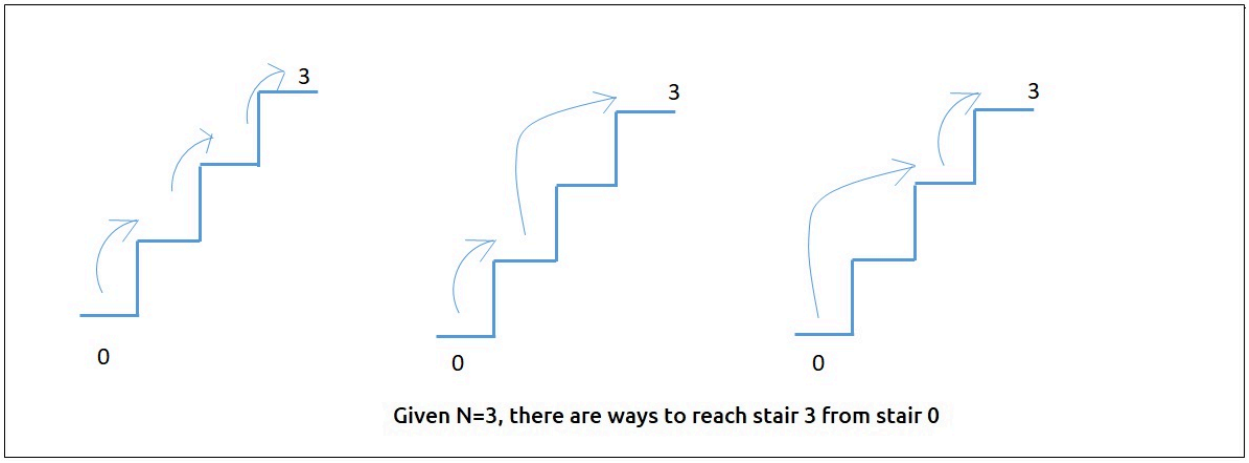
2**Topic: Dynamic Programming**
Problem 001

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P001.cpp [C++ STL] and push to your pvt Github repo.

Problem Statement: Given a number of stairs. Starting from the 0th stair we need to climb to the “Nth” stair. At a time we can climb either one or two steps. We need to return the total number of distinct ways to reach from 0th to Nth stair.

