

# Asymptotic Analysis Assignment Report

## Question 1

### a) Program Code

```
//Question 1

public static int powerIterative(int a,int n){
    int ans = 1;
    for(int i=1;i<=n;i++){
        ans*=a;
    }
    return ans;
}

public static int powerRecursive(int a,int n){
    if (n==0) return 1;
    if (n==1) return a;
    if(n%2==0) return powerRecursive(a, n/2) * powerRecursive(a, n/2);
    else return powerRecursive(a, (n/2) ) * powerRecursive(a, (n/2)+1);
}
```

### b) Determining Asymptotic Time Complexity

For the iterative method we can clearly see the number of steps taken from the diagram, the number of total steps will be  $2n+3$  which is  $\theta(n)$ .

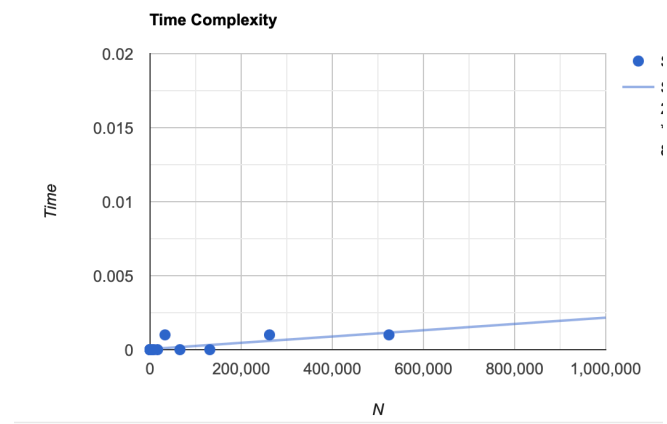
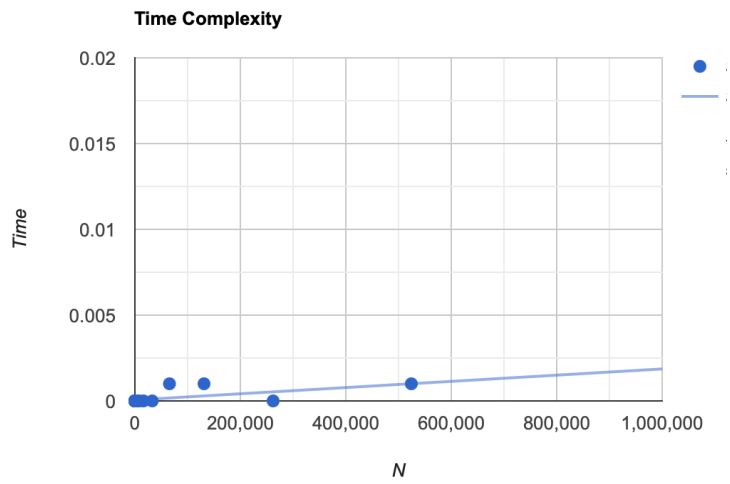
For the divide-and-conquer algorithm, solving the recurrence relation:

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + C & n > 1 \\ 1 & n = 2 \\ 1 & n = 0 \end{cases}$$

The diagram shows a recursion tree for  $T(n)$ . The root node is  $C$ . It has two children, each labeled  $C$ . Each of these children has two children of their own, resulting in a total of 8 leaf nodes. A vertical bracket on the left side of the tree is labeled  $\log_2 n$ , indicating the height of the tree.

Cost of ~~Tree~~ =  $C * 2^{\log_2 n} + \sum_{i=0}^{\log_2 n - 1} C 2^i$   
=  $Cn + Cn$   
Which is clearly  $\theta(n)$

c)



d) Clearly the experimental results confirm the theoretical analysis results in 1.(b).

## Question 2

a) Program Code

```
public static List<int[]> findPairsWithSum(int[] set, int target) {
    List<int[]> result = new ArrayList<>();

    mergeSort(set, left: 0, right: set.length - 1);

    for (int i = 0; i < set.length; i++) {
        int complement = target - set[i];
        if (binarySearch(set, complement, left: i + 1, right: set.length - 1))
            result.add(new int[]{set[i], complement});
    }

    return result;
}
```

(Check repo for rest of code)

b) The recurrence relation is essentially that of binary search:

CSEB B603 Analysis & Design of Algorithms

### Recursion Tree Method - Binary Search

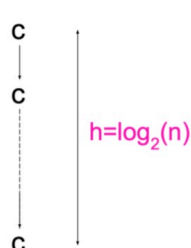


Diagram illustrating the Recursion Tree Method for Binary Search. The tree is a vertical chain of nodes labeled 'C', representing the recursive calls. The height of the tree is labeled as  $h = \log_2(n)$ .

26

---

CSEB B603 Analysis & Design of Algorithms

### Recursion Tree Method - Binary Search

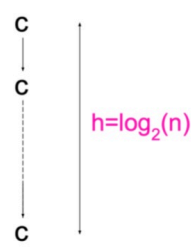


Diagram illustrating the Recursion Tree Method for Binary Search. The tree is a vertical chain of nodes labeled 'C', representing the recursive calls. The height of the tree is labeled as  $h = \log_2(n)$ .

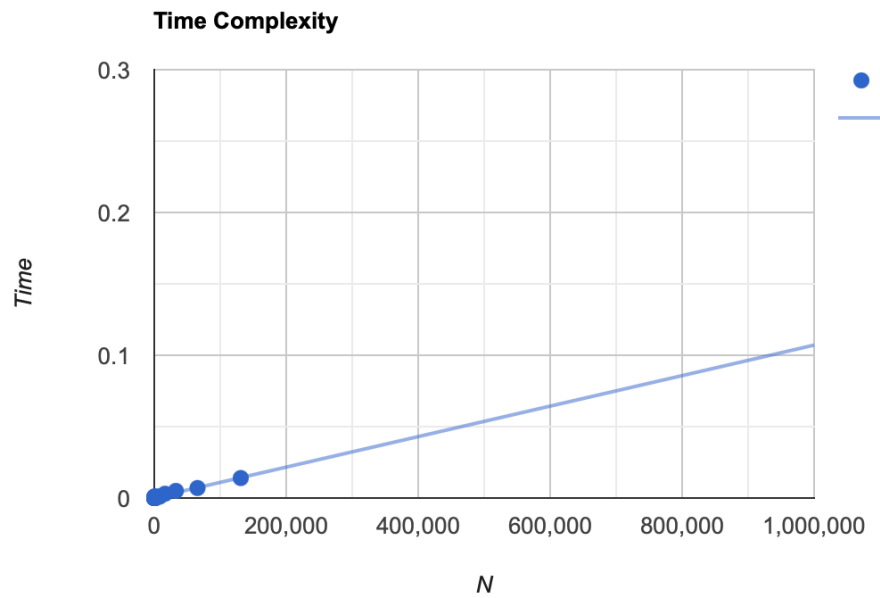
27

$$T(n) = \sum_{i=0}^{\log(n)} c = c(\log(n) + 1) = c \log(n) = \Theta(\log(n))$$

CSEB B603 Analysis & Design of Algorithms

The final time complexity will actually be  $(n * \log(n) + \log(n))$  since Merge Sort is done once and binary search is done  $n$  times. Overall  $\theta(n\log(n))$ .

c)



D) And again the experimental results confirm the theoretical analysis results in 2.(b).