2º Trabalho

Curso: Engenharia da Computação
Disciplina: Estruturas de Dados
Prof. Jarbas Joaci de Mesquita Sá Junior
Universidade Federal do Ceará – UFC/Sobral

Entrega: 15/06/2023 via e-mail para jarbas_joaci@yahoo.com.br

Obs.:

- Não receberei o trabalho após a data mencionada.
- O trabalho é **OPCIONAL,** deverá ser feito **individualmente** e valerá, no máximo, **1,0** ponto.
- Preferencialmente fazer o trabalho usando a IDE Dev-C++.;
- Enviar **todos** os arquivos do projeto, exceto os executáveis (.exe). Organizar os arquivos nas pastas q1 e q2.
- O uso da diretiva #include sem um header file (.h) implicará nota zero no código. Por exemplo, **não** usar `#include "nomearquivo.c"`

1. Implemente a TAD "arvb.h" (Árvore Binária de Buscas) e acrescente as seguintes funções:

a) função que retorne a quantidade de folhas que possuem no campo `info` um número primo. Essa função deve obedecer ao protótipo:

```
int folhas_primos(ArvB* a);
```

b) função que retorne a quantidade de nós que possuem os dois filhos (campos `dir` e `esq` diferentes de `NULL`). Essa função deve obedecer ao protótipo:

```
int dois_filhos(ArvB* a);
```

c) função que retorne a quantidade de nós cujas subárvores esquerda e direita não são vazias e têm igual altura. Essa função deve obedecer ao protótipo:

```
int nos_igual_altura(ArvB* a);
```

d) função que compare se duas árvores binárias de busca são iguais. Essa função deve obedecer ao protótipo:

```
int iguais(ArvB* a, ArvB* b);
```

Obs: 1 – verdadeiro; 0 – falso.

A seguir, execute o seguinte programa.

```
#include <stdio.h>
#include <stdlib.h>
#include "arvb.h"
```

```
int main(void){

  ArvB* arv1 = arvb_cria_vazia();
  arv1=arvb_insere(arv1,9);
  arv1=arvb_insere(arv1,5);
  arv1=arvb_insere(arv1,21);
  arv1=arvb_insere(arv1,4);
  arv1=arvb_insere(arv1,77);
  arv1=arvb_insere(arv1,0);
  arv1=arvb_remove(arv1,4);
  printf(''Altura da árvore %d\n'',arv_altura(arv1));
  printf(''Qtd folhas primos %d\n'',folhas_primos(arv1));
  printf(''Qtd de nós dois filhos %d\n'',dois_filhos(arv1));
  printf(''Nós igual altura %d\n'',nos_igual_altura(arv1));

  ArvB* arv2 = arvb_cria_vazia();

  arv2=arvb_insere(arv2,5);
  arv2=arvb_insere(arv2,6);
  arv2=arvb_insere(arv2,11);

  ArvB* arv3 = arvb_cria_vazia();

  arv3=arvb_insere(arv2,5);
  arv3=arvb_insere(arv2,6);
  arv3=arvb_insere(arv2,11);

  arvb_imprime(arv1); //impressao em ordem simétrica
  arvb_imprime(arv2); //impressao em ordem simétrica

  int comp = iguais(arv1,arv2);
  printf(''Árvores iguais %d\n'',comp);

  comp = iguais(arv2,arv3);
  printf(''Árvores iguais %d\n'',comp);

  arvb_libera(arv1); arvb_libera(arv2); arvb_libera(arv3);

  system(''PAUSE'');

  return 0;

}
```

2. Implemente os algoritmos **BubbleSort**, **InsertionSort**, **QuickSort, MergeSort** e **HeapSort** e calcule o tempo médio de cada um para ordenar vetores de tamanho 10^2, 10^3, 10^4, 10^5 e 10^6 com valores aleatórios. Elabore um pequeno relatório (max. de **3 páginas**) comparando o desempenho dos métodos.

Obs: O tempo deve ser dado em milissegundos.