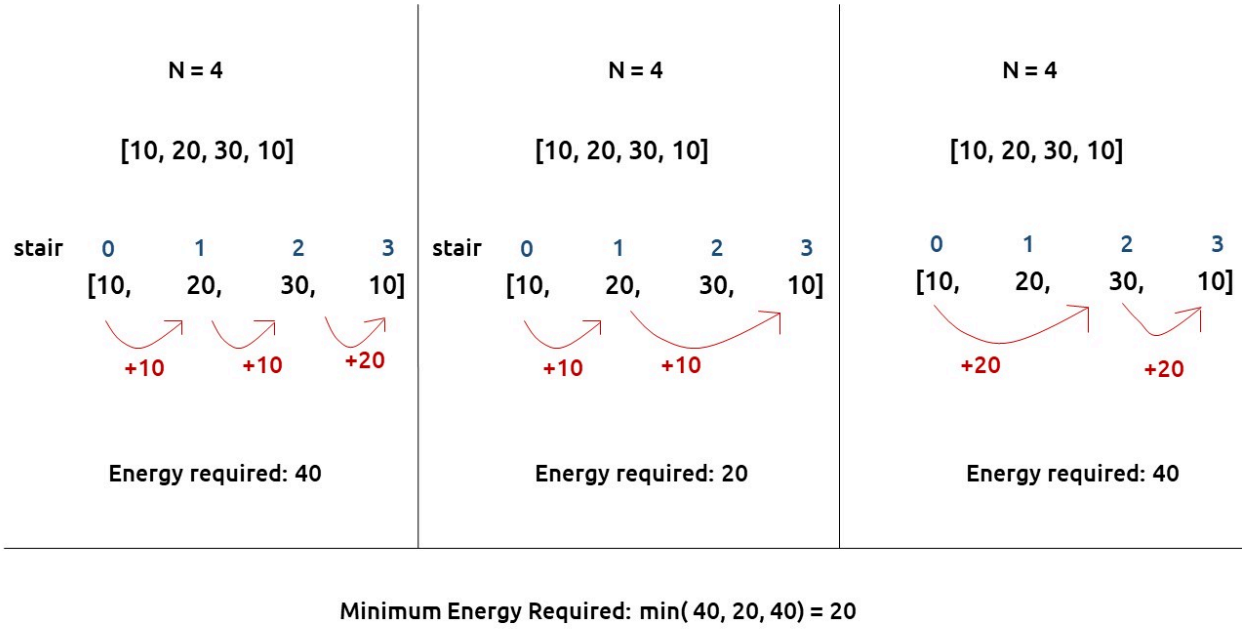


Topic: Dynamic Programming
Problem 002

Handwritten Task:
Describe the following in your notebook
a) Memorization Approach [Briefly as step-wise manner]
b) Tabulation Approach [Briefly as step-wise manner]
c) Space Optimization Approach
d) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P002.cpp [C++ STL] and push to your pvt Github repo.

Problem Statement:
Given a number of stairs and a frog, the frog wants to climb from the 0th stair to the (N-1)th stair. At a time the frog can climb either one or two steps. A height[N] array is also given. Whenever the frog jumps from a stair i to stair j, the energy consumed in the jump is abs(height[i]-height[j]), where abs() means the absolute difference. We need to return the minimum energy that can be used by the frog to jump from stair 0 to stair N-1.



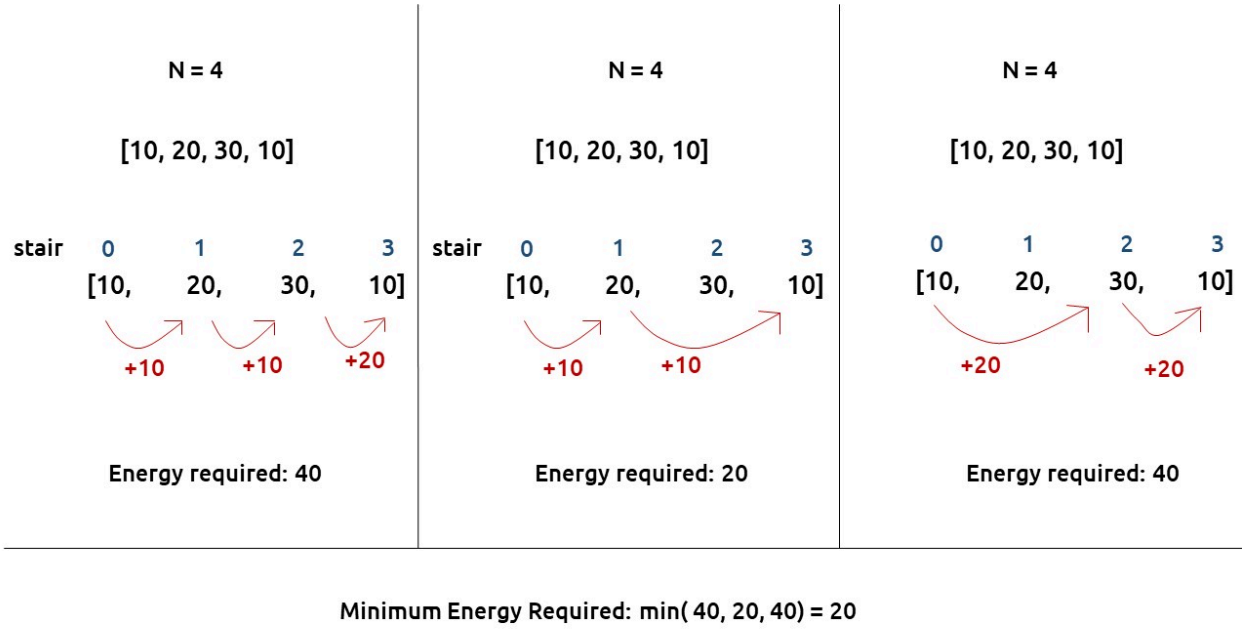
Minimum Energy Required: min(40, 20, 40) = 20

