

INFO 6205
Program Structures & Algorithms
Spring 2019
Assignment 3

The purpose of this assignment is to firstly implement the Height-weighted Union-Find with Path Compression, and secondly develop a Union-Find client which is given an initial sites number as input, and output how many connections needed for the union of all sites. Finally, after implementing the program, testing the cases and running the UF client, I gathered the data of input (original sites number) and output (number of connections needed), and then analyze the functional relationship between the two variables.

My conclusion is: given N as original components number, to reduce the number of components from N to 1, $\sim \frac{1}{2}N \ln N$ (where $\ln N$ is the natural logarithm of N) of connections are needed. The detailed implementation and analysis are as follows:

I. Implementation and Testing

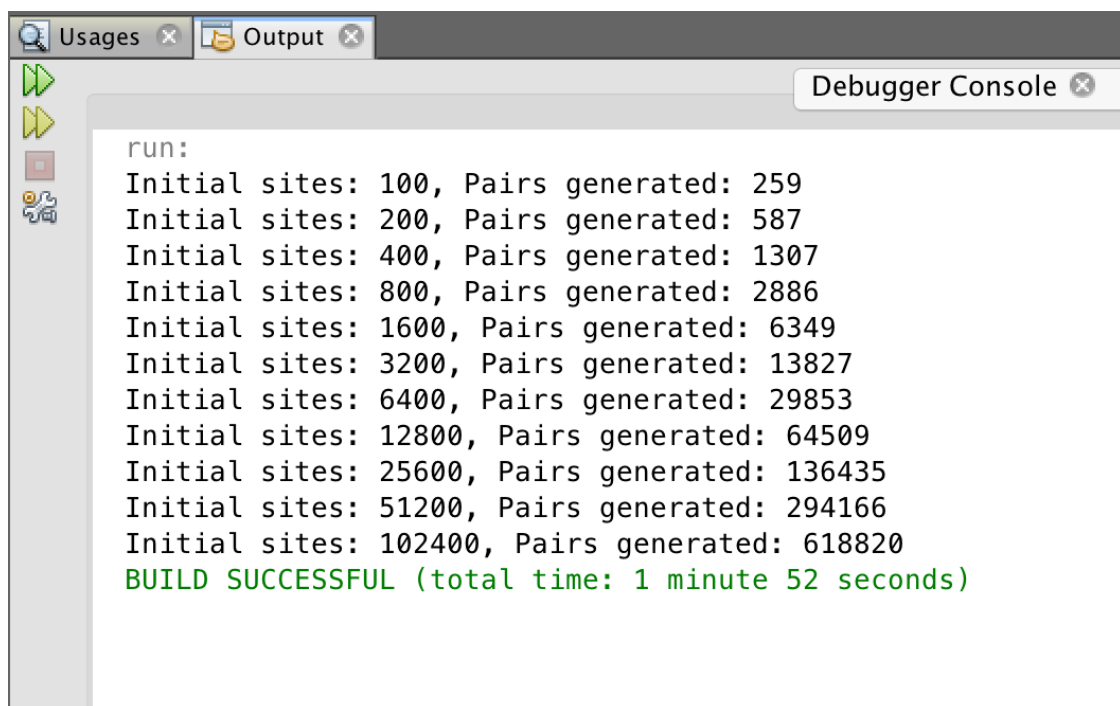
Height-weighted Union-Find with Path Compression uses the same basic strategy as weighted quick-union but has two improvements. One is height-weighted, which means keep track of tree height and always links the shorter tree to the taller one. Another is path compression, which means given a site p and to find its root, add a loop to *find()* function that links every site on the path to the root of p .

Using the above logic, I implemented mainly three functions of the class `UF_HWQUPC`: *find()*, *doPathCompression()*, and *mergeComponents()*. The logic of each function is as follows:

- *find()*: given a site p , returns the root of p , and if path compression needed, then add the loop to update the parents of each site on the path from p to its root.
- *doPathCompression()*: for a given site, update its parent to value of grandparent using single-pass path-halving mechanism of path compression
- *mergeComponents()*: merge the components by always making shorter root point to taller one, and update the height of components by either increasing the height by one (if the two tree are the same height) or leaving the tree heights unchanged (if the two tree have different heights).

Then, using the implementation of UF_HWQUPC, I developed a UF ("union-find") client that takes an integer value N as the number of "sites." Then generates random pairs of integers between 0 and $N-1$, calling the function *connected()* to determine if they are connected and *union()* if not. Loop the process until all sites are connected, the output is the number of connections. The program is packed as a static method *count()*, and called in the *main()* method.

In the *main()* method, to be more convenient and guarantee the accuracy, I set **11 values** of sites count N ranging from 100 to 102400, and for each N , calculated the outputs **1000 times** and get the average as the needed connections for uniting all the sites. My output data is shown as Image 1 below:



```
run:
Initial sites: 100, Pairs generated: 259
Initial sites: 200, Pairs generated: 587
Initial sites: 400, Pairs generated: 1307
Initial sites: 800, Pairs generated: 2886
Initial sites: 1600, Pairs generated: 6349
Initial sites: 3200, Pairs generated: 13827
Initial sites: 6400, Pairs generated: 29853
Initial sites: 12800, Pairs generated: 64509
Initial sites: 25600, Pairs generated: 136435
Initial sites: 51200, Pairs generated: 294166
Initial sites: 102400, Pairs generated: 618820
BUILD SUCCESSFUL (total time: 1 minute 52 seconds)
```

Image 1 Running Output

II. Observation and Conclusion

The hypothesis was that the number of pairs generated to accomplish the height-weighted quick union with path compression (i.e. to reduce the number of components from n to 1) is $\sim \frac{1}{2} N \ln N$ where $\ln N$ is the natural logarithm of N .

