Hands On 3

1.1 Introduction

I have implemented solutions for both problems using dynamic programming.

1.2 Holiday Planner

To solve this problem, I first identified the base cases. For a single city, the maximum attraction that can be visited is simply the sum of all attractions in that city.

For `n` cities, once the optimal solution for the previous `n-1` cities is known, the optimal solution for the `d`-th day in the `n`-th city can be derived by considering the maximum combination of:

- `x` days spent in the current city, and
- `d-x` days spent in the optimal solution of the previous cities.

At the end of each day's computation for the `n`-th city, I compared the outcome with the previous optimal solution to determine the best approach.

The memoization matrix is indexed by the `i`-th city and `j`-th day. The dimension of this matrix is $O(n \times D)$

Where:

- `n` is the number of cities,
- `D` is the number of days.

To insert a new item in the matrix, I had to scan through each row, which has `D` elements. Thus, the overall time complexity of this solution is: $O(n \times D2)$

Where `n` is the number of cities and `D` is the number of days.

1.3 Design a Course

For this problem, the approach was structured as follows:

1. **Sorting Topics**: First, I sorted the topics based on increasing beauty. If two topics had the same beauty, I sorted them by decreasing difficulty.

2. **Reducing to LIS (Longest Increasing Subsequence)**: The next step was to reduce the problem to a LIS calculation on the difficulty levels of the topics. The sorting ensures that beauty is increasing as I select increasingly difficult topics.

The time complexity for both sorting the topics and calculating the LIS is:

$O(n \times \log n)$