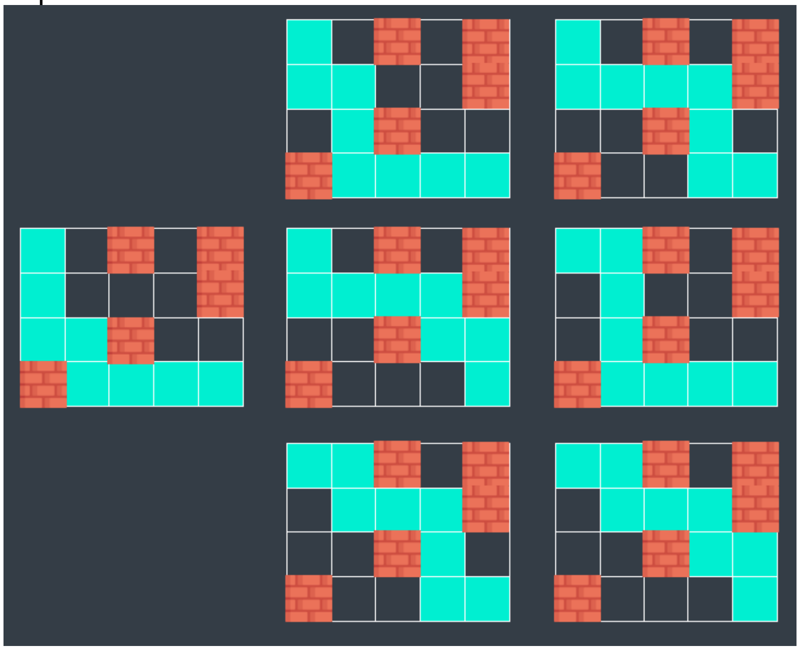
Paths In Matrix (Problem)
Given A Matrix Where A Cell Has The Value Of 1 If It's A Wall And 0 If Not, Find The Number Of Ways To Go From The Top-Left Cell To The Bottom-Right Cell, Knowing That It's Not Possible To Pass From A Wall And We Can Only Go To The Right Or To The Bottom.
Example:
Input:
```
Matrix = [
   [0, 0, 1, 0, 1],
   [0, 0, 0, 0, 1],
   [0, 0, 1, 0, 0],
   [1, 0, 0, 0, 0]
]
```
Output: 7
Explanation:


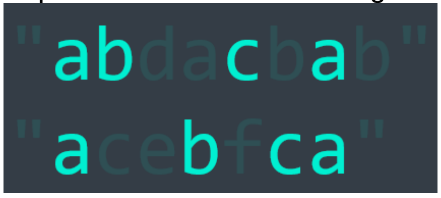
Constraints:
1. Matrix[I][J] Can Either Be 0 Or 1
2. Len(Matrix) >= 1
3. Len(Matrix[I]) >= 1

House Robber (Problem)
Given An Array Of Integers **Arr** Where Arr[I] Represents The Amount Of Money In The House I, You Are Asked To Find The Maximum Amount Of Money That A Robber Can Steal Knowing That He Can't Steal Two Adjacent Houses Because The Security Systems Would Automatically Call The Police.

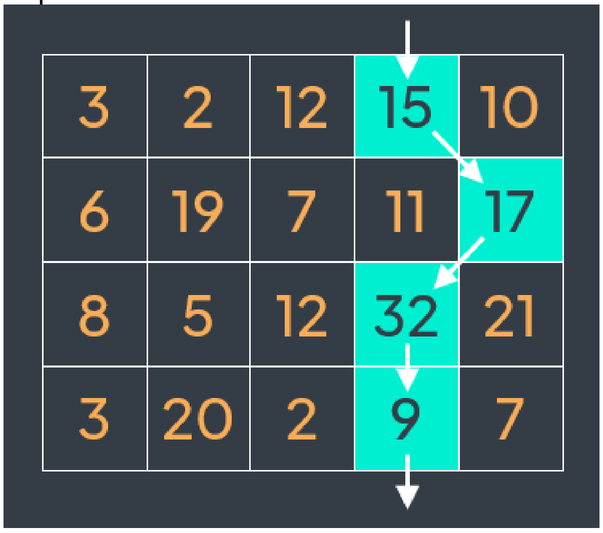
Example:
Input: Arr = [2, 10, 3, 6, 8, 1, 7]
Output: 25
Explanation: The Greatest Amount Of Money That A Robber Can Get Is 25, By The Stealing The House 1, 4, And 6 (Arr[1]+Arr[4]+Arr[6] = 10+8+7 = 25)

