Algoritmos de Programação II - Trabalho 1

Prof. Lucas Reis September 26, 2021

Ordenando Estudantes

A Universidade Federal de Mato Grosso do Sul possui um banco de dados que armazena todos os estudantes regulares cursando a disciplina de Algoritmos e Programação II. Frequentemente, o professor precisa acessar essa lista de alunos de diferentes maneiras, incluindo ações como buscar um aluno e imprimir a lista em uma determinada ordem. Infelizmente, o sistema da universidade foi implementado em 1943 (dois anos antes da publicação do primeiro algoritmo de ordenação $O(n \log_2 n)$). Portanto, o sistema atual costuma ser extremamente lento.

Sabendo que você está cursando a disciplina e os conhecimentos de ordenação e busca eficientes estão bem sedimentados em sua mente, ficou a seu cargo reimplementar o backend deste sistema, agora utilizando algoritmos mais eficientes.

Os estudantes

Os dados de cada estudante são armazenados em um registro (struct) que contém os seguintes campos:

```
struct Estudante
{
    int RGA;
    char nome[];
    double media;
};
```

`RGA` é um campo que contêm o RGA de cada aluno. `nome` é o campo que armazena o nome completo de cada aluno, **que pode conter espaços**[1]. Considere que nenhum aluno possui nome com mais de 100 caracteres. `media` é um campo que armazena a média final de cada aluno na disciplina de Algoritmos e Programação II. Este valor está sempre entre 0.0 e 10.

O Sistema

O sistema que você deve estruturar deve ser capaz de ordenar os estudantes em ordem **crescente** baseado em cada um dos critérios: `RGA`, `media` e `nome` (para esse último, considere a ordem lexicográfica). Além disso, o sistema também deve ser capaz de realizar várias buscas por estudantes baseado em cada um desses fatores.

A sua função de busca deve ter eficiência $O(\log_2 n)$, e sua ordenação deve ter eficiência $O(n \log_2 n)$, considerando n como o tamanho do banco de dados dos estudantes.

[1] para ler uma string com espaços, utilize scanf("%[^\n]s", nome);

As funções

O seu trabalho **deve** implementar as seguintes funções:

Busca

```
int buscaRGA(Estudante v[], int n, int RGA);
int buscaNome(Estudante v[], int n, char nome[]);
int buscaMedia(Estudante v[], int n, double media);
```

Que realiza a busca pelo estudante que contém o nome, RGA o média designada, respectivamente. Essas funções devem devolver o índice do vetor onde o estudante se encontra, caso o estudante seja encontrado. Caso contrário, a função deve retornar -1.

Ordenação

```
void ordenaEstudantes(Estudante v[], int n, char op);
```

que ordena os estudantes baseado em uma opção informada em op, da seguinte maneira:

- `r` - ordenar os estudantes por RGA;
- `n` - ordenar os estudantes por nome (lexicograficamente);
- `m` - ordenar os estudantes por média.

O algoritmo implementado é de escolha do aluno, contanto que sua eficiência seja $O(n \log_2 n)$.

Você **não deve** implementar uma função diferente para cada uma das opções, e sim criar um algoritmo de ordenação que adapta a comparação de valor com base na opção informada em op.

Impressão

```
void imprimeEstudante(Estudante v[], int k);
void imprimeEstudantes(Estudante v[], int n);
```

A primeira função deve imprimir as informações de um único estudante e, localizado no índice k do vetor.

A segunda função deve imprimir o vetor de estudantes na ordem em que ele se encontra assim que o código é chamado, **não podendo** realizar nenhum tipo de ordenação dentro de sua chamada.

