Tamir Krief, Iaian Milton, Blessing Abumere
COSC 336
9/26/2024

Assignment 3

Exercise 1.

(a) $T(n) = 3T(\frac{n}{4}) + 3$. a = 3, b = 4, $f(n) = 3$. $n^{\log_4^3}$ vs. 3. $n^{\log_4^3}$ is the winner because $f(n) = 3$ is a constant function and doesn't grow. Therefore the Theta evaluation for the recurrence is $T(n) = \Theta(n^{\log_4^3})$.

(b) $T(n) = 2T(\frac{n}{2}) + 3n$. a = 2, b = 2, $f(n) = 3n$. $n^{\log_2^2}$ or n vs. $3n$. This is a tie because $f(n) = 3n$ has proportional growth to n. Therefore the Theta evaluation for the recurrence is $T(n) = \Theta(n \log n)$.

(c) $T(n) = 9T(\frac{n}{3}) + n^2$. a = 9, b = 3, $f(n) = n^2$. $n^{\log_3^9}$ or n^2 vs. n^2. This is a tie because both functions are equivalent. Therefore the Theta evaluation for the recurrence is $T(n) = \Theta(n^2 \log n)$.

Exercise 2:

(a) $T(n) = \Theta(2^n)$

(b) $T(n) = \Theta(n)$

(c) $\Theta(n \log n)$

(d) $\Theta(n)$

Programming Task.

[ht]

Table 1: Programming Task Results

7,3,8,1,5	4
input-3.4.txt	248876
input-3.5.txt	24792876

```java
/** Tamir Krief, Iaian Milton, Blessing Abumere  */

//key: L[I] < R[j]
import java.io.FileReader;
import java.util.Scanner;

public class Assignment3 {
    public static void main(String[] args){
        System.err.println(UP_Pairs(new int[]{7,3,8,1,5}) + " UP pairs in [7,3,8,1,5]");
        System.out.println("input-3.4.txt UP_Pairs: " + UP_Pairs(inputFile("input-3.4.txt")));
        System.out.println("input-3.5.txt UP_Pairs: " + UP_Pairs(inputFile("input-3.5.txt")));
    }

    public static int UP_Pairs(int[] arr){
        return merge(arr,0,arr.length);
    }

    /** modified merge method from book to return UP pairs
     * @param A the main array
     * @param  p left index
     * @param  r right index
     * @return UP pair count
     */
    public static int merge(int[] A, int p , int r){
        if (A == null || A.length <= 1 || p >= r) return 0; //if null or length less than 1 or left
index bigger than right index,

        final int MID = A.length/2;
        final int n1 = MID - p ;
        final int n2 = r - MID;

        int[] L = new int[n1];
        int[] R = new int[n2];

        //puts values in L and R from main Aay
        for (int i = 0; i < n1; i++)
            L[i] = A[p + i];

        for (int j = 0; j < n2; j++)
            R[j] = A[MID + j];

        //sorts L and R
        int pairs = merge(L,0,L.length) + merge(R,0,R.length);

        //i =left index, j = right index, k = main array index
        int i = 0;
        int j = 0;
        int k = p;

        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                A[k++] = L[i++];
                pairs += n2 - j;
            }
            else {
                A[k++] = R[j++];
            }

        }

        while (i < n1) {
            A[k++] = L[i++];
        }
```

```java
        while (j < n2) {
            A[k++] = R[j++];
        }

        return pairs;
    }

    /** reads the file and converts it to an int array */
    public static int[] inputFile(String filename) {
        try (Scanner input = new Scanner(new FileReader(filename))) {

            //size is the first number
            final int n = input.nextInt();

            //reads ints in the file
            int[] A = new int[n];
            for (int i = 0; i < n; i++)
                A[i] = input.nextInt();

            return A;
        }catch(java.io.FileNotFoundException e) {
            System.err.println("File not found: '" + filename + "'");
        }

        return null;
    }


}
```