Constraints:
1. Len(Arr) >= 1
2. Arr[I] >= 0

Longest Common Subsequence (Problem)
Given Two Strings S1 And S2, Find The Length Of Their Longest Common Subsequence. A Subsequence Of A String S Is A Subset Of Its Characters That Are Not Necessarily Adjacent But Have To Be In The Right Order.

Example:
Input:
S1 = "Abdacbab"
S2 = "Acebfca"
Output: 4
Explanation: A Possible Longest Common Subsequence Of S1 And S2 Is "Abca"


Gold Mine (Problem)
Given A Mine Of N Rows And M Columns Where Mine[I][J] Represents The Amount Of Gold That Is Present There, We Want To Enter From The Top Of The Mine And Take As Much Gold As Possible When Exiting From The Bottom, Knowing That We Can Only Move To The Bottom, To The Bottom-Left, Or To The Bottom-Right. We Can Exit From Anywhere From The Last Row.

Example:
Input:
Mine = [
  [3, 2, 12, 15, 10],
  [6, 19, 7, 11, 17],
  [8, 5, 12, 32, 21],
  [3, 20, 2, 9, 7]
]
Output: 73
Explanation: 15+17+32+9 = 73


Constraints:
1. Len(Matrix) >= 1
2. Len(Matrix[I]) >= 1
3. Matrix[I][J] >= 0

Edit Distance (Problem)
Given Two Strings Word1 And Word2, Calculate Their Edit Distance.
The Edit Distance In This Problem Is Defined As The Minimum Number Of Insertions, Deletions, And Substitutions Of Characters To Go From Word1 To Word2.

Example:
Input:
Word1 = "Inside"
Word2 = "Index"
Output: 3
Explanation: To Go From "Inside" To "Index", We Can Delete The Character 'S', Delete The Second Character 'I', And Insert A Character 'X' At The End, In Total We Need 3 Operations
"Inside" -> "Inide" -> "Inde" -> "Index"

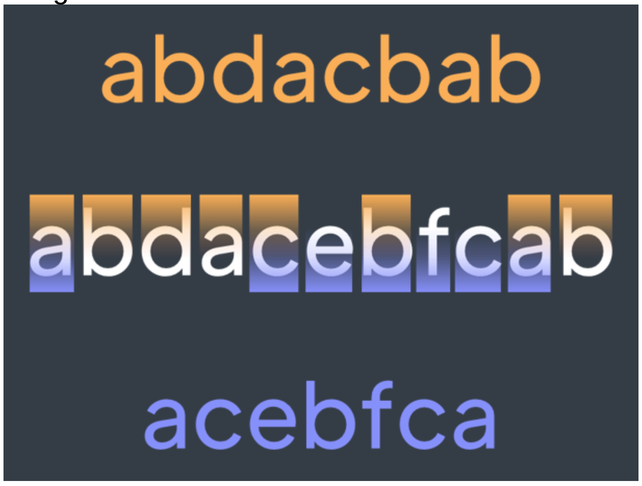
Ways To Climb (Problem)
Given A Stairs With N Steps And A List Of Possible Ways To Jump (For Example If Jumps[I] = 4 Then It's Possible To Jump By 4 Steps), Find The Total Number Of Ways To Reach The Nth Step Starting From The Floor.

