Quiz 2 Report

Keshav Saravanan, EE23B035

September 2023

1 Problem Statement

Post the events of Thor (2011), the bifrost is shattered into pieces and the realms are disconnected with no way to get to each other. The Norse Gods decide to reconnect the worlds using roads and they hire Svadilfari to build these roads. The gods decide to send a set of roads they'd like to be built. Svadilfari will only build roads when necessary. Essentially, if a pair of planets are already connected, we need not actually build the road there. A pair of realms are connected, if there exists a sequence of roads you can take to get from one planet to another.
Input: Run the program, your input will be of the following format, to be read from a file provided as a command line argument.
The first line will denote the number of planets and the number of requests from the Norse gods, to be denoted by N and M respectively. The planets follow a 0 based number system, meaning they are numbered by *0* to N *- 1*.
The next M lines contains a pair of numbers, denoted by x and y. This denotes the pair of planets that a road must be built between.
Output: For each of the M lines, in a single concatenated line, print a 1 if a road will be built between those 2 realms and a 0 if a road will not be built between them. It should appear as a sequence of 1's and 0's in the first line of the file.
The second line needs to contain the amount of time it took your program to run in milliseconds, rounded up to the nearest integer. The output is to be written to the file rollnumber_quiz2_q1_output.txt.

2 Solution Implementations

There are three approaches to this question -

1. Grouping planets into collections - two planets are connected if they are in the same collection.
2. Quick Weighted Union Find Method, using pointers and structs - Constructing a tree of planets, with each tree representing a group of connected planets.

3. Quick Weighted Union Find Method, using a roots and weights array.

Grouping planets into collections is the slowest of the three implementations when there are a large number of planets. This is because for each request, the program needs to loop over every single planet to perform the union of two collections when building a bridge. Hence, it's time complexity is *O(N)* for each request.

The Quick Weighted Union Find method is a much faster solution. It works by assigning connected planets into a tree, so two planets are connected if they have the same root node. Given a planet, the program can reach the root of the tree in at most, $log_2(N)$ iterations, since the minimum branching at each node is 2. Hence, the time complexity for each request is $O(log_2(N))$.

3 Comparison of Execution Times

Number of Cities	Number of Requests	Q1	Q3	Q4
100	100	1 ms	1 ms	1 ms
1e3	1e7	1393 ms	1542 ms	1509 ms
1e5	1e5	24756 ms	32 ms	22 ms
1e6	1e6	Killed	438 ms	313 ms

You can see that the brute force solution of grouping planets into collections executes in comparable time to the Quick Weighted Union Find Method when the number of planets is low. The gains in performance are seen when the number of planets is large, since $O(N)$ increases much faster than $O(log_2(N)$.

4 Worst Case Scenario for Grouping Into Collections

The worst case scenario for first algorithm is when every request returns a 1. This is because the program needs to loop over the whole array of planets if it wants to construct a bridge, but returns a 0 without any other code being executed if it knows that a bridge need not be constructed.
So, the worst case input for 16 cities and 16 requests is:

```
16 16
0 1
1 2
2 3
3 4
4 5
5 6
6 7
```

7 8
8 9
9 10
10 11
11 12
12 13
13 14
14 15