

Universidade Federal de Ouro Preto  
Bacharelado em Ciência da Computação

Gerenciamento de Repúblicas Estudantis

Trabalho Prático I  
Estrutura de Dados I

Daniel Monteiro Valério  
Fabio Henrique Alves Fernandes

Professora Amanda Nascimento

Ouro Preto, 2 de setembro de 2019

<b>Introdução</b>	<b>2</b>
<b>Problema Proposto</b>	<b>3</b>
<b>Objetivos</b>	<b>3</b>
<b>Modelagem do Sistema</b>	<b>3</b>
<b>Fluxograma</b>	<b>7</b>
<b>Discussões</b>	<b>9</b>
<b>Referências Bibliográficas</b>	<b>9</b>

## ● Introdução

As chamadas repúblicas de Ouro Preto e Mariana englobam as repúblicas estudantis, públicas ou particulares, localizadas nas cidades mineiras de Ouro Preto e Mariana, onde vivem mormente alunos da UFOP, a Universidade Federal de Ouro Preto. Uma pesquisa realizada em 2011 pela Associação Nacional dos Dirigentes das Instituições Federais de Ensino Superior mostrou que os alunos da UFOP, no Brasil, são os que mais admitem o consumo de bebidas alcoólicas.

No caso das repúblicas estudantis federais (REFOP), ou seja, casas adquiridas com verba do governo federal, os estudantes têm à sua disposição uma forma de moradia peculiar, um modelo que não é visto em nenhum outro lugar do Brasil, tendo apenas uma situação análoga nas Repúblicas Estudantis de Coimbra que pertencem a Universidade de Coimbra. Trata-se de 58 repúblicas em Ouro Preto que o governo assegura a um grupo de estudantes o poder de autogestão, em que, os estudantes mantêm a moradia e cada casa tem autonomia para estabelecer seu próprio regimento interno. Grande parte delas bem como repúblicas particulares da cidade de Ouro Preto estabelecem um sistema de seleção de novos moradores conhecido como "Batalha".

A "Batalha de vaga" consiste em um período de avaliação dos candidatos a moradores na república pelos moradores mais antigos. Cada república tem critérios específicos de escolha dos candidatos, mas a participação nas atividades de manutenção da casa e em atividades acadêmicas e extra-acadêmicas pelos chamados "bichos" são critérios comuns à maioria das repúblicas nessa escolha. Com isso, de acordo com os próprios alunos, os estudantes conseguiriam manter esse patrimônio histórico com seu trabalho organizando eventos para arrecadação de fundos que são investidos na manutenção das repúblicas. Os alunos alegam que a universidade teria um gasto muito alto para mantê-las, perdendo poder de investimento na qualidade de ensino. Dessa maneira, os moradores acordaram com a própria universidade a autonomia de escolha de moradores dando em contrapartida manutenção dos imóveis pertencentes a ela.

As repúblicas mantêm uma hierarquia por ordem de chegada à casa. A hierarquia serve principalmente para melhor organização do grupo. Porém, por se tratar de entidades autônomas e possuírem seu próprio regimento interno, as regras de organização, como a "Batalha" e a hierarquia sofrem variações de república para república. Há casos, por exemplo, onde as mesmas não possuem hierarquia.

- **Problema Proposto**

Pensando em uma forma de aprimorar a organização interna das repúblicas, criamos um sistema que, de acordo com a hierarquia de cada morador de uma determinada república, cada tarefa de cada morador fosse organizada e apresentada.

- **Objetivos**

Usando a Linguagem C, faremos um sistema para ler, ordenar e mostrar cada morador e seus afazeres dentro de cada república. Contendo 4 Tipos Abstratos de Dados, o sistema se utiliza do método Merge Sort para a ordenação e do Busca Linear para busca.

- **Modelagem do Sistema**

O sistema é composto de 4 Tipos Abstratos de Dados: o relacionado a cara morador, o de cada tarefa de faxina dos moradores, o dos endereços de cada morador e da própria república, além do relacionado a república em si.

Cada TAD tem suas funções para criação, atualização, ler e deletar.

