Seth Barberee
CSCE 313 - 501
9/14/18

Analysis

For PA 1, my code worked for all Ackerman (1,x), (2,x), and (3,x). As I profiled the various M and N values, I noticed a gradual increase in time for when N = 1 and N = 2. However, N = 3 had more of a exponential growth according to my profiling. The data shown below was taken with the following procedure:

- Start program with appropriate arguments
- Input M and N value for testing
- Let computer finish and not do anything else on it during its operation
- Note the time
- Close out the program with N = 0 and M = 0
- Repeat as needed

System

For comparison, my computer that I used for profiling had the following specs

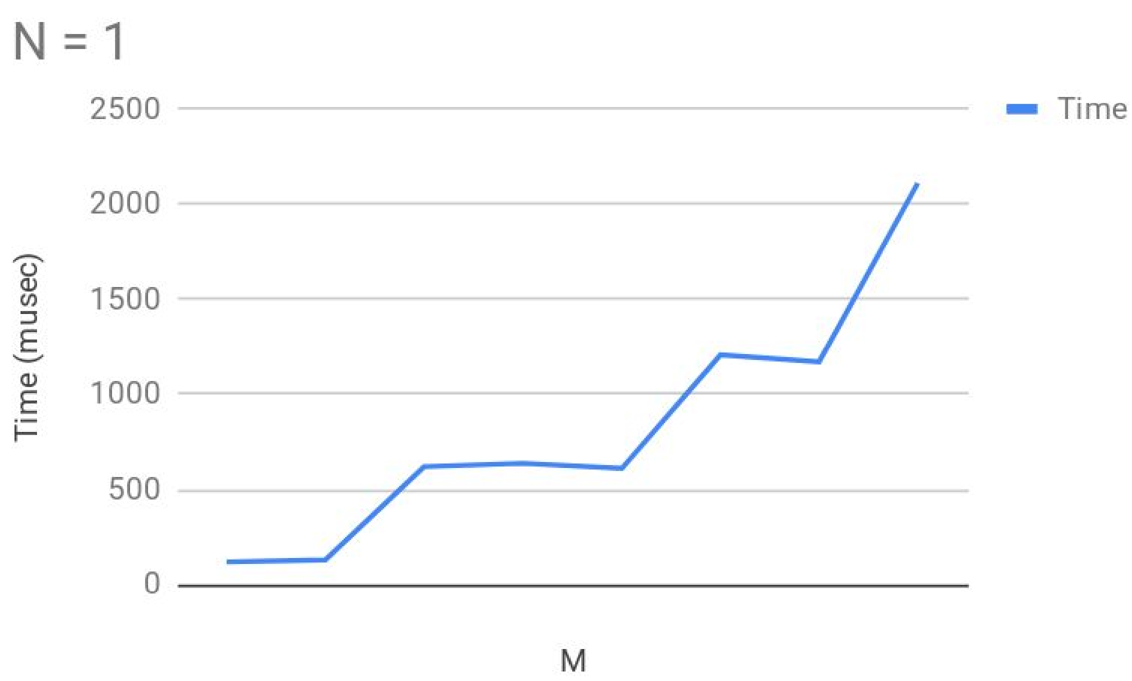
- Arch Linux x86_64
- Intel Core i7 6820HQ 4 cores/8 threads
- 16GB of DDR4 Memory

