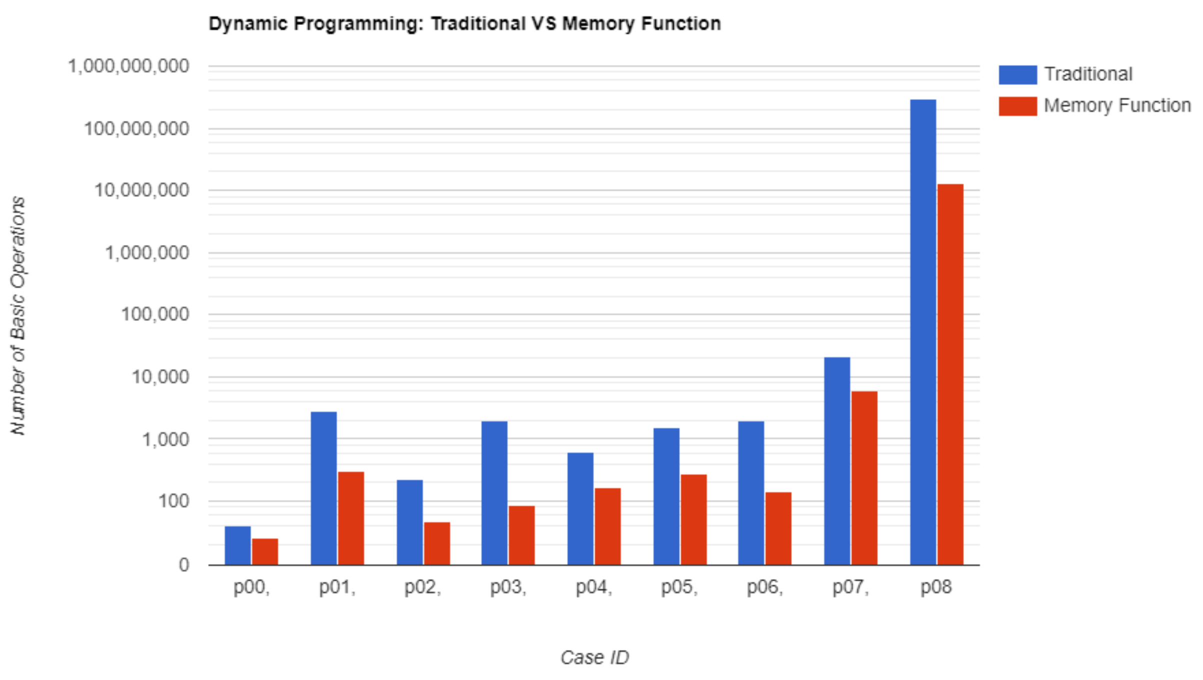
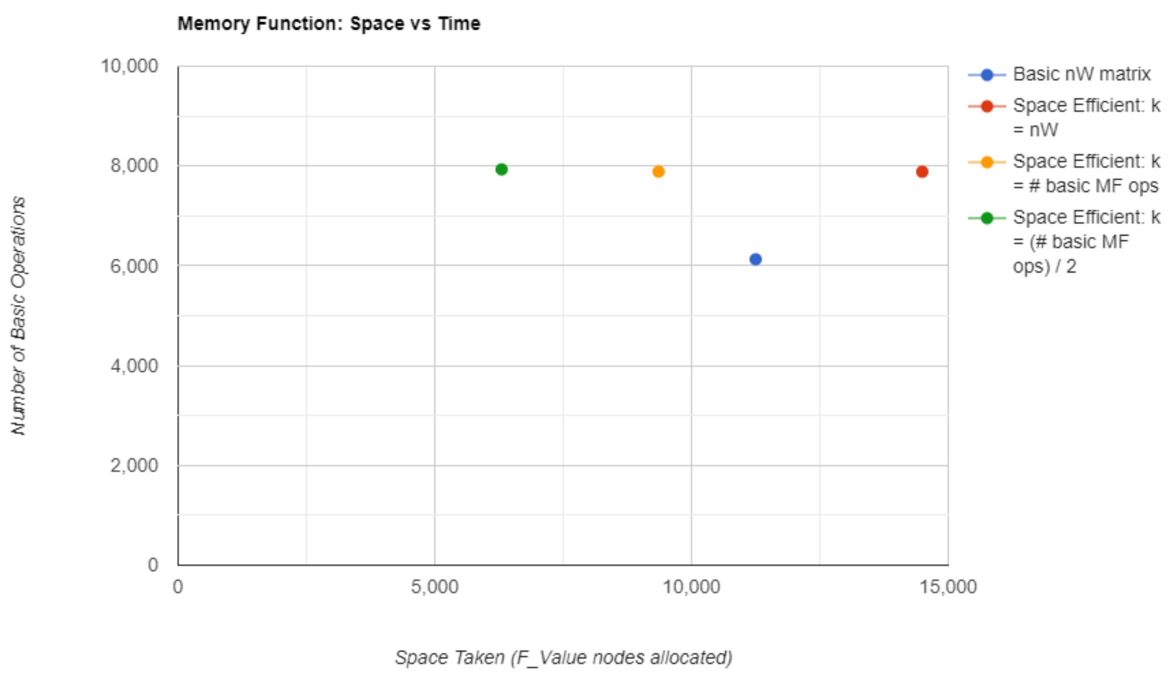
CS 415 Project 04 Knapsack

