CMP-5014Y Coursework Assignment 1

100175876 (eau16vfu)

Sun, 19 May 2019 15:48

PDF prepared using PASS version 1.15 running on Windows 10 10.0 (amd64).

☑ I agree that by submitting a PDF generated by PASS I am confirming that I have checked the PDF and that it correctly represents my submission.

Contents

Algorithm-Analysis.pdf

 (Included PDF starts on next page.)

Algorithm Design and Analysis Q1

Informal Algorithm:

Initialize an array of integers which has a size of the length of the current matrix, then loop through this array and set the current position in the matrix to the maximum value of an integer, this prevents null values from being returned. Then get the minimum value in the current matrix and add it to the integer array. Loop through until all distances have been added then return the array of integers.

Formal Algorithm:

```
static int[] printCity(int mat[][], final int cityCount)
{
    int[] result = new int[cityCount];
    for(int i = 0; i < cityCount; i++) {
        mat[i][i] = Integer.MAX_VALUE;
        result[i] = Arrays.stream(mat[i]).min().getAsInt();
    }
    return result;
}
```

Worst Case Analysis:

Fundamental Operation:

```
result[i] = Arrays.stream(mat[i]).min().getAsInt();
```

Describe the Case:

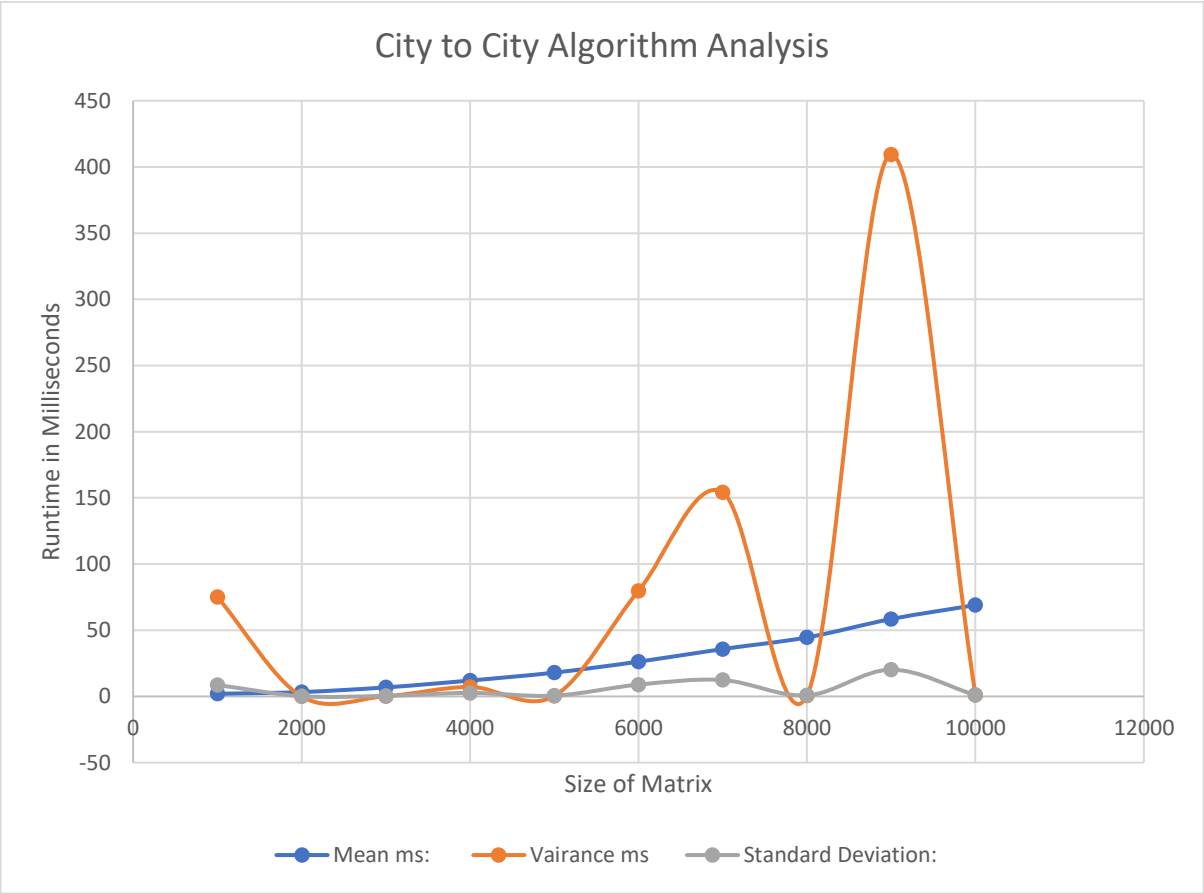
As this algorithm uses a simple array that is not dynamic it has a worst-case of O(n) and average case of O(1). This means it is a linear algorithm as the number of operations is equal to the size of n.

