Ugur AKYEL
20190808020

HOMEWORK 2 Algorithms
javac hw2_20190808020.java
java hw2_20190808020 <inputsize>

2) my greedy function.

```
public static void greedy_min_weight(Tree t, String s, int min_weight){
    int i = 0;
    StringBuilder sBuilder = new StringBuilder(s);
    while (t.getList(i).getRight_node() != null && t.getList(i).getLeft_node() != null){
        int l_weight = t.getList(i).getLeft_node().getWeight()
                + t.getList(i).getLeft_edge().getWeight();
        int r_weight = t.getList(i).getRight_node().getWeight()
                + t.getList(i).getRight_edge().getWeight();
        if (l_weight <= r_weight){
            min_weight += l_weight;
            sBuilder.append("-").append(t.getList(i).getLeft_node().getId());
             i = i*2+1;
        }else {
            min_weight += r_weight;
            sBuilder.append("-").append(t.getList(i).getRight_node().getId());
            i = i*2+2;
        }
    }
    s = sBuilder.toString();
    System.out.println("GREEDY: Min total weight path includes nodes " + s + " with total weight " + min_weight);
}
```

If our nodes is not leaf we are going to down according to which childs weight and edge's weight summation is minimum with respect to other. And also always we choose minimum summation path with greedily.

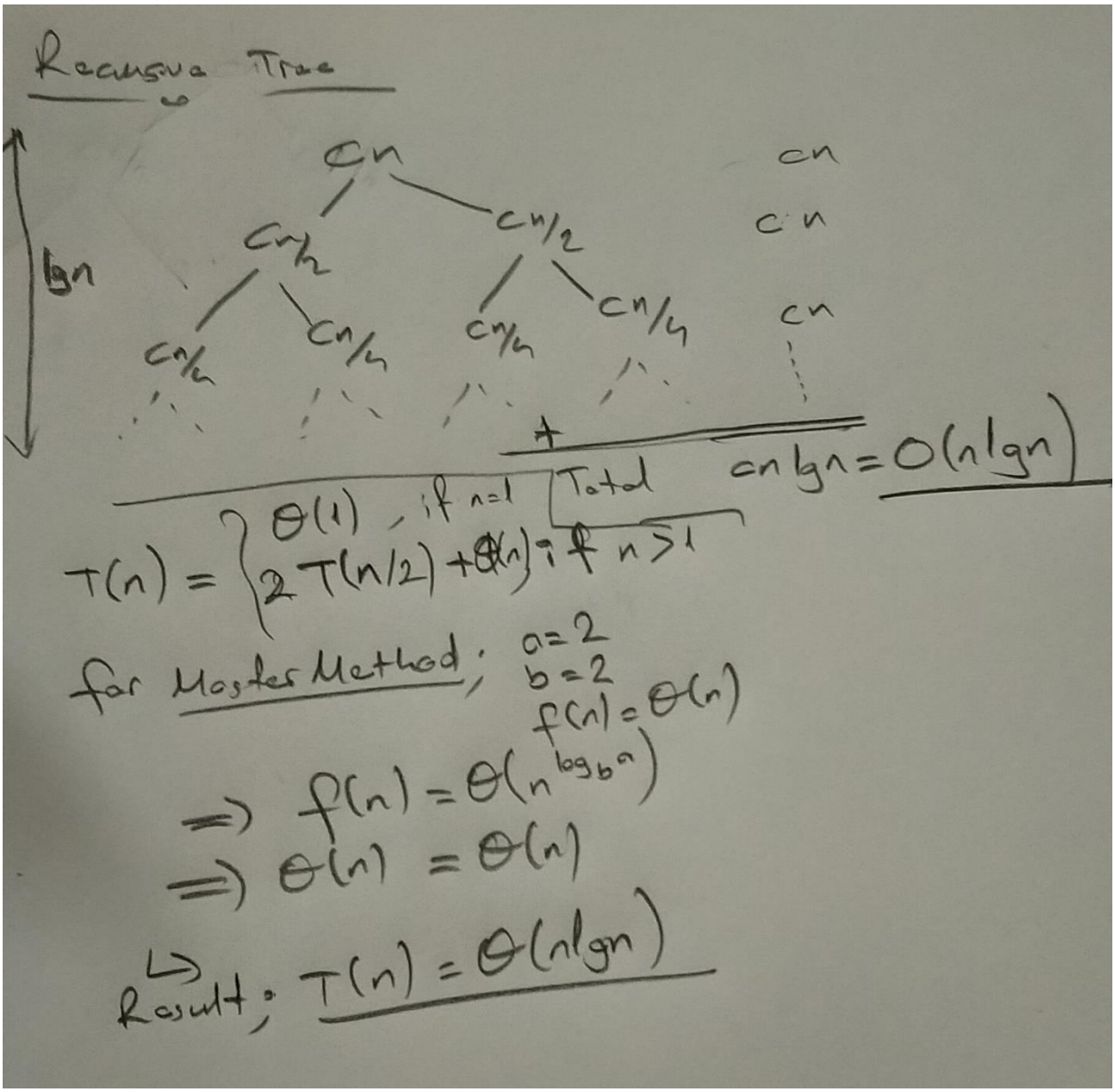
3)

```
public static int recursive_min_weight_allpaths(Node root, String s, int weight, HashMap<String, Integer> paths){

