***Analysis Report***

***Marwan Walid 52-0824***

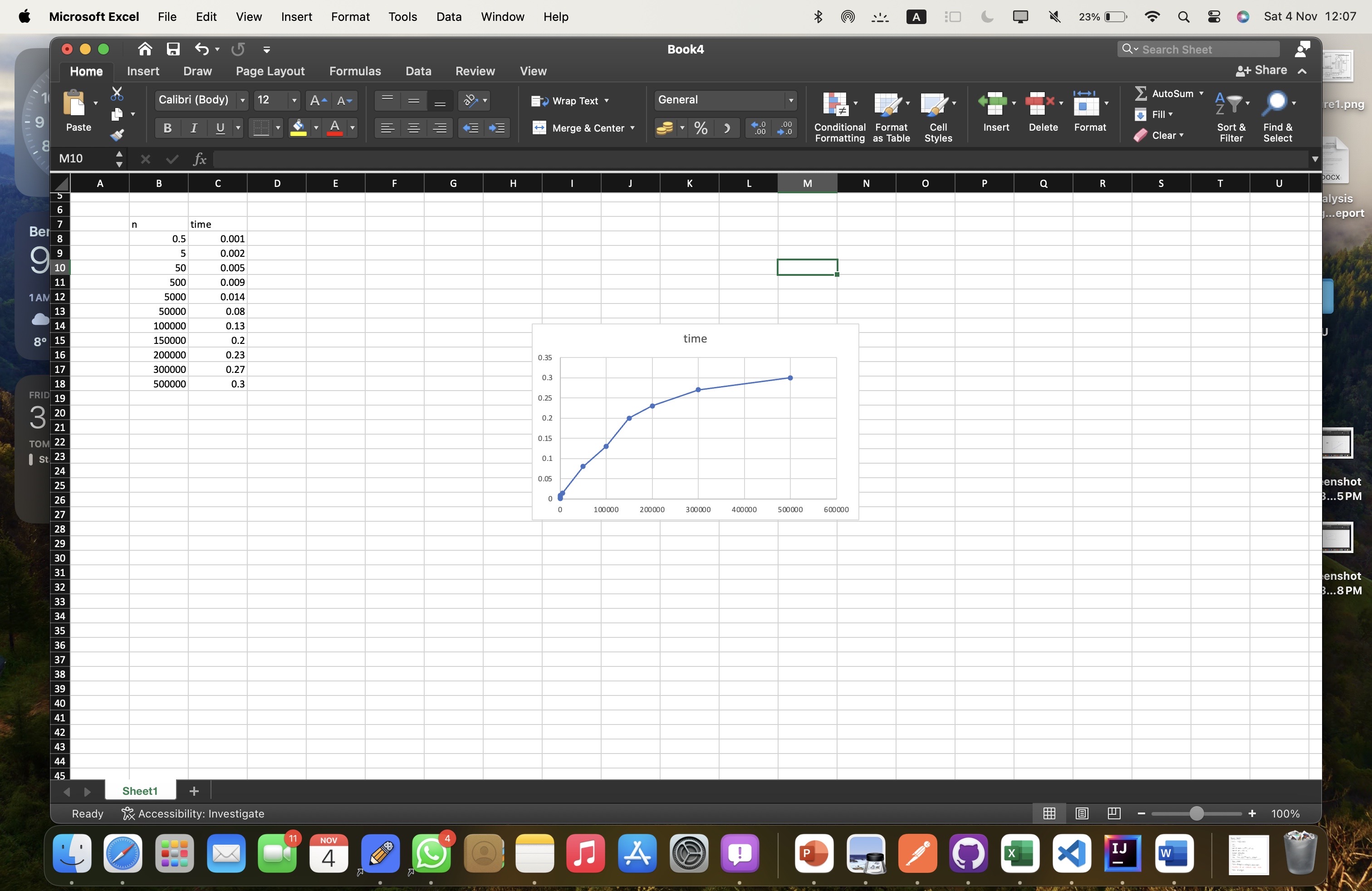
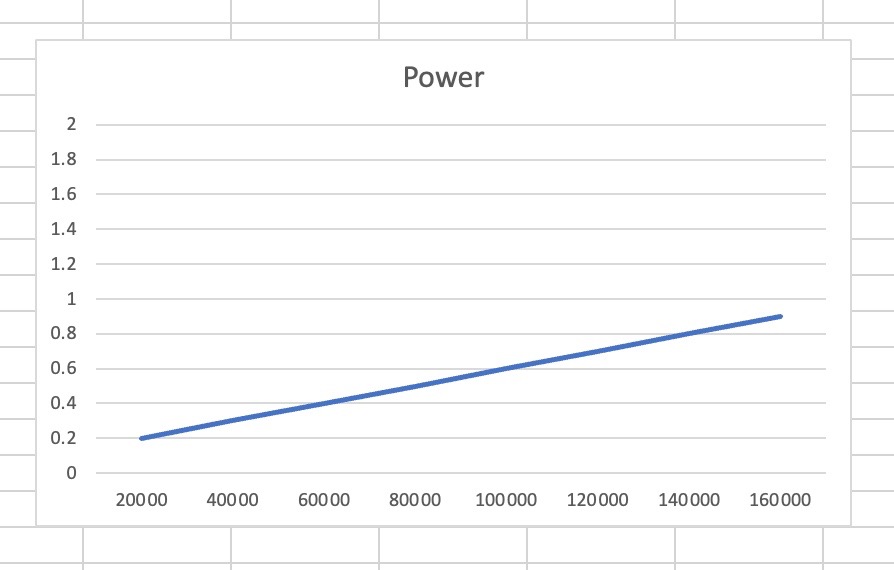
******

