Algorithm Design and Analysis

Assignment 3 : Dynamic Programming

Question 1

> Assume that we want to develop a shopping APP in which a consumer can provide a wish list of items with preferences in the range of 1~100. Then given the current market prices of all items and a budget cap, find a set of items that maximize the sum of preferences with total spending under(≤) the budget cap.
>
> - Explain why the problem is (or not) good for DP.
> - Design and implement an algorithm for the problem.
> - Analyze the complexity of your algorithm.

Solution

a) Dynamic programming is good for this problem because dynamic programming requires an optimal substructure and overlapping sub-problems, both which are present in the problem.

b)In the dymaic programming DP[][] table lets consider all the possible prices from '1' 'budget' as the columns and prices that can be kept as the rows.
The state DP[i][j] will denote maximumm value of 'j-price' considering all values from '1 to ith'. So if we consider 'pi' (price in 'ith' row) we can fill it in alll columns which have 'price values > pi'. Now two possibilities can take place:

->Fill 'pi' in the given column.
->Do not fill 'pi' in the given column.

Now we have to take a maximum of these two possibilities, formally if we do not fill 'ith' price in 'jth' column the DP[i][j] state will be same as DP[i][j] but if we fill the price, DP[i][j] will be equal to the value of 'pi'+ value of the column pricing 'j-pi' in the previous row. So we take the maximum of these two possibilities to fill the current state.

shopping_app.cpp > main()

```cpp
#include<iostream>
using namespace std;

//return max of two integers
int max(int a, int b){
    return (a > b) ? a : b;
}

//return the maximum value that can be put in budget
int solution(int budget, int price[], int preference[], int n){
    int i, w;
    int dp[n+1][budget+1];
    //build table dp[][] in bottom up manner
    for(i=0; i<=n; i++){
        for(w=0; w <= budget; w++){
            if(i==0 || w==0){
                dp[i][w] = 0;
            }
            else if(price[i-1] <= w){
                dp[i][w] = max(preference[i-1] + dp[i-1][w-price[i-1]], dp[i-1][w]);
            }
            else{
                dp[i][w] = dp[i-1][w];
            }
        }
    }

    for(int i=0; i<=n; i++){
        for(int j=0; j<=budget; j++){
            cout << dp[i][j] << " ";
        }
        cout << endl;
    }
    return dp[n][budget];
}

//Driver code
int main(){

    int preference[] = {5, 3, 2, 6};
    int price[] = {4, 2, 3, 5};
    int budget = 10;
    int n = sizeof(preference)/sizeof(preference[0]);

    cout << solution(budget, price, preference, n) << endl;
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
musa.official@134-208-43-101 Algorithms % cd "/Users/musa.official/Desktop/Algorithms/" && g++ --std=c++17 s
official/Desktop/Algorithms/"shopping_app
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 5 5 5 5 5 5 5
0 0 3 3 5 5 8 8 8 8 8
0 0 3 3 5 5 8 8 8 10 10
0 0 3 3 5 6 8 9 9 11 11
11
musa.official@134-208-43-101 Algorithms %
```

c)

- **Time complexity** : O(N*B)
  where 'N' is the number of items and 'B' is budget. As for every item we traverse through all prices possibilities 1<=p<=B.
- **Space complexity**: O(N*B)
  The use of 2-D array of size 'N x B'

Textbook Exercise

Problem 3-4

410921335_A3 > bin_coefficient.cpp > main()

```cpp
//Modify Algorithm 3.2 (Binomial coefficient using dynamic programming) so that
//it uses only one-dimensional array indexed from 0 to k

#include<iostream>
using namespace std;

int min(int a, int b){
    return (a < b) ? a : b;
}

int bin(int n, int k){
    int i, j;
    int B[k+1];
    memset(B, 0, sizeof(B));
    B[0] = 1;

    for(i=1; i<=n;i++){
        for(j=min(i,k); j>0; j--){
                B[j] = B[j] + B[j-1];
        }
    }
    return B[k];
}

int main(){

    cout << bin(4, 2)<< endl;
    return 0;
}
```