- **TAD Morador:** Daniel

A TAD Morador é composta pelo nome, apelido na república, cidade de onde veio, idade e a data de ingresso na república.

```
struct morador {  
    char nome[102];  
    char apelido[22];  
    char cidade[32];  
    int idade;  
    char curso[52];  
    char dataIngresso[12];  
};
```

- **TAD Endereço:** Fabio

A TAD Endereço é composta pela rua, número, complemento, cidade, estado, país e CEP.

```
struct endereco {  
    string rua;  
    string complemento;  
    string bairro;  
    string cidade;  
    string estado;  
    string país;  
    int numero;  
    long int cep;  
};
```

- **TAD Faxina:** Daniel

A TAD Faxina é composta pela área a ser faxinada, o morador destinado àquela tarefa e o estado em que a área se encontra (limpa ou suja).

```
struct faxina{  
    char area[32];  
    Morador* mrd;  
    int situacao;  
};
```

- **TAD República:** Fabio

A TAD República, última e mais importante do sistema, é a que une todas as outras TAD's. Ela é composta pelo nome da república, a quantidade de moradores, um vetor de moradores, um vetor de faxinas e a data de criação.

```
struct republica{  
    Morador Moradores;  
    string nomeRep;  
    Endereco endRep;  
    string dataCriação;  
    int quantMorador;  
    int quantMax;  
    Faxina faxinas;  
};
```

- **Algoritmo de Ordenação:** Daniel

Para mantermos os moradores ordenados pela hierarquia (do mais velho até o mais novo em relação ao tempo de moradia) usamos o método Merge Sort de ordenação que, além de recursivo é rápido e de simples implementação

```
void mergeSort ( int* v, int inicio, int fim ) {
    if ( inicio < fim ) {
        int meio = ( inicio + fim ) / 2;
        mergeSort(v, inicio, meio);
        mergeSort(v, meio + 1, fim);
        merge(v, inicio, meio, fim);
    }
}

void merge ( int* v, int inicio, int meio, int fim ) {
    int i, j, k;
    int n1 = meio - inicio + 1;
    int n2 = fim - meio;
    int* v1 = malloc ( n1 * sizeof(int));
    int* v2 = malloc ( n2 * sizeof(int));
    for ( i = 0; i < n1; i++ )
        v1[i] = v[inicio + i];
    for ( j = 0; j < n2; j++ )
        v2[j] = v[meio + 1 + j];
    i = 0, j = 0, k = inicio;
    while ( i < n1 && j < n2 ) {
        if ( v1[i] <= v2[j] ) {
            v[k] = v1[i];
            i++;
        }
        else {
            v[k] = v2[j];
            j++;
        }
        k++;
    }
    while ( i < n1 ) {
        v[k] = v1[i];
        i++;
        k++;
    }
    while ( j < n2 ) {
        v[k] = v2[j];
        j++;
        k++;
    }
}
```

```

    }
    free(v1);
    free(v2);
}