Soren Richenberg, Owen Mastropietro

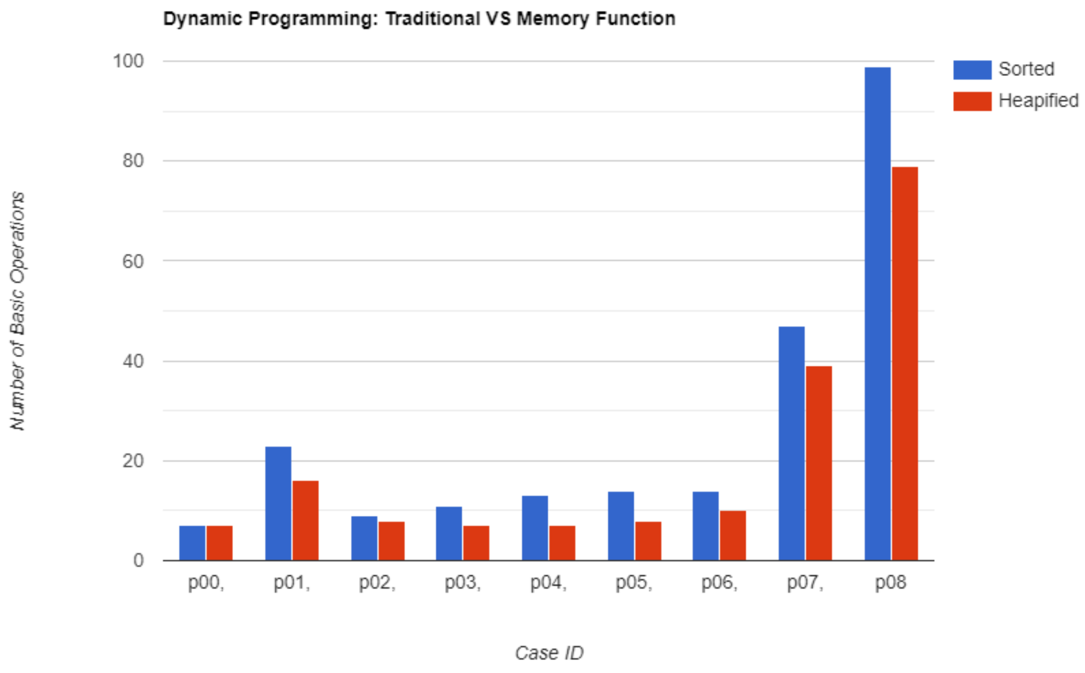


- The graph demonstrates that the Memory Function approach is strictly faster than the Traditional Dynamic Programming approach to the Knapsack problem.
- The Memory Function approach to Task 1 is more efficient because it only needs to calculate the F_Values used directly in the backtracking step. This results in fewer basic operations and a faster execution time, however it does not improve the overall asymptotic complexity from Theta(n*W).



- The Space Efficient Memory Function approach with a hash table of size k = n*W is shown, along with the Basic Memory Function, to demonstrate that this approach is not space or time efficient unless a meaningful value for k is selected, due to the impact of collisions.
- The number of unused (wasted) cells in the Basic Memory Function's matrix is approximately equal to the number of operations that are conserved by choosing the Basic Memory Function approach over the Traditional Dynamic Programming approach. This is because the Memory Function leverages the fact that only specific cells of the matrix need to be computed. Therefore, the space utilized (not wasted) by the Basic Memory Function is approximately equal to (n*W) minus the number of basic operations used by the Memory Function.
- The Basic Memory Function makes a comparison whenever it:
    i. Makes an inquiry into its matrix
    ii. Uses max() to compare two inquiries to its matrix
    iii. Makes a comparison during the construction of its optimal subset
    - Since only one value is written to the matrix for each inquiry, capturing the number of operations from case i, and omitting cases ii and iii, should give the optimal value for k.

- Given the prior insights about the space utilization of the Memory Function, we selected the following values for k:
    - k = # of basic ops in MF : This value provides a minor improvement to space efficiency at the cost of a minor increase to the number of operations.
    - k = (# of basic ops in MF) / 2 : This value attempts to capture the number of times that the Basic Memory Function makes inquiries into its matrix. It provides a major (approximately 2x) improvement to space efficiency at the same cost of a minor increase to the number of operations.
- The advantage of 1b is that it is the fastest Dynamic Programming approach to the Knapsack problem, and it does not require overhead for collision resolution or dynamic memory allocation. Its disadvantage is that it tends to only utilize about half of the memory that it requires.
- The advantage of 1c using k = (# of basic ops in MF) / 2 is that it requires about half the space used by the Basic Memory Function, while maintaining a comparable increase in speed. The disadvantage is that it requires more basic operations to manage the hash table and its collisions, and requires overhead for dynamic memory allocation when open hashing is used.



- Merge Sort was selected for a Theta(n log n) sorting algorithm. Bottom-Up Max-Heap construction was selected for a Theta(n) heapification algorithm.
- The graph shows that using a max-heap to choose the first *k* locally optimal value/weight ratios is always going to be faster or equal to pre-sorting to find the first *k* locally optimal value/weight ratios.
- Greedy Approach Max-Heap: Theta(n) + Theta(k log n).
- Greedy Approach (Merge) Sort: Theta(n log n) + Theta(k).
- Since *k* is strictly less than or equal to *n*, the Max-Heap approach has an advantage because of its smaller scalar multiple on (log n) in the dominant factor:
    - k log n <= n log n
- Another aspect of the Max-Heap's advantage is that in the greedy approach, the first step (heapification or sorting) must run to completion, while the second step (sifting or iterating through values from 0 to k) is likely to exit early. Since the asymptotic complexity of the Max-Heap approach is governed by its second step (sifting), this likelihood to exit early provides a more meaningful speedup to the combined execution time of these two steps.