Introduction To Algorithms CS430
Spring 2024
Project Report

Sudipta Banerjee
CWID: A20460632
Email: sbanerjee5@hawk.iit.edu

Union-Find and Union-Find with Path Compression

This project implements two algorithms: Union-Find and Union-Find with Path Compression. These data structures are used to maintain disjoint sets and perform operations such as finding the representative element of a set and merging (unioning) two sets.

Union-Find Without Path Compression

Algorithm

Input: Size of the Union-Find data structure: *size*
Output: Union-Find data structure

1. **Initialization:**
   Create a parent array of size *size*. Initialize each element of the parent array to its index.
   Create a rank array of size *size*. Initialize each element of the rank array to 0.

2. **Find Root Function:**
   Input: Node
   Output: Root of the set containing the node
   While the parent of the node is not equal to the node itself, update the node to its parent. Return the node.

3. **Union Function:**
   Input: Two elements x and y
   Output: None
   Find the root of the set containing x (rootX). Find the root of the set containing y (rootY).
   If $rootX$ is not equal to $rootY$:

   - Compare the ranks of $rootX$ and $rootY$.
     - If the rank of $rootX$ is less than the rank of $rootY$, set the parent of $rootX$ to $rootY$.
     - Else if the rank of $rootX$ is greater than the rank of $rootY$, set the parent of $rootY$ to $rootX$.
     - Else, set the parent of $rootY$ to $rootX$ and increment the rank of $rootX$.

Time Complexity Analysis:

The time complexity of both FIND and UNION operations in the UNION-FIND data structure with rank optimization is $O(\alpha(n))$, where $\alpha(n)$ is the inverse Ackermann function. This complexity is very close to $O(1)$ for all practical purposes, making the UNION-FIND data structure efficient for disjoint set operations.

Union-Find with Path Compression

Input: Size of the Union-Find data structure: *size*
Output: Union-Find data structure with path compression

1. **Initialization:**
   Create a parent array of size *size*. Initialize each element of the parent array to its index.
   Create a rank array of size *size*. Initialize each element of the rank array to 0.

2. **Find Root Function with Path Compression:**
   Input: Node
   Output: Root of the set containing the node
   If the parent of the node is not equal to the node itself, set the parent of the node to the root of the set containing the parent node recursively (path compression). Return the parent of the node.

3. **Union Function (same as Union-Find):**
   Input: Two elements x and y
   Output: None
   Find the root of the set containing x using the Find Root Function with Path Compression.
   Find the root of the set containing y using the Find Root Function with Path Compression.
   If the roots are not equal, perform the union operation based on the ranks of the roots (same as Union-Find).

Time Complexity Analysis:

- The time complexity of the FIND operation with path compression is amortized $O(\alpha(n))$, where $\alpha(n)$ is the inverse Ackermann function.
- The time complexity of the UNION operation remains the same as in the UNION-FIND algorithm, i.e., $O(\alpha(n))$.
- Overall, the UNION-FIND-WITH-PATH-COMPRESSION data structure achieves efficient amortized time complexity for both FIND and UNION operations.

Running the Main Program

To run the program, use the following command:

```
python3 main.py -i <input_file>
```

Replace `<input_file>` with the desired input file name. For example:

```
python3 main.py -i input.txt
```

The program will read the input from the specified file, process the operations, and generate an output file, with name as `output.txt`

Input Format:

```
4  ---> n
6  ---> m
F2
U12
F2
U23
F3
F2
F4
U14
F4
```

Output Format:

```
2
1
1
1
4
1
```

Plot Performance Analysis

The `plot_operations.py` file contains code to analyze the performance of the Union-Find with Path Compression algorithm for multiple inputs. It measures the time taken for operations from input files and plots the results.

Running the Plot Performance Graph

To run the plot performance analysis, use the following command:

```
python3 plot_operations.py
```

This will generate plots showing the running times for different values of `m` and `n`. The plots will be displayed and saved as image files in the current directory.

Plotting Graph

The plotting function in `plot_operations.py` generates a plot similar to Figure 1, which displays the running times of the Union-Find with Path Compression algorithm for various values of `m` (number of FIND operations) and `n` (number of elements).

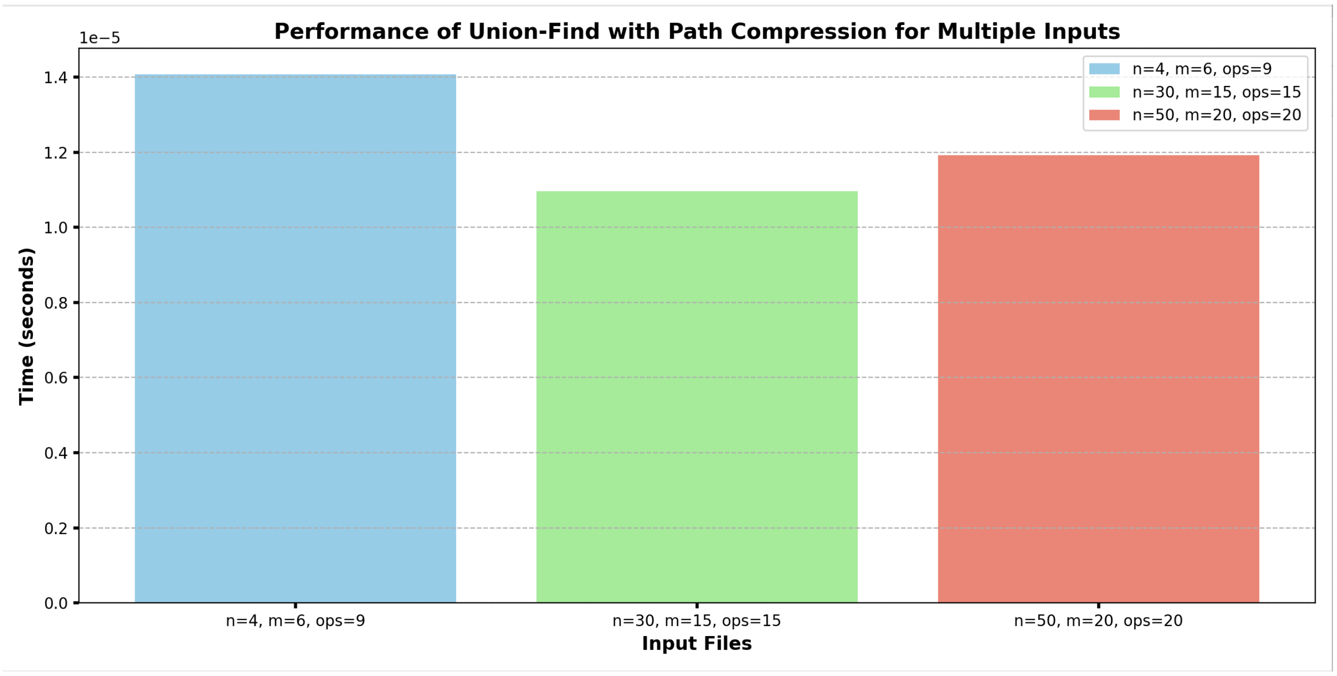

Figure 1: Plot for Union-Find with Path Compression