CS 4720/5720 Design and Analysis of Algorithms

Homework #2

Due: Thursday, October 4, 2018

If written: due in class

If typed and submitted on Canvas: 11:59 pm

Submission requirements:

1. You may implement the programming part in any computer language you like, but source code must be submitted digitally to the "Homework 2 Code" assignment on the Canvas site. I may ask you to demo if I have doubts about your results.
2. **5% extra credit** if you submit *digital, typed* writeups in PDF format to the "Homework 2 Writeup" assignment on the Canvas site. However, you may also turn in handwritten assignments in class – but assignments submitted in person will not receive the 5% extra credit. The due dates are the same for either submission method, but I will need handwritten assigments turned in to me in class.

Assignment:

1. Exercise 3.2.3 in the textbook (*Gadget Testing*). Provide pseudocode for your algorithm. Make sure your algorithm is *exact*: that is, if the gadget can survive a drop of 10 stories, your algorithm must not return 9 or 11 stories. Find the best-case and worst-case efficiencies of your algorithm in terms of $\Theta(\cdot)$.

   Graduate students (extra-credit for undergrads): design an algorithm for this problem that has better than $\Theta(n)$ worst-case efficiency, or prove that it cannot be done.

2. Exercise 3.2.8 in the textbook: Consider the problem of counting, in a given text, the number of substrings that start with an `A` and end with a `B`. For example, there are four such substrings in `CABAAXBYA`.

   (a) Design a brute-force algorithm for this problem and determine its best-case and worst-case efficiency in terms of $\Theta(\cdot)$.

   (b) **Graduate students (extra-credit for undergrads)**: Design a more efficient algorithm for this problem.

3. (*Knapsack*) You are on the planning committee for a space voyage, and you have been tasked with determining which combinations of supplies and experiments should be brought on the voyage. There are n total items to pick from, and each one has its own *value* v_i and *weight* w_i, for each $i \in \{1, \dots n\}$. Your rocket cannot lift more than a total weight of $W > 0$.

   Your task: Design an exhaustive search algorithm which decides which items to bring, such that these items maximize the total value without exceeding the weight limit W. That is, given W and arrays $\{v_1, v_2, \dots v_n\}$, $\{w_1, w_2, \dots w_n\}$ your algorithm should find a subset K of the total items that maximises $\sum_{k \in K} v_k$ (out of all possible subsets of items), and such that $\sum_{k \in K} w_k \leq W$. Your algorithm should output three things:

- An array K containing the indices of the items you should bring,
- The total value of the items you are bringing: $\sum_{k \in K} v_k$,
- The total weight of the items you are bringing: $\sum_{k \in K} w_k$.

(a) Write your algorithm in pseudocode and determine its efficiency class in terms of $\Theta(\cdot)$. This algorithm must be *exact*: of all possible subsets of items, it must return the absolute best one that fits in the rocket.

(b) Consider a different algorithm which works in the following way: it first finds the highest-value item that fits in the rocket, and it adds that to K. Then, it finds the next-highest-value item that fits in whatever capacity is left, and adds that to K. It repeats this process until there are no more items which will fit in the rocket. This is an example of what is called a *greedy* algorithm. Write this algorithm in pseudocode and determine its efficiency class in terms of $\Theta(\cdot)$.

(c) Does your greedy algorithm solve the problem exactly (always find the best set of items)? If it does, prove it. Otherwise, come up with an input to the problem for which the greedy algorithm gives a worse answer than the exhaustive search algorithm (with whatever values you want for n, W, $\{v_i\}_{i=1}^n$, and $\{w_i\}_{i=1}^n$).

(d) Implement both of these algorithms (exhaustive and greedy) in a programming language of your choice. In all your experiments, let $W = 10000$. For each[1] n from 3 to 15, generate at least 5 random inputs to the algorithm: the values $\{v_1, v_2, \ldots v_n\}$ can be any random positive numbers and the weights $\{w_1, w_2, \ldots w_n\}$ should be random numbers between 1 and 10000. Run both of your algorithms on each randomly-generated input (don't throw away an input until you have run both of your algorithms on it!). *For each randomly-generated input*, compute the following things:

- The time (or number of steps) it takes for each of your two algorithms to terminate.
- The total weight of items selected by each algorithm (this value should never be more than 10000, because that would violate the weight limit).
- The total value chosen by the exhaustive algorithm, *divided by* the total value chosen by the greedy algorithm (this value should never be more than 1, because the greedy can't do better than exhaustive).

Create three scatter plots, one for each of these bullet points, with n on the horizontal axis. For example, if you generated 5 random inputs for each n, each of your plots should have clusters of 5 dots for each n.

[1]If your computer system is slow, this could take a while on larger inputs. If your system takes longer than 5 minutes to run exhaustive search for some n, you may ignore that value of n and higher. Just make sure to explain in your writeup why you didn't go all the way to $n = 15$.