Assessment 12: Plane Ticket Pricing

CSCI 128

Introduction

This week you will write a program to calculate when airline tickets should be purchased to maximize profit across a series of weeks during which prices fluctuate. You will do this by writing a function that handles much of the work recursively.

Problem

Plane ticket prices fluctuate wildly from week to week, which can be a major source of frustration for travelers. In this assessment, we will look at this problem from the airline's perspective of maximizing profits and practice a new programming paradigm - recursion - along the way.

Consider a situation where there are w weeks left before a flight, and the flight departs *after* the end of week 1. Airlines have collected years of historical data, and this provides an estimate of the number of tickets that will be sold during a certain week at a given price. Every week, there are several price points to consider, each with a different number of tickets that will be sold. Note that higher prices do not necessarily mean a lower number of tickets will be sold: a price increase may result in more tickets being sold if passengers are worried that prices will increase further in later weeks.

Consider an example the week before the flight with 13 seats remaining.

Price	Max Tickets Sold	Potential Revenue
375	5	$8,250
676	15	$10,140
830	6	$4,980

Table 1: Price options for week $w = 2$.

If we tried to maximize revenue in this week only, it would be optimal to sell 13 tickets at a price of $676 for a revenue of $8788. Note there is a higher price option available ($830), but our historical data indicates we would only be able to sell 6 tickets at that price for a revenue of $4980.

Although the $676 price point is optimal for week $w = 2$ in isolation, this may not be optimal for the overall problem. We can still sell tickets in week $w = 1$ (the week of the flight) at the following prices and ticket options:

Price	Max Tickets Sold	Potential Revenue
45	57	$450
355	20	$1,550
575	8	$4,600

Table 2: Price options for week $w = 1$.

The optimal solution for this problem overall is to set the price to $830 in week 2 and $575 in week 1, selling 6 and 7 tickets, respectively, for a total revenue of $9005.

This problem as we've described it lends itself to recursion. We can consider a pricing option for the current week w_c and the rest of the weeks $w_{c-1}, w_{c-2}, \ldots 1$ as a smaller instance of the same problem, also known as a subproblem. We use the optimal answers to the subproblems to find the optimal answer for the overall problem.

We previously showed a counterexample for a "greedy" solution which takes the maximum revenue at each week. In fact, this problem cannot be solved in such a fashion; considering only one week at a time. Instead, we are going to consider a recursive solution that tries all possible prices at every week, and takes only **the maximum number of seats** for each option.

For this program you should define and use two functions:

`ticket_pricing(price_data, week, num_seats_left)` This function should take three parameters: the 2D price grid from the provided data file, the weeks left before the flight departs, and the number of seats left on the flight for the current week.

`parse_price_file(filename)` This function should take one parameter: the path to the data file. This function should return both the 2-D price list that contains all pricing options for each week and the number of remaining seats.

For example:

```
# contents of one_week_1.txt
10
10,999 100,999 1000,999

file_data = parse_price_file("one_week_1.txt")
pricing_options = file_data[0]
remaining_seats = file_data[1]

print(pricing_options) # prints [["10,999", "100,999", "1000,999"]]
print(remaining_seats) # prints 10
```

Strategy

Your solution to this problem will likely be fairly short, but may be difficult to construct while navigating the new concept of recursion. We encourage you to consider the following guiding questions that relate to recursion:

- What is the base case of the function that should not recurse?
- When *should* the function recurse (call itself)?
- How should you utilize the return value of the recursive call?

Use the pseudocode given below to implement this recursive solution.

```
def ticket_pricing(price_data, week, num_seats_left):
    # Check for, and handle the base case

    # Initialize variables to hold the most revenue and num seats used

    # Loop over the pricing options for this week
        # Calculate the revenue for this option, selling all possible seats
        # Calculate the remaining seats after this option is taken
        # If there are seats remaining and more weeks before the flight:
            # Make a recursive call for the next week and remaining seats
            # Add the recursive revenue to this option's total revenue
        # If this option's revenue is the highest so far, store it

    # Return the best revenue you found
```

Input

Your program will take one line of input: the path of a data file that contains flight pricing information.

Within this file, its first line will contain one integer, s, that is the number of seats remaining on the flight. The next w lines in the file provide the estimates for weeks $w, w-1, \ldots 1$ in *decreasing order* i.e. the earliest week is first and the latest week (week 1) is last. The estimates for a week are in the form of pairs of numbers p_i,t_i where p_i gives the ticket price and t_i denotes the number of tickets that will be sold at price p_i. Pairs are separated by spaces.

Consider the example file below.

```
# Contents of two_weeks_1.txt
13
375,5 676,15 830,6
45,57 355,20 575,8
```

This file is covering data for a flight that departs in 2 weeks (which we know because there are two lines of price estimates) and has 13 remaining seats. The 2nd and 3rd lines show there are three price options available each for weeks 2 and 1 respectively.

Output

The first (and only) line of output is the maximum integer revenue the airline can generate for the given data file.

Reflection

Consider the computer glitch that led to massive Southwest Airlines flight cancellations on April 18, 2023. Pick one question below to reflect on.

- What factors should be prioritized when designing national transportation systems that rely on software?
- What could be some possible impacts of prioritizing profit over service?
- Is it right to cancel a flight simply for profit if people have already bought tickets and made plans?
- What responsibilities should companies adhere to in order to produce maximum societal benefits?

Sample Execution

Function-specific call to parse_price_file - Example 1

```
print(parse_price_file("one_week_1.txt"))

# should print:
[[['10,999', '100,999', '1000,999']], 10]
```

Function-specific call to parse_price_file - Example 2

```
print(parse_price_file("two_weeks_1.txt"))

# should print:
[[['375,5', '676,15', '830,6'], ['45,57', '355,20', '575,8']], 13]
```

Function-specific call to ticket_pricing - Example 1

```
print(ticket_pricing([['10,999', '100,999', '1000,999']], 1, 5))

# should print:
5000
```

Function-specific call to ticket_pricing - Example 2

```
print(ticket_pricing([['375,5', '676,15', '830,6'],
                      ['45,57', '355,20', '575,8']], 2, 13))

# should print:
9005
```

Function-specific call to ticket_pricing - Example 3

```
print(ticket_pricing([['375,5', '676,15', '830,6'],
                      ['45,57', '355,20', '575,8']], 1, 7))

# should print:
4025
```

Function-specific call to ticket_pricing - Example 4

```
print(ticket_pricing([['375,5', '676,15', '830,6'],
                      ['45,57', '355,20', '575,8']], 0, 1000))

# should print:
0
```

Function-specific call to ticket_pricing - Example 5

```
print(ticket_pricing([['375,5', '676,15', '830,6'],
                      ['45,57', '355,20', '575,8']], 2, 0))

# should print:
0
```

Sample Execution 1

```
DATA_FILE> one_week_1.txt
OUTPUT 10000
```

Sample Execution 2

```
DATA_FILE> one_week_2.txt
OUTPUT 81
```

Sample Execution 3

```
DATA_FILE> two_weeks_1.txt
OUTPUT 9005
```