/*
 * Name: Chao Huang, YeungKit Wong
 * Date : Aug 30, 2016
 * File   : Report.pdf
 */

PA4 Final Report

1. Which implementation is better and by how much?

Test Pairs(trial)	BFS (ms)	uFind (ms)	(uFind/BFS) %
100 pairs #1	241.524	83.7723	34.68487604
100 pairs #2	244.082	85.5119	35.03408691
100 pairs #3	241.926	83.14	34.36588048
50 pairs #1	1414.49	673.085	47.5849953
50 pairs #2	1423.97	667.305	46.86229345
50 pairs #3	1443	654.235	45.33853084
5 pairs #1	1494.25	18.867	1.26264012
5 pairs #2	1492	18.6015	1.24674933
9 pairs #1	1513.5	30.3464	2.005047902
9 pairs #2	1525.89	30.118	1.973798898
8 pairs #1	1758.58	18.2724	1.039042864
8 pairs #2	1504.54	18.2527	1.213174791
200 pairs #1	436.287	178.56	40.92718784
200 pairs #2	445.688	184.255	41.34170092
200 pairs #3	427.742	179.601	41.98816109

After running BFS and uFind in various cases using movie_cast.tsv(6MB), result shows that the uFind search algorithm is consistently faster than the BFS search algorithm. When searching for few pairs, uFind is much faster than BFS; As the number of test pairs increases, uFind tends to take about 30~40% of BFS's runtime on average.

2. When does the unionfind data structure significantly outperform BFS (if at all)?

In most cases, we expect unionfind data struture would outperform BFS. Since BFS unconditionally searches every possible path while unionfind performs some path reduction. When searching for path between two elements in a large data set with many possible paths, unionfind is a better solution than BFS. However, BFS would beat unionfind in some rare cases such as BFS reaches the target in the first few paths; or in a small data, it is hard to say one is significantly better than another.

3. What arguments can you provide to support your observations?

Considering the way BFS and unionfind were implemented, BFS will take more time as the number of possible path increases; BFS needs to test through each path starting from the beginning when movies of certain year gets added. On contrast, unionfind unions disjoint sets when new movies gets added, compress paths and make children nodes (of a path findRoot exams through) directly point to the root--making the future search of theses nodes to be O(1); Therefore, in most cases unionfind's run time does not increase as much as BFS does when there are a lot of possible paths to search.