Knapsack problem algorithms comparison

In the knapsack recursive algorithm, it takes four inputs, size, value, number of items n, and capacity. The base case is where capacity is used up or there is no more item in the bag. If it does not reach the base case, there will be two possible cases, indexing the lists from back to top, if the value of last item doesn’t fit in the capacity, it will call knapsack function with n-1, without reducing the capacity. Otherwise, the result will be the max of the sum of the value adds up to n-1, with capacity reduced by size[n-1] or not taking the (n-1)th item.

**Performace for dataset 10**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Enumeration | template | Asm 1 program | Recursive |
| Time | 18.31s | 0.064s | 145.3s | 64s |
| Values | 13549094 | 13415886.0 | 13549094 | 13549094 |
| Weight | 6402560 | 6323699.0 | 6402560 |  |

Recursive function and dynamic programing used in assignment 1 take longer time if the dataset is too big, as it needs more memory to store and calculate the result as dimension increases.