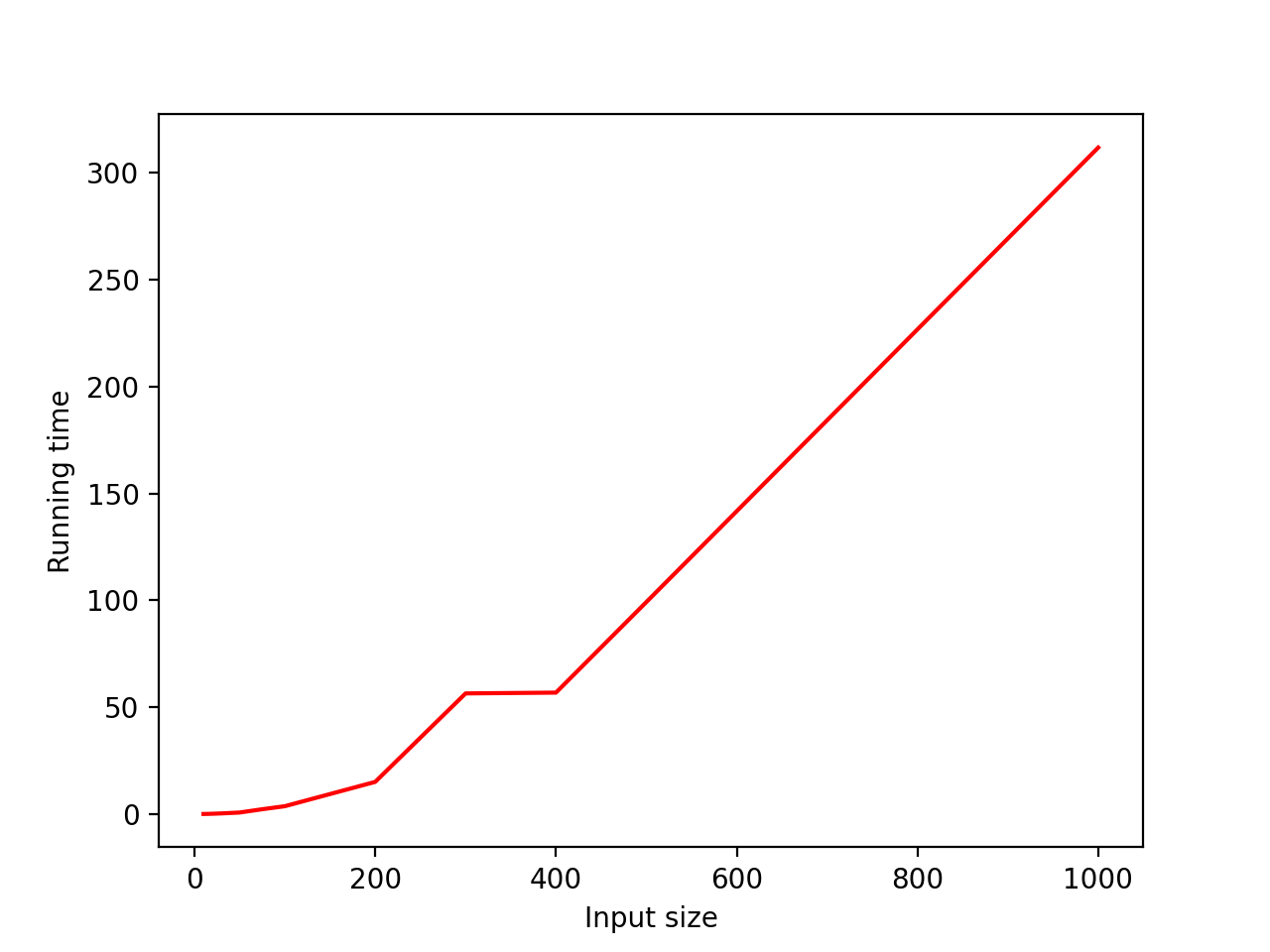
The optimized version uses dynamic programming and significantly reduces the time it takes for a brute-force solution. The horizontal axis of the table is capacity c (0 <= c <= limit) and the vertical axis contains each element in the problem set with the original order they have in the file. Firstly we set the left-most column with 0s because no element can be added to a backpack that has at most 0 weight. Then for the first row, we compare the weight of the first element in the problem set with each c and fill it in the position if weight <= c. For the rest elements, we need to do more computations. For each [index of element][c](starting at second row, second column), if the weight of the element is less or equal to c, then we compare the profits we gain in the following two cases:

1. Including the element: profit = profit of element + [index of element-1][c-weight of element]
2. Not including the element: profit = [index of element-1][c]

Then we compare the two values and keep the larger one in the current position. And if the weight of the element is larger than c, then we simply use the result of the function which calculates the profit of not including the element. In this way, each position in the table contains the best case and thus the best combination of all the items is in the right-down corner of the table.

After the table is completed, we want to get the answer set. We start at the right-down corner which contains the best profit and uses two variables to keep track with the total value and weight. We compare it with the profit right above it(current row-1, same column). If the two profits are the same, it shows that the element in the current row is not used, thus we do not add it to the answer set. If the two profits are different, then we add the element in the current row and subtract its value and weight from the total value and weight. Then we go to position (current row-1, capacity limit-modified total weight). We continue with the same process and stops after finishing with the 2nd row. There is nothing above the first row to compare with, so instead of continuing the process with the first row, we just check if the total value has become 0 at this point. If it is 0, then we have added all the element to the answer set and there is no point to add the first element. If it is not 0, then we add the first element to complete the answer set.