    if (root.getRight_node() == null && root.getLeft_node() == null){
        paths.put(s, weight);

        return weight;
    }else {
        if (root.getRight_node() != null){
            Node right_node = root.getRight_node();

            return Math.min(recursive_min_weight_allpaths(root.getLeft_node(),  s: s + "-" + root.getLeft_node().getId(),
                    weight: weight + root.getLeft_node().getWeight()
                            + root.getLeft_edge().getWeight(), paths),
                    recursive_min_weight_allpaths(root.getRight_node(),  s: s + "-" + root.getRight_node().getId(),
                            weight: weight + root.getRight_node().getWeight() + root.getRight_edge().getWeight(), paths));
        }
        return recursive_min_weight_allpaths(root.getLeft_node(),  s: s + "-" + root.getLeft_node().getId(),
                weight: weight + root.getLeft_node().getWeight()
                        + root.getLeft_edge().getWeight(), paths);
    }
}

public static void recursive_get_min_path(HashMap<String, Integer> paths){
    int minValueInMap=(Collections.min(paths.values()));
    for (Entry<String, Integer> entry : paths.entrySet()) {
        if (entry.getValue()==minValueInMap) {
            System.out.println("RECURSIVE: Min total weight path includes nodes "
                    + entry.getKey() + " with total weight " + minValueInMap);
            break;
```

