You are given an array **arr[]** of non-negative numbers. Each number tells you the **maximum number of steps** you can jump forward from that position.

For example:

- If **arr[i] = 3**, you can jump to index **i + 1**, **i + 2**, or **i + 3** from position **i**.
- If **arr[i] = 0**, you **cannot jump forward** from that position.

Your task is to find the **minimum number of jumps** needed to move from the **first** position in the array to the **last** position.

Note: Return **-1** if you can't reach the end of the array.

Examples :

Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]

Output: 3

Explanation: First jump from 1st element to 2nd element with value 3. From here we jump to 5th element with value 9, and from here we will jump to the last.

Input: arr = [1, 4, 3, 2, 6, 7]
Output: 2

Explanation: First we jump from the 1st to 2nd element and then jump to the last element.

Input: arr = [0, 10, 20]
Output: -1

Explanation: We cannot go anywhere from the 1st element.

Constraints:
$2 \leq arr.size() \leq 10^5$
$0 \leq arr[i] \leq 10^5$

Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]

Output: 3

Explanation: First jump from 1st element to 2nd element with value 3. From here we jump to 5th element with value 9, and from here we will jump to the last.

Input: arr = [1, 4, 3, 2, 6, 7]
Output: 2

Explanation: First we jump from the 1st to 2nd element and then jump to the last element.

Input: arr = [0, 10, 20]
Output: -1

Explanation: We cannot go anywhere from the 1st element.

Input [9 10 1 2 3 4 8 0 0 0 0 0 0 0 1]

Output 2

https://www.geeksforgeeks.org/problems/minimum-number-of-jumps-1587115620/1?page=1&category=Greedy&sortBy=submissions

Stock buy and sell

Difficulty: **Medium**Accuracy: **29.18%**Submissions: **305K+**Points: **4**Average Time: **20m**

Given an array **arr[]** denoting the cost of stock on each day, the task is to find the maximum total profit if we can buy and sell the stocks any number of times.

Note: We can only sell a stock which we have bought earlier and we cannot hold multiple stocks on any day.

Examples:

Input: arr[] = [100, 180, 260, 310, 40, 535, 695]

Output: 865

Explanation: Buy the stock on day 0 and sell it on day 3 => 310 – 100 = 210 Buy the stock on day 4 and sell it on day 6 => 695 – 40 = 655 Maximum Profit = 210 + 655 = 865

Input: arr[] = [4, 2, 2, 2, 4]

Output: 2

Explanation: Buy the stock on day 3 and sell it on day 4 => 4 – 2 = 2

Input: arr[] = [4, 2]

Output: 0

Explanation: Don't Buy the stock.

Constraints:
$2 \leq arr.size() \leq 10^6$
$0 \leq arr[i] \leq 10^6$

https://www.geeksforgeeks.org/problems/stock-buy-and-sell-1587115621/1?page=1&category=Greedy&sortBy=submissions

Given an **infinite supply** of each denomination of Indian currency { **1, 2, 5, 10** } and a target value **n**. Find the **minimum** number of coins and/or notes needed to make the change for Rs **n**.

Examples:

Input: n = 39

Output: 6

Explaination: 39 can be formed using 3 coins of 10 rupees, 1 coin of 5 rupees and 2 coins of 2 rupees so minimum coins required are 6.

Input: n = 121

Output: 13

Explaination: 121 can be formed using 12 coins of 10 rupees and 1 coin of 1 rupees.

Constraints:
$1 \leq n \leq 10^6$

Solutions:

Min Coins:

```
function findMin(n) {


  // Traverse through all denomination

  const denomination = [1, 2, 5, 10];

  let count =0;

  for (let i = denomination.length - 1; i >= 0; i--) {


    // Find denominations

      count+= Math.floor(n/denomination[i]);

      n=n%denomination[i];

}
```

```
    return count;

}
// Driver Code
const n = 39;
console.log(findMin(n));
```

Minimum Jumps

```
// JavaScript program to find the minimum number
// of jumps to reach the end of the array

function minJumpsRecur(i, arr) {

    // Return 0 when last element is reached.
    if (i >= arr.length - 1)
        return 0;

    // Traverse through all the points
    // reachable from arr[i].
    // Recursively, get the minimum number
    // of jumps needed to reach array end from
    // these points.
    let ans = Infinity;
    for (let j = i + 1; j <= i + arr[i]; j++) {
        let val = minJumpsRecur(j, arr);
```

```
        if (val != Infinity)
            ans = Math.min(ans, 1 + val);
    }

    return ans;
}

function minJumps(arr) {
    let ans = minJumpsRecur(0, arr);

    // If end cannot be reached.
    if (ans == Infinity)
        return -1;

    return ans;
}

let arr = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9];
console.log(minJumps(arr));
```

Stocks

```
function maxProfitRec(price, start, end) {
    let res = 0;

    // Try every possible pair of buy (i) and sell (j)
    for (let i = start; i < end; i++) {
```

```
    for (let j = i + 1; j <= end; j++) {
      // Valid transaction if selling price > buying price
      if (price[j] > price[i]) {
        let curr = (price[j] - price[i]) +
            maxProfitRec(price, start, i - 1) +
            maxProfitRec(price, j + 1, end);
        res = Math.max(res, curr);
      }
    }
  }
  return res;
}

function maxProfit(prices) {
  return maxProfitRec(prices, 0, prices.length - 1);
}
// Driver Code
let prices = [100, 180, 260, 310, 40, 535, 695];
console.log(maxProfit(prices));
```