**Jyoti Vaswani (NUID -001005711)**

**INFO 6205**

**Program Structures and Algorithms**

**Fall 2020**

**Assignment No -4**

**Task :** Benchmarking done for path halving and two loop path compression algorithms used in Union-Find.

Note: Kindly follow the package structure for smooth execution.

1. **I have added Benchmark\_Timer\_UF.java for providing input:**

**package edu.neu.coe.info6205.util;**

* **main() –** I have added the loop to generate doubled values of n i.e. number of nodes in the tree.
* **performance()-** this method initiates the benchmarking by initializing fRun for both path halving and two loops method.

1. **I have added UF\_Benchmark\_Client.java for the methods listed below:**

**package edu.neu.coe.info6205.util;**

* **benchmark\_performance\_two\_loops () –** creates random values for find-union operations as per the input n.
* **benchmark\_performance\_path\_halving ()-** creates random values for find-union operations as per the input n.

1. **I have added UF\_path\_halving.java for the methods listed below:**

**package edu.neu.coe.info6205.union\_find;**

* **find() –**finds the root of the input node and performs path – halving on it by connecting every other node encountered while traversing to its respective grandparent node.
* **mergecomponents()** – connects the two input nodes(when not already connected as invoked by union()).
* **averageDepthAllNodes()** – finds average depth of all the nodes -- in the tree formed after all the nodes have been connected and are present in single component.

Note : This method is not benchmarked and is invoked post formation of complete tree.

1. **I have added UF\_HWQUPC.java for the methods listed below:**

**package edu.neu.coe.info6205.union\_find;**

* **find() –**finds the root of the input node and performs path compression on it by connecting every other node encountered while traversing to root node.
* **mergecomponents()** – connects the two input nodes(when not already connected as invoked by union()).
* **averageDepthAllNodes()** – finds average depth of all the nodes -- in the tree formed after all the nodes have been connected and are present in single component.

Note : This method is not benchmarked and is invoked post formation of complete tree.

1. **I have used Timer.java as is.**

**package edu.neu.coe.info6205.util;**

Note: the methods of Timer .java were coded in previous assignment- No. 2:

  

 

1. **Output:**

Below are the screenshots of output generated by Benchmark\_Timer\_UF.java for different input values using doubling on n for path halving and two loops method. (number of nodes-n).

n goes like 1000,2000, 4000, 8000,..8192000

**  **

**  **

****

1. **Relationship Observations and Conclusion:**

* **Space complexity:** For both the union find improvement algorithms- path halving and two loop path compression, the space required is exactly same. As for ‘n’ nodes there would be an array of size ‘n’ required to store their pointing to respective parent node. Hence the space complexity for both is O(n).
* **Time complexity and asymptotic nature**: Both the improvement algorithms exhibit similar (linear) asymptotic nature when it comes to their running time.

For a sequence of M union and find operations over N nodes it takes O(N + M log\* N) time.

Here log\* N is the inverse Ackermann function that grows so slowly that it is considered as a constant for practical values of this world. So both the algorithms show linear growth for doubling values of 'n' used in the experiment.

Refer to evidence sheet for experiment data and graph.

* **Average depth of nodes in the final tree** –

Since the maximum depth ‘d’ of a tree with ‘n’ nodes formed using union by rank is log n i.e. **d<= log n.**

In both the improvement algorithms we are merging the trees by comparing the heights (rank/depth in this case as we are initialising height of every node to 1) i.e. we have used union by rank which complements both the algorithms as the amortized cost of operations gets bounded by Inverse Ackermann function. Hence the log N becomes the upper bound for the depth of the tree with ‘n’ nodes.

Note : Both the path halving and two loop improvement algorithms are opportunistic in nature. We don't actively seek paths that can be compressed/reduced. We perform compression while traversing.

1. **Evidence to support relationship observations and conclusions:**

To support the observations described above, I took 2 rounds by doubling the values of n – 1000, 2000, 4000, 8000,..8192000.

Data thus collected was plotted and graph analysis was done to make the above conclusions.

All the data values are added in the below attached spreadsheet.



1. **Screenshot of Unit test passing:**

No unit test cases were available/asked for this assignment.