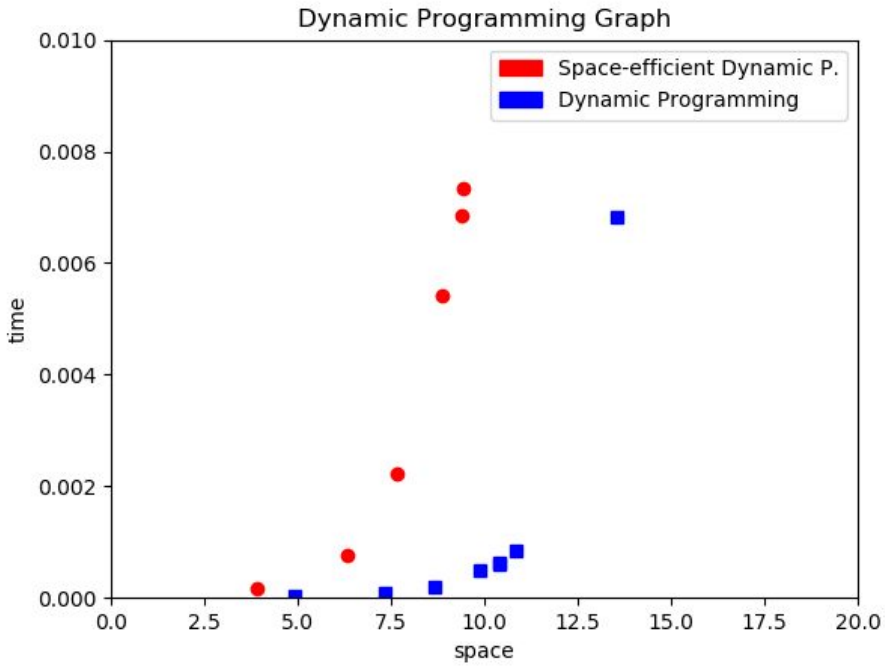
Jevan Smith
Isaac Davidson
CS415 - Algorithm Analysis

Problem Analysis

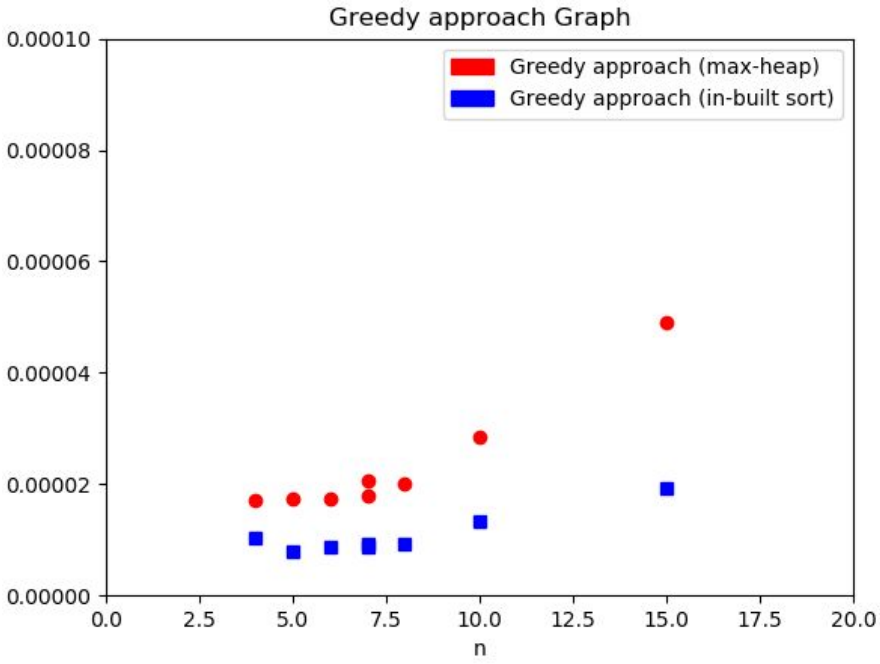
Task 1b implementation details:
When implementing Task 1b we started by creating a HashTable, using the hash function provided we were able to get the result of R_{ij} mod k, which was utilized to spread out our data being inserted. Our hashTable was supported by a fairly simple implementation of a linked list data structure, which helped resolve collisions within our hashTable. Once the hashTable was created implementation of the Dynamic Programming approach to Knapsack was fairly simple, instead of storing our data inside a nW matrix we input that data into our hashTable which helped save space. Lastly we found the most effective k value to be somewhere in the middle of nW, specifically (nW)/2. We found this value to be the best size value of k because it still managed to be significantly smaller than nW, and improve on speed over larger k values.

Dynamic Programing Graph:
Our Space-efficient DP algorithm in red to the right seems to indicate our space-efficient algorithm is running in a more space efficient manner, yet sacrificing speed as compared to the blue traditional DP approach algorithm. This can be seen by the curves in the graph, you'll notice blue's are slightly more clustered at the bottom where as the red is slightly to the left of blue and more spread out, indicating our above analysis to be correct.



Jevan Smith
Isaac Davidson
CS415 - Algorithm Analysis

Greedy Approach Graph:



Our initially assumption on the results of this graph was that our max-heap implementation would be faster than the in-built sorting algorithm within python, to our surprise the graph would seem to indicate the in-built approach to be faster than our max-heap implementation. Curious, we did some research and found pythons in-build sorting algorithm utilizes the “Timsort” algorithm which is essentially a hybrid mergesort algorithm, for more information regarding Timsort we have provided a link below which we found interesting in regards to Timsort.
https://hackernoon.com/timsort-the-fastest-sorting-algorithm-youve-never-heard-of-36b28417f399