary-search is O(log(n)) since it divides the search space in half with each step, and since it loops n times and with each time it uses binary-search the time complexity of this part is O(nlog(n)).

3.Adding the pair to the returned array if the complement is found which takes O(n).

In conclusion the running time complexity of sum-pairs is the maximum running time of all of the above which is O(nlog(n)) , in the best case and worst case scenarios the running complexity of sum-pairs is O(nlog(n)).

ii.

1.The recurrence relation of merge-sort is T(n) = 2T(n/2) + O(n)
using the Master Theorem : T(n)=aT(n/b)+f(n)

So in our case a=2, b=2, f(n)=O(n), $\log_2(2) = 1$
When we compare f(n) with $n^{\log_b(a)}$ we find that O(n)=n , which means that it is the second case of the master theorem, therefore T(n)=Theta($n^{log_b(a)} log(n)$)= Theta($n^1 log(n)$)= Theta(nlog(n) .

2.The recurrence relation of binary-search is T(n) = T(n/2) + O(1)
using the Master Theorem : T(n)=aT(n/b)+f(n)

So in our case a=1, b=2, f(n)=O(1), $\log_2(1) = 0$
When we compare f(n) with $n^{\log_b(a)}$ we find that O(1)=n^0 , which means that it is the second case of the master theorem, therefore T(n)=Theta($n^{log_b(a)} log(n)$)= Theta($n^0 log(n)$)= Theta(log(n)) .

(c) The graph of running time of sum-pairs for running it with random array sizes and random elements in figure 3, represents O(nlog(n)) which confirms the theoretical analysis in (b)

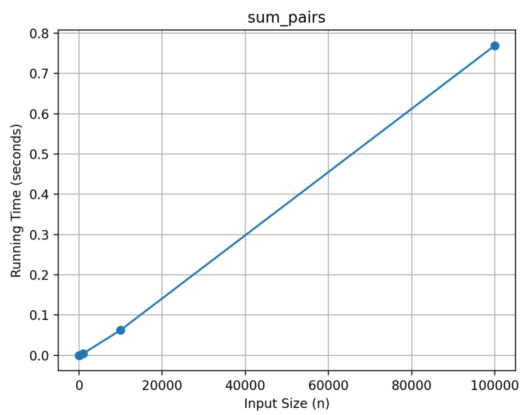

Figure 3: The running time of sum-pairs(array,n) for different values

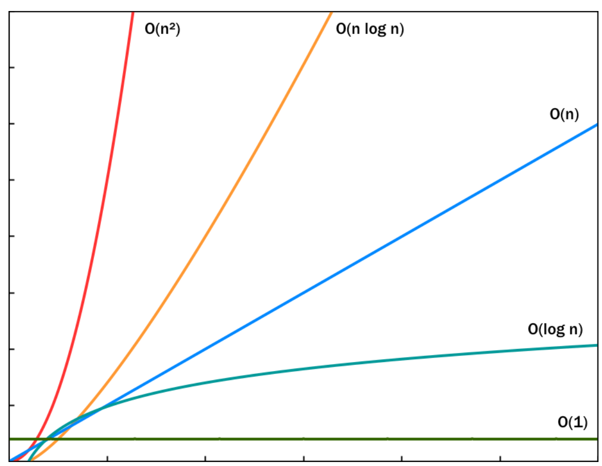

Figure 4: the Asymptotic diagram