Problem 1

/\* Question 2

PART A

"0-4" === 7 ways

1. 1-1-1-1

2. 1-1-2

3. 1-2-1

4. 2-1-1

5. 2-2

6. 1-3

7. 3-1

PART B

BASE CASES

V[0] = 1 ways

V[1] = 1 ways

V[2] = 2 ways

GENERAL CASE

V[K] = V[K-1] + V[K-2] + V[K-3]

e.g

V[4] = V[3] + V[2] + V[1]

V[4] = 4 + 2 + 1

V[4] = 7 ways

\*/

//Question3 - Part A

//carac

/\* Question 3 -Part B

\*/

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  |  | **C** | **H** | **A** | **R** | **A** | **C** | **T** | **E** | **R** |
| 1 | **C** | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | **H** | 0 | 0 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | **A** | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| 4 | **R** | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 2 |
| 5 | **A** | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 |
| 6 | **C** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 2 |
| 7 | **T** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 |
| 8 | **E** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 9 | **R** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **C** | **H** | **A** | **R** | **A** | **C** | **T** | **E** | **R** |
| **C** | / | / | | | ← | ← | ← | ← | ← | ← |
| **H** |  | / | / |  | ← | ← | ← | ← | ← |
| **A** |  |  | / | / | | | | | | | | | | |
| **R** |  |  |  | / | / | | | | | | | | |
| **A** |  |  |  |  | / | / | | | | | | |
| **C** |  |  |  |  |  | / | / | | | | |
| **T** |  |  |  |  |  |  | / | / | | |
| **E** |  |  |  |  |  |  |  | / | / |
| **R** |  |  |  |  |  |  |  |  | / |

Qs 4. for string in k: // k=subset

for letter in string:

// iterate each letter till the length of the word in the

// Sn array

// maintain a variable to track the no of letter matches and store the word in matches[i]

// if length of the word == no of matches in current word AND matches[i]==word

// print the substring

Time Complexity: O(n^2)

Problem 2

Qs1a – **Cap =50**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item No.** | **Weight** | **Price** | **Price/Weight** |
| **1** | 10 | $60 | 6 |
| **2** | 20 | $100 | 5 |
| **3** | 30 | $120 | 4 |

Fractional KnapSack += 60+100+(20/30 \* 120 )

= 240

1B – Pseudocode Fractional Knapsack

1.sort on the basis of price/weight ratio in decreasing order

2.for item in items:

if weight <= cap:

add price to profit

if weight> cap: //whole weight that of the curr item > cap

add the remaining amount of capactiy/total weight (ratio)of that item \* price

of the that item

3.return profit

Qs2

|  |  |  |
| --- | --- | --- |
| **Activity** | **Start Time** | **Finish Times** |
| **1** | 1 | 4 |
| **2** | 2 | 3 |
| **3** | 3 | 6 |
| **4** | 5 | 7 |
| **5** | 4 | 5 |

a) In decreasing order of finish times will be inefficient as after picking activity 4 with finish time 7 and then looking for a non overlapping activity, no other activity can be picked

b)In increasing order of start times will be inefficient as after picking activity 4 with start time 5 and then looking for a non overlapping activity, no other activity can be pciked

Qs3

1.Sort on descending order of finish times the set of lectures

2.Map each lecture to each of the n rooms

3.Now we have the remaining set of lectures that are unmapped

4.Pick the first lecture and compare if the finish time of that lecture < starting time of the first lecture in the remaining sets assign that room to it otherwise assign a new room to it, and add the newly alloted room to the list of rooms of lectures so that as soon as it gets free it can be reused.

Qs4.

1.Sort on increasing order of air fares

2.Pick the first country and add it to the visited list + update total air fare

3.Select the next item in air fares table

4.Now check if the next destination is not in the visited list already

5.if not add to the visited list and update total air fare

6.Repeat until all countries in visited list

7.Return total air fare and visited list

Time Complexity = O(n)