**DESAFIO LÓGICA**

**As questões abaixo podem ser solucionadas na linguagem de sua preferência. Tente**

**resolver o maior numero de questões que conseguir.**

**01 - QUESTÃO**

Dado um array de números inteiros, retorne os índices dos dois números de forma que eles se

somem a um alvo específico.

Você pode assumir que cada entrada teria exatamente uma solução, e você **não** pode usar o

mesmo elemento duas vezes.

**Exemplo:**

Dado nums = [2, 7, 11, 15], alvo = 9,

Como nums[**0**] + nums[**1**] = 2 + 7 = 9,

return [**0**, **1**].

function QuestaoUm(nums, alvo){

    return new Promise( resolve => {

        nums.forEach( (A, indexA) => {

            nums.forEach( (B, indexB) => {

                if( A + B == alvo) resolve([indexA, indexB])

            })

        })

        resolve("Sem correspondencia")

    })

}

(async () => {

    console.log( "Resposta questão Um:", await QuestaoUm([1,1,2,3,5,8,13], 21) )

// $ node teste.js

// Resposta questão Um: [ 5, 6 ]

})();

**02 – QUESTÃO**

Um bracket é considerado qualquer um dos seguintes caracteres: (, ), {, }, [ ou ].

Dois brackets são considerados um par combinado se o bracket de abertura (isto é, (, [ou {) ocorre à esquerda de um

bracket de fechamento (ou seja,),] ou} do mesmo tipo exato. Existem três tipos de pares de brackets : [], {} e ().

Um par de brackets correspondente não é balanceado se o de abertura e o de fechamento não corresponderem entre

si. Por exemplo, {[(])} não é balanceado porque o conteúdo entre {e} não é balanceado. O primeiro bracket inclui o

de abertura, (, e o segundo inclui um bracket de fechamento desbalanceado,].

Dado sequencias de caracteres, determine se cada sequência de brackets é balanceada. Se uma string estiver

balanceada, retorne SIM. Caso contrário, retorne NAO.

**Exemplo:**

{[()]} SIM

{[(])} NAO

{{[[(())]]}} SIM

function QuestaoDois( brackets ){

    brackets = brackets.split("")

    const relacao = { "{":"}", "(":")", "[":"]" }

    for( let p = 0, f = brackets.length-1; p < brackets.length/2; p++, f--){

        if( relacao[brackets[p]] != brackets[f] ) return "NÃO"

    }

    return "SIM"

}

(async () => {

    console.log( "Resposta questão Dois:",

        QuestaoDois( "{[()]}" ),

        QuestaoDois( "{[(])}" ),

        QuestaoDois( "{{[[(())]]}}" )

    )

    // $ node teste.js

    // Resposta questão Dois: SIM NÃO SIM

})();

**03 – QUESTÃO**

Digamos que você tenha um array para o qual o elemento i é o preço de uma determinada ação

no dia i.

Se você tivesse permissão para concluir no máximo uma transação (ou seja, comprar uma e

vender uma ação), crie um algoritmo para encontrar o lucro máximo.

**Note que você não pode vender uma ação antes de comprar**.

**Exemplo:**

**Input:** [7,1,5,3,6,4]

**Output:** 5 (Comprou no dia 2 (preço igual a 1) e vendeu no dia 5 (preço igual

a 6), lucro foi de 6 – 1 = 5

**Input:** [7,6,4,3,1]

**Output:** 0 (Nesse caso nenhuma transação deve ser feita, lucro máximo igual a

0)

function QuestaoTres( historico ){

    let Transacao = {

        compra: {

            valor: 1^40,

            dia: null

        },

        venda: {

            valor: 0,

            dia: null

        }

    }

    historico.forEach( ( valor, dia ) => {

        if( valor < Transacao.compra.valor ) Transacao.compra = { valor, dia: dia+1 }

    })

    for( let dia = Transacao.compra.dia; dia < historico.length; dia++ ){

        const valor = historico[dia]

        if( valor > Transacao.venda.valor ) Transacao.venda = { valor, dia: dia+1 }

    }

    if( !Transacao.venda.dia || !Transacao.venda.valor )

        return '0 (Nesse caso nenhuma transação deve ser feita, lucro máximo igual a 0 )'

    else {

        const lucroMax = Transacao.venda.valor - Transacao.compra.valor

        return `${lucroMax} ` +

            `(Comprou no dia ${Transacao.compra.dia} (preço igual a ${Transacao.compra.valor}) e ` +