A impressão de cada estudante deve seguir o seguinte formato:

```
RGA: rga
Nome: nome
Media: media
------------
```

onde `rga`, `nome` e `media` são o RGA, nome e média do estudante citado, respectivamente. **O valor da média deve sempre conter duas casas decimais.**

Caso o índice informado não exista no vetor (estudante não encontrado), sua função deve imprimir:

```
RGA: null
Nome: null
Media: null
------------
```

A função main

A função `main` do seu programa deve, primeiramente, ler um inteiro $n \leq 10000$. Em seguida, você deve ler n estudantes, onde cada leitura de estudante deve ser feita em três linhas. A primeira delas deve informar o RGA, a segunda o nome e por fim a média do estudante na disciplina.

Logo após, sua main deve ler uma opção op, indicando se queremos ordenar os estudantes por RGA, nome ou média. Então, um inteiro $c \leq 10000$ deve ser lido, indicando a quantidade de buscas a serem realizadas entre estudantes, todas elas da mesma opção lida em op. Logo após, devem ser lidos os valores específicos para a busca (RGA, nome ou média), que dependem da opção escolhida. Para cada estudante, suas informações devem ser impressas conforme listado na função de impressão. Isso inclui também a impressão especial de estudantes que não foram encontrados na sua busca.

Após encerradas as `c` linhas de busca, seu programa deve imprimir todos os estudantes ordenados.

Exemplo 1

Entrada

```
5
12
marcelo
5.6
23
julia
9.3
13
anderson
4.5
83
selton
3.2
9
alice
5.3
n
2
alice
bob
```

Saída

```
RGA: 9
Nome: alice
Media: 5.30
------------
RGA: null
Nome: null
Media: null
------------
RGA: 9
Nome: alice
Media: 5.30
------------
RGA: 13
Nome: anderson
Media: 4.50
------------
RGA: 23
```

```
Nome: julia
Media: 9.30
------------
RGA: 12
Nome: marcelo
Media: 5.60
------------
RGA: 83
Nome: selton
Media: 3.20
------------
```

Exemplo 2

Entrada

```
5
12
marcelo
5.6
23
julia
9.3
13
anderson
4.5
83
selton
3.2
9
alice
5.3
r
2
83
12
```

Saída

```
RGA: 83
Nome: selton
Media: 3.20
```

```
------------
RGA: 12
Nome: marcelo
Media: 5.60
------------
RGA: 9
Nome: alice
Media: 5.30
------------
RGA: 12
Nome: marcelo
Media: 5.60
------------
RGA: 13
Nome: anderson
Media: 4.50
------------
RGA: 23
Nome: julia
Media: 9.30
------------
RGA: 83
Nome: selton
Media: 3.20
------------
```

Ponto Extra

Além da pontuação do trabalho, um ponto extra será creditado caso seja implementada uma comparação de tempo de ordenação entre o algoritmo utilizado e um dos algoritmos de execução $O(n^2)$. Você deve implementar, após a linha final de execução do programa principal, uma ordenação do vetor de maneira ingênua (com tempo quadrático), e logo após imprimir a razão do desempenho entre a versão $O(n \log_n)$ e a versão $O(n^2)$. A impressão deve seguir o seguinte formato:

```
EXTRA -> razao
```

onde `razao` é um valor em double com 6 casas decimais calculado na seguinte equação:

$$razao = \frac{t_q}{t_o}$$

onde t_q é o tempo de duração do algoritmo quadrático, e t_l é o tempo de duração do algoritmo otimizado. Ambos os tempos devem ser computados em microssegundos.

Atenção: Atente-se para não aplicar essa ordenação no vetor original já ordenado. Caso contrário, você estará fazendo uma comparação injusta (pior caso x melhor caso).

Nota

A nota do trabalho será contabilizada da seguinte maneira:

1. Ordenação - 3 pontos
2. Busca - 2 pontos
3. Impressão - 1 ponto
4. Estrutura da main - 1 ponto
5. Casos de teste - 3 pontos

Atenção: Não esqueça de **comentar seu código** de maneira concisa e clara!

Entrega

A entrega deve ser feita até o último minuto (23:59) do dia 25/10.