**Name: Mohamed Hazem**

**ID:18107076**

**Q1-** **Consider a country having monetary coins of values (2, 3, and 7).**

**a. Using dynamic programming, write an algorithm that finds the number of ways to**

**construct an amount N.**

**b. What is the complexity of your algorithm?**

**c. Show the dynamic programming table for an input of N=10. For N=10 the solution**

**is 3: (2,2,2,2,2) (2,2,3,3) (3,7).**

**Answer:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C/P | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | i/j |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 3 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 2 |
| 7 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |

**ALGO**

**Function DP(){**

**Int coins[]={0,2,3,7};**

**Int n=10;**

**Int dptable[coins.length][n+1];**

**For(i=0;i<=coins;i++){**

**Dptable[i][0]=1;**

**}**

**For(i=1;i<=n;i++){**

**Dptable[0][i]=0;**

**}**

**For(i=0;i<coins;i++)**

**For(j=0;j<=n;j++){**

**If(j<coins[i])**

**Tab[i][j]=tab[i-1][j]**

**Else**

**Tab[i][j]=tab[i][j-coins[i] + tab[i-1][j]**

**}**

**Complexity (C\*N)**

**Q2-**

Consider the 0/1 knapsack problem. Given N objects where each object is specified by a weight and a profit, you are to put the objects in a bag of capacity C such that the sum of weights of the items in the bag does not exceed C and the profits of the items is maximized. Note that you cannot use an item type more than once.

1. a. Using dynamic programming, write an algorithm that finds the maximum total value according to the above constraints.
2. b. What is the complexity of your algorithm?
3. c. Show the dynamic programming table for the following data: W= { 2 , 7 , 1} , P={ 3 ,15 , 2 } and C=8.

**Answer:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Weight** | **Profit** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **2** | **3** | **0** | **0** | **3+0** | **3+0** | **3** | **3** | **3** | **3** | **3** |
| **7** | **15** | **0** | **0** | **3** | **3** | **3** | **3** | **3** | **15** | **15** |
| **1** | **2** | **0** | **2+0** | **3** | **2+3=5** | **5** | **5** | **5** | **15** | **17** |

**Algorithm**

**Function TabulationApproach ( capacity )**

**{ Weight []={0,2,7,1}**

**; Profit[]={0,3,15,2};**

**Dp[weight.length][capacity+1];**

**For i=0 to weight.length**

**For j=0 to capacity+1;**

**If(i==0 || j==0)**

**{ Dp[i][j]=0; Else if (j<weight[i])**

**{**

**Dp[i-1][j];**

**Else**

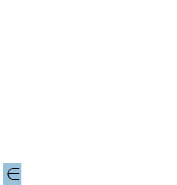
**Max( Dp[i-1][j-weight[i]]+profit[i],Dp[i-1][j]);**

**Complexity O(W\*C)**

**Q3**- Design a dynamic programming algorithm to solve the following problem (Floyd’s Algorithm):

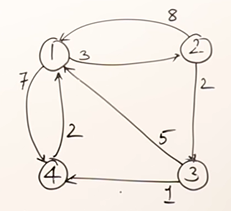
ALL-PAIRS SHORTEST PATH (APSP)

**INSTANCE**: Directed graph G = (V, E); edge cost function c: V × V → R.

**SOLUTION**: Distances d (i, j) for all i, j  V.

**Answer:**

**Example to make DP on it**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **D(i,j)** | **1** | **2** | **3** | **4** |
| **1** | **0** | **3** | **5** | **6** |
| **2** | **5** | **0** | **2** | **3** |
| **3** | **3** | **6** | **0** | **1** |
| **4** | **2** | **5** | **7** | **0** |

**D(I,j)=min{D(I,J),D(I,K},D(K,J)}**

**K🡺 stands for changing in intermediate matrix (DP table effecter)**

**Q4-**

Consider the Longest Common Subsequence (LCS) problem. You are given two strings A and B and you are to return the longest common subsequence in A and B. A subsequence of a string is defined to be the initial string with 0 or more characters removed. For example if A=”MOHAMED”, B=”AHMED”, the length of the LCS of A and B is 4 (“HMED” or “AMED”)

1. a. Using dynamic programming, write an algorithm that finds the length of the LCS of two given strings A and B.
2. b. What is the complexity of your algorithm?
3. c. Show the dynamic programming table for A=”MOHAMED” and B=”AHMED”.

