Mestrado Integrado em Engenharia Informática e Computação
2ºAno 2ºSemestre

Tema 3: Atribuição de projetos de dissertação

2MIEIC2:
Fábio Manuel Marques Carneiro – ei08000@fe.up.pt
200801774
João Ricardo Pintas Soares – ei12000@fe.up.pt
201200740
Mário André Macedo Ferreira – ei12105@fe.up.pt
201208066

Professor Dr. Rosaldo Rossetti
Professora Dra. Ana Paula Rocha

Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
28 de Abril de 2014

Índice

1. Introdução

Este trabalho foi realizado no contexto da unidade curricular Concepção e Análise de Algoritmos, CAL, do 2º semestre do 2º ano do ano lectivo 2013/2014 do curso de Mestrado Integrado em Engenharia Informática e Computação (MIEIC).

O projecto surge na continuação da matéria leccionada ao longo do semestre sobre estruturação de dados sobre a forma de grafos e desenvolvimento de algoritmos sobre essa mesma estrutura. O ínicio do projecto foi marcado com um desenho inicial do diagrama de classes, utilizando a ferramenta *Enterprise Architect*, que foi sendo actualizado e alterado conforme as necessidades no decorrer da codificação. Para além disto foi ainda gerado documentação *Doxygen* de todo o código.

No que respeita ao conteúdo do projecto, o objectivo deste passou por desenvolver um programa em C++ que permita ao utilizador fazer a atribuição de temas de tese a estudantes do MIEIC, segundo o algoritmo *Stable Marriage Problem*. Desta forma o programa permite:

- Carregar de ficheiros *.txt* as informações sobre estudantes, proponentes e supervisores;
- Atribuir um supervisor a um projecto caso o seu proponente não seja docente, segundo o *Hungarian Algorithm*;
- Atribuir uma tese a um estudante segundo o *Stable Marriage Problem,* tendo em conta as preferências envolvidas;
- Gerar um grafo que mostre todas as relações entre os elementos envolvidos;

De um modo geral, pensamos ter implementado todas as funcionalidades pedidas, não esquecendo que nos foram dadas poucas informações relativamente a toda a estruturação e implementação de código do projecto.

2. Concepção e Implementação da Solução

2.1. Implementação das Classes

A divisão do projecto em ficheiros foi feita da seguinte forma de forma a facilitar a implementação do projecto:

- *Person.h/.cpp*
  - *Person (class)*

- *Student.h/.cpp*
  - *Student (class)*

- *Master.h/.cpp*
  - *Master (class)*

- *Owner.h/.cpp*
  - *Owner (class)*

- *Graph.h*
  - *Graph (class)*
  - *Vertex (class)*
  - *Edge (class)*

- *connection.h/.cpp*
  - *Connection (class)*

- *edgetype.h*
  - *EdgeType (class)*

- *graphviewer.h/.cpp*
  - *GraphViewer (class)*

- *main.cpp*

2.2. Implementação do Grafo

A implementação do grafo que representa as relações entre os intervenientes do desenvolvimento de uma tese foi feita utilizando a API fornecida pelos docentes da unidade curricular, *GraphViewer*.

No grafo, os vertices são representados por estudantes, supervisores e projectos, cuja implementação se encontra nos ficheiros *Student.h/.cpp, Master.h/.cpp* e *Project.h/.cpp* respectivamente. As arestas representam as preferências dos elementos, isto é, as arestas que saem dos estudantes ou dos supervisores em direção aos projectos têm associado um número inteiro que representa a preferência do estudante ou supervisor pelo projecto a que se liga sendo que quanto menor esse inteiro, maior é a preferência por esse projecto.

2.3. Descrição das Classes Implementadas

- Classe *Person* é a classe mãe de *Student, Owner* e *Master* e guarda o nome da pessoa, e um número *ID* que é único;

- Classe *Student* guarda uma lista de preferências de Projectos definida pelo estudante e guarda também uma lista que diz a que projectos o estudante está ligado

- Classe *Owner* guarda uma lista de preferências de Estudantes e guarda também uma lista que diz a que projectos o estudante está ligado. Para além disto guarda o nome do projecto que chefia;

- Classe *Master* guarda o número máximo que de projectos que o supervisor pode supervisionar e uma lista de preferência de projectos, assim como uma lista de possíveis projectos que pode vir a estar associado;

- Classes *Graph, Vertex* e *Edge* implementadas nas aulas prácticas e que gerem a construção do grafo e seu manuseamento (inserção, remoção, pesquisa de elementos). É de salientar que todas as classes são classes *template*;

- Classes *connection* e *GraphViewer* fornecidas pelos docentes da unidade curricular.

