B Fuentes

Ely Merenstein

Stephen Sigman

CP307: Final Project Write-up

# 0-1 Knapsack Problem

Our team's overall strategy was to implement multiple algorithms and figure out which implementation was fastest. We started by implementing a brute force algorithm to get a baseline timing and to have an algorithm we knew would compute the correct answer. From here we moved on to implementing a backtracking algorithm including our own stack data structure. This proved to be much faster than the brute force, as expected, but similarly struggled to solve problems larger than 15 items, taking approximately 2.08 seconds total to solve the entire file of 15-item problems (compared to approximately 68.5 seconds using the brute force algorithm). Next, we implemented a branch-and-bound algorithm, also built off of our implementation of a stack. This algorithm was slower than the backtracking algorithm, taking approximately 15.7 seconds to complete the 15-item file. Finally we tried a meet-in-the-middle algorithm, an implementation combining divide-and-conquer strategies with dynamic programming strategies. We spent the most time working to optimize this algorithm. This implementation was significantly faster than the others, solving the 15-item file in approximately 0.825 seconds, roughly 1% of the time of our brute force algorithm. We mostly pair programmed and met up to code together, using Github for version control, which worked well for us. We had a hard time incorporating our own data structures because we either could not see an efficient way to use them or we found a better solution; the algorithm that ended up being fastest, our meet-in-the-middle algorithm, didn't necessitate the use of a data structure from our library. If

we were participating in a similar project competition in the future, we would spend more time on optimizations instead of (or at least in addition to) trying as many different implementations as we did. However, we found that working on multiple implementations was a valuable use of time because it improved our understanding of the problem, helping us work towards faster solutions. Our fastest algorithm was the meet-in-the-middle method, and we have documented the timings we were able to measure in the graphs below.