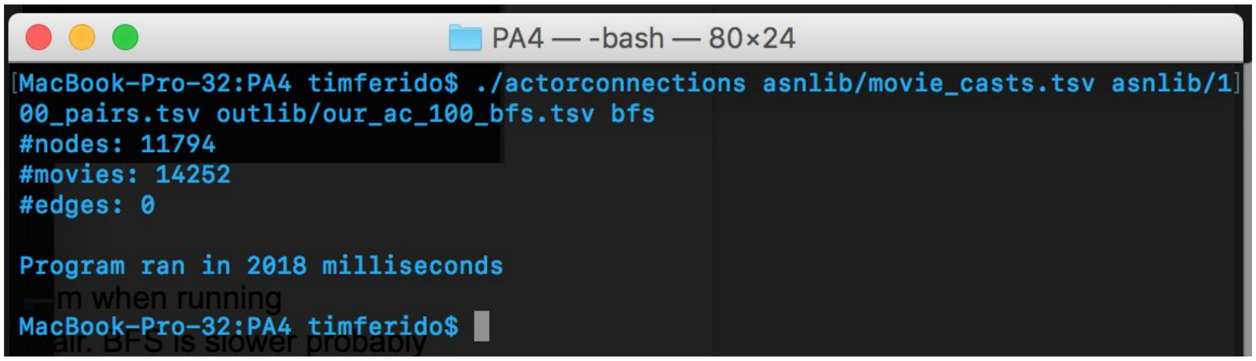
Tim Ferido A12880086
Kent Nguyen A12164917

PA4 Report

After implementing the BFS and UFind algorithms, we ran the actorconnections with the following inputs:

1. BFS with inputs:
    a. movie_casts.tsv
    b. 100_pairs.tsv
        i. 100 pairs of "LOPEZ, GEORGE (I) SHELLEY, RACHEL"

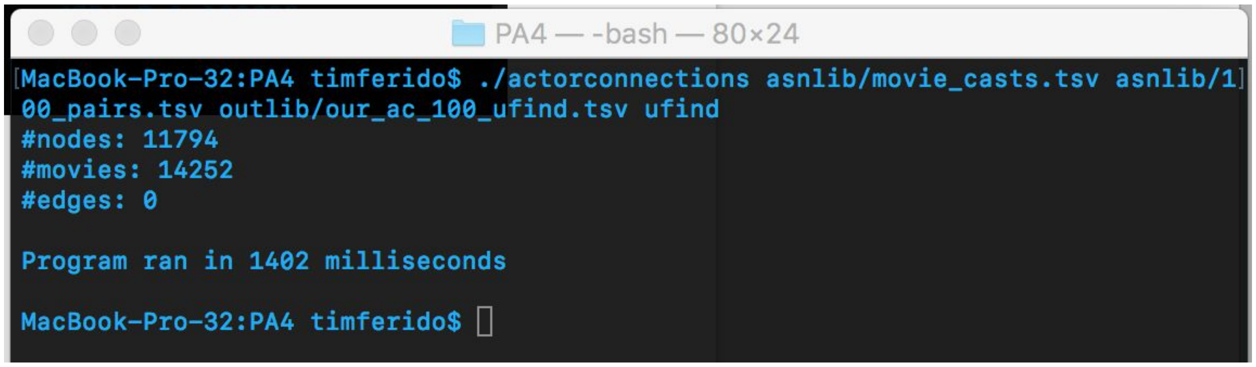


```
PA4 — -bash — 80×24
[MacBook-Pro-32:PA4 timferido$ ./actorconnections asnlib/movie_casts.tsv asnlib/1]
00_pairs.tsv outlib/our_ac_100_bfs.tsv bfs
#nodes: 11794
#movies: 14252
#edges: 0

Program ran in 2018 milliseconds

MacBook-Pro-32:PA4 timferido$ 
```

2. UFind with inputs:
    a. movie_casts.tsv
    b. 100_pairs.tsv



```
PA4 — -bash — 80×24
[MacBook-Pro-32:PA4 timferido$ ./actorconnections asnlib/movie_casts.tsv asnlib/1]
00_pairs.tsv outlib/our_ac_100_ufind.tsv ufind
#nodes: 11794
#movies: 14252
#edges: 0

Program ran in 1402 milliseconds

MacBook-Pro-32:PA4 timferido$ 
```

Analysis:

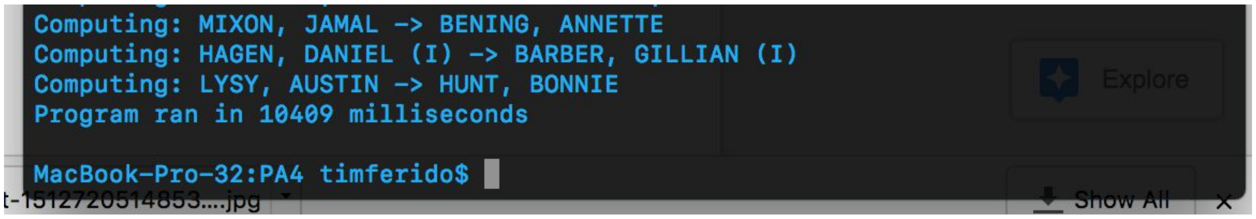
Our UFind algorithm was a smidgen faster than our BFS algorithm when running actorconnections program for 100 instances of the same actor pair. BFS is slower probably because it must iterate through more nodes to find a path between two nodes. UFind must only manage connections which is much less operations.

Questions:

1. UFind implementation was much better than BFS by about 100 nanoseconds in our test case with 100 of the same. However, with the full movie_casts and 100 of the provided

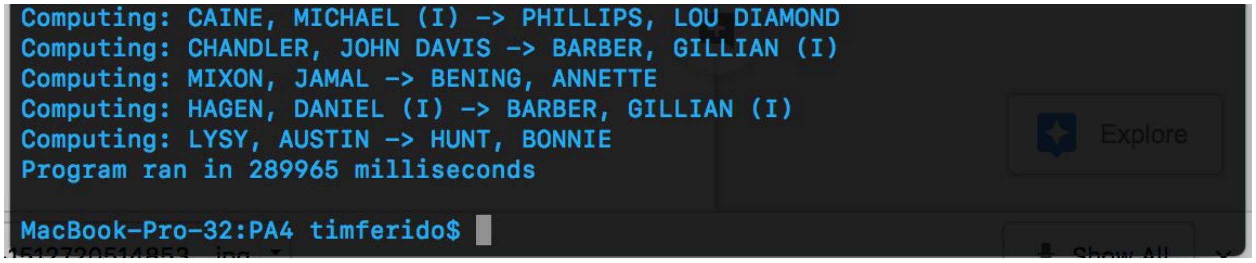
pairs in pair.tsv, UFind was **MUCH** faster than our BFS. Here is the program runtime with the following inputs:

a. movie_casts.tsv
b. Pair.tsv
    i. **UFIND:** 10409 milliseconds
    ii. **BFS:** 289965 milliseconds



```
Computing: MIXON, JAMAL -> BENING, ANNETTE
Computing: HAGEN, DANIEL (I) -> BARBER, GILLIAN (I)
Computing: LYSY, AUSTIN -> HUNT, BONNIE
Program ran in 10409 milliseconds

MacBook-Pro-32:PA4 timferido$
```



```
Computing: CAINE, MICHAEL (I) -> PHILLIPS, LOU DIAMOND
Computing: CHANDLER, JOHN DAVIS -> BARBER, GILLIAN (I)
Computing: MIXON, JAMAL -> BENING, ANNETTE
Computing: HAGEN, DANIEL (I) -> BARBER, GILLIAN (I)
Computing: LYSY, AUSTIN -> HUNT, BONNIE
Program ran in 289965 milliseconds

MacBook-Pro-32:PA4 timferido$
```

2. The UFind Data structure significantly outperforms BFS when the number of total nodes is large, this is because the UFind union function is not dependent on number of nodes O(n), while BFS gets slower as number of nodes increase O(|E|).
3.
    a. This is true because the smaller test file contained 100 instances of the same pair, this means that n (# of nodes in graph) is small. The runtimes for both BFS and UFind were similar. The second input file contained 100 random pairs, this means that n is large. The runtimes are much different when processing this input because UFind is O(1), compressing the paths to the root, and BFS is O(|E|), because the algorithm must search through E edges to find connection.
    b. BFS has more nested loops due to the nature of the algorithm, so as n (# of nodes) increases, the runtime becomes increasingly longer. UFind is a simpler algorithm and the runtime does not grow at a similar rate.
    c. UFind is not perfect because of the compression before finding if two nodes were connected, however it is still much faster because once compressed, the paths are just one iteration away from the root or sentinel node.