Proposta de Trabalho

Uma Rede Social Profissional

Aplicação de Estruturas de dados não lineares e Programação Orientada aos Objetos (POO)

Linguagens Programação II

Rui Silva Moreira
rmoreira@ufp.edu.pt
Beatriz Gomes
argomes@ufp.edu.pt

Algoritmia Estruturas Dados II

José Torres
jtorres@ufp.edu.pt
André Pinto
arpinto@ufp.edu.pt
Beatriz Gomes
argomes@ufp.edu.pt

Março 2019

Universidade Fernando Pessoa
Faculdade de Ciência e Tecnologia

1. Definição do problema

Neste projeto pretende-se que os alunos modelizem, implementem, testem e documentem uma aplicação Java para manipular e gerir informação relativa a uma rede social Profissional de pessoas, ligações entre pessoas, empresas e encontros programados. Mais concretamente, pretende-se que combinem a utilização de estruturas de dados orientadas a objetos (e.g. tabelas de símbolos e grafos) para armazenar e gerir a informação necessária à rede social.

As estruturas do tipo *Symbol Table* (e.g. *Hashmaps*, *Binary Search Trees*, *Redblack Trees,* etc.) deverão permitir armazenar e gerir a informação relativa às entidades que se pretendem manipular (e.g. pessoas, encontros, empresas, históricos, competências). Por exemplo, para cada pessoa, deve registar-se as suas competências, histórico de empresas onde colaborou, áreas de interesse, empresa atual, localização, entre outros atributos.

As empresas podem promover relações entre profissionais criando encontros (meetings) no seu domínio de atuação. Deverá ser possível sugerir encontros a profissionais com base nas suas áreas de interesse. A participação em encontros, competências e interesse em empresas em comum, contribuem para o aumento do peso da ligação (rating) no grafo de relacionamentos entre profissionais.

Através das coordenadas das empresas e encontros, é possível adicionar os mesmos a um mapa previamente carregado de forma a ser possível pesquisar caminhos (paths) para alcançar uma determinada empresa ou encontro.

As diferentes estruturas do tipo grafo permitirão armazenar a informação relativa às ligações entre profissionais, entre profissionais e empresas e mapa de localização/caminhos.

No grafo de localização/caminhos, cada empresa, encontro e ponto de passagem corresponderão a um nó (vértice) do grafo e poderão ter um conjunto de ligações (arestas) a outras entidades. As ligações serão dirigidas e poderão caracterizar-se por vários pesos (e.g., distância, tempo, etc.). As ligações entre empresas e encontros são sempre através de pontos de passagem, não sendo possível ligar diretamente empresas a outras empresas ou encontros.

Existe também a possibilidade de profissionais seguirem empresas e vice-versa. A aplicação deverá permitir combinar e gerir a informação das diferentes ligações e entidades.

Embora a solução pudesse utilizar uma arquitetura cliente-servidor, para facilitar a implementação, irá utilizar-se uma arquitetura *standalone*, ou seja, uma implementação que deverá funcionar num único PC. Os alunos deverão utilizar pacotes de software pré-existentes que oferecem estruturas de dados genéricas (cf. grafos, árvores, tabelas de símbolos etc.), que possam ser reutilizadas na implementação do problema proposto. Desta forma não terão que implementar as estruturas de dados básicas, podendo concentrar-se na lógica e requisitos funcionais da aplicação proposta.

1.1. Requisitos funcionais

Pretende-se que os alunos sigam uma abordagem orientada aos objetos na modelização e implementação do problema proposto. Em concreto deverão desenhar os diagramas de classes necessários que permitam modelizar o problema, reutilizando pacotes/classes pré-existentes (cf. grafo, árvores, *hashmap*, etc.) através de herança, composição ou agregação.