Topic: Dynamic Programming
Problem 003

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P003.cpp [C++ STL] and push to your pvt Github repo.

Problem Statement:
This is a follow-up question to “Frog Jump” discussed in the previous question. In the previous question, the frog was allowed to jump either one or two steps at a time. In this question, the frog is allowed to jump up to ‘K’ steps at a time. If K=4, the frog can jump 1,2,3, or 4 steps at every index.



Minimum Energy Required: min(40, 20, 40) = 20

Topic: Dynamic Programming
Problem 004

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P004.cpp [C++ STL] and push to your pvt Github repo.

Problem Statement:
Given an array of 'N' positive integers, we need to return the maximum sum of the subsequence such that no two elements of the subsequence are adjacent elements in the array.
Note: A subsequence of an array is a list with elements of the array where some elements are deleted (or not deleted at all) and the elements should be in the same order in the subsequence as in the array.

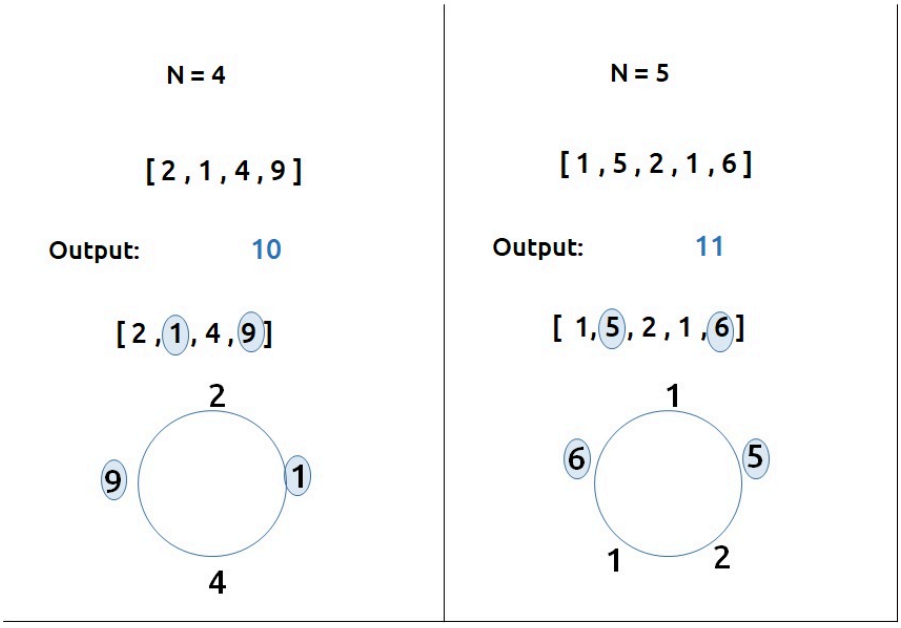
N = 3	N = 4	N = 9
[1 , 2 , 4]	[2 , 1 , 4 , 9]	[1 , 2 , 3 , 1, 3, 5, 8, 1, 9]
Output: 5	Output: 11	Output: 24
[1, 2 , 4]	[2, 1 , 4 , 9]	[1, 2 , 3 , 1, 3, 5, 8, 1, 9]

Topic: Dynamic Programming
Problem 005

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P005.cpp [C++ STL] and push to your pvt Github repo.