**Answer:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A/B** | **0** | **M** | **O** | **H** | **A** | **M** | **E** | **D** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **A** | **0** | **0** | **0** |  | **1** | **1** | **1** | **1** |
| **H** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** |
| **M** | **0** | **1** | **1** | **1** | **1** | **1+1=2** | **2** | **2** |
| **E** | **0** | **1** | **0** | **1** | **1** | **2** | **2+1=3** | **3** |
| **D** | **0** | **1** | **0** | **1** | **1** | **2** | **3** | **3+1=4** |

**For(i=0;I<n;i++)**

**For(j=0;j<m;j++)**

**If (i==0 || j==0)**

**Dp[i][j]=0**

**Else if (array[i]==array[j])**

**Dp[i][j]=1+dp[i-1][j-1]**

**Else**

**Max(dp[i][j-1],dp[i-1][j])**

**Complexity(n\*m)**

**Q5-**The Longest Increasing Subsequence (LIS) problem is to find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order. For example, the length of LIS for {10, 22, 9, 33, 21, 50, 41, 60, and 80} is 6

and LIS is {10, 22, 33, 50, 60, and 80}.

**Answer :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I(Iterations)** | **Arr/list** | **10** | **22** | **9** | **33** | **21** | **50** | **41** | **60** |
| **1** | **\** | **1** | **2** | **1** | **1** | **1** | **1** | **1** | **1** |
| **2** | **\** | **1** | **2** | **1** | **1** | **1** | **1** | **1** | **1** |
| **3** | **\** | **1** | **2** | **1** | **3** | **1** | **1** | **1** | **1** |
| **4** | **\** | **1** | **2** | **1** | **3** | **2** | **1** | **1** | **1** |
| **5** | **\** | **1** | **2** | **1** | **3** | **2** | **4** | **1** | **1** |
| **6** | **\Final** | **1** | **2** | **1** | **3** | **2** | **4** | **4** | **5** |

**Q6-**Given a set of integers, the task is to divide it into two sets S1 and S2 such that the absolute difference between their sums is minimum.

If there is a set S with n elements, then if we assume Subset1 has m elements, Subset2 must have n-m elements and the value of abs (sum (Subset1) – sum (Subset2)) should be minimum.

**Answer:**

{1,6,11,5}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **N/C** | **0** | **1** | **2** | **3** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |
| **1** | **T** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **5** | **T** | **T** | **F** | **F** | **T** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **6** | **T** | **T** | **F** | **F** | **T** | **T** | **T** | **F** | **F** | **F** | **T** | **T** | **F** |
| **11** | **T** | **T** | **F** | **F** | **T** | **T** | **T** | **F** | **F** | **F** | **T** | **F** | **F** |

**Q7-**Given a distance ‘dist, count total number of ways to cover the distance with 1, 2 and 3 steps.

**Answer:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 3 (Distance of 3) |
| A=1 | A | A | A | (A)=1 |
| B=2 | A,B | A | A||B | (A+B||B+A||A)=3 |
| C=3 | A,B,C | A | A||B | (A+B||B+A||A||C)=4 |

**Q8-**Given a n\*n matrix where all numbers are distinct, find the maximum length path (starting from any cell) such that all cells along the path are in increasing order with a difference of 1. We can move in 4 directions from a given cell (i, j), i.e., we can move to (i+1, j) or (i, j+1) or (i-1, j) or (i, j-1) with the condition that the adjacent cells have a difference of 1.

**Answer:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **#** | **#** | **i/j** |
| **1** | **2** | **9** | **#** |
| **5** | **3** | **8** | **#** |
| **4** | **6** | **7** | **#** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **2** | **1+2** | **2+3||2+1** | **3+2** | **4+5** | **5+4** | **6+7** | **7+6||7+8** | **8+7||8+9** | **9+8** |
| **3** | **1+2+3** | **#5||#3** | **3+2+1** | **#9** | **#9** | **6+7+8** | **#13||7+8+9** | **8+7+6||#17** | **9+8+7** |
| **4** | **#6** | **##** | **#6** | **#** | **#** | **6+7+8+9** | **#13||#24** | **#21||#17** | **9+8+7+6** |
| **#** | **#** | **##** | **#** | **#** | **#** | **#30** | **##** | **##** | **#30** |

**Q9-**Given a set of non-negative integers, and a value *sum*, determine if there is a subset of the given set with sum equal to given *sum*.