Problem 3-5

Initial state for Matrix D

$$\begin{bmatrix} 0 & 4 & \infty & \infty & \infty & 10 & \infty \\ 3 & 0 & \infty & 18 & \infty & \infty & \infty \\ \infty & 6 & 0 & \infty & \infty & \infty & \infty \\ \infty & 5 & 15 & 0 & 2 & 19 & 5 \\ \infty & \infty & 12 & 1 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ \infty & \infty & \infty & 8 & \infty & \infty & 0 \end{bmatrix}$$

Initial state for Matrix D

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

D1

$$\begin{bmatrix} 0 & 4 & \infty & \infty & \infty & 10 & \infty \\ 3 & 0 & \infty & 18 & \infty & 13 & \infty \\ \infty & 6 & 0 & \infty & \infty & \infty & \infty \\ \infty & 5 & 15 & 0 & 2 & 19 & 5 \\ \infty & \infty & 12 & 1 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ \infty & \infty & \infty & 8 & \infty & \infty & 0 \end{bmatrix}$$

Matrix P after the 1st iteration

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

D2

$$\begin{bmatrix} 0 & 4 & \infty & 22 & \infty & 10 & \infty \\ 3 & 0 & \infty & 18 & \infty & 13 & \infty \\ 9 & 6 & 0 & 24 & \infty & 19 & \infty \\ 8 & 5 & 15 & 0 & 2 & 18 & 5 \\ \infty & \infty & 12 & 1 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ \infty & \infty & \infty & 8 & \infty & \infty & 0 \end{bmatrix}$$

Matrix P after the 2nd iteration

$$\begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 2 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

D3

$$\begin{bmatrix} 0 & 4 & \infty & 22 & \infty & 10 & \infty \\ 3 & 0 & \infty & 18 & \infty & 13 & \infty \\ 9 & 6 & 0 & 24 & \infty & 19 & \infty \\ 8 & 5 & 15 & 0 & 2 & 18 & 5 \\ 21 & 18 & 12 & 1 & 0 & 31 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ \infty & \infty & \infty & 8 & \infty & \infty & 0 \end{bmatrix}$$

Matrix P after the 3rd iteration

$$\begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 2 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 & 2 & 0 \\ 3 & 3 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

D4

$$\begin{bmatrix} 0 & 4 & 37 & 22 & 24 & 10 & 27 \\ 3 & 0 & 33 & 18 & 20 & 13 & 23 \\ 9 & 6 & 0 & 24 & 26 & 19 & 29 \\ 8 & 5 & 15 & 0 & 2 & 18 & 5 \\ 9 & 6 & 12 & 1 & 0 & 19 & 6 \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ 16 & 13 & 23 & 8 & 10 & 26 & 0 \end{bmatrix}$$

Matrix P after the 4th iteration

$$\begin{bmatrix} 0 & 0 & 4 & 2 & 4 & 0 & 4 \\ 0 & 0 & 4 & 0 & 4 & 1 & 4 \\ 2 & 0 & 0 & 2 & 4 & 2 & 4 \\ 2 & 0 & 0 & 0 & 0 & 2 & 0 \\ 4 & 4 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 0 & 4 & 4 & 0 \end{bmatrix}$$

D5

$$\begin{bmatrix} 0 & 4 & 36 & 22 & 24 & 10 & 27 \\ 3 & 0 & 32 & 18 & 20 & 13 & 23 \\ 9 & 6 & 0 & 24 & 26 & 19 & 29 \\ 8 & 5 & 14 & 0 & 2 & 18 & 5 \\ 9 & 6 & 12 & 1 & 0 & 19 & 6 \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ 16 & 13 & 22 & 8 & 10 & 26 & 0 \end{bmatrix}$$

Matrix P after the 5th iteration

$$\begin{bmatrix} 0 & 0 & 5 & 2 & 4 & 0 & 4 \\ 0 & 0 & 5 & 0 & 4 & 1 & 4 \\ 2 & 0 & 0 & 2 & 4 & 2 & 4 \\ 2 & 0 & 5 & 0 & 0 & 2 & 0 \\ 4 & 4 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 5 & 0 & 4 & 4 & 0 \end{bmatrix}$$