Problem Statement
A thief needs to rob money in a street. The houses in the street are arranged in a circular manner. Therefore the first and the last house are adjacent to each other. The security system in the street is such that if adjacent houses are robbed, the police will get notified.
Given an array of integers “Arr" which represents money at each house, we need to return the maximum amount of money that the thief can rob without alerting the police.



Topic: Dynamic Programming
Problem 006

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P006.cpp [C++ STL] and push to your pvt Github repo.

Problem Statement: A Ninja has an 'N' Day training schedule. He has to perform one of these three activities (Running, Fighting Practice, or Learning New Moves) each day. There are merit points associated with performing an activity each day. The same activity can't be performed on two consecutive days. We need to find the maximum merit points the ninja can attain in N Days. We are given a 2D Array POINTS of size 'N*3' which tells us the merit point of specific activity on that particular day. Our task is to calculate the maximum number of merit points that the ninja can earn.

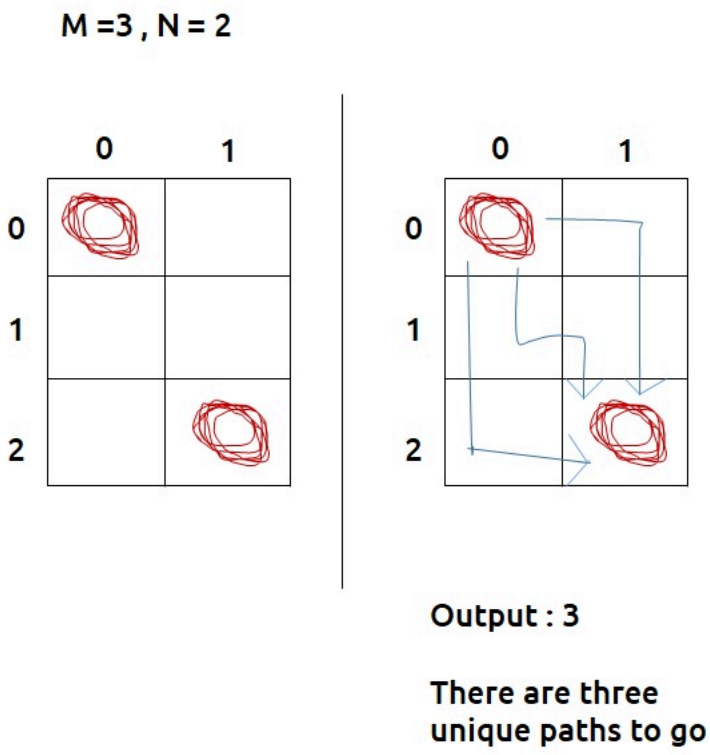
```
Days = 3

Points =    10, 40, 70    // Day 0
            20, 50, 80    // Day 1
            30, 60, 90    // Day 2

Output: 210    // 70 (Day 0 )+ 50 (Day 2)+ 90 (Day 3)
```

Topic: Dynamic Programming
Problem 007

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P007.cpp [C++ STL] and push to your pvt Github repo.

Given two values M and N, which represent a matrix[M][N]. We need to find the total unique paths from the top-left cell (matrix[0][0]) to the rightmost cell (matrix[M-1][N-1]).
At any cell we are allowed to move in only two directions:- bottom and right.



Topic: Dynamic Programming
Problem 008

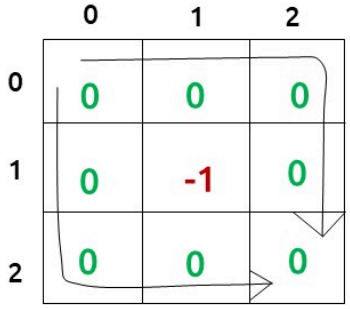
Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P008.cpp [C++ STL] and push to your pvt Github repo.

