Mini SISU - Estrutura de Dados

Breno Poggiali de Sousa - 2018054800

Universidade Federal de Minas Gerais

Mini SISU para Arendelle

Introdução

Esse projeto foi desenvolvido especialmente para o Ministério da Educação do reino de Arendelle. A ideia, solicitada por Elsa - rainha de Arendelle - é fazer algo parecido com o SISU brasileiro, porém reduzido a um único dia. Em que os alunos das escolas de gelo, que realizaram o ENEM de gelo, possam se inscrever nas universidades de gelo. De forma justa, os que tiveram maiores notas tem preferência, os que optaram pela matéria como primeira opção possuem preferência quando comparados com alguém que teve a mesma nota e colocou a matéria como segunda opção e, por fim, os que se cadastraram antes, possuem preferência quando comparados a alguém que possui a mesma nota que eles e a mesma preferência de curso (primeira ou segunda opção).

Implementação

Para implementar esse MINI sisu para o reino de Elsa, utilizamos de uma lista duplamente encadeada por dois motivos:

1. Com uma lista duplamente encadeada, podemos navegar mais facilmente entre os alunos
2. Realizar uma lista duplamente encadeada foi ótimo para treinar as estruturas de dados ensinadas na matéria de Estrutura de Dados e que Elsa tanto ouviu falar bem.

   Funções:

   - Arrays: criamos 3 arrays para navergamos mais facilmente entre os dados. Um array que guarda os nomes dos cursos e outro array que guarda as vagas de cada curso (para um mesmo curso, eles possuem o mesmo índice). Além disso, usamos um array com variáveis do tipo Aluno que guarda todos os alunos na ordem que eles foram se cadastrando e, por fim, um array com variáveis do tipo ListaCursos com variáveis do tipo ListaCursos.
   - **bubbleSort(Aluno *alunos, int n)**: essa função recebe a lista de todos os alunos cadastrados (por ordem de cadastro) e faz um bubbleSort com eles de acordo com a nota, deixando as maiores notas acima.

- ListaCursos: se trata de uma lista encadeada de um curso que contém o nome e a quantidade de vagas de cada curso. Primeiro distribuímos a primeira opção de cada aluno (da maior pra menor nota). E depois, também da maior pra menor nota, conferimos se esse aluno passou na primeira opção dele e, caso não tenha passado, colocamos ele na segunda opção e ordenamos.

ListaCursos()

- **append():** Insere um elemento no fim da lista.
- **get_iesimo(int):** Retorna um elemento na posição i.
- **set_iesimo(int, Aluno, dif):** Altera um elemento na posicao i e sua classificacao.
- **get_nota_de_corte():** Retorna nota de corte do curso.
- **get_vagas():** Retorna vagas do curso.
- **size():** Retorna o tamanho da lista.

Alunos()

- **string get_nome(): Retorna o nome do Aluno**
- **float get_nota():** Retorna a nota do Aluno
- **int get_opcao0():** Retorna a opcao0 do Aluno
- **int get_opcao1():** Retorna a opcao1 do Aluno
- **int get_classificacao_opcao0():** Retorna a classificação do Aluno em sua opcao.
- **void dif_classificacao_opcao0(int dif):** Altera a classificacao do aluno em sua opção 0.
- **void set_classificacao_opcao0(int classificacao):** Define a classificacao de um aluno em sua opcao 0.

Compilador : o compilador utilizado foi o g++, como seguinte comando "básico":

*g++ *.cpp -o main*

Isso pode ser simplificado apenas com o seguinte comando:

make

Mini SISU para Arendelle

Entrada e saída de dados: A entrada e a saída do programa ocorrem por meio da entrada padrão. Sendo assim, para executar o programa com a entrada "entrada.in" e a saída "saída.out", por exemplo, após compilado, fazemos:

./main < entrada.in > saida.out

Testes: após compilar o programa o seguinte comando pode ser executado:

make tests

Esse comando executa alguns testes (passados pelo professor) que, retornam a diferença entre a saída desejada e a saída do programa em 6 casos. Caso ele não retorne nada , é porque está tudo funcionando normalmente. :)

Análise de Complexidade

Em todos os testes que eu executei (e os testes dados no moodle pelo professor e monitores), o tempo para que o código terminasse e obtivessemos o arquivo file.out foi inferior a 1 segundo. O que por mim, foi visto com muitos bons olhos, já que Arendelle possui uma população de bem pequena.

Já quanto ao espaço do algoritmo, fizemos uma implementação sucinta que resolvia particulmarmente o problema apresentado por Elsa. Focamos em utilizar uma lista encadeada (no caso, duplamente), quando ela foi necessária, precisamente. Ou seja, na lista de cada curso.

Por fim, quanto à complexidade do algoritmo, fizemos um algoritmo que resolve o problema com **$O(n^2)$**. Isso porque, nossa operação mais custosa é um bubble sort que é executado sempre em cima da nossa lista de alunos, executamos um bubble sort nela que possui complexidade n^2 e é a maior complexidade que existe em nosso código. Ou seja, no pior caso, sempre teremos $O(n^2)$.

Sendo assim, temos um **pior caso $O(n^2)$** e, no melhor caso, como o bubble sort só passa n vezes e não precisamos ordenar nada, temos O(n) operações.

Conclusão

Acredito que resolvemos bem o problema de Elsa e, assim, o SISU de gelo já pode começar a acontecer no reino de Arendelle. Fiquei feliz, também, com a confiança de Elsa no

potencial dos alunos da Universidade Federal de Minas Gerais. Por fim, quanto ao trabalho solicitado por ela, assim como quase todos os trabalhos solicitado a estudantes de universidades, foi ótimo para colocar em prática conceitos visto na matéria de Estrutura de Dados e, também, rever alguns conceitos de Programação e Desenvolvimento de Software II.

Referências

Chaimowicz, L. e Prates, R. (2019). Análise de Complexidade de Algoritmos. https://virtual.ufmg.br/20191/pluginfile.php/283587/mod_resource/content/1/ED_Aula1_AnaliseComplexidade.pdf

Ziviani, N. (2010). Projeto de Algoritmos com Implementações em Pascal e C

Ziviani, N. (2019). Listas Lineares. https://virtual.ufmg.br/20191/pluginfile.php/304915/mod_resource/content/1/Aula5_ListasLineares%20-%20LUIZ.pdf

Disney, (Once Upon a Time). Frozen. https://disney.fandom.com/wiki/Frozen