Tamir Krief, Iaian Milton, Blessing Abumere
COSC 336
08/29/2024

Assignment 4

Exercise 1.

For each of the following functions, give a $\Theta(t(n))$ estimation with the simplest possible $t(n)$ (for example $3n^2 + 5n \log n = \Theta(n^2)$).

1. $13n^2 - 2n + 56$
   $t(n) = \Theta(n^2)$

2. $2.5 \log n + 2$
   $t(n) = \Theta(\log n)$

3. $n(12 + \log n)$
   $n(12 + \log n) = 12n + n \log n$
   $t(n) = \Theta(n \log n)$

4. $1 + 2 + 3 + \ldots + 2n$

   $S_n = \frac{n(n+1)}{2}$
   $S_{2n} = \frac{2n(2n+1)}{2}$
   $\frac{2n(2n+1)}{2} = n(2n + 1) = 2n^2 + n$
   $t(n) = \Theta(n^2)$

5. $1 + 2 + 3 + \ldots + n^2$

   $S_n = \frac{n(n+1)}{2}$
   $S_{n^2} = \frac{n^2(n^2+1)}{2}$
   $\frac{n^2(n^2+1)}{2} = \frac{n^4+n^2}{2}$
   $t(n) = \Theta(n^4)$

6. $\log(n^3) + 10$
   $\log(n^3) = 3 \log n$
   $t(n) = \Theta(\log n)$

7. $\log(n^3) + n \log n$
   $\log(n^3) = 3 \log n$
   $t(n) = \Theta(n \log n)$

8. $n \log(n^3) + n \log n$
   $3n \log n + n \log n = 4n \log n$
   $t(n) = \Theta(n \log n)$

9. $2^{2 \log n} + 5n + 1$
   $2^{2 \log n} = (2^{\log n})^2 = n^2$
   $t(n) = \Theta(n^2)$

Exercise 2.

1. Evaluate the following postfix arithmetic expression: 10 3 4 − 5 ∗ /

   10 3 4 − 5 ∗ /
   = 10 −1 5 ∗ /
   = 10 −5 /
   = −2

2. Convert the following infix arithmetic expression to postfix notation: $(((2 + 3) * 5) - 15)$

   $= (((2\ 3\ +) * 5) -\ 15)$
   $= (((2\ 3\ +)5\ *) -\ 15)$
   $= (((2\ 3\ +)5\ *)15\ -)$
   $= 2\ 3\ +\ 5\ *\ 15\ -$

Exercise 3.

1. Write the recurrence for the runtime $T_A(n)$ of algorithm A, and solve the recurrence to find a $\Theta(\cdot)$ estimation of $T_A(n)$.

   $T_A(n) = T_A(\frac{n}{2}) + O(1)$
   $a = 1,\ b = 2,\ k = 0,\ p = 0; f(n) = 1, f(n) = \Theta(n^k\ log^p\ n)$
   $a = b^k,\ T(n) = \Theta(n^{\log_b a}\ \log^{p+1} n)$
   $T(n) = \Theta(n^{\log_b a}\ \log^{p+1} n)$
   $T(n) = \Theta(n^0 \log^{0+1} n)$
   $T(n) = \Theta(\log n)$

2. Write the recurrence for the runtime $T_B(n)$ of algorithm B, and solve the recurrence to find a $\Theta(\cdot)$ estimation of $T_B(n)$.

   $T_B(n) = 2T_B(\frac{n}{2}) + O(1)$
   $a = 2,\ b = 2,\ k = 0,\ p = 0,\ f(n) = 1,\ f(n) = \Theta(n^k\ log^p\ n)$
   $a > b^k,\ T(n) = \Theta(n^{\log_b a})$
   $T(n) = \Theta(n^1)$
   $T(n) = \Theta(n)$

3. Which algorithm is faster? (Note: There is a huge difference between T_A and T_B.)
   Algorithm A is faster than Algorithm B logarithmic growth $\Theta(\log n)$ is much slower than linear time $\Theta(n)$

Exercise 4.

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

Table 1: Programming Task 1 Results

Input	Max Revenue	Cuts yielding Revenue
1, 5, 8, 9, 10, 17, 17, 20, 24, 30	141	[10]
input-4.2.txt:	123	[9, 5, 1]
input-4-3.txt	59	[2, 2, 2, 2, 2, 2, 2, 1]

```java
/** Tamir Krief, Iaian Milton, Blessing Abumere  */
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Scanner;
import java.io.FileNotFoundException;

public class Assignment4{
    /** reads the file and converts it to an int array */
    public static int[] inputFile(String filename) {
        try (Scanner console = new Scanner(new FileReader(filename))) {
            //size is the first number
            final int n = console.nextInt();

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

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

        return null;
    }

    //Extended-Bottom-Up-Cut-Rod(p,n) method to compute max revenue and cuts
    public static int[] extBottomUpCutRod(int[] p, int n) {
        int[] r = new int[n + 1];
        int[] s = new int[n + 1];

        r[0] = 0; //base case

        for (int j = 1; j <= n; j++) {
            int q = Integer.MIN_VALUE;
            for (int i = 1; i <= j; i++) {
                if (q < p[i - 1] + r[j - 1]) {
                    q = p[i - 1] + r[j - 1];
                    s[j] = i;
                }
            }
            r[j] = q;
        }

        return new int[]{r[n], s[n]};
    }

    //Print-Cut-Rod-Solution(p,n) method to print the solution
    public static void printCutRodSolution(int[] p, int n) {
        int[] r = new int[n + 1];
        int[] s = new int[n + 1];
        ArrayList<Integer> cuts = new ArrayList<>();

        r[0] = 0;

        for (int j = 1; j <= n; j++) {
            int q = Integer.MIN_VALUE;
            for (int i = 1; i <= j; i++) {
                if (q < p[i - 1] + r[j - 1]) {
                    q = p[i - 1] + r[j - 1];
                    s[j] = i;
                }
            }
            r[j] = q;
```

```
        }

        while (n > 0) {
            cuts.add(s[n]);
            n = n - s[n];
        }

        System.out.println("Maximum Revenue: " + r[r.length - 1]);
        System.out.println("Cuts yielding this revenue: " + cuts);
    }

    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("Enter the filename: ");
            String filename = console.nextLine();
        int[] prices = inputFile(filename);

        console.close();
        if (prices == null) {
            System.out.println("Error reading the input file.");
            return;
        }

        int rodLength = prices.length;

        printCutRodSolution(prices, rodLength);

    }
}
```