# COMP26120 Lab 5 Report

Ziyi Li

February 22, 2023

Written with assistance from chatGPT

## 1 What makes a problem hard for Dynamic Programming?

### 1.1 Hypothesis

When using a dynamic programming algorithm to solve the knapsack problem, there should be a positive relation between the backpack's size and the program's runtime, when the input data size is fixed. This means that the program takes more time to process when the knapsack size increases.

### 1.2 Design

In order to prove the hypothesis, when designing the experiment, I set the number of items to be the same. I set up a total of two sets of experiments, 10,000 items and 5,000 items.

All input data are generated by script provided (kp_generate.py). For each value of knapsack capacity, two sets of item data were generated to reduce errors in the experimental results. The ranges of size are between 500 to 15000 when items size is 10000 and 500 to 5000 when items size is 5000, both with spacing 500. The program used here is dp_kp.py. In order to ensure that the output does not affect the running time of the program, all outputs are turned off. Python Jupyter Notebook is used here to run test and collect data.

## 1.3 Results

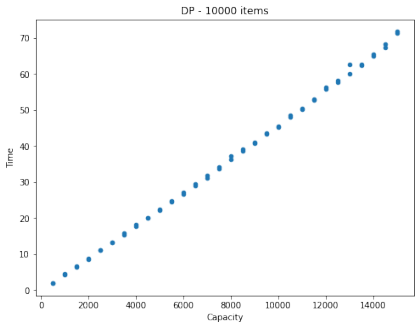


Figure 1: 10000 Items

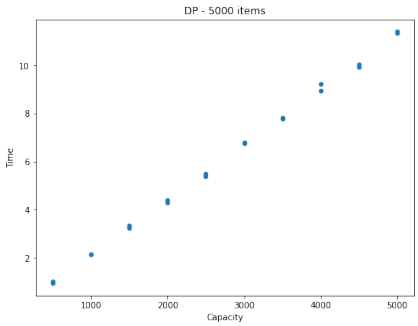


Figure 2: 5000 Items

## 1.4 Discussion

According to the experimental results, we can observe a direct proportionality between the knapsack capacity and the runtime of the program. This means that as the knapsack capacity increases, the program's running time also increases proportionally. This is due to the amount of data that the program needs to process also increasing, so more time is required to calculate the solution.

This shows that the hypothesis holds. Therefore we can conclude that the larger knapsack size makes the knapsack problem more difficult to solve.

However in the situation when the capacity of the knapsack is greater than the number of items, all items can be packed into the knapsack, and the problem becomes trivial. This is because the algorithm is designed to handle general instances of the knapsack problem, where the capacity is not necessarily larger than the number of items. Therefore, in order to optimize the implementation, it is often helpful to add special cases for trivial instances, such as returning the sum of values when the capacity is larger than the number of items. By doing so, the algorithm can avoid unnecessary computations and run more efficiently.

## 1.5 Data Statement

All files mention below are located in test_data folder.

Experiment scripts is test_dp.py. The scripts will use os.system to run the algorithm file, and store the runtime to pandas DataFrame, the results for dynamic programming are in result_dp_50000items.csv and result_dp_100000items.csv .

## 1.6 Appendix

I do give permission for this report to be anonymised and circulated within the University of Manchester to allow staff to better understand how chatGPT can be used in coursework. I do give permission for anonymous excerpts from my report to be quoted in reports or papers about the use of chatGPT in coursework.