Example:
Input:
N = 10
Jumps = [2, 4, 5, 8]
Output: 11
Explanation: We Have 11 Possible Ways To Reach The Step 10 Starting From The Floor Are:
2 2 2 2 2 (Jump By 2 Steps, Then 2 Steps, Then 2 Steps, Then 2 Steps, Then 2 Steps)
2 2 2 4 (Jump By 2 Steps, Then 2 Steps, Then 2 Steps, Then 4 Steps)
2 2 4 2 (Jump By 2 Steps, Then 2 Steps, Then 4 Steps, Then 2 Steps)
2 4 2 2 (Jump By 2 Steps, Then 4 Steps, Then 2 Steps, Then 2 Steps)
4 2 2 2 (Jump By 4 Steps, Then 2 Steps, Then 2 Steps, Then 2 Steps)
2 4 4 (Jump By 2 Steps, Then 4 Steps, Then 4 Steps)
4 2 4 (Jump By 4 Steps, Then 2 Steps, Then 4 Steps)
4 4 2 (Jump By 4 Steps, Then 4 Steps, Then 2 Steps)
5 5 (Jump By 5 Steps, Then 5 Steps)
2 8 (Jump By 2 Steps, Then 8 Steps)
8 2 (Jump By 8 Steps, Then 2 Steps)

Constraints:
1. N >= 1
2. Len(Jumps) >= 1
3. Jumps[I] >= 1

Shortest Common Super sequence (Problem)
Given Two Strings S1 And S2, Find The Length Of Their Shortest Common Super sequence, The Shortest String That Has Both S1 And S2 Subsequence's.
A String **A** Is A Super sequence Of A String **B** If **B** Is A Subsequence Of **A.**

Example:
Input:
S1 = "Abdacebfcab"
S2 = "Acebfca"
Output: 11
Explanation: The Shortest Common Super sequence Of S1 And S2 Is "Abdacebfcab", Its Length Is 11



Coin Change (Problem)
Given An Integer That Represents An Amount Of Money And A List Of Possible Coins, Find The Minimum Number Of Coins To Make That Amount.
Return -1 If It's Possible To Make The Amount With The Given Coins.

Example:
Input:
Amount = 15
Possible_Coins = [2, 3, 7]
Output: 4
Explanation: We Can Make The Amount 15 By Taking 4 Coins Only, One Coin Of Value 2, Two Coins Of Value 3, And One Coin Of Value 7
2+3+3+7 = 15

Constraints:
1. Amount >= 0
2. Len(Possible_Coins) >= 1
3. Possible_Coins[I] >= 1

0-1 Knapsack (Problem)
Given The Value And The Weight Of Each One Of N Items, We Want To Maximize The Value Of Items We Take In Our Knapsack Without Exceeding Its Capacity K. You Are Asked To Find The Maximum Total Value That We Can Get Without Exceeding A Total Weight Of K.
Note That Each Item Can Be Took At Most Once.

Example:
Input:
Values = [20, 30, 15, 25, 10]
Weights = [6, 13, 5, 10, 3]
K = 20
Output: 55
Explanation: The Maximum Total Value That We Can Get Is 55, By Taking Items 0, 3, And 4, Their Total Weight Doesn't Exceed K
Values[0]+Values[3]+Values[4] = 20+25+10 = 55
Weights[0]+Weights[3]+Weights[4] = 6+10+3 = 19 <= 20

Constraints:
1. Len(Values) == Len(Weights)
2. Len(Values) >= 1
3. K >= 0
4. Values[I] >= 0
5. Weights[I] > 0

Subset Sum (Problem)
Given An Array Arr Of Strictly Positive Integers, And An Integer K, Create A Function That Returns The Number Of Subsets Of Arr That Sum Up To K.