i. In the Naive Iterative Method, the time complexity is O(n) since it involves 'n' multiplication operations.

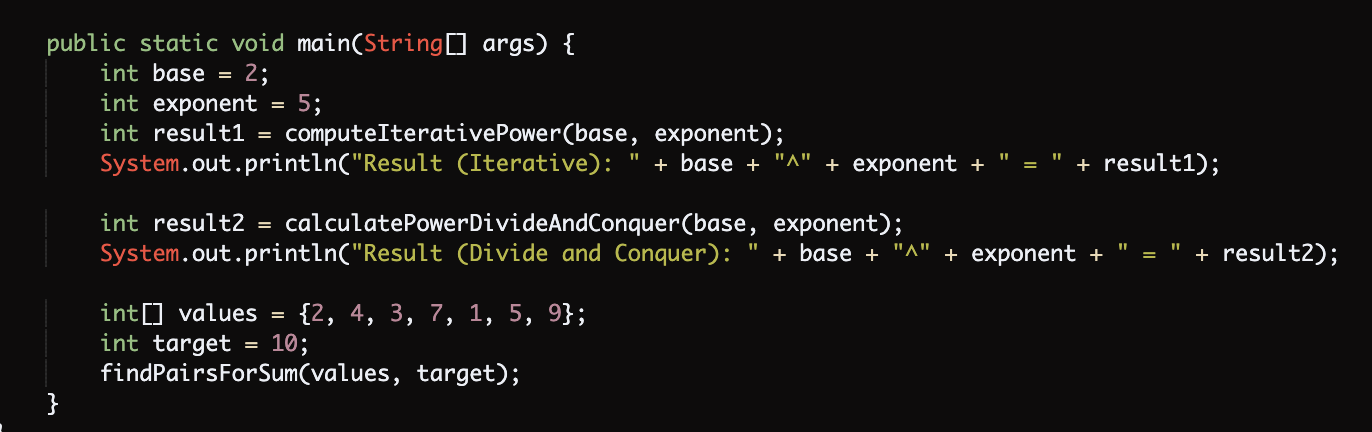
ii. Regarding the Divide and Conquer Method, we can express the time complexity as follows: T(n) = T(n/2) + O(1). According to the Master Theorem, it falls into the second case, resulting in a time complexity of O(log(n)).

Nhkjh

Xefxrfrfc

******

1D) The graph is like the predicted output

******

import java.util.Arrays;

public class AnalysisAssignment {

public static int computeIterativePower(int base, int exponent) {

int result = 1;

for (int i = 0; i < exponent; i++) {

result \*= base;

}

return result;

}

public static int calculatePowerDivideAndConquer(int base, int exponent) {

if (exponent == 0) {

return 1;

} else if (exponent % 2 == 0) {

int halfPower = calculatePowerDivideAndConquer(base, exponent / 2);

return halfPower \* halfPower;

} else {

int halfPower = calculatePowerDivideAndConquer(base, (exponent - 1) / 2);

return halfPower \* halfPower \* base;

}

}

public static void findPairsForSum(int[] data, int targetSum) {

Arrays.sort(data);

for (int i = 0; i < data.length; i++) {

int diff = targetSum - data[i];

int index = searchNumber(data, diff);

if (index != -1 && index != i) {

System.out.println(data[i] + " + " + data[index] + " = " + targetSum);

}

}

}

public static int searchNumber(int[] data, int target) {

int left = 0;

int right = data.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (data[mid] == target) {

return mid;

}

if (data[mid] < target) {

left = mid + 1;

} else {a

right = mid - 1;

}

}

return -1;

}

public static void main(String[] args) {

int base = 2;

int exponent = 5;

int result1 = computeIterativePower(base, exponent);

System.out.println("Result (Iterative): " + base + "^" + exponent + " = " + result1);

int result2 = calculatePowerDivideAndConquer(base, exponent);

System.out.println("Result (Divide and Conquer): " + base + "^" + exponent + " = " + result2);

int[] values = {2, 4, 3, 7, 1, 5, 9};

int target = 10;

findPairsForSum(values, target);

}

}