# Design and Analysis of Algorithms

**Q1)** Imagine that you are moving to a new location and you want to find an apartment that you want to live in. You are given a list a list of street blocks and at each block there's an apartment that you might want to consider living in. You want to make sure that the apartment you live in is really good for you and the way that you're going to determine if it's good for you is if the apartment is close to some buildings that you really value so for example you might really value being close to a gym or being close to the office etc.

You are given information about at each block whether or not there's a building like an office or like not a school or a gym and you want to find the apartment that minimizes the farthest distance that you would have to walk to to get to all the buildings that you value.

Sample Input:

```
 Blocks = [
{
“gym”: false
“school”: true,
“store” = false,
},
{
“gym”: true
“school”: false,
“store” = false,
},
{
“gym”: true
“school”: true,
“store” = false,
},
{
“gym”: false
“school”: true,
“store” = false,
},
{
“gym”: false
“school”: true,
“store” = true,
},
]
```

The second input is a list of required buildings. It can have any number of required buildings as given in the following sample input.

Reqs = [“gym”, “school”, “store”]
Reqs = [“gym”, “school”,]

Every block has a building. You have to find the block that will minimize the farthest distance that you would have to go to reach all of your requirements. For example in the above sample input if Reqs = [“gym”, “school”, “store”], then block 4 is correct answer as the farthest distance to any required building is 1. (In block 4, distance to school is 0, distance to gym is 1 and distance to store is 1.) Input size is n, where n is number of input blocks,

(a) Write $O(n^2)$ brute force program for solving this problem.
(b) Write O(n) dynamic programming program using memorization.

**Q2)** There is a grid of farms with x*y dimensions. A farmer has to go from top left most farm to bottom right most farm. How many distinct paths the farmer can take to reach his destination? He can only move right and down. Write the most efficient program to solve this problem and also write its time complexity.

**Q3)** You are given a set of substrings of DNA sequence. A DNA sequence is defined as a string of 4 alphabets (A, G, C, T) as follows:
GCAACGTTAGA….
A substring of DNA sequence is a consecutive string. For example, ACG is a substring of above DNA sequence but GGA is not substring of this sequence.
Given a new DNA sequence $S_n$ with length n and a set of substrings K of another DNA sequence, find out if the new DNA sequence $S_n$ can be divided into substrings of the given set K. For example:
Let $S_n$ = GCAACGTTAGA
K = {AGA, GT, GC, AACG, TT}
$S_n$ can be divided into following substrings from K
GC, AACG, TT, AGA
Second Example:
Let $S_n$ = GCAGCCTGTACT
K = {AG, GT, AACG, CC}
$S_n$ cannot be divided into substrings from K
Given a DNA sequence $S_n$ of length n and a set K of substrings, describe an efficient algorithm that will detect whether or not the input DNA sequence $S_n$ can be split into substrings of set K. Your algorithm should also print the substrings. Analyze the time complexity of your algorithm.
You can assume that there is a function that given a substring, checks in O(1) time if the substring belongs to input set K of substrings.
**Input to algorithm:** A DNA sequence $S_n$ of length n and a set K of substrings

**Q4)** A person is booking a flight to bring back Pakistanis stranded in some other country due to COVID-19 pandemic. The flight has fixed passenger capacity M (M passengers can travel other than the airline crew). He wants to book as many passengers as possible for this flight without exceeding the passenger capacity of the aircraft but there is a constraint. Some people have families with them and they only want to book the flight if all family members get a seat. Every person has different number of family members with him. The number of family members with 1 person can range from 0 to k (0 means he will travel alone and k means he needs k+1 seats in order to travel). There are n person interested in booking flights and each person has different number of family members (0 to k). Give an efficient algorithm for this problem. Be sure to prove that your algorithm yields an optimal solution and analyze the time complexity.

**Input to algorithm:** Number of family members (0 to k) for each of the n persons interested in booking the flight.

**Q5)** Now at this point you have become an algorithm expert and it's the right time to increase the difficulty level. You have given an array M = {1 ,2, 3, 8, 6, 7, 9, 10, 2, 1}. You have to calculate the longest increasing with decreasing subsequence in the array. Let's understand what a subsequence is through an example.

1, 3, 6, 10, 1 → Subsequence

In Subsequence we can skip some elements but it should be done in increasing order, like 1 3 2 6 is not subsequence. Order of elements in parent array should remain same in subsequence.

In the problem you have to find such subsequence which has 2 parts, whose first part is **increasing** in order and second part is **decreasing** in order and it is the ***longest*** subsequence.

Here, (red is increasing subsequence and green is decreasing subsequence)

i) 1 2 3 8 9 10 2 1
ii) 1 2 3 6 7 9 10 2 1

are one of such subsequences but we can see that second subsequence is longest so we will choose this one

1) Define a subproblem for it.
2) Provide its recurrence.
3) Pseudo code for the algorithm devised.
4) Run time complexity

**Q6)** Given a matrix M * N of integers where each cell has a cost associated with it., find the minimum cost to reach the last cell (M-1, N-1) of the matrix from its first cell (0,0). We can only move one unit right (Column No + 1) and one unit down (Row No + 1)

For example from index (i , j) you can move to (i , j+1) and (i+1 , j)

where i = row no and j = column no.

| | | | | |
|---|---|---|---|---|
| 4 | 7 | 8 | 6 | 4 |
| 6 | 7 | 3 | 9 | 2 |
| 3 | 8 | 1 | 2 | 4 |
| 7 | 1 | 7 | 3 | 7 |
| 2 | 9 | 8 | 9 | 3 |

| | | | | |
|---|---|---|---|---|
| **4** | 7 | 8 | 6 | 4 |
| **6** | **7** | **3** | 9 | 2 |
| 3 | 8 | **1** | **2** | 4 |
| 7 | 1 | 7 | **3** | **7** |
| 2 | 9 | 8 | 9 | **3** |

Path = 4 -> 6 -> 7 -> 3 -> 1-> 2 -> 3 ->7-> 3 = 36 cost

Provide a DP solution for it, also provide the following things with it.

1) Sub Problem definition
2) Recurrence for Subproblem
3) Time Complexity