Example 1:
Input:
Arr = [1, 2, 3, 1]
K = 4
Output: 3
Explanation: Subsets That Sum Up To K Are [1, 2, 1], [1, 3], And [3, 1]

Example 2:
Input:
Arr = [1, 2, 3, 1, 4],
K = 6
Output: 4
Explanation: Subsets That Sum Up To K Are [1, 2, 3], [1, 1, 4], [2, 3, 1], And [2, 4]

Example 3:
Input:
Arr = [2, 4, 2, 6, 8]
K = 7
Output: 0
Explanation: There Are No Subsets That Sum Up To K

Constraints:
1. K >= 1
2. Arr[I] >= 1

Longest Increasing Subsequence (Problem)
Given An Array Of Integers Arr, Find The Length Of Its Longest Increasing Subsequence, Its Longest Subsequence Where Each Element Is Strictly Greater Than The Previous One.

Example 1:
Input:
Arr = [7, 5, 2, 4, 7, 2, 3, 6, 4, 5, 12, 1, 7]
Output: 5
Explanation: A Possible Longest Increasing Subsequence Is [2, 3, 4, 5, 7], Its Length Is 5

Example 2:
Input:
Arr = [8, 5, 5, 3]
Output: 1
Explanation: The Longest Increasing Subsequence That We Can Make Are Those That Contain One Element Only, Like [8]

Constraints:
Len(Arr) >= 1

Ways To Decode (Problem)
A String Made Of Letters Can Be Encoded By Replacing Each Letter By Its Position In The Alphabet (E.G.: Eld -> 5124), But When Decoding, A Same Encoded String Can Have Multiple Ways To Be Decoded (E.G.: 5124 Can Be Decoded As 5 1 2 4 (Eabd), Or 5 12 4 (Eld), Or 5 1 24 (Eax)).
Given An Encoded String S Made Of Numbers, Find The Number Of Possible Ways To Decode It By Following This Decoding Map:
1 -> A,   2 -> B,   3 -> C,   4 -> D,   5 -> E,   6 -> F,   7 -> G,   8 -> H,   9 -> I,   10 -> J,   11 -> K,   12 -> L,   13 -> M,   14 -> N,   15 -> O,   16 -> P,   17 -> Q,   18 -> R,   19 -> S,   20 -> T,   21 -> U,   22 -> V,   23 -> W,   24 -> X,   25 -> Y,   26 -> Z

Example:
Input:
S = "512810120129"
Output: 4
Explanation: There Are 4 Possible Ways To Decode S:
```
5   1   2   8   10   1   20   1   2   9
5   1   2   8   10   1   20   12   9
5   12   8   10   1   20   1   2   9
5   12   8   10   1   20   12   9
```

Constraints:
1. Len(S) >= 1
2. '0' <= S[I] <= '9'

Partition (Problem)
Given An Array Of Strictly Positive Integers Arr, Check If We Can Split It Into Two Subsets That Have The Same Sum Of Elements.

Example 1:
Input:
Arr = [4, 5, 3, 2, 5, 1]
Output: True
Explanation: We Can Split Arr Into [4, 3, 2, 1] And [5, 5], And They Have The Same Sum 4+3+2+1 = 5+5 = 10

Example 2:
Input:
Arr = [5, 6, 2, 3, 8, 1]
Output: False
Explanation: We Can't Split Arr Into Two Subsets That Have The Same Sum