Problem Description:
We are given an “N*M” Maze. The maze contains some obstacles. A cell is ‘blockage’ in the maze if its value is -1. 0 represents non-blockage. There is no path possible through a blocked cell.
We need to count the total number of unique paths from the top-left corner of the maze to the bottom-right corner. At every cell, we can move either down or towards the right.

N =3 , M = 3

	0	1	2
0	0	0	0
1	0	-1	0
2	0	0	0



Output : 2

There are two unique paths to go

Topic: Dynamic Programming
Problem 009

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P009.cpp [C++ STL] and push to your pvt Github repo.

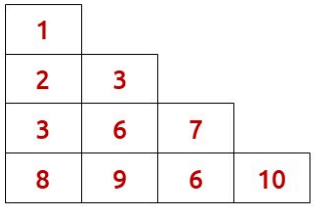
Problem Description:
We are given an “N*M” matrix of integers. We need to find a path from the top-left corner to the bottom-right corner of the matrix, such that there is a minimum cost past that we select.
At every cell, we can move in only two directions: right and bottom. The cost of a path is given as the sum of values of cells of the given matrix.

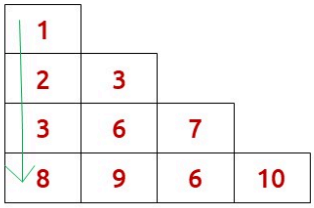
Topic: Dynamic Programming
Problem 010

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P010.cpp [C++ STL] and push to your pvt Github repo.

Problem Description:
We are given a Triangular matrix. We need to find the minimum path sum from the first row to the last row.
At every cell we can move in only two directions: either to the bottom cell (↓) or to the bottom-right cell(↘)





Minimum Path Sum = 14
(1 + 2 + 3 + 8)

Topic: Dynamic Programming
Problem 011

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P011.cpp [C++ STL] and push to your pvt Github repo.

Problem Description:
We are given an 'N*M' matrix. We need to find the maximum path sum from any cell of the first row to any cell of the last row.
At every cell we can move in three directions: to the bottom cell (↓), to the bottom-right cell(↘), or to the bottom-left cell(↙).

N=4, M=4

1	2	10	4
100	3	2	1
1	1	20	2
1	2	2	1

N=4, M=4

1	2	10	4
100	3	2	1
1	1	20	2
1	2	2	1

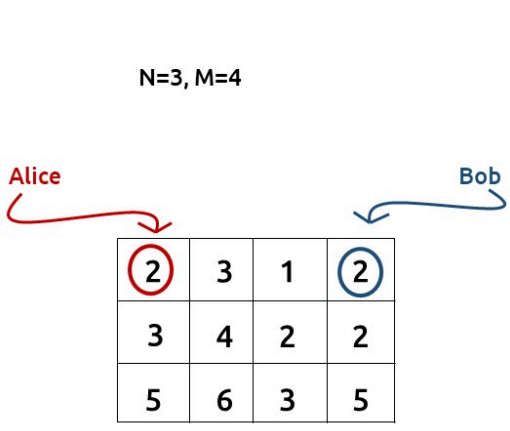
Maximum Path Sum: 105
(2+100+1+2)

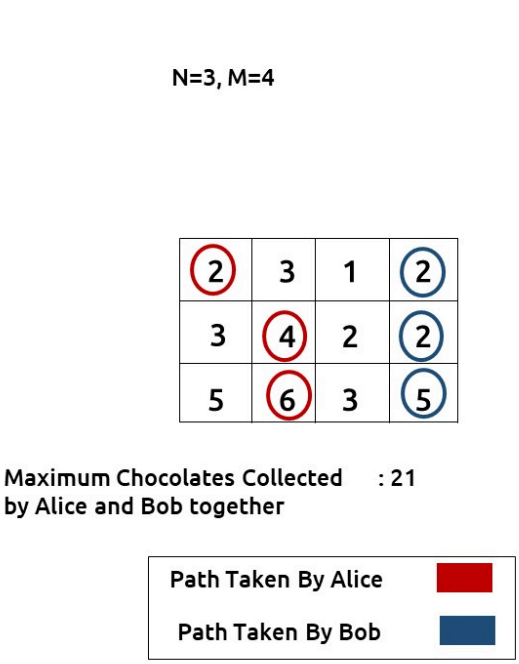
Topic: Dynamic Programming
Problem 012

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P012.cpp [C++ STL] and push to your pvt Github repo.