Pretende-se que desenvolvam uma API de classes (biblioteca/conjunto de métodos em classes) que satisfaçam os requisitos listados a seguir. Pretende-se, também, que se implementem casos de teste dessa API (funções de teste) para as várias funcionalidades/requisitos implementados. Cada caso de teste deverá ser devidamente documentado numa função *static* que deverá ser caracterizada pelo conjunto de funções a testar, pelos valores de input a utilizar no teste (preferencialmente de ficheiro ou, em alternativa, editando diretamente o código, mas nunca provenientes de valores interactivamente inseridos pelo utilizador), e por valores de output/resultado do teste enviados para a consola e/ou escritos em ficheiro.

Em concreto a aplicação deverá cumprir os seguintes requisitos:

Fase 1

R1. O modelo de dados deverá permitir representar todas a redes e respetivas ligações, bem como toda a meta-informação associada a estas entidades; profissionais, empresas, encontros, competências; As ligações dos diferentes grafos, devem ser representadas através dos vários pesos (e.g., classificação, distancia, tempo).

R2. Devem suportar modelos de dados para os valores (classe genérica *Java Value* das *Symbol Tables* (ST)) dos elementos das várias tabelas de símbolos consideradas no projeto. Deverão utilizar funções e tabelas de *hash* em, pelo menos, uma ST cuja chave não tenha que ser ordenável. Deverão usar BSTs balanceadas (*redblack*) nas STs cuja chave é ordenável (e.g., tempos, ordem, etc.).

R3. Devem criar funções para inserir, remover, editar e listar toda a informação, para cada uma das várias STs consideradas na base de dados.

R4. Deverão validar a consistência de toda a informação como, por exemplo, que todos os profissionais que são mencionados noutras STs, existam na ST pessoas.

R5. Deverão considerar a remoção de informação e nestes casos deve-se arquivar a informação. Por exemplo, ao remover uma empresa da base de dados deve garantir-se a sua remoção total do sistema e respetivo arquivamento (em ficheiro ou em estruturas auxiliares).

R6. Deve-se popular as diversas STs da aplicação com o conteúdo de ficheiros de texto de entrada (carregar/gravar a informação em ficheiro txt).

R7. Deve garantir-se o *output* (*dump*) de toda a informação para ficheiros de texto, isto é, de profissionais, empresas, dos diversos grafos e as suas ligações e de todas as outras STs.

R8. Devem implementar-se diversas pesquisas à base de informação como, por exemplo: profissionais com determinadas competências; profissionais sem empresa; encontros ocorridos num dado intervalo de tempo; encontros realizados por determinada empresa; encontros com mais de n participantes, etc.

Fase 2

R9. Cada nó ou vértice do grafo de localização/caminhos representa uma empresa, encontro ou ponto de passagem. Deverá ter um conjunto de atributos/propriedades principais (e.g., nome, localização, etc.) e outros atributos que possam vir a ser necessários;

R10. O modelo de dados deve prever a utilização de algoritmos genéricos de gestão e verificação de grafos, nomeadamente:

a) Algoritmos de cálculo: do caminho mais curto entre um ponto de passagem e um encontro ou empresa selecionados (com base na distância entre nós); do caminho mais rápido (com base no custo temporal das viagens); do caminho mais direto (com base no número de saltos).
b) Verificar se o grafo de ligações entre pessoas é conexo;
c) Verificar se o grafo de ligações entre pessoas e empresas é bipartido.
d) Selecionar um subgrafo e aplicar-lhe os mesmos algoritmos ou funcionalidades descritas anteriormente.

R11. Deverá ser possível efetuar e combinar vários tipos de pesquisas sobre os grafos como, por exemplo:

a) Listar as pessoas (profissionais) que seguem uma determinada empresa.
b) Listar as pessoas (profissionais) que participaram um determinado encontro.
c) Listar as pessoas (profissionais) que chegaram a um encontro num determinado intervalo de tempo.
d) Procurar e listar profissionais com determinadas competências e que não tenham empresa associada.
e) Sugerir profissionais a empresas com base em competências. O resultado deve ser ordenado por experiência profissional.
f) Sugerir através do círculo de amizades de um determinado profissional quem pode facilitar a relação com uma determinada empresa, i.e., saber quem ou qual caminho determinado profissional deve seguir para chegar a uma determinada empresa.
g) Efetuar pesquisas com vários critérios, combinados por operadores booleanos (cf. *and*, *or*); Por exemplo, procurar profissionais com determinadas competências e sem empresa atual, e com uma experiência profissional de mais de 2 anos;
h) Procurar num determinado intervalo de tempo os encontros realizados ou agendados.