```

- **Algoritmo de Busca: Fabio**

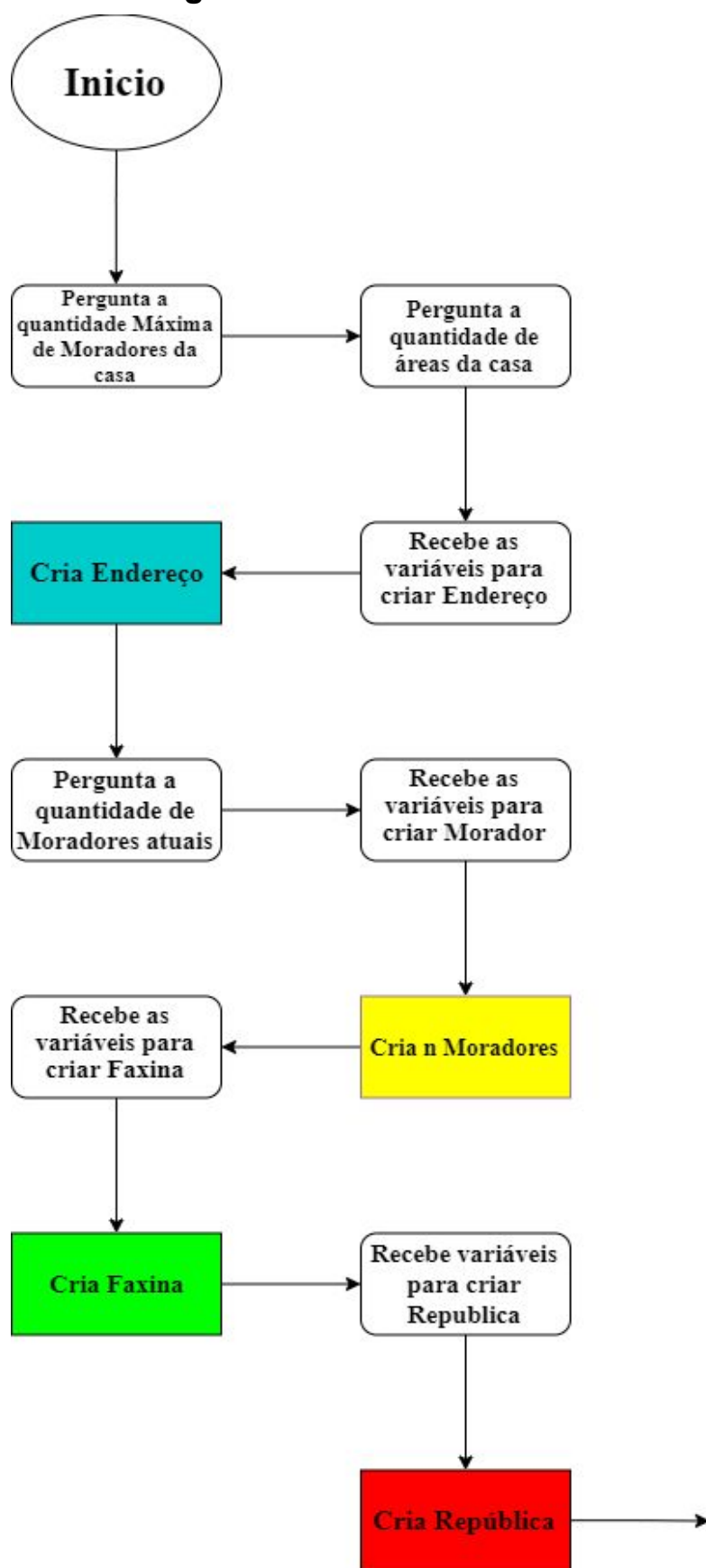
O algoritmo de busca utilizado foi o Busca Linear, por ser simples e objetivo e usando sua forma recursiva. Sua complexidade, no pior caso é  $f(n) = O(n)$ , por ele ter que percorrer todo o vetor de strings.

```

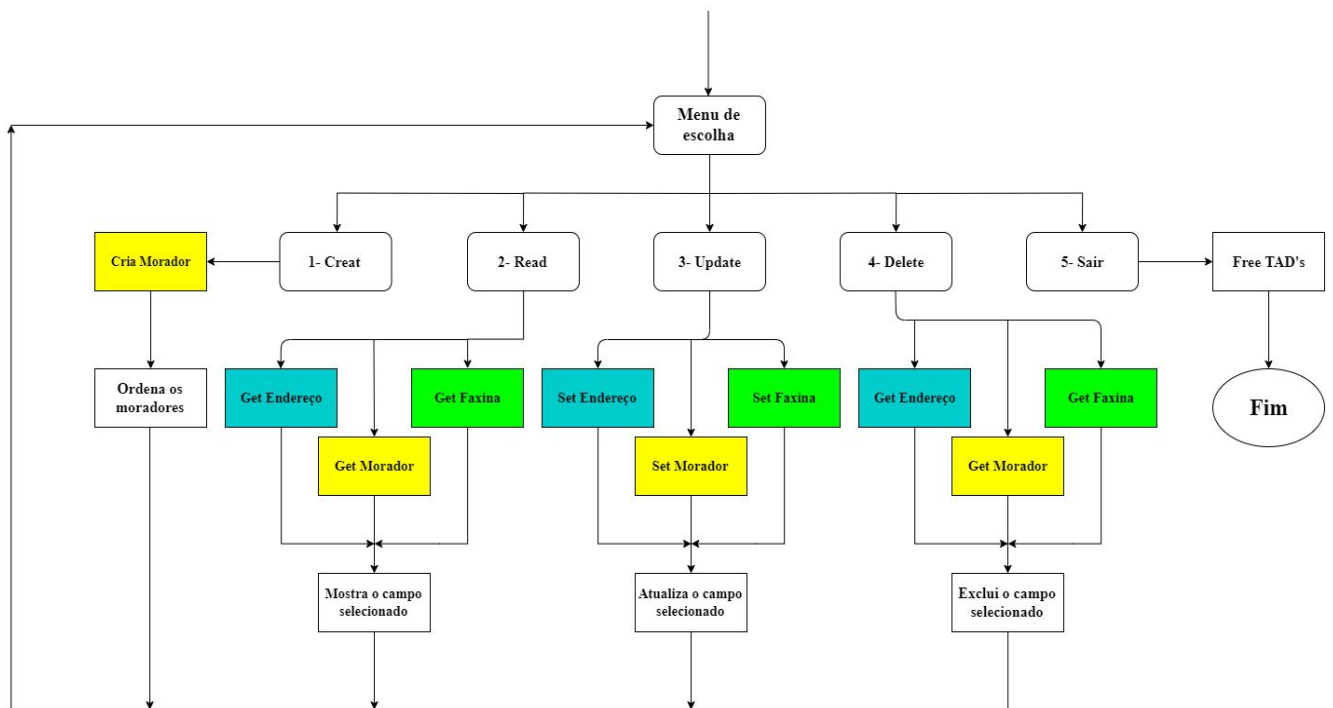
int busca_nome (char nome[], char **vetorNomes, int esq, int dir){
    if (esq > dir){
        return -1; //não encontrado
    }else{
        if (strcmp(nome, vetorNomes[esq]) == 0){
            return esq;
        }
        else{
            return busca_nome(nome, vetorNomes, esq+1, dir);
        }
    }
}