Problem Description:
We are given an 'N*M' matrix. Every cell of the matrix has some chocolates on it, mat[i][j] gives us the number of chocolates. We have two friends 'Alice' and 'Bob'. initially, Alice is standing on the cell(0,0) and Bob is standing on the cell(0, M-1). Both of them can move only to the cells below them in these three directions: to the bottom cell (↓), to the bottom-right cell(↘), or to the bottom-left cell(↙).
When Alica and Bob visit a cell, they take all the chocolates from that cell with them. It can happen that they visit the same cell, in that case, the chocolates need to be considered only once.
They cannot go out of the boundary of the given matrix, we need to return the maximum number of chocolates that Bob and Alice can together collect.



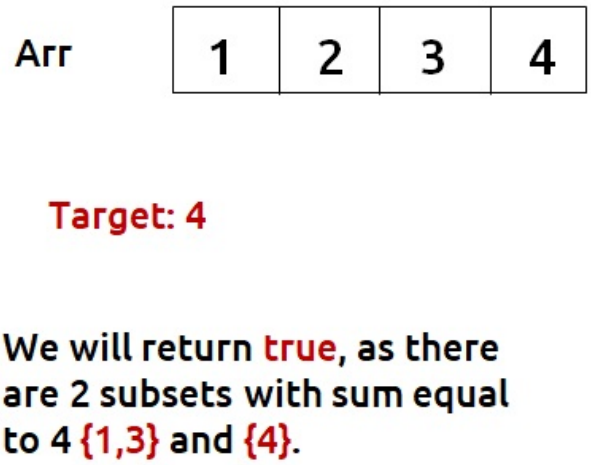


Topic: Dynamic Programming
Problem 013

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P013.cpp [C++ STL] and push to your pvt Github repo.

A subset/subsequence is a contiguous or non-contiguous part of an array, where elements appear in the same order as the original array.
For example, for the array: [2,3,1] , the subsequences will be [{2},{3},{1},{2,3},{2,1},{3,1},{2,3,1}} but {3,2} is not a subsequence because its elements are not in the same order as the original array.
We are given an array 'ARR' with N positive integers. We need to find if there is a subset in "ARR" with a sum equal to K. If there is, return true else return false.

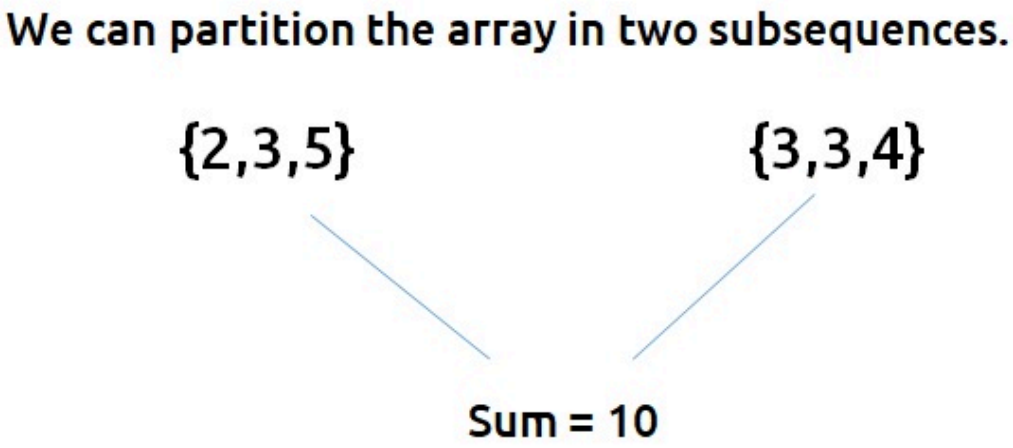


Topic: Dynamic Programming
Problem 014

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P014.cpp [C++ STL] and push to your pvt Github repo.

