CSC 114 Algorithms Analysis and Design

Travel Recommendation Project

Submitted To: Dr. Ahmed Bayoumi

Fayrouz Ahmed Ismail Abd El Rahman
ID: 320210131
05-22-2023

Travel Recommendation Project using Dynamic Algorithms and Knap-sacking using python language.

- **Goal:** Maximize the number of cities the user enters as much as possible within his chosen budget by the knapsack using the given dataset which has three files' users, flights, and hotels

Input: User, Budget, and Number of cities to be visited.

Output: List of recommended cities in details including flights and hotels details.

Firstly,

I divided the work into two code files, one for merging the three csv files to only one file to make it easier for searching and using the mentioned algorithm, (I only used the flights and hotels'), and the other one for the main part, the program itself.

1. CSV Merging File:

```python
import pandas as pd

# Read dataset files

users_df = pd.read_csv("users.csv")
flights_df = pd.read_csv("flights2.csv", usecols=['travelCode', 'userCode', 'from','place','flightType','Fprice', 'time', 'distance','agency'])
hotels_df = pd.read_csv("hotels2.csv", usecols=['travelCode', 'userCode', 'name', 'place', 'days', 'price', 'total'])

# Merge the two dataframes based on the common keys

HF_df_merged = pd.merge(flights_df, hotels_df, on=['travelCode', 'userCode', 'place'])

# Add the Fprice to total if there is a match

HF_df_merged['FHtotal'] = HF_df_merged['Fprice'] + HF_df_merged['total']

# Drop duplicates

HF_df_merged.drop_duplicates(inplace=True)

# Write the updated dataframe to a new CSV file

HF_df_merged.to_csv('FinalMerged2.csv', index=False)
```

a. I started by importing the pandas Library, and then reading the three given csv files and specifying the columns I am going to use in my project.
b. Then I started merging the three data frames based on their common keys ('Travel code', 'User Code', and 'Place') to make it easier in searching.
c. Based on the previous step, the Flight price is added to the Total price of spent days in the hotel to be then stored in a new data frame that's named FHtotal so I can then search using the total cost of each city and choose the best option based on the given budget of each user.
d. Dropping the duplicates is one of the most important steps, so for every whole row if there's another duplicate of each variable (data frame), it gets dropped in an in-place way.
e. All of this is then saved to a (one) new csv file that's name "Final Merged 2.csv".

2. Main Part:

```python
import pandas as pd
df = pd.read_csv('FinalMerged2.csv') # O(n)
df = df.sort_values(by=['Fprice', 'FHtotal', 'days'], ascending=True) # O(n log n).

# Calculate the value-to-weight ratio for each city
df['FHtotal_to_Fprice'] = df['FHtotal'] / df['Fprice'] # O(n)

# Sort the cities by value-to-weight ratio in descending order
df = df.sort_values(by='FHtotal_to_Fprice', ascending=False)
name = input("Enter your name: ")
budget = float(input("Enter your budget: "))
num_cities = int(input("Enter the number of cities you want to visit: "))

memo = {}
if (budget, num_cities) in memo:
    selected_cities = memo[(budget, num_cities)]

else:
    selected_cities = []
    total_cost = 0
    for index, row in df.iterrows():
        if total_cost + row['FHtotal'] <= budget and row['place'] not in selected_cities:
            selected_cities.append(row['place'])
            total_cost += row['FHtotal']
        if len(selected_cities) == num_cities:
            break
    memo[(budget, num_cities)] = selected_cities
print(f"Dear {name}, we recommend the following cities within your budget of {budget}:")

for num_cities in selected_cities:
    flightType = df[df['place'] == num_cities]['flightType'].iloc[0]
    agency = df[df['place'] == num_cities]['agency'].iloc[0]
    days = df[df['place'] == num_cities]['days'].iloc[0]
    hotel = df[df['place'] == num_cities]['name'].iloc[0]
    FHtotal = df[df['place'] == num_cities]['FHtotal'].iloc[0]
    print(f"{num_cities} (flight type: {flightType}, agency: {agency}, days: {days}, hotel: {hotel}, total price: {FHtotal} ):")
```

a) We start by importing the Pandas library, reading from the csv file we have created from the previous part and then start sorting the values of the three main data frames in an ascending order.

(This operation depends on the size of the file but can generally be considered to have a complexity of O(n), where n is the number of rows in the file, The complexity of sorting a data frame using the "sort_values" function is typically O(n log n), where n is the number of rows in the data frame. Since the sorting is performed twice in this code, the overall complexity is O(n log n).)

b) We calculate the "value-to-weight" (FHtotal-to-Fprice) ratio of each city by dividing the total price of hotel accommodation and flight price by the flight price and then putting the result into "FHtotal_to_Fprice" and then start sorting the cities in a descending order.