R12. Deverá ser criada uma interface gráfica para:

a) Visualizar os grafos de caminhos/localização e de amizades bem como as respetivas ligações, distinguindo de alguma forma os tipos de nós;
b) Gerir o grafo através da adição e remoção de nós/vértices e arcos/ramos bem como edição dos seus atributos;
c) A manipulação e gestão de toda a informação e das respectivas pesquisas deverá ser também suportada pela interface gráfica;

R13. Todos os dados referentes à aplicação e respectivas pesquisas deverão poder ser gravados em ficheiros de texto para consulta posterior dos utilizadores;

R14. Deverá ser ainda possível importar e exportar os dados dos modelos de dados (cf. Grafo e STs relacionadas) para ficheiros binários e de texto.

2. Objectivos

Pretende-se que os alunos modelizem e implementem a aplicação descrita, cumprindo todos os objectivos propostos. Deverão nomeadamente:

- Modelizar o problema através de diagramas de classes (UML), reutilizando estruturas de dados base (e.g. grafo, árvore, hashmap, etc.) e respectivos métodos/operações (cf. propriedades, travessias, pesquisas, etc.) que permitam representar e manipular os dados necessários;
- Implementar os algoritmos principais para a pesquisa e processamento da informação de acordo com as funcionalidades solicitadas;
- Implementar o modelo de dados OO utilizando a linguagem Java; em particular os requisitos funcionais enumerados e outros que se revelem úteis ou necessários;
- Implementar um conjunto de casos de teste que recorram a dados de *input* devidamente selecionados;
- Implementar uma interface gráfica que suporte a visualização e gestão de toda a informação (e.g., visualização da rede, gestão e edição dos nós, execução de pesquisas, etc.);
- Implementar a leitura e escrita de informação baseada em ficheiros de texto e binários.

3. Ficheiros e documentos a entregar

O projeto proposto deve ter uma implementação orientada aos objetos. Recomenda-se que todo o código (algoritmos e estruturas de dados) seja complementado com os comentários apropriados, que facilitem a compreensão do mesmo e a respectiva geração automática de documentação. Deve ser incluída uma explicação dos algoritmos implementados e uma menção ao desempenho dos mesmos assim como dos testes efetuados/implementados. Devem ainda ser realizados testes unitários que demonstrem o bom funcionamento das classes desenvolvidas.

Irão existir duas fases/momentos de entrega e avaliação presencial:

- Na fase 1 de entrega, serão considerados os requisitos R1 a R8.
- Na fase 2 serão considerados todos os requisitos mencionados.

NB: para AED2 alguns dos requisitos são opcionais:

- R1 (não é obrigatório o modelo de dados em UML)
- R12 (não é obrigatório a implementação de uma GUI em Java)
- R14 (não é obrigatório a utilização de ficheiros binários, apenas de texto).

As entregas devem incluir os seguintes componentes complementares:

i) **Modelos de classes** definidos para o projeto (ficheiro **zargo**);
ii) **Código fonte** (diretório **src**) das classes implementadas;
iii) **Ficheiros de teste** e respectivos dados *input/output* utilizados;
iv) **Documentação** do código (páginas de HTML geradas com *JavaDoc*).

Estes componentes do projeto devem ser entregues num **único ficheiro zip** através da plataforma de *elearning* até ao dia registado nos *assignments* da fase 1 e 2.

As duas fases do projeto deverão ser apresentadas e defendidas de “viva voz”, a meio do semestre (fase 1) e no final do semestre (fase 2), em datas a anunciar pelos docentes. Projetos entregues fora do prazo ou não apresentados presencialmente, não serão considerados para classificação.