Form the Runtime Function:

t(n) = n is the worst-case for this algorithm, as the fundamental operation Is performed n times.

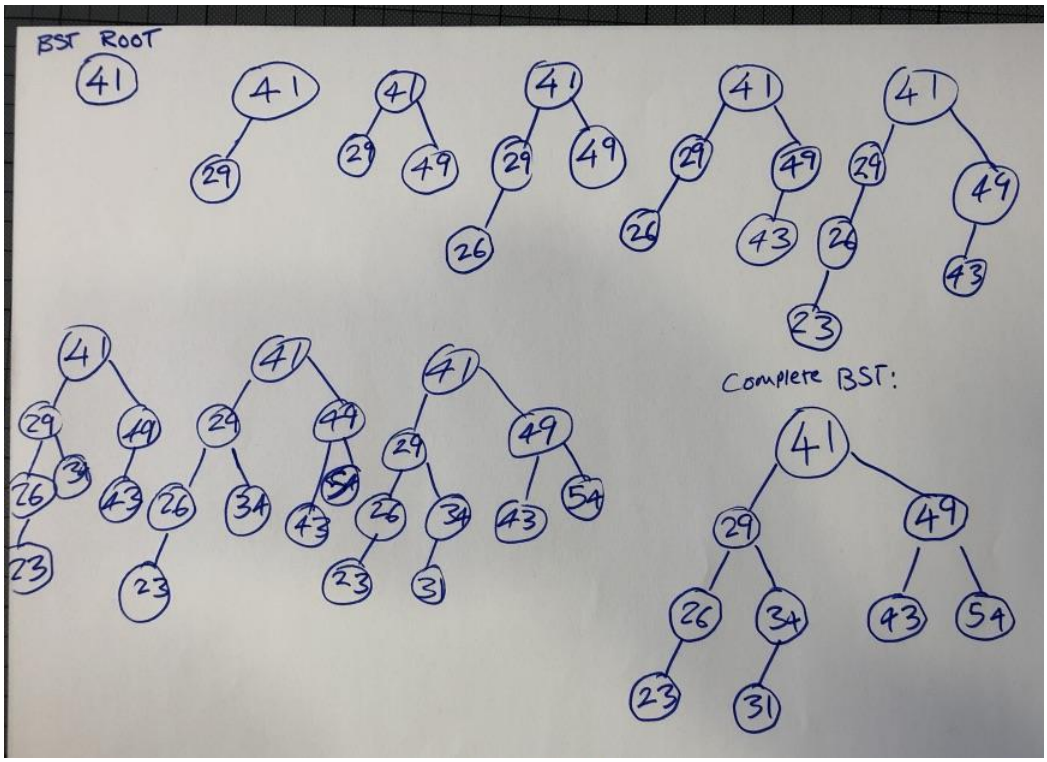
Characterise the Runtime Function:

Linear O(n)

