Proposta de Trabalho

Rede de distribuição de energia eléctrica

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

Linguagens Programação II

Rui Silva Moreira
rmoreira@ufp.edu.pt
Christophe Soares
csoares@ufp.edu.pt
Beatriz Gomes
argomes@ufp.edu.pt
Alessandro Moreira
afmoreira@ufp.edu.pt

Algoritmia Estruturas Dados II

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

Março 2018

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 à gestão e distribuição de energia numa rede eléctrica. Mais concretamente, pretende-se que combinem a utilização de tabelas de símbolos e grafos para armazenar e gerir a informação necessária.

As estruturas do tipo *Symbol Table* (e.g. *Hashmaps*, *Redblack BST Trees,* etc.) deverão permitir armazenar e gerir a informação relativa às entidades que se pretendem manipular, como por exemplo: os contadores de energia eléctrica, os postos de transformação (PT), os equipamentos domésticos eléctricos e suas respectivas categorias de consumo, as fontes produtoras de energia, etc.

Deste modo, a estrutura do tipo grafo permitirá armazenar a informação relativa às ligações na rede eléctrica. Por exemplo, cada moradia poderá dispor de um contador que será um nó (vértice) do grafo e terá uma ligação (aresta) para a rede de energia eléctrica. Esta ligação será dirigidas e poderá caracterizar-se por vários pesos (e.g., distância, potência e tensão de energia eléctrica suportada, etc.). A aplicação deverá permitir gerir a informação dos nós e das ligações.

A rede eléctrica pode ser alimentada por diversas fontes de energia, como por exemplo: painéis solares, eólicas, barragens, etc. As habitações podem conter vários electrodomésticos (consumidores de energia) assim como painéis solares (geradores de energia). Desta forma, uma casa pode produzir energia suficiente para se tornar auto sustentável, e não necessitar de consumir energia distribuída na rede eléctrica. Uma casa pode também gerar mais energia do que consome, e nesse caso pode fornecer o excesso para a rede. A rede pode conter várias zonas. Por exemplo, uma zona habitacional é composta pelas contadores das casas e um posto transformador (nó agregador). Uma Zona deve recorrer à energia da rede, disponível a montante do PT, sempre que não conseguir gerar energia suficiente para alimentar as suas casas. Globalmente, a potência disponível não pode ser inferior à potência consumida nas vária zonas.