To confirm the hypothesis, I calculated $\frac{1}{2} N \ln N$ for each of my input N , and then gather the data of N , the program running output, and the calculated $\frac{1}{2} N \ln N$ into a sheet, shown as Table 1.

Initial Sites Count (N)	Count of Pairs to be United	$1/2(N*\ln N)$
100	259	230.2585093
200	587	529.8317367
400	1307	1198.292909
800	2886	2673.844691
1600	6349	5902.207127
3200	13827	12913.44974
6400	29853	28044.97046
12800	64509	60526.08288
25600	136435	129924.4497
51200	294166	277593.4672
102400	618820	590676.07

Table 1 Data Comparison

According to the data shown in Table 1, I figured out the relationship between $1/2 N \ln N$ and the actual computed pair counts as the following Chart 1 shows:

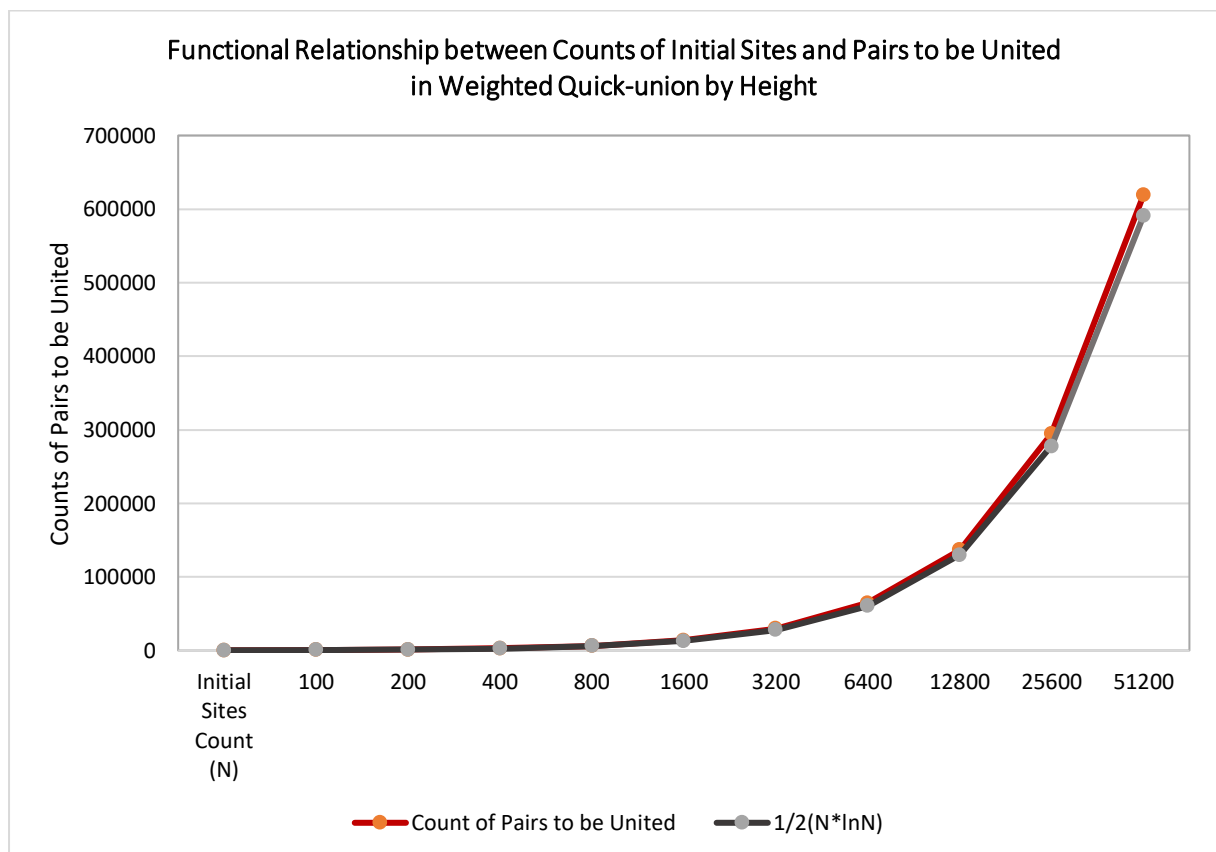


Chart 1 Functional Relationship between Counts of Initial Sites and Pairs to be United in Weighted Quick-union by Height

Zigeng Fu (NUID: 001448458)

The Chart 1 shows obviously that the growing trends of $1/2N\ln N$ and the count of pairs needed for sites union are nearly the same as the initial sites count grows. Thus, the hypothesis can be confirmed, i.e. in the height-weighted quick union with path compression, the pair needed for uniting all initial sites is $\sim 1/2N\ln N$, where N is the original sites count and $\ln N$ is the natural logarithmic of N .