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

MACHINE PROBLEM 1

(CMSC 142 - Design and Analysis of Algorithms)

General Objective. Investigate the time efficiency of the exhaustive search algorithm for solving the 0-1 Knapsack problem. In particular, perform empirical analysis by running the algorithm on different input sizes and measuring the time taken to find an optimal solution.

Guidelines

1. Knapsack Problem Definition

   - The knapsack capacity is fixed at 1000.
   - The weight of the items are random positive integers ranging from 50 to 100. Similarly, 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 =10 to n =50.

   You *may* pregenerate a 50-items set and just use the first k items for each of the $k-$instance of the problem, where $10 \leq k \leq 50$.

2. Implement the exhaustive search algorithm solving the 0-1 Knapsack problem.

   - Write either a recursive or nonrecursive algorithm for generating all 2^n subsets of the $n-$item set. You can use either (i) the *Decrease-by-one* algorithm, that stores all the subsets in memory, or (ii) the algorithm based on the one-to-one correspondence between all the subsets and all the bit strings of length n.
   - If you opt for the second algorithm, you may consider the minimal-change algorithm discussed in class and leverage the procedure for determining whether a subset is feasible and computing the subset's total value *faster*.
   - Test for correctness of your program by solving *small* instances of the problem and comparing it with the program's output. Make sure to do this before performing experiments for large values of n.
   - Make sure that you have means of "timing" your program.

   Note. 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 the algorithms 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 input per run.
   - 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

   *If you think that your computer/laptop cannot handle the higher test cases, you can **extrapolate** the result. Make sure you explicitly state the method used for extrapolation. In such case, indicate in your write up the maximum number of items, n_{max}, for which the program was successfully terminated.

5. Investigate the (in)efficiency and effectiveness of the overall algorithm for small values of n and for sufficiently large values of n. How does the observed time complexity compare to 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 or extensions to the analysis or alternatives to solving the 0-1 Knapsack problem.

   (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_MP1.<extension>`

   The due date is on November 4, 2024. A printed copy of the write up must be submitted in class by November 6, 2024.

---

kmgonzales.ay24-25.sem1