Comp 251 (Fall 2022): Assignment 3

Answers must be submitted online by December 2nd (11:59 pm), 2022.

General instructions (Read carefully!)

- **Important:** All of the work you submit must be done by only you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously.
- To some extent, collaborations are allowed. These collaborations should not go as far as sharing code or giving away the answer. **You must indicate on your assignments the names of the people with whom you collaborated or discussed your assignments (including members of the course staff). If you did not collaborate with anyone, write "No collaborators". If asked, you should be able to orally explain your solution to a member of the course staff.**
- It is your responsibility to guarantee that your assignment is submitted on time. We do not cover technical issues or unexpected difficulties you may encounter. Last minute submissions are at your own risk.
- Multiple submissions are allowed before the deadline. We will only grade the last submitted file. Therefore, we encourage you to submit as early as possible a preliminary version of your solution to avoid any last minute issue.
- Late submissions can be submitted for 24 hours after the deadline, and will receive a flat penalty of 20%. We will not accept any submission more than 24 hours after the deadline. The submission site will be closed, and there will be no exceptions, except medical.
- In exceptional circumstances, we can grant a small extension of the deadline (e.g. 24h) for medical reasons only. However, such request must be submitted before the deadline, justified and approved by the instructors.
- Violation of any of the rules above may result in penalties or even absence of grading. If anything is unclear, it is up to you to clarify it by asking either directly the course staff during office hours, by email at (`cs251@cs.mcgill.ca`) or on the discussion board on Ed (recommended). Please, note that we reserve the right to make specific/targeted announcements affecting/extending these rules in class and/or on the website. It is your responsibility to monitor Ed for announcements.
- The course staff will answer questions about the assignment during office hours or in the online forum. We urge you to ask your questions as early as possible. We cannot guarantee that questions asked less than 24h before the submission deadline will be answered in time. In particular, we will not answer individual emails about the assignment that are sent sent the day of the deadline.
- Unless specified, **all answers must be justified!**

1. Knapsack Problem

We have seen in class the Knapsack problem and a dynamic programming algorithm. One could define the Knapsack problem as:

Knapsack Problem. Let $n > 0$ be the number of distinct items and $W > 0$ be the knapsack capacity. For each item i, $w_i > 0$ denotes the item weight and $v_i > 0$ denotes its value. The goal is to maximize the total value

$$\sum_{i=1}^{n} v_i x_i$$

while

$$\sum_{i=1}^{n} w_i x_i \leq W$$

where $x_i \in \{0, 1\}$ for $i \in \{1, \ldots, n\}$.

Algorithm. We recall the recursive form of the dynamic programming algorithm. Let $OPT(i, w)$ be the maximum profit subset of items $1, \ldots, i$ with weight limit w. If OPT does not select the item i, then OPT selects among items $\{1, \ldots, i-1\}$ with weight limit w. Otherwise, OPT selects among items $\{1, \ldots, i-1\}$ with weight limit w. We could formalize as

$$OPT(i, w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1, w) & \text{if } w_i > w \\ \max\{OPT(i-1, w), v_i + OPT(i-1, w-w_i)\} & \text{otherwise} \end{cases}$$

(a) (40 points) **Correctness of dynamic programming algorithm**
Usually, a dynamic programming algorithm can be seen as a recursion and proof by induction is one of the easiest way to show its correctness. The structure of a proof by strong induction **for one variable**, say n, contains three parts. First, we define the **Proposition** $P(n)$ that we want to prove for the variable n. Next, we show that the proposition holds for **Base case(s)**, such as $n = 0, 1, \ldots$ etc. Finally, in the **Inductive step**, we assume that $P(n)$ holds for any value of n strictly smaller than n', then we prove that $P(n')$ also holds.

Use the proof by strong induction **properly** to show that the algorithm of the Knapsack problem above is correct.

(b) (40 points) **Bounded Knapsack Problem**
Let us consider a similar problem, in which each item i has $c_i > 0$ copies (c_i is an integer). Thus, x_i is no longer a binary value, but a non-negative integer at most equal to c_i, $0 \leq x_i \leq c_i$. Modify the dynamic programming algorithm seen at class for this problem.
Note: One could consider a new set, in which item i has c_i occurrences. Then, the algorithm seen as class can be applied. However, this could be costly since c_i might be large. Therefore, the algorithm you propose should be different than this one.

(c) (20 points) **Unbounded Knapsack Problem**
In this question, we consider the case where the quantity of each item is unlimited. Thus, x_i could be any non-negative integer. Provide a dynamic programming algorithm for this problem.