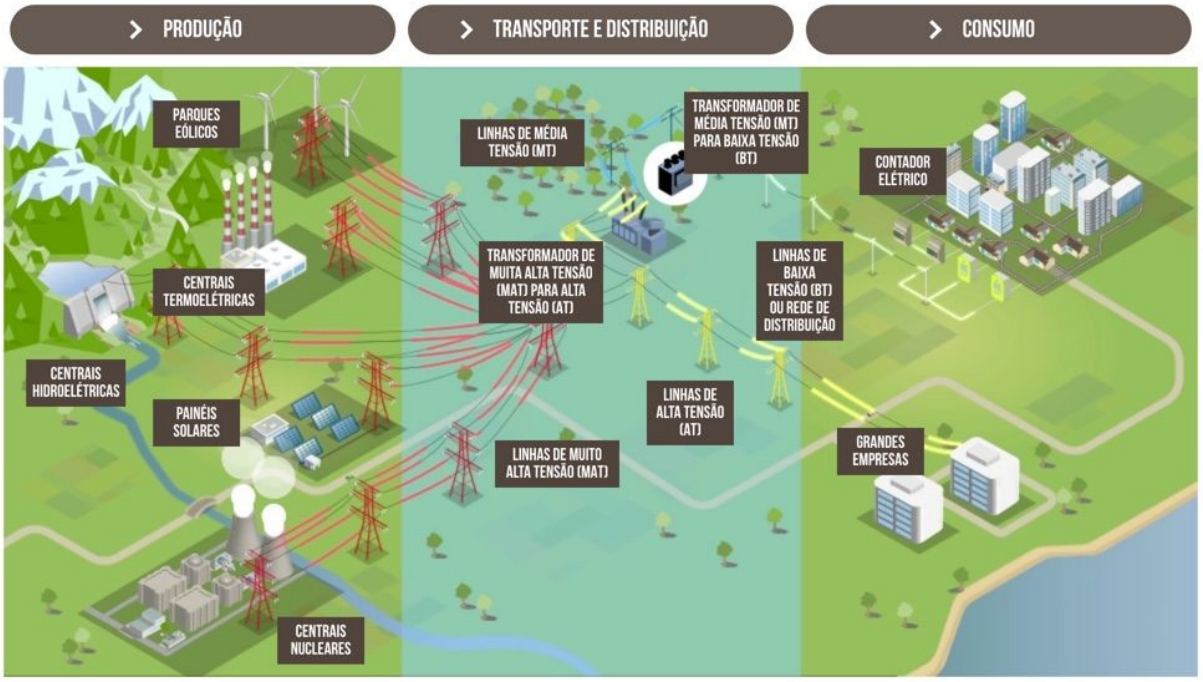

Figura 1 - Rede de distribuição de energia eléctrica (Guia da ERSE)

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 e árvores), 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, re-utilizando 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 (funções de teste) para as várias funcionalidades/requisitos implementados nessa API. Cada caso de teste deverá ser devidamente documentado numa função *static* caracterizada por um 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 ficheiro ou para a consola.

Em concreto a aplicação deverá cumprir os seguintes requisitos, que por questões de ordem prática deverão ser organizados em duas fases:

Fase I:

R1. O modelo de dados deverá permitir representar a rede de distribuição eléctrica, bem como toda a meta-informação associada às entidades identificadas anteriormente;

deverá ser ainda integrada a informação sobre os contadores (casas), postos de distribuição eléctrica, equipamentos de consumo eléctrico; a ligação entre os nós deverá refletir vários pesos (e.g., custos de consumo, distâncias espaciais, potência energética veiculada, etc.);

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, validar se a potência contratada pela casa suporta o número de aparelhos consumidores que nela são instalados.

R5. Deverão considerar a remoção e alteração de informações das estruturas de armazenamento (e.g. ST, BST, etc.) e nestes casos deve-se arquivar a informação. Por exemplo, ao remover um aparelho do sistema deve garantir-se a sua remoção total do sistema e respectivo arquivamento (em ficheiro ou estruturas auxiliares), ou alterar a potência de uma aparelho deve-se atualizar todo os sistema.

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

R7. Deve garantir-se o *output* (*dump*) de toda a informação para ficheiros de texto, isto é, de toda a informação de todas as STs e Grafos de maneira a permitir o restauro de dados.

R8. Devem implementar-se diversas pesquisas à base de informação como, por exemplo: quais as casas e respectivos aparelhos de todas as zonas ou de uma determinada zona; quais as casa com aparelhos produtores; top x de casas com maior auto-sustentabilidade (geram x% da energia que consomem); entre outras pesquisas.

R9. Deve ser possível consultar o histórico de consumo energético de um dado aparelho da casa.

Fase II:

R10. Cada nó ou vértice do grafo representa uma casa ou uma fonte de energia. O posto de transformação interliga as zonas à rede eléctrica. Cada nó deverá ter um conjunto de atributos/propriedades principais (e.g., potência contratada, capacidade em Watts de fornecimento de energia, cidade, localidade, etc.), entre outros atributos que possam vir a ser necessários;

