Problem Set 6

Discrete Mathematics

Due: 29th of March, 2023

1. Suppose that we are given a sorted list L of length $n \in \mathbb{N}$ and we are asked to determine whether or not $(\exists i \in n)\big(L(i) = x\big)$. The *binary search* algorithm solves this problem by comparing the middle element $L\big(n/2\big)$ of the list to x and either returning immediately (if they are equal) or recursively searching the appropriate sublist (if they are unequal).

   (a) Provide a *recursive* implementation in `Python` of *binary search*. Name your function `ps06pr1a`.

   - **Input:** a list $L : n \to \mathbb{Z}$ and an integer $x \in \mathbb{Z}$.
   - **Output:** `True` if $(\exists i)\big(L(i) = x\big)$; `False` otherwise.
   - **Constraints:** if `len(`L`) == 0`, your function should make 0 comparisons; if `len(`L`) == 1`, your function should make 1 comparison; otherwise, your function should make 2 comparisons.

   (b) Find a recurrence relation for the number of *comparisons* your function makes.

   (c) Prove that your recurrence relation has the closed form $T(n) = 2\log_2(n) + 1$.

2. In this problem, we want an efficient way of recursively *merging* two sorted lists into one sorted list.

   (a) Provide a *recursive* implementation in `Python` of *merge*. Name your function `ps06pr2a`.

   - **Input:** a sorted list $L_1 : n_1 \to \mathbb{Z}$ and a sorted list $L_2 : n_2 \to \mathbb{Z}$.
   - **Output:** a sorted list $L : (n_1 + n_2) \to \mathbb{Z}$ containing all of the elements of L_1 and L_2.
   - **Constraints:** if the length of either input list is 0, your function can return immediately; otherwise, your function should make 1 comparison.

   (b) Find a recurrence relation for the number of *comparisons* your function makes.
   Hint: your recurrence should be a function of *one* variable, which is the *size of the problem*.

   (c) Prove that your recurrence relation has the closed form $T(n) = n$.

3. The towers of Hanoi are an arrangement of three wooden pegs, labelled *start*, *middle*, and *end*, along with a collection of n rings of distinct sizes. The rings are all stacked on the *start* peg in ascending order based on their sizes. The goal is to move all of the rings from the *start* peg to the *end* peg without violating the following constraints:

   - A *move* consists of moving the top-most ring from one peg to the top of the stack on another peg.
   - A larger ring can never be stacked on top of a smaller ring.
   - Rings can only be moved one-at-a-time.

   The question is: what is the minimum number of moves required to win the game with $n \in \mathbb{N}_+$ rings?

(a) Initial configuration. (b) After two moves. (c) Final configuration.

   (a) Provide a *recursive* implementation in `Python` of *towers of Hanoi*. Name your function `ps06pr3a`.

   - **Input:** a positive natural number $n \in \mathbb{N}_+$ representing the number of rings.
   - **Output:** a number $n \in \mathbb{N}_+$ denoting the minimum number of moves required to win the game.
   - **Constraints:** none.

   (b) Find a recurrence relation for the *minimum number of moves required to win the game* with n rings.

   (c) Prove that your recurrence relation has the closed form $T(n) = 2^n - 1$.
   Hint: recall that $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all $n \in \mathbb{N}$.

Remark. Formally, a *list* of length $n \in \mathbb{N}$ with elements from A is just a function $L : n \to A$. The k^{th} term in the list is given by $L(k)$ for $k \in \{0, \ldots n-1\}$, so this corresponds to 0-indexed arrays when programming.

Code Submission Instructions:

Several of the problems in this problem set have a programming component. The `Python` functions you define must be named as the instructions for each problem indicate, and they *must be recursive.* You are not permitted to use any internal or external libraries (*i.e.*, no `import <...>` statements). Your functions should all be implemented in one file, with the filename `ps06-<lastname>-<firstname>.py`; for example, a possible file name would be `ps06-gonzalez-cedre-daniel.py`.

If you are submitting the rest of your solutions to this problem set electronically, then attach your `Python` file *in the same email* as the rest of your solutions.

If you are submitting your proofs in-person on paper, then email your code separately.