We are given an array 'ARR' with N positive integers. We need to find if we can partition the array into two subsets such that the sum of elements of each subset is equal to the other.
If we can partition, return true else return false.

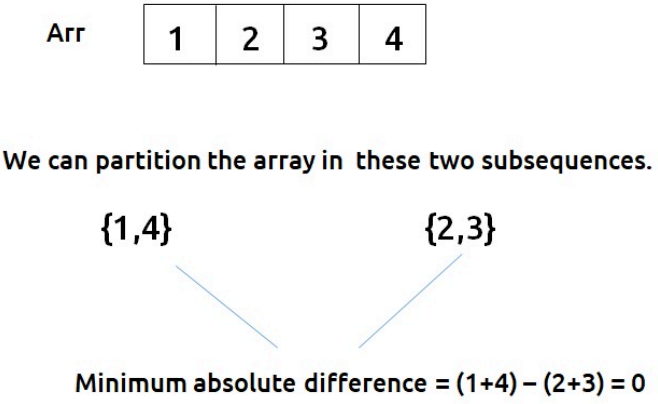


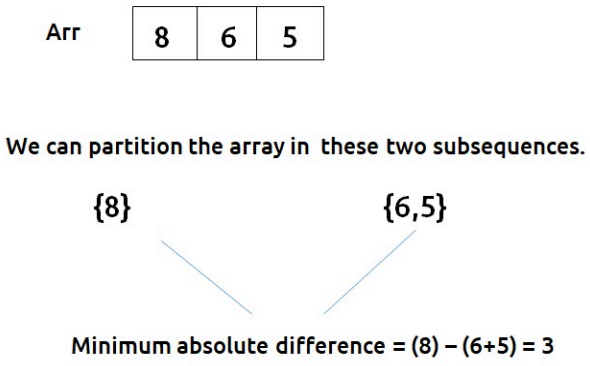
Topic: Dynamic Programming
Problem 015

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P015.cpp [C++ STL] and push to your pvt Github repo.

We are given an array 'ARR' with N positive integers. We need to partition the array into two subsets such that the absolute difference of the sum of elements of the subsets is minimum. We need to return only the minimum absolute difference of the sum of elements of the two partitions.





Topic: Dynamic Programming
Problem 016

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P016.cpp [C++ STL] and push to your pvt Github repo.

We are given an array 'ARR' with N positive integers and an integer K. We need to find the number of subsets whose sum is equal to K.

We can have the following unique subsets with target Sum of 3

[1,2] Arr[0] Arr[1]

[1,2] Arr[0] Arr[2]

[3] Arr[3]

Total Count of subsets: 3

Topic: Dynamic Programming
Problem 017

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P017.cpp [C++ STL] and push to your pvt Github repo.

Count Partitions with Given Difference
This article will be divided into two parts:

- First, we will discuss an extra edge case of the problem discussed in Count Subsets wKith Sum , and then,
- we will discuss the problem for this article: Partitions with Given Difference.

