SCC0220 - Laboratório Introdução à Ciência da Computação II

Relatório de execução do trabalho prático 8

Alunos	**NUSP**
Alec Campos Aoki	15436800
Juan Henrique Passos	15464826

Trabalho 8 – Buscando problema

Buscando problema

Comentário

O trabalho de hoje consiste em dado uma quantidade de cartas e um valor de uma carta buscada, encontrar a primeira carta do vetor da esquerda para direita que será fornecido na entrada. Para isso, foi necessário ordenar o vetor, e aplicar a busca binária, utilizando uma estrutura “carta” que guarda o index do vetor original e o valor da carta. O algoritmo usado para a ordenação foi o mergesort, escolhido por conta da complexidade O(n log n) e sua ordenação estável.

A busca binária possui complexidade O(log n), tendo em vista que em cada busca, o vetor é dividido pela metade, até possuir tamanho 1, tendo em vista que buscamos o primeiro elemento. Assim, ao encontrar o elemento do meio, verificamos se é o que buscamos, caso seja, ele é uma possível resposta, assim atualizamos o vetor para incluir esse elementos e os anteriores a ele, caso contrário, corta-se a outra metade do vetor, isolando a parte esquerda sem incluir o número visualizado tal processo se repete até os ponteiros “L” (esquerdo) e “R” (direito) se encontrarem.

Código

```c
#include<stdio.h>
#include<stdlib.h>

typedef struct carta_{
    int valor;
    int index;
}carta;

void dividir(carta *vet, int l, int r);
void conquistar(carta *vet, int l, int meio, int r);

int main(){
    int quant_cartas, carta_buscada;
```

```
    scanf("%d %d", &quant_cartas, &carta_buscada);

    carta *baralhos = (carta*) malloc(quant_cartas*sizeof(carta));

    for(int i = 0; i <  quant_cartas; i++){
        scanf("%d", &baralhos[i].valor);
        baralhos[i].index = i;
    }

    dividir(baralhos, 0, quant_cartas-1);

    int l = 0, r = quant_cartas-1;

    while(l < r){
        int mid = (l+r)/2;
        if(carta_buscada <= baralhos[mid].valor)
            r = mid;
        else
            l = mid+1;
    }

    printf("%d", baralhos[l].index + 1);

    return 0;
}

void dividir(carta *vet, int l, int r){
    if(l<r){
        int meio = (l+r)/2;

        dividir(vet, l, meio);
        dividir(vet, meio+1, r);

        conquistar(vet, l, meio, r);

    }
}
```

```
void conquistar(carta *vet, int l, int meio, int r){
    int tam1 = meio-l+1;
    int tam2 = r-meio;

    carta L[tam1];
    carta R[tam2];

    for(int i = 0; i < tam1; i++){
        L[i] = vet[i+l];
    }

    for(int i = 0; i < tam2; i++){
        R[i] = vet[i+meio+1];
    }

    int posL = 0, posR = 0, posVet = l;

    while(posL < tam1 && posR < tam2){
        if(L[posL].valor <= R[posR].valor){
            vet[posVet] = L[posL];
            posL++;
        }
        else{
            vet[posVet] = R[posR];
            posR++;
        }
        posVet++;
    }

    while(posL < tam1){
        vet[posVet] = L[posL];
        posVet++;
        posL++;
    }
    while(posR < tam2){
        vet[posVet] = R[posR];
        posVet++;
```

```
        posR++;
    }
}
```

Saída

Caso 7 do run codes

```
rm -f bb
gcc -std=c99 -Wall bb.c -o bb
./bb < 10.in
131308
Tempo de execução: 0.155000 segundos
```

Gráfico