On the above function shows us how to find recursively each paths how to choose mintotal weight path.
Pseuodocode;

Recursive_get_all_paths(node, s, w, map)

```
If node.right == null and node.left == null
        map.put(s, w)
return w
else
        if node.right != null
                return min(recursive_all_paths(node.left, s+leftid, w+node.left+node.leftedge)+
                  recursive_all_paths(node.right, s+rightid, w+node.right+node.rightedge) )
        else
                return recursive_all_paths(node.left, s+leftid, w+node.left+node.leftedge)+
```

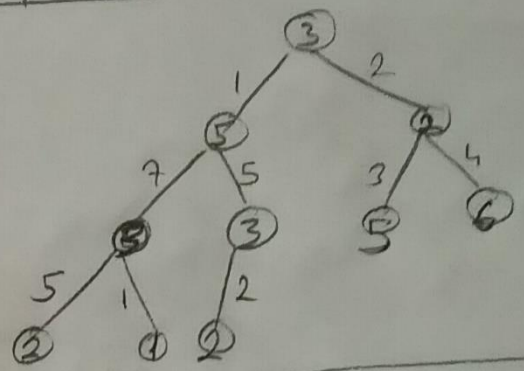


4)Dynamic solution; I show code, pseudocode my tabulation work and time complexity analysis..

```
public static void dynamic_min_path(Tree t){
    int n = t.getList().length;
    int min = Integer.MAX_VALUE;
    String nodes = "";
    int[][] c = new int[n][n];

    for (int i = n-1; i >= n/2; i--) {
        int j = i;
        StringBuilder s = new StringBuilder("" + t.getList(j).getId());
        int weight = t.getList(j).getWeight();
        do{
            weight += t.getList(j).getParent().getWeight() + t.getList(j).getParent_edge().getWeight();
            c[i][t.getList(j).getParent().getId()] = weight;
            j = (j-1)/2;
            s.append("-").append(j);
        }while (t.getList(j).getParent() != null);

        min = Math.min(min, weight);
        if (min == c[i][0]){
            nodes = s.toString();
        }
    }
    System.out.println("DYNAMIC: Min total weight path includes nodes " + reverse(nodes) + " with total weight " + min);
}
```

Dynamic Solution

example tree



c[i][j]

↳ c[leaf][leaf's parent]

C	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5	14		9							
6	18		13							
7	23	19		7						
8	21	17		5						
9	21	17			7					

c[9][4]

↳ leaf olan

9'.node'dan 4'e gidiş parentına gidişin değeri: d (edge ile aradaki)

c[9][0]

↓

9.node'dan root'a gidişin weight'i idi.

⇒ c[i][0] root'a giden yollarin weightni verir.

Psaudocode:

```
Dynamic_sum_Path(Tree t)
    n ← t.size
    min ← Infinity(Positive)
    nodes ← ""
    for i ← n-1 to n/2
        j ← i
        s ← t.id
        weight ← t(j.weight
        do
            w ← t(parent(j)).w + t(parent(j).w
            c[i][t(j).parent] = w
            j ← (j-1)/2
            s ← s
        while t(j).parent != null
        min ← Min(min, w)
        if (min ← c[i][0]
            nodes ← s
```

$$T(n) = 1+1+1+ n/2 + n/2 + n/2 + n/2$$

$$+ (n/2 \cdot \log n) \times 4 + n/2$$

$$T(n) = 3 + \frac{5n}{2} + 2n\log n = \underline{O(n\log n)}$$

5)**Greedy algorithm fail to for optimal solution in this work**

```
GREEDY: Min total weight path includes nodes 0-2-6-13-28-58-118-237-475-951-1903-3808-7617 with total weight 207
RECURSIVE: Min total weight path includes nodes 0-2-5-11-24-49-100-202-406-814-1629-3259-6519 with total weight 183
DYNAMIC: Min total weight path includes nodes 0-2-5-11-24-49-100-202-406-814-1629-3259-6519 with total weight 183
```

This is the table for 2*3 matrix, for solution different input sizes, my recursive time analyses beat the dynamic solution and I investigated this stiation so reached some optimal timing capacity process for about the compiler. (nanoseconds)

```
----------FOR DIFFERENT 3 INPUT SIZES SOLUTION-----------
RECURSIVE: Min total weight path includes nodes 0-1-3-8-18-37-76 with total weight 78
DYNAMIC: Min total weight path includes nodes 0-1-3-8-18-37-76 with total weight 78
RECURSIVE: Min total weight path includes nodes 0-2-6-13-28-58-118-237-475-952 with total weight 138
DYNAMIC: Min total weight path includes nodes 0-2-6-13-28-58-118-237-475-952 with total weight 138
RECURSIVE: Min total weight path includes nodes 0-1-3-7-16-34-69-140-282-566-1133-2268-4537-9075 with total weight 160
DYNAMIC: Min total weight path includes nodes 0-1-3-7-16-34-69-140-282-566-1133-2268-4537-9075 with total weight 160
-------------------------------------------------------------
Input Size |         100    |        1000    |       10000    |
-------------------------------------------------------------
Recursive  |      164700 ns |      504800 ns |     4213700 ns |
-------------------------------------------------------------
Dynamic    |      189300 ns |     1051200 ns |   120685600 ns |
-------------------------------------------------------------
```