**Answer:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I/N** | **1** | **2** | **3** | **True Color** |
| **0** | **0** | **0+0** | **0+0+0** | **False Color** |
| **2** | **2** | **2+0** | **2+0+0** | **Stop Color** |
| **3** | **3** | **3+2** | **3+2+0** |  |
| **4** | **4** | **4+3** | **4+3+2** |  |
| **5** | **5** | **5+4** |  |  |
| **12** |  |  |  |  |
| **34** |  |  |  |  |

**Q10-**Given two strings str1 and str2, find the shortest string that has both str1 and str2 as subsequences.

**Answer:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Str1/Str2** | **#** | **g** | **e** | **e** | **k** | **True Color** |
| **I=1** | **e** | **g≠e** | **e=e geke∈geek** | **e=e geeke∈geek** |  | **False Color** |
| **I=2** | **k** |  |  |  |  | **Stop Color** |
| **I=3** | **e** |  |  |  |  |  |
| **I=4** | **#** |  |  |  |  |  |

**Q11-**Partition problem is to determine whether a given set can be partitioned into two subsets such that the sum of elements in both subsets is same

**Answer:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sum/arr** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **True Color** |
| **1&5,5,11** | **T&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **False Color** |
| **1,5&5,11** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **T&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **Stop Color** |
| **5,5&1,11** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **T&F** | **F&F** |  |
| **1,5,5&11** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **F&F** | **T&T** |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**Q12-**Given a rod of length n inches and an array of prices that contains prices of all pieces of size smaller than n. Determine the maximum value obtainable by cutting up the rod and selling the pieces. For example, if length of the rod is 8 and the values of different pieces are given as following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6)

**Anwser:**

**DP table for another example**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Len** | **1** | **2** | **3** | **4** | **5** |
| **Val** | **2** | **7** | **10** | **11** | **12** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Len** | **0** | **1** | **2** | **3** | **4** | **5** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **2** | **4** | **6** | **8** | **10** |
| **1 2** | **0** | **2** | **7** | **9** | **14** | **16** |
| **1 2 3** | **0** | **2** | **7** | **10** | **14** | **17** |
| **1 2 3 4** | **0** | **2** | **7** | **10** | **14** | **17** |
| **1 2 3 4 5** | **0** | **2** | **7** | **10** | **14** | **17** |

**Q13-**Given a value N, if we want to make change for N cents, and we have infinite supply of each of S = {S1, S2,... ,Sm} valued coins, how many ways can we make the change? The order of coins doesn’t matter.

For example, for N = 4 and S = {1,2,3}, there are four solutions: {1,1,1,1},{1,1,2},{2,2},{1,3}. So output should be 4. For N = 10 and S = {2, 5, 3, 6}, there are five solutions: {2,2,2,2,2},

{2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}. So the output should be 5.

**Answer:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N/S** | **0** | **1** | **2** | **3** | **4** |
| **0** | **1** | **0** | **0** | **0** | **0** |
| **1** | **1** | **1+0** | **1+0** | **1+0** | **1+0** |
| **1 2** | **1** | **0+1** | **1+1** | **1+1** | **2+1=3** |
| **1 2 3** | **1** | **0+1** | **0+2** | **1+2=3** | **1+3=4** |

**Q14-**Given a rope of length n meters, cut the rope in different parts of integer lengths in a way that maximizes product of lengths of all parts. You must make at least one cut. Assume that the length of rope is more than 2 meters.

**Answer:**

**Example N=7**

|  |  |
| --- | --- |
| **Max/N** | **Max=1** |
| **N=7** | **N-3>4? Max\*=3**  **N-3** |
| **N=4** | **N-3>4?**  **Max\*=N** |
|  | **Max=12** |

**Example N=20**

|  |  |
| --- | --- |
| **Max/N** | **Max=1** |
| **N=20** | **N-3>4? Max\*=3**  **N-3** |
| **N=17** | **N-3>4?**  **Max\*=3**  **N-3** |
| **N=14** | **N-3>4?**  **Max\*=3**  **N-3** |
| **N=11** | **N-3>4?**  **Max\*=3**  **N-3** |
| **N=8** | **N-3>4?**  **Max\*=3**  **N-3** |
| **N=5** | **N-3>4?**  **Max\*=3**  **N-3** |
| **N=2** | **N-3>4?**  **Max\*=N** |
|  | **Max=1,458** |

**Q15-**Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words. See following examples for more details.

This is a famous Google interview question, also being asked by many other companies now a days.