Data

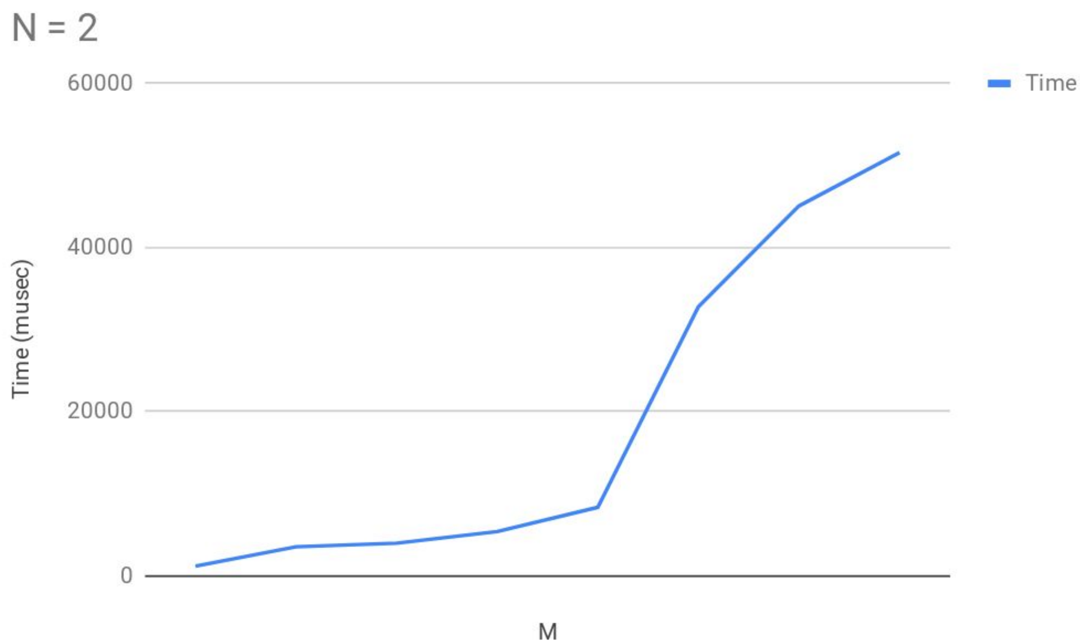
N Value	M Value	Time (averaged over 3 runs) (musec)
1	1	121.333
1	2	131.333
1	3	620.667
1	4	638.667
1	5	611.333
1	6	1207.333
1	7	1172
1	8	2111
2	1	1186.333
2	2	3556
2	3	3991.667
2	4	5407.333

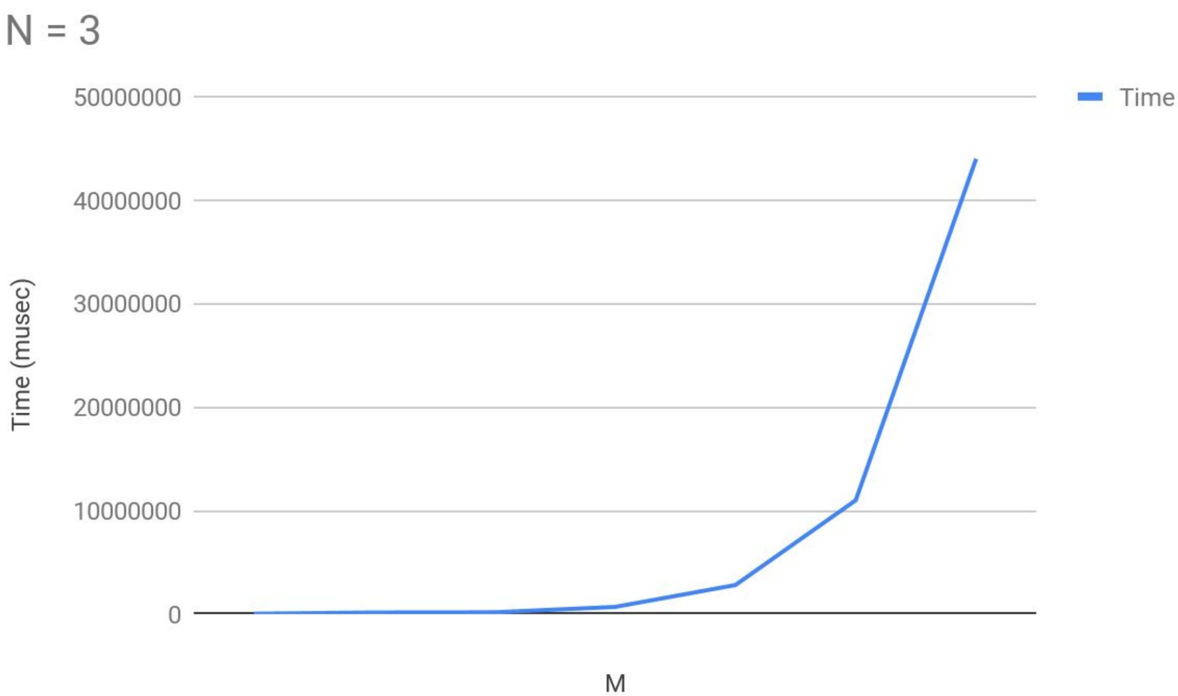
Seth Barberee
CSCE 313 - 501
9/14/18

2	5	8363
2	6	32811.333
2	7	45091
2	8	51571
3	1	32211.66667
3	2	149211.3333
3	3	169478.6667
3	4	689468.3333
3	5	2803197
3	6	10997689.87
3	7	43987158.33



Seth Barberee
CSCE 313 - 501
9/14/18





Bottlenecks/Alternate Designs
An alternate design that came to mind was a binary search tree, specifically a red-black tree. This would minimize the search from worst case runtime of O(n) to O(log n) however at the cost of insertion runtime going from O(1) to O(log n).