D6

$$\begin{bmatrix} 0 & 4 & 36 & 22 & 24 & 10 & 20 \\ 3 & 0 & 32 & 18 & 20 & 13 & 23 \\ 9 & 6 & 0 & 24 & 26 & 19 & 29 \\ 8 & 5 & 14 & 0 & 2 & 18 & 5 \\ 9 & 6 & 12 & 1 & 0 & 19 & 6 \\ \infty & \infty & \infty & \infty & \infty & 0 & 10 \\ 16 & 13 & 22 & 8 & 10 & 26 & 0 \end{bmatrix}$$

Matrix P after the 6th iteration

$$\begin{bmatrix} 0 & 0 & 5 & 2 & 4 & 0 & 6 \\ 0 & 0 & 5 & 0 & 4 & 1 & 4 \\ 2 & 0 & 0 & 2 & 4 & 2 & 4 \\ 2 & 0 & 5 & 0 & 0 & 2 & 0 \\ 4 & 4 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 4 & 5 & 0 & 4 & 4 & 0 \end{bmatrix}$$

D7

$$\begin{bmatrix} 0 & 4 & 36 & 22 & 24 & 10 & 20 \\ 3 & 0 & 32 & 18 & 20 & 13 & 23 \\ 9 & 6 & 0 & 24 & 26 & 19 & 29 \\ 8 & 5 & 14 & 0 & 2 & 18 & 5 \\ 9 & 6 & 12 & 1 & 0 & 19 & 6 \\ 26 & 23 & 32 & 18 & 20 & 0 & 10 \\ 16 & 13 & 22 & 8 & 10 & 26 & 0 \end{bmatrix}$$

Matrix P after the 7th iteration

$$\begin{bmatrix} 0 & 0 & 5 & 2 & 4 & 0 & 6 \\ 0 & 0 & 5 & 0 & 4 & 1 & 4 \\ 2 & 0 & 0 & 2 & 4 & 2 & 4 \\ 2 & 0 & 5 & 0 & 0 & 2 & 0 \\ 4 & 4 & 0 & 0 & 0 & 4 & 4 \\ 7 & 7 & 7 & 7 & 7 & 0 & 0 \\ 4 & 4 & 5 & 0 & 4 & 4 & 0 \end{bmatrix}$$

Problem 3-6

> path(7,3) = 5
>
> > path(7, 5) = 4
> >
> > > path(7, 4) = 0
> > > **v4**
> >
> > path(4, 5) = 0
> > **v5**
>
> path(5, 3) = 0
> path(4, 5) = 0

RESULT: v4 v5. (The shortest path from v7 to v3 is **v7->v4->v5->v3**.)

Problem 3-13

$A_1 = 10 \times 4$
$A_2 = 4 \times 5$
$A_3 = 5 \times 20$
$A_4 = 20 \times 2$
$A_5 = 2 \times 50$

Input : Let N be the number of matrix to be multiplied.

Let the total number of time is
$\sum_{diagonal=1}^{n-1} [(n - diagonal) \times (diagonal)]$

$\frac{n(n-1)(n+1)}{6} \epsilon O(n^3)$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ & 2 & 3 & 4 & 5 \\ & & 3 & 4 & 5 \\ & & & 4 & 5 \\ & & & & 5 \end{bmatrix}$$

Done fractorization,
$A_1[[[(A_2A_3)A_4]A_5]A_6]$

The final matrix M and P produce by minimum multiplication alogorithm.

$$\begin{bmatrix} 0 & 200 & 1200 & 320 & 1320 \\ 0 & 0 & 400 & 240 & 640 \\ 0 & 0 & 0 & 200 & 700 \\ 0 & 0 & 0 & 0 & 2000 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 4 \\ 0 & 0 & 2 & 2 & 4 \\ 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$\therefore [[A_1(A_2(A_3A_4))]A_5]$