Trabalho segundo semestre, parte 2: “O problema de Josephus”

Código para resolução utilizando lista encadeada circular:

function CircularLinkedList(){

    let Node = function(element){

        this.element = element;

        this.next = null;

    }

    let length = 0;

    let head = new Node();

    head.next = head;

    this.append = function(element){

        let node = new Node(element)

        current = head

        while(current.next != head){

            current = current.next

        }

        current.next = node

        node.next = head

        length++

    }

    this.getAllElements = function(){

        let atual = head

        listaDeElementos = []

        do{

            listaDeElementos.push(atual.element)

            listaDeElementos.push(atual.next != head ? " --> " : "")

            atual = atual.next

        }while(atual != head)

        listaDeElementos = listaDeElementos.join("")

        return listaDeElementos

    }

    this.getHead = function(){

        return head

    }

    this.createList = function(variavel, n){

        let i=1;

        while(i<=n){

            variavel.append(i)

            i++

        }

        return variavel

    }

    this.josephusResolution = function(m){

        let current = head

        let andar = m

        while(length>1){

            while(andar>1){

                andar--

                if(current.next == head){

                    current = current.next.next

                }

                else{

                    current = current.next

                }

            }

            if(current.next == head){

                current.next.next = current.next.next.next

            }

            else{

                current.next = current.next.next

            }

            andar = m

            length --

        }

    }

    this.getSize = function(){

        return length

    }

}

let variavel = new CircularLinkedList();

let n = 41, m =7;

variavel.createList(variavel, n)

console.log(variavel.getAllElements())

variavel.josephusResolution(m)

console.log("Ultimo elemento que sobrou: ")

console.log(variavel.getAllElements())

Código para resolução utilizando vetor:

function createVetor(vetor, n){

    for(let i=1; i<=n; i++){

        vetor.push(i)

    }

}

function josephusResolution(vetor, m){

    let tamanho = vetor.length;

    let andar = m -1;

    let posicaoAtual = 0;

    let saldo;

    while(tamanho>1){

        if(posicaoAtual + andar < tamanho){

            posicaoAtual = posicaoAtual + andar

        }

        else{

            if(posicaoAtual - tamanho >= 0){

                saldo = posicaoAtual - tamanho

                posicaoAtual = saldo

            }

            for(let i = 0; i < andar; i++){

                if(posicaoAtual + 1 < tamanho){

                    posicaoAtual++

                }

                else{

                    posicaoAtual = 0

                }

            }

        }

        vetor.splice(posicaoAtual, 1)

        tamanho--

    }

}

let vetor = []

let n = 24, m = 7;

createVetor(vetor, n)

josephusResolution(vetor, m)

console.log("Número que sobrou: " + vetor)

**Vantagens e Desvantagens:**

**Lista Encadeada Circular:**

Vantagens:

Tempo de processamento constante para adicionar ou remover valores na posição atual;

Elementos armazenados de forma adjacente, gerando menor uso de memória;

Entendimento do processo de remoção mais simples, uma vez que depende apenas de seguir o ponteiro para a próxima posição;

Desvantagens:

Tempo de busca lento, uma vez que se deve passar por todos os elementos anteriores a ele;

Adicionar e remover elementos em posições especificas possui uma logica mais complexa, comparado ao vetor;

Eficiência baixa para acesso a elementos aleatórios;

**Vetor:**

Vantagens:

Acesso direto a qualquer elemento em tempo constante;

Adicionar e remover elementos em qualquer posição possui um tempo de execução também constante;

Implementação mais simples, pois não usa conceitos de loops circulares ou ponteiros;

Uso de memória eficiente e previsível;

Desvantagens:

Exige atualização manual do índice do elemento atual, gerando uma simulação menos intuitiva, na qual o índice e tamanho do vetor estão em constante mudança;

Uma vez que vetores possuem tamanho pré-definido, pode haver um desperdício de memoria ao utilizar um vetor muito grande para problemas pequenos;

**Conclusão:**

A lista encadeada circular é recomendada caso o intuito seja economizar memória e priorizar a simplicidade de entendimento das simulações da aplicação.

Enquanto isso, para priorizar a velocidade de acesso a elementos e utilização de operações dinâmicas para inserir, remover e modificar as pessoas no círculo, o qual pode ser feita utilizando métodos build-in de vetores e apenas os índices armazenados nas operações.