No ficheiro *AlgorithmImplementation.h* apesar de não estar implementada nenhuma classe é o local onde foram desenvolvidas as funções mais importantes e que dizem respeito à atribuição dos projectos de tese aos estudantes e aos supervisores. Algumas dessas funções são:

- ***void thesisAttribuitiontoStudent****(Graph<Person>* graph);* Atribui os projetos de tese aos estudantes de acordo com as preferências dos estudantes e de acordo com a preferência dos proponentes, relativamente aos estudantes.
- ***Void thesisAttribuitionMasters****(Graph<Person>* graph, vector<vector<**int**>> IDmatrix);* Atribui aos supervisores projectos para supervisionar tendo em conta as suas preferências, as preferências dos proponentes no que respeita ao supervisores e o número máximo de projectos que cada supervisor pode supervisionar.

2.4. Algoritmos Utilizados e Respectiva Complexidade

Para resolver o problema de atribuição de teses foi utilizado o algoritmo sugerido pelos docentes da unidade curricular, nomeadamente, *Stable Marriage Problem*.

No caso concreto deste projeto começou-se por associar a um estudante o proponente cujo o projecto é o mais pretendido pelo estudante. Contudo se um segundo estudante pretender o mesmo projecto a decisão de alterar a atribuição do projecto a um estudante passa pelo proponente que se preferir mais o 1º estudante que o segundo não defaz a associação, contudo caso isso não aconteça associa-se ao 2º estudate ficando o 1º novamente sem projecto atribuído. Fez-se este processo ciclicamente até todos os estudantes terem um projeto atribuído.

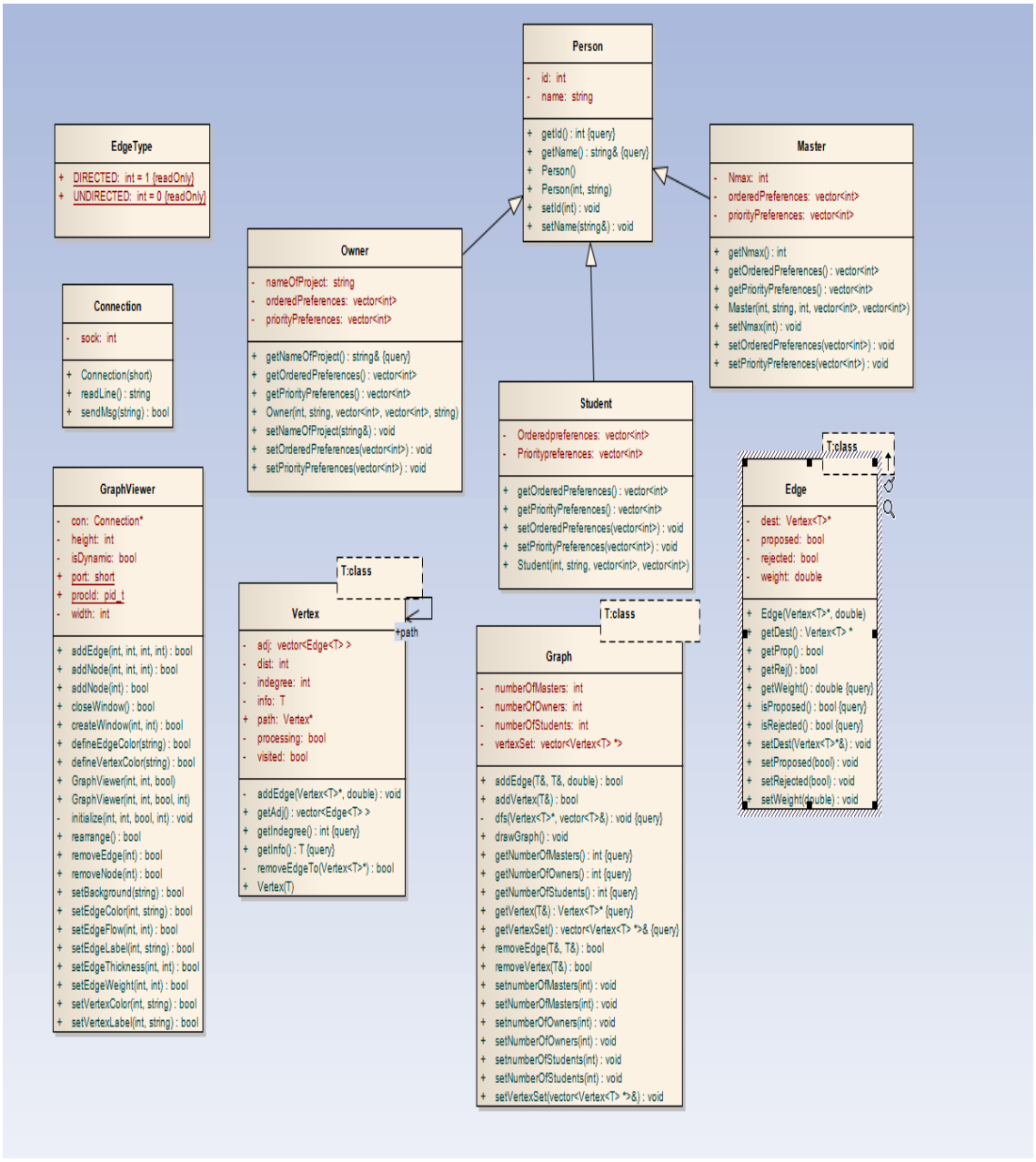
Numa segunda fase, atribui-se supervisores aos projectos cujos proponentes não sejam docentes pelo *Hungarian Algorithm*. Este algoritmo consiste em colocar os *id's* dos supervisores e dos proponentes numa matriz quadrada de tamanho 'a' e subtrair a cada linha da matriz o elemento menor dessa mesma linha e de seguida fazer o mesmo com as colunas. De seguida tenta-se enquadrar o maior número possível de zeros e se o esse número de zeros for 'a' então está-se perante uma associação óptima. Caso isto não