```

- Fluxograma


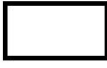







### Legenda:

#### Formas geométricas:

-  Indica ações do usuário
-  Indica processos do programa
-  Indica início ou fim do programa

#### Cores:

-  Morador
-  Endereço
-  Faxina
-  Republica

- **Discussões**

O Algoritmo de Ordenação usado (Merge Sort) usa do método dividir para conquistar. O vetor é dividido em subvetores, e vão sendo ordenados de forma recursiva até que seja encontrada uma solução completa para a ordenação, onde os subvetores são unidos em um único vetor ordenado.

Dependendo da quantidade de elementos, o uso de memória e de gasto de tempo pode ser alta por ser uma função recursiva. Sua complexidade é  $f(n) = \Theta(n \log n)$ .

O Algoritmo de Busca usado (Busca Linear) percorre o vetor completamente até encontrar o elemento procurado.

Por ser uma função que somente percorre o vetor, seu gasto de memória e tempo dependem da quantidade de elementos, e por ela ter sido usada na sua forma recursiva, o gasto de memória aumenta ainda mais por conta das chamadas sucessivas da função. Sua complexidade é  $f(n) = O(n)$  no pior caso e  $f(n) = 1$  no melhor caso.

- **Referências Bibliográficas**

[https://pt.wikipedia.org/wiki/Rep%C3%BAblicas\\_de\\_Ouro\\_Preto\\_e\\_Mariana](https://pt.wikipedia.org/wiki/Rep%C3%BAblicas_de_Ouro_Preto_e_Mariana)

[http://www.decom.ufop.br/anascimento/site\\_media/uploads/bcc202/Aula%2003%20-%20TAD.pdf](http://www.decom.ufop.br/anascimento/site_media/uploads/bcc202/Aula%2003%20-%20TAD.pdf)

<https://pt.stackoverflow.com/questions/383134/busca-bin%C3%A1ria-recursiva-por-string-e-m-c>

[https://pt.wikipedia.org/wiki/Merge\\_sort](https://pt.wikipedia.org/wiki/Merge_sort)

[https://pt.wikipedia.org/wiki/Busca\\_linear](https://pt.wikipedia.org/wiki/Busca_linear)