ECEN 521
10/20/15
Matthew Wyman
Matthew James
Connor Smith

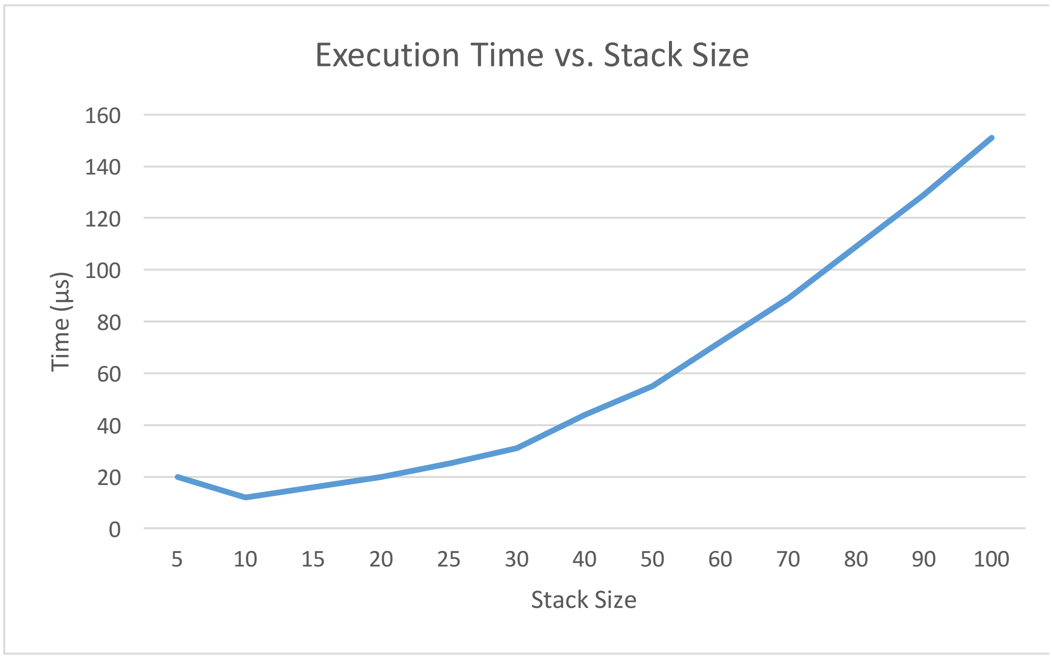
Design Problem #2: Pancake Sort

For this design problem we used a simple algorithm based off selection sort. The algorithm works as follows: first, find the location of the biggest pancake. If it is not in the correct location, flip from the largest element's location and above, resulting in the largest element being placed at the top. Then flip the entire stack so the largest pancake ends up at the bottom. The algorithm continues by finding the next largest pancake that is out of order and repeating the process on the part of the stack that is unsorted. Once this sub-stack has no more remaining elements, the entire stack will be sorted. A flip is accomplished by swapping the order of all the elements in the desired sub-stack. The entire algorithm requires $2n - 3$ flips, each of which requires at most n swaps. Thus the algorithm runs in $O(2n * n) = O(n^2)$ time.

We didn't have too much difficulty solving this problem. The hardest part was understanding how the flip operation worked and how we could use it to insert pancakes into the positions we desired. We chose an algorithm that would insert pancakes into the correct position as it progressed. This allows the number of pancakes that need to be flipped to decrease each time. One other confusion we had was on the numbering system used to index into the pancake stack and determining how the output should be formatted.

A table and graph of the performance is shown below. The recorded time increases quadratically with stack size, as predicted by the asymptotic analysis above. There is one outlier to this behavior, when the stack size is 5. We believe this is due to startup factors like having a cold cache or page faults in simulations with larger stack size, or perhaps simply due to random variation for small data sets. Other than this one case, the execution time does increase as expected.

Stack Size	Time (µs)
5	20
10	12
15	16
20	20
25	25
30	31
40	44
50	55
60	72
70	89
80	109
90	129
100	151


Execution Time vs. Stack Size
Time (µs)
Stack Size