EE3980 Algorithms

Homework 3. Network Connectivity Problem

Due: Mar. 27, 2021

Given a network, the connectivity problem is to determined if two given nodes in the network are connected or not. An example of a network is shown on page 12 of the class handout, `lec23.pdf`.

A general algorithm to solve the network connectivity problem is shown in Algorithm (2.3.1). This algorithm is modified below to generate connected sets, which are embedded into the array R such that $R[v]$ is the node number of the root of the set. Once this is done, one can test the connectivity of two nodes, i, j, by checking if $R[i\,] = R[j\,]$.

```
// Given G(V, E) find connected vertex sets, generic version.
// Input: G(V, E)
// Output: Disjoint connected sets R[1 : n].
1  Algorithm Connectivity(G, R)
2  {
3      for each v_i∈V do S_i := {v_i} ; // One element for each set.
4      NS := |V| ; // Number of disjoint sets.
5      for each e = (v_i, v_j) do { // Connected vertices
6          S_i := SetFind(v_i) ; S_j := SetFind(v_j) ;
7          if S_i ≠ S_j then { // Unite two sets.
8              NS := NS − 1 ; // Number of disjoint sets decreases by 1.
9              SetUnion(S_i, S_j) ;
10         }
11     }
12     for each v_i ∈ V do { // Record root to R table.
13         R[i] := SetFind(v_i) ;
14     }
15 }
```

For this homework, we are going to study the impacts of the disjoint set operations, `SetFind` and `SetUnion`, to the performance. Specifically, three functions with different `SetFind` and `SetUnion` are to be implemented.

1. `void Connect1(void)`: This function uses the `SetFind` function listed in Algorithm (2.3.3) and the `SetUnion` function as Algorithm (2.3.2) given in class handout.
2. `void Connect2(void)`: This function replaces the `SetUnion` function by Algorithm (2.3.5), `WeightedUnion` but keep the same `SetFind` function.

3. `void Connect3(void)`: This function not only uses `WeightedUnion` but also replaces the `SetFind` on line 6 by Algorithm (2.3.8) `CollapsingFind`. But, the `SetFind` function on line 13 remains as it is.

Please also implement the `main` function as shown below to measure the performance. Note that the graph is read in and stored as global variables such that each function can have efficient access. The repetition number $Nrepeat$ should be set to 100. At the end of program execution, the CPU time and the number of disjoint sets for each function should be printed out.

```
// Driver function to measure 3 Connect functions.
// Input: network file contains G(V, E)
// Output: Disjoint connected sets R[1 : n].
1 Algorithm main()
2 {
3     readGraph(); // Read a network from stdin.
4     t0 := GetTime(); // Record time.
5     for i := 1 to Nrepeat do Connect1();
6     t1 := GetTime(); Ns1 := NS; // Record time and number of sets found.
7     for i := 1 to Nrepeat do Connect2();
8     t2 := GetTime(); Ns2 := NS; // Record time and number of sets found.
9     for i := 1 to Nrepeat do Connect3();
10    t3 := GetTime(); Ns3 := NS; // Record time and number of sets found.
11    write((t1 − t0)/Nrepeat, (t2 − t1)/Nrepeat, (t3 − t2)/Nrepeat, Ns1, Ns2, Ns3);
12 }
```

To test the performance of those functions, 10 sets of network files, `g1.dat` to `g10.dat` are provided. The first line of each file contains two integers: $|V|$, the number of nodes, and $|E|$, the number of edges in the network, followed by $|E|$ lines, which contain two integers indicating the two nodes connected by an edge. Using these 10 data files please compare the performance of those three functions against your complexity analyses.

Example of program output is as follows:

```
$ a.out < g1.dat
|V| = 100, |E| = 143
Connect1 CPU time: 1.62411e-05, Disjoint sets: 1
Connect2 CPU time: 2.17915e-06, Disjoint sets: 1
Connect3 CPU time: 2.89917e-06, Disjoint sets: 1
```

Notes.

1. One executable and error-free `C` source file should be turned in. This source file should be named as `hw03.c`.
2. A report file in `pdf` format is also needed. This file should be named as `hw03a.pdf`.
3. Submit your `hw03.c` and `hw03a.pdf` on EE workstations using the following command:

   ```
   ~ee3980/bin/submit hw03 hw03.c hw03a.pdf
   ```

   where `hw03` indicates homework 3.
4. Your report should be clearly written such that I can understand it. The writing, including English grammar, is part of the grading criteria.