aconteça, cobre-se os zeros enquadrados com número de traços igual ao número de zeros. Depois disto volta-se ao início do algoritmo.

Este algoritmo apresenta complexidade temporal constante e equivalente a $O(n^3)$.

3. Estrutura das Classes

O esquema abaixo representa o diagrama de classes resultante do projecto. Diagrama este que foi sendo atualizado conforme a estruturação do código sofreu alterações.



4. Conclusão

Em síntese pensamos termos resolvido todos os problemas que nos foram propostos e que nos autopropusemos. É também importante salientar que foi tudo realizado dentro do período definido pelos docentes. Contudo, foi precisamente neste campo que sentimos as maiores dificuldades, ou seja, não gerimos o tempo da forma mais correcta.

Para além desta adversidade, a outra principal dificuldade foi inicialmente perceber como implementar os métodos necessários, *Stable Marriage Problem* e *Hungarian Algorithm*.

Durante a realização do projecto a participação na sua realização foi quantitivamente semelhante, uma vez que a sua realização utilizou ferramentas que permitem trabalho de grupo, nomeadamente *Skype*, *Saros* e *VS Anywhere*. Claro está que não deixa de ser importante um agradecimento aos docentes da disciplina, professor Dra. Ana Paula Rocha e professor Dr. Rosaldo Rossetti.

5. Biliografia

- Wikipedia, 2 April 2014 at 21:57, http://en.wikipedia.org/wiki/Stable_marriage_problem

- Apontamentos das Aulas Teóricas https://moodle.up.pt/course/view.php?id=1497

- *Juan Soulie*, "C++ Language Tutorial", cplusplus.com, 2000-2014, http://www.cplusplus.com/doc/tutorial/

- Ana Paula Tomás, "Emparelhamentos, Casamentos Est□aveis e Algoritmos de Colocação de Professores", Março 2005, http://www.dcc.fc.up.pt/Pubs/TR05/dcc-2005-02.pdf

- Wikipedia, 23 April 2014 at 02:12, http://en.wikipedia.org/wiki/Hungarian_algorithm