Comp 251 (Fall 2022): Assignment 2

Answers must be submitted online by November 10th (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!**

Least common multiple

1. This problem aims to study an algorithm that computes, for an integer $n \in \mathbb{N}$, the least common multiple (LCM) of all integers $\leq n$.

   For a given integer $n \in \mathbb{N}$, let $P_n = p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$, where $p_1, p_2, \cdots, p_k$ is a strictly increasing sequence of prime numbers between 2 and n and for each $i \in \{1, \cdots k\}$, x_i is the integer such that $p_i^{x_i} \leq n < p_i^{x_i+1}$. For example, $P_9 = 2^3 \cdot 3^2 \cdot 5 \cdot 7$.

   More precisely, we're going to compute all P_j, $j \in \{1, \cdots, n\}$ and store pairs of integers (p^α, p) in a heap, a binary tree where the element stored in the parent node is **strictly smaller** than those stored in children nodes. For two given pairs of integers (a, b) and (a', b'), $(a, b) < (a', b')$ if and only if $a < a'$. Let h denotes the tree height, we admit that $h = \Theta(\log n)$. All levels of the binary tree are filled with data except for the level h, where elements are stored from the left to the right. After computing P_j, all pairs (p^α, p) are stored in the heap such that p is a prime number smaller or equal to j and α is the **smallest** integer such that $\mathbf{j < p^\alpha}$. For instance, after computing P_9, we store $(16, 2)$, $(27, 3)$, $(25, 5)$, and $(49, 7)$ in the heap.

   The algorithm is iterative. We store in the variable **LCM** the least common multiple computed so far. At first, **LCM**$= 2$ is the LCM of integers smaller than 2 and the heap is constructed with only one node with value $(4, 2)$. After finish the $(j-1)$-th step, we compute the j-th step as follows:

   1. If j is a prime number, multiply **LCM** by j and insert a new node (j^2, j) in the heap.
   2. Otherwise, if the root (p^α, p) satisfies $j = p^\alpha$, then we multiply **LCM** by p, change the root's value by $(p^{\alpha+1}, p)$, and reconstruct the heap.

   We're going to prove, step by step, that the time complexity of this algorithm is $O(n\sqrt{n})$.

   (a) (5 points) In operation 1, a new node is inserted. What is the complexity of this operation? There will be no partial credit for this question.

   (b) (5 points) In operation 2, the heap is reconstructed. What is the time complexity of this operation? There will be no partial credit for this question.

   (c) (20 points) The number of prime numbers smaller than n concerned in the operation 2 is less than $\sqrt{n}$. Prove that the number of times N we need to execute operation 2 to compute P_n is asymptotically negligible compared to n. Tip: you can prove this by proving N is $o(n)$, where o (little o) denotes a strict upper bound.

   (d) (20 points) Assume the complexity of assessing whether an integer is a prime number is $\sqrt{n}$ and suppose multiplication has a time complexity of 1. Prove that the algorithm's complexity is $O(n\sqrt{n})$.

   (e) (5 points) Prove that, for a given heap of height h with n nodes, we have $h = \Theta(\log n)$. No partial credit will be awarded.

Change-making problem

2. In this problem, we aim to answer the question: *"How to return a given amount of change using the minimum number of coins (including bills) for a given coin system."* For example, the best way to return 7 dollars is returning a 5 dollars and a 2 dollars.

   First, let us define formally the problem. A coin system is a m-tuplet $c = (c_1, c_2, \ldots, c_m)$ such that $c_1 > c_2 > \cdots > c_m = 1$. For a given coin system c and a positive integer x, we want to find a solution (a m-tuple of non-negative integers) $k = (k_1, k_2, \ldots, k_m)$ such that $x = \sum_{i=1}^{m} k_i c_i$ so as to minimize $\sum_{i=1}^{m} k_i$.

   In general, the problem is NP-complete. However, there exists a greedy algorithm to find the optimal solution for some coin systems. The algorithm is simple. For a given x, we select the largest coin $c_i \leq x$. Then we repeat it for $x - c_i$. The procedure halts when x becomes 0. For instance, with the coin system $(10, 5, 2, 1)$, the algorithm decomposes $x = 27$ into $10, 10, 5$, and 2.

   From now, we say a coin system is canonical if and only if the solution given by the greedy function above is optimal for any positive integer x. For example, all systems $(a, 1)$ with $a > 1$ are canonical. For any positive integer x, we can write it in the form of Euclidean division, $x = aq + r$ with $r < a$. The solution returned by the greedy algorithm described above is then $g = (q, r)$. To prove that the solution is optimal, we consider $g' = (q', r')$ different than g such that $x = aq' + r'$. We have $q' < q$, otherwise $g' = g$. Since $(q' + r') - (q + r) = (q' + x - aq') - (q + x - aq) = (a - 1)(q - q') > 0$, the solution g is optimal. Therefore, the system $(a, 1)$ is canonical.

   (a) (5 points) Design a non-canonical system of $3 - tuple$ $c = (c_1, c_2, c_3)$ and justify it.

   (b) (20 points) Let q and n be two integers ≥ 2. Prove that the system $c = (q^n, q^{n-1}, \ldots, q, 1)$ is canonical.

   (c) (20 points) Prove that the Euro system $c = (200, 100, 50, 20, 10, 5, 2, 1)$ is canonical.

   Further reading, for those who are interested, there exists an algorithm to verify if a system is canonical (Kozen and Zaks, 1994).