Department of Mathematics and Computer Science
College of Science
University of the Philippines Baguio

MACHINE PROBLEM 2

(CMSC 142 - Design and Analysis of Algorithms)

General Objective. Revisit the 0/1 Knapsack problem by exploring alternative approaches to solving the problem such as **dynamic programming** and **approximation or greedy methods**. Specifically, investigate the algorithms' correctness and efficiency in terms of the time- and space- resources by running the algorithms on different input sizes and measuring the time taken to find an optimal/approximate solution.

Guidelines

1. Knapsack Problem Definition
   - The knapsack capacity is fixed at 1000.
   - The weight of the items are random positive integers ranging from 100 to 1500.
     The values of the items are random positive integers from 100 to 500.
   - For the experiment, the number of items n to choose from is n =100 to n =100000.

   You *may* pregenerate the different set of items to be used in the experiment.

2. Implement the alternative algorithms/approaches for solving the 0-1 Knapsack problem presented in class:
   (i) Dynamic Programming (Bottom-up implementation)
   (ii) Dynamic Programming with Memoization (Top-down implementation)
   (iii) Greedy algorithms
      (a) *Greedy Algorithm 1*: Take item of **largest value first**
      (b) *Greedy Algorithm 2*: Take the item of **smallest size first**
      (c) *Backpack Approximation Algorithm*: Take item of **greatest worth ratio**

   - Make sure to check the accuracy of your program (solve *small* instances of the problem and validate the program's output manually or using the exhaustive search algorithm) before performing experiments for large values of n.
   - Make sure that you have means of "timing" your program.
   - You may either implement your own version of the algorithm or use a preexisting code. In the latter case, please cite your source.

3. State all the parameters used during your experiments including, but not limited to,
   - Programming language used
   - Computer/Laptop specifications
   - Background applications running

4. Run each algorithm on the **same set of inputs** and record the **running times***.
    - Divide your test cases *evenly* and *appropriately.*
    - Each test case must have *at least* 3 runs. Use a different set of items per run in each algorithm. Tabulate the results of the 3 runs and their averages.
    - Visualize results using graphs showing the averages of the running times vs number of items

    * **Do not include in the timing the generation of the items.**

5. Investigate the efficiency and effectiveness/correctness of the algorithms for sufficiently large values of n. **Analysis is not limited to the following:**
    - Compare the time efficiencies of the bottom-up implementation and the memoized version of the dynamic programming algorithm. Visualize by how much is the improvement of the memoized version over the bottom-up approach. You can compare the number of nontrivial entries in the dynamic table that were computed (and those retrieved rather than recomputed) in the two algorithms.
    - What is time efficiency of backtracing algorithm for determining the composition of optimal solution in the dynamic programming algorithms?
    - Which of the three greedy algorithms gave the closest approximation to the optimal solution? You can use either the total value of items in the knapsack or the utilization of the knapsack's capacity as metric.
    - How does the observed time complexity of each of the algorithms compare to their theoretical expectations?

6. Create a write up (in PDF format) of your results. The write up must have **at least** the following chapters/sections:
    (i) *Abstract* - short summary that concisely reports your aims, methods, results, and conclusions (150-250 words)
    (ii) *Introduction* - brief introduction, background to the problem, and objectives
    (iii) *Methodology* - detailed description of how the empirical analysis will be performed including the following: algorithm overview, experiment setup, generation of inputs, tools/methods in measuring runtime, recording of the results, etc.
    (iv) *Results and Discussion* - present and discuss the results of the experiment, other significant observations, possible sources of error
    (v) *Conclusion* - summarize the results and give a conclusion based on the results. Recommendations *may* also be included e.g. improvements to the algorithms or extensions to the analysis.
    (vi) *References* if there are any

7. Submission will be done in Google Classroom (Write up and program codes). For all files to be submitted online, follow the filename format:

    `lastname(s)_CS142_MP2.<extension>`

    The soft deadline is on December 18, 2024. The hard deadline is on December 27, 2024.