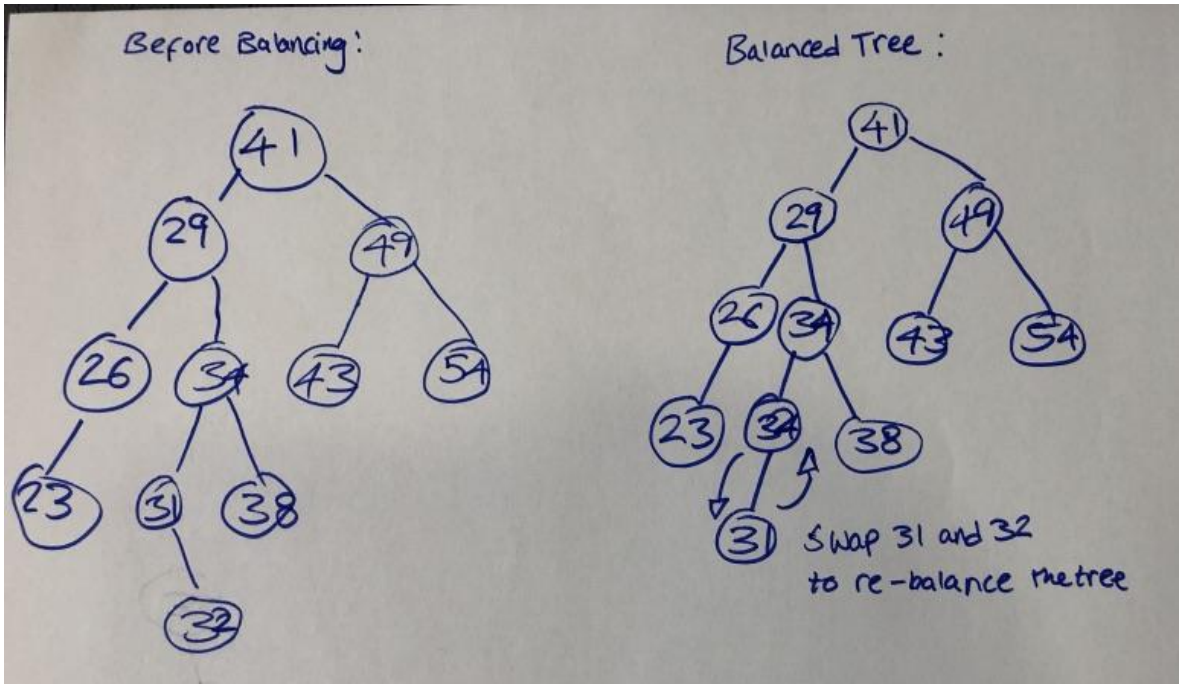


Binary Search Trees Q2

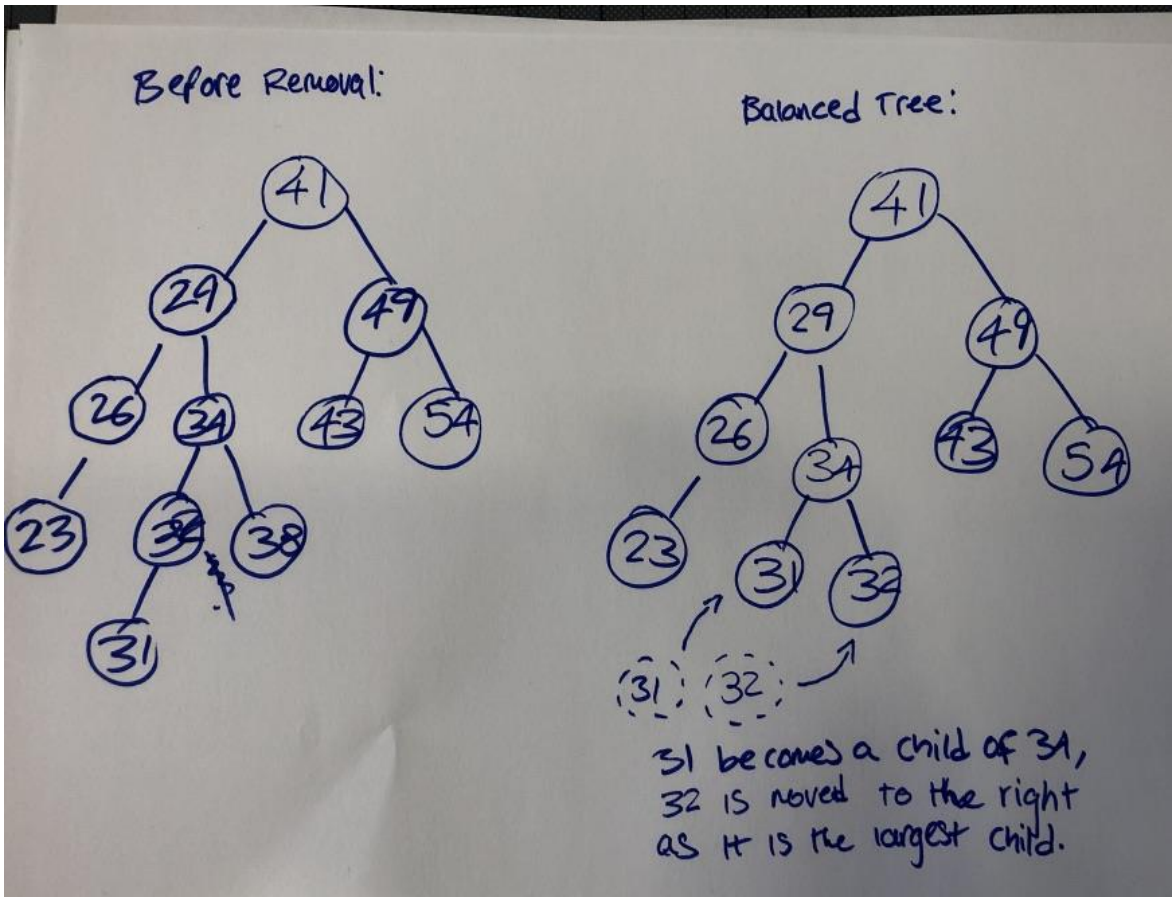
1.