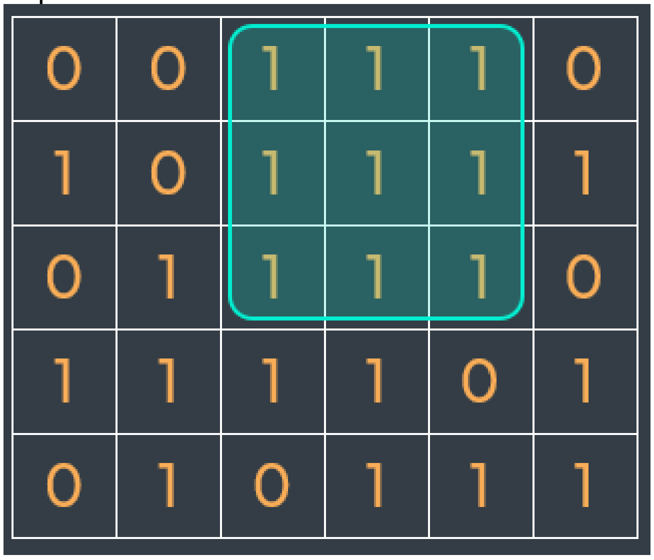
Constraints:
1. Len(Arr) >= 1
2. Arr[I] >= 1

Rod Cutting (Problem)
We Want To Sell A Rod Of Size N, And To Maximize The Profit, We Can Cut It Into Multiple Pieces. The Price Of A Piece Depends On Its Length, To Know It, We Are Given An Array Prices Where Prices[I] Represents The Price Of A Piece Of Length I.
You Are Asked To Find The Maximum Price That We Can Get By Cutting It Into Pieces. Note That You Are Allowed To Sell It As A Single Piece Of Length N If It's The Most Profitable Choice.

Example:
Input:
N = 8
Prices = [0, 1, 3, 5, 6, 7, 9, 10, 11]
Output: 13
Explanation: The Most Profitable Way Of Cutting A Rod Of Length 8 With The Given Prices Is By Cutting It Into A Piece Of Length 2 And Two Pieces Of Length 3
Prices[2]+Prices[2]+Prices[3] = 3+5+5 = 13

Constraints:
1. N > 0
2. Len(Prices) == N+1
3. Prices[0] == 0
4. Prices[I] >= 0

Square Matrix Of Ones (Problem)
Given A Matrix Of Ones And Zeros, Find The Area Of The Greatest Square Submatrix Full Of Ones.
A Square Matrix Is A Matrix Whose The Number Of Rows Is Equal To The Number Of Columns.

Example:
Input:
Matrix = [
  [0, 0, 1, 1, 1, 0],
  [1, 0, 1, 1, 1, 1],
  [0, 1, 1, 1, 1, 0],
  [1, 1, 1, 1, 0, 1],
  [0, 1, 0, 1, 1, 1]
]
Output: 9
Explanation:


Constraints:
1. Len(Matrix) >= 1
2. Len(Matrix[I]) >= 1
3. Matrix[I][J] Is Equal To 0 Or 1

Minimum Cost For Tickets (Problem)
We Are About To Travel For A Period Of N Days, During Which We Will Use The Train In Some Days That You Can Find In A Given Set Train_Days.
And To Use The Train, We Have To Possess Either A 1-Day Pass, A 7-Days Pass, Or A 30-Days Pass, Each One Of Them Has A Price That You Can Find In A Given Array Prices, Where Prices[0] Represents The Price Of A 1-Day Pass, Prices[1] Represents The Price Of A 7-Days Pass, And Prices[2] Represents The Price Of A 30-Days Pass.
You Are Asked To Find The Minimum Amount Of Money That We Need To Use The Train During All The Train Days.

Example:
Input:
Train_Days = [1, 3, 8, 9, 22, 23, 28, 31]
Costs = [4, 10, 25]
N = 32
Output: 28
Explanation: The Cheapest Way Is By Buying

- A 1-Day Pass On Day 1, We Use It On Day 1
- A 7-Days Pass On Day 3, We Use It During Days 3, 8, And 9
- A 7-Days Pass On Day 22, We Use It During Days 22, 23, And 28
- A 1-Day Pass On Day 31, We Use It On Day 31

Costs[0]+Costs[1]+Costs[1]+Costs[0] = 4+10+10+4 = 28

Constraints:

1. N > 0
2. Train Days Are Between 0 Inclusive And N Exclusive
3. Len(Costs) == 3