R11. 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 dois nós do grafo;
b) Deverá poder-se verificar se o grafo de ligações da rede energética é conexo;
c) Selecionar um subgrafo e aplicar-lhe os mesmos algoritmos ou funcionalidades descritas;
d) Deverá associar-se a cada nó uma ou mais árvores para armazenamento de dados dos aparelhos e posteriormente permitir a manipulação conjunta de forma

a facilitar as pesquisas e as ações de gestão de informação; Poderá usar-se, por exemplo, uma árvore Red-Black (RB) para guardar a lista de logs dos aparelhos assim como toda a informação associada de forma a facilitar a pesquisa bem como a manipulação da informação acerca do perfil desses nós;

R12. Deverá ser possível efetuar e combinar vários tipos de pesquisas sobre a rede, como por exemplo:
   a) Listar informação detalhada do consumo de uma casa num determinado período de tempo;
   b) Procurar e listar os aparelhos e a informação arquivada dos mesmos numa determinada casa ou zona;
   c) Listar as casa com mais consumo registado.
   d) Identificar se a potência contratada pela casa está em conformidade com o número de aparelhos existentes. A cada inserção de um aparelho na ST de aparelhos da casa, verificar se existe a possibilidade (quando todos estiverem ligados) de ultrapassar a potência contratada da casa. Caso isso aconteça, podemos sugerir aparelhos geradores (e.g. Painéis Solares); Deve ser dada a possibilidade de não inserir o aparelho.
   e) Efetuar pesquisas com vários critérios, combinados por operadores booleanos (cf. *and*, *or*); Por exemplo, pesquisar casas na zona 1 *and* com mais de 5 aparelhos consumidores;
   f) Listar os caminhos possíveis e mais curtos de uma dada fonte de alimentação até uma determinada casa (e.g. por custo, potência, etc.); deverão aplicar-se algoritmos de cálculo de caminhos mais curtos em função dos pesos selecionados;
   g) Procurar num determinado dia e hora todos as casas que consumiram mais do que um dado valor de potência (W/h);
   h) Deverá ser criada uma interface gráfica para:
      i) Visualizar o grafo da rede e respectivas ligações bem como da árvore de aparelhos, distinguindo de alguma forma os tipos de nós;
      ii) Gerir o grafo através da adição de nós/vértices e arcos/ramos bem como edição dos seus atributos;
      iii) 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.

R15. Deverá ser possível identificar a(s) casa(s) mais eficiente(s), i.e., casas auto sustentáveis. Adicionalmente deve-se sugerir aparelhos geradores de energia com base em históricos de consumo e número de aparelhos consumidores da casa, nas casas menos eficientes.

R16. Deverá ser possível identificar Zonas Sustentáveis, determinar a percentagem de autonomia e perceber quais as necessidades de fornecimento de energia proveniente da rede.

R17. Calcular a MST (Árvore de Custos Mínimos) para realizar a manutenção a toda a rede.

R18. No caso da Zona não ser auto sustentável terá de recorrer a energia fornecida pela rede. Por isso, deverá poder calcular-se o caminho mais curto para o fornecimento da energia. As casas por omissão apenas têm uma ligação dirigida (PT -> Casa). Ao inserir um aparelho gerador, deve ser criada também a ligação dirigida inversa (Casa -> PT).

R19. Devem usar o algoritmo Ford-Fulkerson para calcular o fluxo da rede de um ou vários nós (produtores) para um ou vários terminais consumidores.
*(https://algs4.cs.princeton.edu/64maxflow/)

R20. Deverá ser possível consultar o estado da rede.

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:

- Na fase 1 de entrega, serão considerados obrigatórios os seguintes requisitos:
    - R1, R2, R3, R4, R5, R6, R7, R8, R9.

- Na fase 2 serão considerados todos os requisitos mencionados.

Em 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 do projeto (diretório **src** das classes implementadas e testadas);
iii) Ficheiros de teste e de 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.

No final do projecto deverá ser realizada uma apresentação/defesa de “viva voz” em datas a anunciar pelos docentes. Projetos entregues fora do prazo ou não apresentados presencialmente, não serão considerados para classificação. O projeto poderá ser elaborado com grupos compostos no máximo por três elementos.