DSC 190 - Homework 05

Due: Wednesday, February 9

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope at 11:59 PM.

Problem 1.

Determine whether each of the following statements is True or False. If the statement is False, provide a counterexample.

a) In the activity selection problem, assuming unique finish times, any optimal solution must contain the event with the earliest finish time.

> **Solution:** This is false. Suppose the events are $[0, 2], [0, 1]$, and $[3, 4]$. Then both $\{[0, 1], [3, 4]\}$ and $\{[0, 2], [3, 4]\}$ are optimal. Namely, the second solution is optimal but does not contain the event with earliest finish time.

b) In the minimal spanning tree problem, assuming unique edge weights, any optimal solution must contain the graph edge whose weight is the smallest.

> **Solution:** This is True.
>
> You didn't need to provide an argument, but here's one in case you were curious. Suppose (u, v) is the lightest edge in a graph, and let T^* be the MST. Suppose (u, v) is not in T^*. Let e be any edge in T^* along the path from u to v. Removing this edge disconnected T^* into two connected components, one containing u and the other v. Adding the edge (u, v) re-connects the tree, and since (u, v) is the lightest edge, doing so decreases the total edge weight of T^*. Therefore T^* wasn't optimal (this is a proof by contradiction).

Programming Problem 1.

In a file named `make_change.py`, create a function `make_change{t}` which computes the number of ways of combining American quarters (worth 25 cents), dimes (10 cents), nickels (5 cents), and pennies (1 cent) to make t cents.

It may be possible that there is no way to make change for the target, t, in which case your function should return 0. You should assume that your are very rich and have an unlimited number of each type of coin.

For example: `make_change(11)` should be 4, since we can either use:

- 11 pennies,
- 1 nickel and 6 pennies, or
- 1 dime and 1 penny, or
- 2 nickels and 1 penny.

There is starter code on the course page, and you should submit your solution to the autograder.

Solution:

```
# the value of a quarter, dime, nickel, penny
```

```python
VALUES = (25, 10, 5, 1)


# BEGIN REMOVE
def _make_change_helper(t: int, coin_ix=0):
    """Computes the number of ways to make change using only VALUES[coin_ix:]"""
    if t == 0:
        return 1

    if coin_ix >= len(VALUES):
        return 0

    if t < 0:
        return 0

    # either we use coin coin_ix at least once
    n_with_coin_ix = _make_change_helper(t - VALUES[coin_ix], coin_ix)

    # or we don't
    n_without_coin_ix = _make_change_helper(t, coin_ix + 1)

    return n_with_coin_ix + n_without_coin_ix


# END REMOVE
def make_change(t: int):
    """Count the number of ways to make change.

    Assumes an unlimited number of quarters, dimes, nickels, pennies.

    Parameters
    ----------
    t: int
        The total number of cents.

    Example
    -------
    >>> make_change(5) # 5 pennies or 1 nickel
    2
    >>> # (1 dime + 1 penny) or (2 nickels + 1 penny) or (11 pennies)
    >>> # or (6 pennies + 1 nickel)
    >>> make_change(11)
    4

    """
    # BEGIN PROMPT
    return _make_change_helper(t, 0)
    # END PROMPT
```

Programming Problem 2.

Consider the same problem of computing the number of ways to make change, but now we are not rich – we only have a certain number of quarters, dimes, nickels, and pennies.

In `constrained_make_change.py`, write a function named `constrained_make_change(t, coins)`, where t is the target sum that we are trying to reach, and `coins` is a list of four elements containing the number of quarters, dimes, nickels, and pennies we have. The function should return then number of ways of adding up to t using only the coins we have.

For example, `constrained_make_change(25, [2, 0, 0, 25])` should return 2, since there are only two ways with the coins we have:

- 1 quarter, or
- 25 pennies.

There is starter code on the course page, and you should submit your solution to the autograder.

Solution:

```
VALUES = (25, 10, 5, 1)


# BEGIN REMOVE
def _get_a_coin(coins):
    """Returns the first coin we have available, or None if we're out of coins."""
    for i in range(4):
        if coins[i] > 0:
            return i

    return None

# END REMOVE
def constrained_make_change(t, coins):
    """Compute the number of ways to make change, given a set number of each coin.

    Examples
    --------
    >>> constrained_make_change(11, [1, 1, 2, 5])
    2
    >>> constrained_make_change(11, [99, 99, 99, 99])
    4
    >>> constrained_make_change(25, [2, 0, 0, 25])
    2

    """
    # BEGIN PROMPT
    if t == 0:
        return 1

    if t < 0:
        return 0

    coin = _get_a_coin(coins)

    if coin is None:
        return 0

    original_number = coins[coin]
```

```
    # either we use this coin at least once
    coins[coin] -= 1
    n_with_coin = constrained_make_change(t - VALUES[coin], coins)

    # or we don't use it at all
    coins[coin] = 0
    n_without_coin = constrained_make_change(t, coins)

    coins[coin] = original_number

    return n_with_coin + n_without_coin
    # END PROMPT
```