ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 336E

Analysis of Algorithms II
HOMEWORK REPORT

HOMEWORK NO : 3

HOMEWORK DATE : 16.05.2023

150200915 : ADIL MAHMUDLU

Spring 2023

Contents

1 EXPLANATION OF THE CODE AND THE SOLUTION

The code consists of four parts. First part is reading and filling the relevant data structures. The second part includes interval scheduling for a day in a place. The third part consists of the same interval scheduling method applied to the places itself, for maximum gain throughout the tour. The last part is getting the most value out of assets using knapsack algorithm. The algorithms used are instances of dynamic programming algorithms. The most important parts of the program are the interval scheduling and knapsack algorithms.

1.1 Interval Scheduling

```
IntervalScheduling(sessions, selectedSessions):
    s <- integer, size of sessions
    sort(sessions)
    M <- array of size s, holding integer values
    sSeq <- matrix of size s filled with arrays, holding sessions
    M[0] <- sessions[0].capacity
    sSeq[0].insert(sessions[0])

    for i from 1 to s:
        p <- findSessionBefore(sessions, i)
        c <- sessions[i].capacity
        if p != -1 :
            c <- c + M[p]
        if c > M[i-1]:
            if p != -1:
                sSeq[i] <- sSeq[p]
            sSeq[i].insert(sessions[i])
            M[i] <- c
        else:
            sSeq[i] <- sSeq[i-1]
            M[i] <- M[i-1]
        }
    }
    selectedSessions = sSeq[s-1]
    return M[s-1]
```

The interval scheduling algorithm is one of the dynamic programming algorithms that utilize overlapping subproblems with memoization for optimal solution. As the overlapping subproblems use the same data from time to time, recalculation the data slows down the program, and therefore memoization is critical for the best performance. The subproblems find the maximum among current and previous intervals. As previous maxes are kept in memory, the extra calculations are redundant. Nevertheless, because of sorting and finding p with binary search, the program works in $O(nlogn)$ time. The second part of the program makes use of the same interval scheduling algorithm with different data type, so I'll skip the explanation for that part

1.2 Knapsack

```
knapsack(totalRevenue, assets, selectedAssets) {
    a <- integer, size of assets
    K <- matrix of size (totalRevenue+1, a+1) filled with zeros
    for x from 1 to totalRevenue+1
        for j from 1 to a+1
            K[x][j] <- K[x][j-1];
            if assets[j-1].price <= x
                K[x][j] <- max(K[x][j], K[x-assets[j-1].price][j-1] +
                    assets[j-1].value)
    x <- totalRevenue, j <- a
    while j > 0 AND x > 0
        if K[x][j] != K[x][j-1]
            selectedAssets.insert(assets[j-1])
            x <- x - assets[j-1].price
        j <- j-1
    return K[totalRevenue][a];
```

The Knapsack algorithm is another Dynamic Programming algorithm. It is used in last part of the program to calculate the best value gained from assets. The algorithm utilizes overlapping subproblems with memoization. The subproblems use the current and previous data. The previous data was calculated and kept in memory, and thus it can be used without extra processing time. The run time of the algorithm is $O(W * N$ where W is totalRevenue and N is the number of assets.

2 How would the result be if you discard these constraints for the case 1?

1.

Torium 5 May - 9 May: $4 \times 235 = 940$
Cevahir 10 May - 11 May: $1 \times 210 = 210$
Torium 11 May - 14 May: $3 \times 235 = 705$
Trump 14 May - 17 May: $3 \times 300 = 900$
Cevahir 17 May - 18 May: $1 \times 210 = 210$
Torium 18 May - 20 May: $2 \times 235 = 470$
Trum 20 May - 25 May: $5 \times 300 = 1500$
Torium 25 May - 30 May: $5 \times 235 = 1175$
Cevahir 30 May - 1 June: $2 \times 210 = 420$
Trump 1 June - 8 June: $7 \times 300 = 2100$
Torium 8 June - 11 June: $3 \times 235 = 705$
Trump 11 June - 16 June: $5 \times 300 = 1500$
Torium 16 June - 19 June: $3 \times 235 = 705$
Trump 19 June - 24 June: $5 \times 300 = 1500$

Total: 13040

2.

Trump Salon1 8 - 10 : 60
Trump Salon2 10 - 12 : 80
Torium Salon2 12 - 13 : 25
Trump Salon2 13 - 15: 80
Torium Salon1 15 - 16: 20
Trump Salon1 16 - 18: 80
Cevahir Salon2 18 - 21: 70

Total daily: 415