CS302 - Problem Set 3

1. Suppose you are living in a country that has a coin worth 6 cents, a coin worth 4 cents, and a coin worth 1 cent. This society also values having thin wallets, so whenever they make change, they want to use as few coins as possible. For example, if they need to make 8 cents worth of change, they would rather return two 4-cent coins (2 total coins), rather than a 6-cent coin, and two 1-cent coins, (3 total coins). You are tasked with creating a program for cash registers that will help make change using the least number of coins.

   - **Input:** An integer n
   - **Output:** A list $M = (m_6, m_4, m_1)$ of non-negative integers such that

   $$n = 6m_6 + 4m_4 + m_1 \tag{1}$$

   and

   $$S(M) = m_6 + m_4 + m_1 \tag{2}$$

   is minimized.

   (In other words, we like to make the correct change for n using the fewest number of coins and m_i represents the number of i-cent coins that should be returned.)

   (a) **Challenge:** Come up with a dynamic programming algorithm from scratch to solve this problem without my guidance by following the same process we went through in class for MWIS. (If you would like guidance continue to the next question).

   (b) To design a dynamic programming algorithm, we first decide which type of objects we should use for our recurrence (for MWIS on a line, we used the sets S_i), and we thought about options for a "final choice" that allowed us to come up with a recurrence for subproblems (for MWIS on a line, we considered $v_n \in S_n$ and $v_n \notin S_n$). What objects should we use for our recurrence relation for our coin problem? What are our options for a "final choice?"

   (c) Please check the final page to see if you have the same idea as me for your recurrence and "final choices." (If not, please use my moving forward.) Create a recurrence relation (you do not need to include base cases in this part) that gives an option based on each "final choice." Write a brief explanation describing why your recurrence relation is correct.

   (d) What are the base cases of your recurrence?

   (e) Based on your recurrence relation from the previous part, fill in the pseudocode below:

Algorithm 1: `AFill`(n)

Input : $n \geq 1$
Output: Array A such that $A[i]$ is the mininimum number of coins needed to make i cents (for all i up to n).

```
1 YOUR CODE HERE
```

(f) Write pseudocode that takes as input the array A that is output from Algorithm 1, and works backwards through it to find the number of coins of each type that should be returned:

Algorithm 2: `OptCoins`(A)

Input : Array A such that $A[i]$ is the mininimum number of coins needed to make i cents (for all i up to n).
Output: (m_6, m_4, m_1), the optimal number of 6-, 4- and 1-cent coins to make n cents.

```
1 YOUR CODE HERE
```

(g) What are the runtimes of each algorithms? Then what is the runtime of the whole algorithm (both Algorithm 1 and Algorithm 2 together)?

2. Suppose you have n events, each with a start time s_i and end time f_i, for $i \in \{1, \ldots, n\}$. Unfortunately, you only have one auditorium, and you can't schedule conflicting events (events where a start time of one is between the start time and end time of another.) We would like to maximize the number of events that are held. For each of the following greedy algorithms, create an example of a series of events (with start and end times) where the algorithm does not perform optimally.

   (a) At each iteration, pick the remaining event with the earliest start time.
   (b) At each iteration, pick the remaining event that has the shortest time ($f_i - s_i$ is smallest.)

3. Let `SelfReference` be an algorithm that takes as input a sorted (in increasing order) array A of n distinct (non repeating) positive and negative integers, and returns an index i such that $A[i] = i$, or returns 0 otherwise. (Assume the indices of A start at 1 and go to n.) Some hints are on the last page.

   (a) Write psuedocode for a recursive version of `SelfReference` that is as fast as possible.
   (b) Prove your algorithm is correct.
   (c) Write a recurrence relation for the runtime of your algorithm. Solve your recurrence relation (use the tree method formula if you can.)

4. Approximately how long did you spend on this assignment (round to the nearest hour)?

Hints

1(c)
Let $M_n = (m_{(n,6)}, m_{(n,4)}, m_{(n,1)})$ be an optimal list of coins to make n cents worth of change. We will create a recurrence relation for M_n in terms of M_i with $i < n$. For the "final choice" we can think about whether the last coin added to the set was a 6-cent coin, 4-cent coin, 1-cent coin. (Since the order of the coins in the set doesn't matter, this is really saying, whether the set contains a 6-cent coin, contains a 4-cent coin, or contains a 1-cent coin.)

3
You should create a divide and conquer algorithm similar to binary search.

Instead of having the size of the input array shrink, it can be easier to just keep track of the boundaries (s for starting index and f for final index) of an effectively shrinking array, as in the following pseudocode:

Algorithm 3: `SelfReference`(A, s, f)

Input : Sorted (in increasing order) array A of distinct positive and negative integers, a starting index s and an ending index f such that $1 \leq s \leq f \leq length(A)$

Output: Value i such that $A[i] = i$, and $s \leq i \leq f$, or 0 if no such i exists.

The most challenging part of the proof is explaining which side of the array the self-referencing element is in. Make sure you explain this clearly.