(This operation involves performing element-wise division on two columns of the data frame, which can be considered to have a complexity of O(n).)

c) The program starts taking the inputs from user (name, budget, and the number of cities he is willing to visit).

(considered to have a complexity of O(1) since it doesn't depend on the input size.)

d) For the dynamic part (the memoization part) I preferred to choose creating a dictionary rather than a 2d array, So I created a dictionary named “memo”, then an if statement is used to check if the selected cities for the given budget and the number of cities have already been calculated previously or not. If so, it retrieves the previously calculated list of the selected cities, The memoization algorithm works by storing previously calculated results in a dictionary with keys representing the input parameters (budget and number of cities). If those same input parameters are used again in future iterations of the algorithm, it can retrieve the previously calculated result instead of recalculating it. This can save time for larger datasets or when running multiple iterations with similar input parameters.

(The complexity of this is O (1) because it simply checks if a key-value pair exists in a dictionary (memo) and retrieves the value associated with the key if it exists or not.)

e) Otherwise, it starts iterating through the sorted list of cities and adds them to the “selected_cities” list and check whether they fit within the budget and have not already been added.
f) Once the desired number of cities have been added, the loop breaks out.

(The complexity of this algorithm (knap sacking) depends on the number of cities (“num_cities”) and the number of rows in the data frame (“n”). In the worst case, where “num_cities” is equal to “n”, the complexity would be O (n^2). However, since the algorithm breaks out of the loop as soon as it reaches the desired number of cities, the average case complexity would be less than O (n^2).)

g) The new non-stored cities get stored in the memoization dictionary.
h) Finally, I created a for loop that starts looping through the list of selected cities (stored in the variable "selected_cities") and for each city, it is extracting information from the pd data frame (stored in the variable "df").
i) For each city (place), it is finding the row in the data frame where the "place" column matches the current city (stored in the variable "num_cities"). It then extracts specific needed information from that row, including the flight type, agency, number of days, hotel name, and the total price. Finally, it prints out a formatted string that includes all this information for each desired city and the output will show the selected cities along with their corresponding flight type, agency, number of days, hotel name and total price.

(Therefore, the overall complexity of the code is dominated by the sorting operations and can be approximated as O (n log n), where n is the number of rows in the data frame.)

3. Sample Test Cases:

a)

Input:
Enter your name: fayrouz
Enter your budget: 10000
Enter the number of cities you want to visit: 5
Output:
Dear fayrouz, we recommend the following cities within your budget of 10000.0:
Natal (RN) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel BD, total price: $ 1273.03)
Recife (PE) (flight type: economic, agency: Rainbow, days: 4, hotel: Hotel AU, total price: $ 1657.68)
Aracaju (SE) (flight type: economic, agency: Rainbow, days: 4, hotel: Hotel Z, total price: $ 1133.77)
Brasilia (DF) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel BP, total price: $ 1379.95)
Rio de Janeiro (RJ) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel CB, total price: $ 977.58)

b)

Input:
Enter your name: shaher
Enter your budget: 6000
Enter the number of cities you want to visit: 3
Output:
Dear shaher, we recommend the following cities within your budget of 6000.0:
Natal (RN) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel BD, total price: $1273.03)
Recife (PE) (flight type: economic, agency: Rainbow, days: 4, hotel: Hotel AU, total price: $1657.68)
Aracaju (SE) (flight type: economic, agency: Rainbow, days: 4, hotel: Hotel Z, total price: $1133.77)

c)

Input:
Enter your name: lara
Enter your budget: 15000
Enter the number of cities you want to visit: 9
Output:
Dear lara, we recommend the following cities within your budget of 15000.0:
Natal (RN) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel BD, total price: $1273.03)
Recife (PE) (flight type: economic, agency: Rainbow, days: 4, hotel: Hotel AU, total price: $1657.68)
Aracaju (SE) (flight type: economic, agency: Rainbow, days: 4, hotel: Hotel Z, total price: $1133.77)
Brasilia (DF) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel BP, total price: $1379.95)
Rio de Janeiro (RJ) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel CB, total price: $977.58)
Salvador (BH) (flight type: economic, agency: Rainbow, days: 4, hotel: Hotel K, total price: $1668.9)
Florianopolis (SC) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel A, total price: $2013.28)
Sao Paulo (SP) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel AF, total price: $931.91)
Campo Grande (MS) (flight type: economic, agency: CloudFy, days: 4, hotel: Hotel BW, total price: $576.72)

Note: source code files and the new csv are attached within the word document in the same zip file.