Here is the code of the given question:

function helper(arr, i, ans) {

if (i == arr.length - 1) {

return 0;

}

if (i >= arr.length || arr[i] == 0) {

return Number.MAX\_VALUE;

}

if (ans[i] != undefined) {

return ans[i];

}

let min\_jumps = Number.MAX\_VALUE;

for (let j = i + 1; j <= i + arr[i]; j++) {

let cost = helper(arr, j, ans);

if (cost != Number.MAX\_VALUE) {

min\_jumps = Math.min(min\_jumps, cost + 1);

}

}

return (ans[i] = min\_jumps);

}

function solution(arr) {

if (arr.length == 0)

return 0;

var ans = new Array(arr.length);

let r = helper(arr, 0, ans);

if (r == Number.MAX\_VALUE)

return -1;

return r;

}

let arr = [1, 6, 3, 4, 5, 0, 0, 0, 6];

console.log(solution(arr));

The explanation of the code:

The main function, solution(arr), is a wrapper function that initializes an array to store the minimum number of jumps required to reach a given index in the input array arr, and then calls the helper function helper(arr, i, ans) to find the solution.

The helper function helper(arr, i, ans) has three main parts:

It checks if the current index i is the last element of the array, in which case it returns 0, as no further jumps are required.

It checks if the current index i is out of bounds of the array or the value at the current index is 0, in which case it returns Number.MAX\_VALUE which is a very large number, signifying that it is impossible to reach the end from this position.

If the minimum number of jumps required to reach the current index has already been calculated and stored in ans[i], the function simply returns the stored value.

Otherwise, the function enters a loop where it will iterate through all the possible jumps starting from the current index i, and for each jump, it will recursively call the helper function to find the minimum number of jumps required to reach the end of the array, starting from the index after the jump. The function will then take the minimum of these values and add 1 to it, since the current jump is also counted as a jump. This minimum value will be stored in ans[i], which will be used next time if that index is encountered

The final result is returned to the wrapper function, which checks if the result is equal to Number.MAX\_VALUE, in which case it returns -1 to signify that it is impossible to reach the end of the array, or it returns the result as the minimum number of jumps required to reach the end of the array.