**Answer:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dictionary** | **I** | **Like** | **sam** | **sung** | **samsung** | **mobile** | **ice** | **cream** | **icecream** | **man** | **go** | **mango** |
| **Word** |  |  |  |  |  |  |  |  |  |  |  |  |
| **ILikesamsung** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **I LikeSamsung** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **I Like Samsung** | **T** | **T** | **F** | **F** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **I Like Sam sung** | **T** | **T** | **T** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ilikeicecreamandmango** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **I likeicecreamandmango** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **I like icecreamandmango** | **T** | **T** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** | **F** |
| **I like ice creamandmango** | **T** | **T** | **F** | **F** | **F** | **F** | **T** | **F** | **F** | **F** | **F** | **F** |
| **I like ice cream andmango** | **T** | **T** | **F** | **F** | **F** | **F** | **T** | **T** | **F** | **F** | **F** | **F** |
| **I like ice cream and mango** | **T** | **T** | **F** | **F** | **F** | **F** | **T** | **T** | **F** | **F** | **F** | **T** |
| **I like ice cream and man go** | **T** | **T** | **F** | **F** | **F** | **F** | **T** | **T** | **F** | **T** | **T** | **F** |

**Q16-** The maximum subarray problem is the task of finding the contiguous subarray within a one dimensional array of numbers which has the largest sum. For example, for the sequence of values −2, 1, −3, 4, −1, 2, 1, −5, 4; the contiguous subarray with the largest sum is 4, −1, 2, 1,

with sum 6. Given the following recurrence, write a dynamic programming algorithm to find the maximum subarray for the elements in array A.

**S [0] = A [0]**

Required to do:

1. Write a complete dynamic programming algorithm.

2. Find the complexity of the algorithm.

3. Fill in the dynamic programming table for input -2, 1, -3, 4, -1, 2, 1, -5, 4 and show where the answer is in the table.

**Answer:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **I** | **N/S** | -2 | 1 | -3 | 4 | -1 | 2 | 1 | -5 | 4 |
| 0 | **Max\_S=** | 0 |  |  |  |  |  |  |  |  |
| 1 | **Max\_S=** |  | 1>0 |  |  |  |  |  |  |  |
| 2 | **Max\_S=** |  |  | 0 |  |  |  |  |  |  |
| 3 | **Max\_S=** |  |  |  | 4&4>1 |  |  |  |  |  |
| 4 | **Max\_S=** |  |  |  |  | 3&3!>4 |  |  |  |  |
| 5 | **Max\_S=** |  |  |  |  |  | 5&5>4 |  |  |  |
| 6 | **Max\_S=** |  |  |  |  |  |  | 6&6>5 |  |  |
| 7 | **Max\_S=** |  |  |  |  |  |  |  | 1&1!>6 |  |
| 8 | **Max\_S=** |  |  |  |  |  |  |  |  | 5&5!>6 |

**Symbol !> mean not more than**

**Kadane’s Algorithm:**

**Initialize:**

**max\_so\_far = INT\_MIN**

**max\_ending\_here = 0**

**Loop for each element of the array**

**(a) max\_ending\_here = max\_ending\_here + a[i]**

**(b) if(max\_so\_far < max\_ending\_here)**

**max\_so\_far = max\_ending\_here**

**(c) if(max\_ending\_here < 0)**

**max\_ending\_here = 0**

**return max\_so\_far**

**Complexity O(N)**

Lets take the example:

**{-2, 1, -3, 4, -1, 2, 1, -5, 4 }**

**for i=0, a[0] = -2**

**max\_ending\_here = max\_ending\_here + (-2)**

**Set max\_ending\_here = 0 because max\_ending\_here < 0**

**for i=1, a[1] = 1**

**max\_ending\_here = max\_ending\_here + (1)**

**Set max\_ending\_here = 1**

**max\_so\_far is updated to 1 because max\_ending\_here greater**

**than max\_so\_far which was 0 till now**

**for i=2, a[2] = -3**

**max\_ending\_here = max\_ending\_here + (-3)**

**max\_ending\_here = 0**

**for i=3, a[3] = 4**

**max\_ending\_here = max\_ending\_here + (4)**

**Set max\_ending\_here = 4**

**max\_so\_far is updated to 4 because max\_ending\_here greater**

**than max\_so\_far which was 1 till now**

**for i=4, a[4] = -1**

**max\_ending\_here = max\_ending\_here + (-1)**

**max\_ending\_here = 3**

**for i=5, a[5] = 2**

**max\_ending\_here = max\_ending\_here + (2)**

**max\_ending\_here = 5**

**max\_so\_far is updated to 5 because max\_ending\_here greater**

**than max\_so\_far which was 4 till now**

**for i=6, a[6] = 1**

**max\_ending\_here = max\_ending\_here + (1)**

**max\_ending\_here = 6**

**max\_so\_far is updated to 6 because max\_ending\_here greater**

**than max\_so\_far which was 5 till now**

**for i=7, a[7] = -5**

**max\_ending\_here = max\_ending\_here + (-5)**

**max\_ending\_here = 1**

**for i=8, a[8] = 4**

**max\_ending\_here = max\_ending\_here + (4)**

**max\_ending\_here = 5**

**max\_ending\_here at end equal to 6**