2.


Before Balancing:
41
29
49
26
34
43
54
23
31
38
32
Balanced Tree:
41
29
49
26
34
43
54
23
34
38
31
Swap 31 and 32
to re-balance the tree

3.


Before Removal:
41
29
49
26
34
43
54
23
38
31
Balanced Tree:
41
29
49
26
34
43
54
23
31
32
31
32
31 becomes a child of 34,
32 is moved to the right
as it is the largest child.

Main.java

```
import java.util.*;
import java.util.Arrays;
import java.util.stream.*;

class Main {


    static int[] printCity(int mat[][], final int cityCount)
    {
        int[] result = new int[cityCount];
        for(int i = 0; i < cityCount; i++) {
            mat[i][i] = Integer.MAX_VALUE;
            result[i] = Arrays.stream(mat[i]).min().getAsInt();
        }
        return result;
    }

    private static void testRuntime(int n) {
        Random r = new Random();
        int reps = 100; //Initialize number of repetitions to 100

        while(n <= 10000){
            System.out.println("--------------------------------");
            System.out.println("Current Matrix Size: " + n);
            System.out.println("--------------------------------");
            int[][] numbers = new int[n][n];

            for(int j = 0; j < n; j++) {
                for(int i = 0; i < n; i++) {
                    if(i == j) numbers[i][j] = 0;
                    else numbers[i][j] = Math.abs(r.nextInt());
                }
            }

            // Record mean and std deviation of performing an operation
            // reps times
            double sum = 0;
            double sumSquared = 0;

            for (int i = 0; i < reps; i++) {
                long t1 = System.nanoTime(); //Initialize start time

                printCity(numbers, n);

                //Get runtime in nanoseconds
                long t2 = System.nanoTime() - t1;

                // Convert to milliseconds to make the result more readable
                sum += (double) t2 / 1000000.0;
                sumSquared += (t2 / 1000000.0) * (t2 / 1000000.0);
            }
            //Calculate the mean time taken for each rep
            double mean = sum / reps;

            //Calculate the variance to see the range of the set from its mean
            double variance = sumSquared / reps - (mean * mean);

            //Calculate standard deviation to see how the mean runtime
            //can variate
            double stdDev = Math.sqrt(variance);

```

```
                // Print results to console
63              System.out.println("Mean: " + mean);
                System.out.println("Variance: " + variance);
65              System.out.println("Standard Deviation: " + stdDev);
                System.out.println();
67
                n+= 1000; //increase size of the matrix
69
//                if (n < 1000) {
71//                   n += 100;
//                }
73//                else {
//                   n += 1000;
75//                }
            }
77          System.out.println("Exit");
        }
79
        // Driver program to test above
81      public static void main(String args[])
        {
83          int mat[][] = { { 0, 58, 184, 271, 378, 379 },
                            { 58, 0, 167, 199, 351, 382 },
85                          { 184, 167, 0, 43, 374, 370 },
                            { 271, 199, 43, 0, 394, 390 },
87                          { 378, 351, 374, 394, 0, 47 },
                            { 379, 382, 370, 390, 47, 0 },
89                  };
            //Size of Matrix
91          int n = mat.length;

93//          printCity(mat, n);

95          testRuntime(1000);

97      }
    }
```