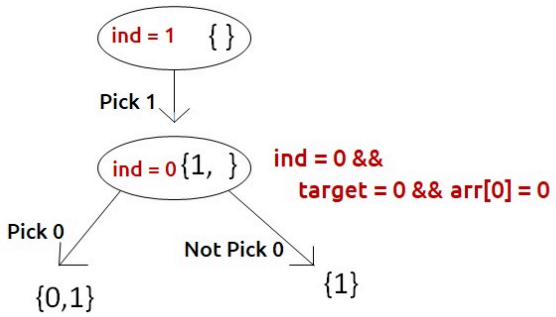
Part 1: Extra edge case for the problem Count Subsets with Sum K
In the problem Count Subsets with Sum K, the problem constraints stated that an array element is greater than 0, so the code we have written there works perfectly for the given constraints.
If the constraints mentioned that an array element can also be equal to 0 and the target sum can also be 0, then that code will fail. To understand it we will take an example:
Let the target arr = [0,0,1] and the target = 1.
The previous code will give us the answer 1 as it first takes the element arr[2] and then finds the answer by picking it. Then from the base condition, we will return 0 (as the target will become 0 by picking 1). But for this question, the answer will be 4 with the following subsets({0,1},{0,1},{0,0,1} and {1}).
Therefore we need to modify the base conditions in order to handle the changes. These are the base conditions of that problem.

```
f(ind,target) {

    if( target == 0) return 1
    if( ind == 0)    return arr[ind]==target


    }
```

```
f(ind,target) {

    if( ind == 0)    return arr[ind]==target


    }
```

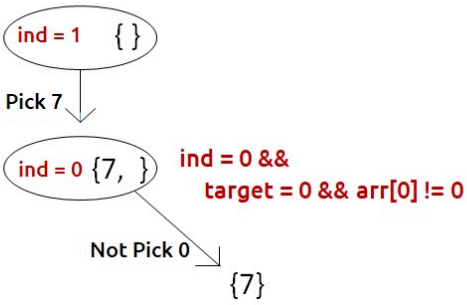
```
f(ind,target) {

    if( ind == 0) {
        if( target ==0 && arr[0] == 0)
            return 2



    }
```

arr: [0,1] target :1



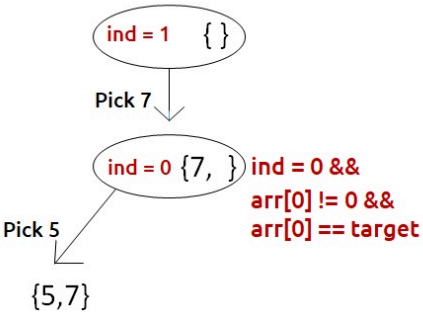

```
f(ind,target) {

    if( ind == 0) {
        if( target ==0 && arr[0] == 0)
            return 2
        if(target==0)
            return 1

    }
```

arr: [5,7] target :7




```
f(ind,target) {

    if( ind == 0) {
        if( target ==0 && arr[0] == 0)
            return 2
        if(target==0 || arr[0] == target)
            return 1

    }
```

arr: [5,7] target :12



Topic: Dynamic Programming
Problem 018

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P018.cpp [C++ STL] and push to your pvt Github repo.

Problem Description:
We are given an array ‘ARR’ of size ‘N’ and a number ‘Target’. Our task is to build an expression from the given array where we can place a ‘+’ or ‘-’ sign in front of an integer. We want to place a sign in front of every integer of the array and get our required target. We need to count the number of ways in which we can achieve our required target.

Target: 3

+1 -2 +3 +1 = 3

+1 -2 +3 +1 = 3

There are **2** ways

Problem 019

Handwritten Task:
Describe the following in your notebook
a) Brute Approach [Briefly as step-wise manner]
b) Optimal Approach [Briefly as step-wise manner]
c) Time and Space complexity in both the cases

Github Task:
Using C++ STL, write the code for the above approaches. Make two functions (one each of for bruteforce and optimal). Save the code in P019.cpp [C++ STL] and push to your pvt Github repo.

Problem Statement: Wildcard Matching
We are given two strings 'S1' and 'S2'. String S1 can have the following two special characters:

- '?' can be matched to a single character of S2.
- '*' can be matched to any sequence of characters of S2. (sequence can be of length zero or more).

We need to check whether strings S1 and S2 match or not.