            `vendeu no dia ${Transacao.venda.dia} (preço igual a ${Transacao.venda.valor}), ` +

            `lucro foi de ${Transacao.venda.valor} – ${Transacao.compra.valor} = ${lucroMax}`

    }

}

console.log( "Resposta questão tres:",

        QuestaoTres( [7,1,5,3,6,4] ),

        QuestaoTres( [7,6,4,3,1] )

    )

Resposta questão tres:

5 (Comprou no dia 2 (preço igual a 1) e vendeu no dia 5 (preço igual a 6), lucro foi de 6 – 1 = 5

0 (Nesse caso nenhuma transação deve ser feita, lucro máximo igual a 0 )

**04 – QUESTÃO**

Dados n inteiros não negativos representando um mapa de elevação onde a largura de cada barra é 1, calcule quanta água é capaz de reter após a chuva.



**Exemplo:**

**Input:** [0,1,0,2,1,0,1,3,2,1,2,1]

**Output:** 6

function QuestaoQuatro( elevacao ) {

    const PegarAltura = () => {

        let max = 0;

        for(let index = 0; index < elevacao.length; index++){

            if( max < elevacao[index] ) max = elevacao[index];

        }

        return max

    };

    const MontarGrade = (altura) => {

        let result = []

        for( let y = 1; y <= altura; y++){

            result.push(

                elevacao.reduce( (acc, item) => {

                    acc.push( item >= y ? 1 : 0 )

                    return acc

                }, [])

            )

        }

        return result.reverse()

    };

    const ExisteOcorrenciaDeParedeADireita = ( item, indiceInicial ) => {

        if( item.length == indiceInicial ) return false

        for(let linhaIndex = indiceInicial; linhaIndex < item.length; linhaIndex++){

            if( item[linhaIndex] == 1 ) return true

        }

        return false

    };

    const ExisteOcorrenciaDeParedeAEsquerda = ( item, indiceInicial ) => {

        if( !indiceInicial ) return false

        for(let linhaIndex = indiceInicial; linhaIndex >= 0; linhaIndex--){

            if( item[linhaIndex] == 1 ) return true

        }

        return false

    };

    const SimularReservatorio = ( grade ) => {

        return grade.reduce( (acc, item, itemGrade) => {

            acc.push(

                item.reduce( (itemAcc, ponto, indexPonto) => {

                    if( ponto == 0 ){

                        const anterior = ExisteOcorrenciaDeParedeAEsquerda( item, indexPonto )

                        const proximo = ExisteOcorrenciaDeParedeADireita( item, indexPonto )

                        itemAcc.push( anterior && proximo ? 2 : ponto )

                    }else itemAcc.push(ponto)

                    return itemAcc

                }, [])

            )

            return acc

        }, [])

    };

    const ContarReservatorios = ( grade ) => {

        return grade.reduce( ( acc, item ) => {

            return acc + item.filter( i => i == 2).length

        }, 0)

    };

    const altura = PegarAltura()

    let grade = MontarGrade( altura )

    let gradeSimulada = SimularReservatorio( grade )

    console.log(gradeSimulada)

    return ContarReservatorios(gradeSimulada)

}

    console.log( "Resposta questão quatro:",

       QuestaoQuatro( [0,1,0,2,1,0,1,3,2,1,2,1] ),

       QuestaoQuatro( [0,4,0,2,5,0,3,3,2,4,2,1,5,0,0,5] )

    )

[

[ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0 ],

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

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

]

[

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

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

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

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

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

]

Resposta questão quatro: 6 36