**Documentação Trabalho Prático 01**

**DCC206 - Algoritmos 1**

**Nome: Heloiza Aparecida dos Santos**

**Matrícula: 2017086953**

Departamento de Ciencia da Computação – Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte – MG – Brasil

heloizas@ufmg.br

1. **Introdução:**

Este trabalho prático teve como objetivo desenvolver conhecimentos a cerca de Casamento Estável, assunto estudado nas aulas da disciplina Algoritmos 1 - DCC206. Foi solicitado o desenvolvimento de um programa para auxiliar na alocação de pessoas para postos de vacinação para auxiliar na distribuição das vacinas contra o vírus Sars-CoV-2. Foi necessário respeitar a capacidade de cada posto e seguir a condição de estabilidade ótima no ponto de vista das pessoas, o que significa que a pessoa mais velha deve ser atendida no posto mais próximo de sua localização.

1. **Implementação – Modelagem Computacional:**

* A estratégia utilizada para desenvolver o algoritmo foi a implementação de classes para Gerenciador, Pessoa, Posto e Casamento Estável. A classe principal é a CasamentoEstavel, que possui toda a lógica para gerar combinação estável, que foi baseada no algoritmo de Gale-Shapley.
* As classes estão organizadas da seguinte maneira:

|  |  |
| --- | --- |
| **Classes:** | **Principais responsabilidades:** |
| Gerenciador | Responsável por ler o arquivo e inicializar objetos para criação dos postos e pessoas. Cria a lista de preferência de cada pessoa com base na distância euclidiana e cria a lista de preferência dos postos em ordem decrescente de idade.  Principais funções:  - criarPostos();  - criarListaPreferenciaPosto();  - criarPessoas();  - criarListaPreferenciaPessoas(); |
| CasamentoEstavel | Responsável por gerar a combinação estável levando em consideração o caso 1 ou caso 2.  Caso 1: Há mais vagas que aplicações ou há um número igual de vagas. Caso 2: Há menos vagas que aplicações.  Principais funções:  - gerarCombinacaoEstavel();  - gerarCombinacaoEstavelCaso1();  - gerarCombinacaoEstavelCaso2();  - checarSeTemPostoCheio();  - checarSeTemPessoaSemAlocar();  - imprimirSaida(); |
| Pessoa | Armazena o id, idade, lista de preferência, posição (x,y), se a pessoa já foi alocada ou não. |
| Posto | Armazena o id, capacidade, lista de preferência, posição (x,y), se o posto está cheio ou não. |

No arquivo Main do programa, um objeto da classe Gerenciador é instanciado: gerenciador.

Um objeto da classe CasamentoEstavel também é instanciado: casamentoEstavel(gerenciador).

O programa segue o seguinte pseudocódigo:

|  |
| --- |
| **INÍCIO**  **(Classe Gerenciador)**  Lê o arquivo;  Cria array de postos;  Cria array de pessoas;  Cria array de lista de preferência para cada posto;  Cria array de lista de preferência para cada pessoa;  **(Classe CasamentoEstavel)**  Identifica caso:  Caso 1: + vagas que aplicações || número igual.  Caso 2: - vagas que aplicações).  Dependendo do cenário, condição do while é alterada:  Caso 1: while continua enquanto houver algum posto vazio.  Caso 2: while continua enquanto houver alguma pessoa sem alocação.  **Pseudocódigo** (Baseado no Gale-Shapley):  **Enquanto** (não houver posto cheio || houver pessoa sem alocação)  Percorra o array de postos[i].  Percorra a lista de pessoas[j].  **Se** (pessoa[j] não estiver alocada)  Aloca a pessoa[j] no posto[i].  **Senão se** (pessoa[j] já está alocada)  Verifica se a alocação candidata é preferível.  **Se** (alocação candidata é preferível)  Quebra a alocação atual.  Efetiva a nova alocação posto[i] com pessoa[j].  **Senão**  Pessoa[j] rejeita posto[i]. |

O código está organizado seguindo a seguinte estrutura de pastas:

projeto

- src: códigos .cc.

- obj: arquivos .o.

- include: headers .h.

Makefile  
tp01

casoTeste01.txt

* O trabalho foi desenvolvido e testado utilizando a seguinte configuração:  
  Sistema Operacional: macOS Catalina 10.15.7  
  Linguagem de programação implementada: C++  
  Compilador utilizado: GCC  
  Dados do processador: Intel Iris Plus Graphics 640  
  Memória RAM: 8 GB

1. **Análise de complexidade das principais funções:**

|  |  |
| --- | --- |
| **Funções:** | **Complexidade:** |
| Classe: Gerenciador  criarPostos();  criarPessoas(); | **Complexidade de Tempo:**  Complexidade constante Θ(1). Apenas atribuição.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |
| Classe: Gerenciador  criarListaPreferenciaPosto();  criarListaPreferenciaPessoas(); | **Complexidade de Tempo:**  Complexidade constante Θ(1). Apenas atribuição.  **Complexidade de Espaço:**  Complexidade constante Θ(1).  Cria lista de preferência com base na idade das pessoas (para postos) e com base na distância (para pessoas).  Para ordenar as listas, foi utilizado o algoritmo de ordenação XXX que foi desenvolvido na disciplina de Estrutura de Dados. O pior caso desse algoritmo é XXX e YYY no melhor caso. |
| Classe: CasamentoEstavel  gerarCombinacaoEstavel (); | Complexidade constante Θ(1). Apenas atribuição. |
| Classe: CasamentoEstavel  gerarCombinacaoEstavelCaso1();  gerarCombinacaoEstavelCaso1(); |  |
| Classe: CasamentoEstavel  checarSeTemPostoCheio ();  checarSeTemPessoaSemAlocar (); | Complexidade Θ(n) dependendo da quantidade de postos ou quantidade de pessoas, pois percorre os arrays de postos ou pessoas e retorna se encontrar pelo menos um posto cheio ou pelo menos uma pessoa sem alocar.  No melhor caso, o posto cheio ou pessoa sem alocar é o primeiro da lista, e no pior caso é o último. |
| Classe: CasamentoEstavel  imprimirSaida(); | Complexidade Θ(n2) pois são 2 loops aninhados que percorrem o array de postos juntamente com o array de pessoas. |

1. **Instruções de compilação e execução:**

Na pasta principal do programa, executar o comando make. Em seguida, rodar o programa fornecendo o arquivo de entrada.

Exemplo:

|  |
| --- |
| make  ./tp01 < casoTeste01.txt |

1. **Conclusão:**

Esse trabalho lidou com o problema de Casamento Estável. Foi necessário ler o arquivo de entrada, criar classes, atribuir responsabilidades a cada uma delas, criar listas de preferências e implementar o algoritmo de Casamento Estável para alocar as pessoas nos postos de saúdes. Foi respeitado a capacidade de cada posto seguindo a condição de estabilidade ótima no ponto de vista das pessoas.

**Referências:**

1. Jussara Almeida. (2021). Slides virtuais da disciplina de Algoritmos 1. Disponibilizado via moodle. Departamento de Ciencia da Computação. Universidade Federal de Minas Gerais. Belo Horizonte.
2. https://www.geeksforgeeks.org/stable-marriage-problem/