Estruturas de Dados — Projeto Final

Prof. Samuel Plaça de Paula

Centro Universitário Senac — São Paulo, junho de 2020

Instruções para a realização e entrega

Este documento contém a descrição do que você deverá fazer para o projeto final da disciplina!

A tarefa aqui descrita deve ser feita em grupos de **até três pessoas**. Caso seja feita em trio, será exigida uma funcionalidade adicional, descrita adiante. Sua entrega consistirá de:

- Um programa em linguagem Java, sendo que apenas os arquivos fonte (arquivos `.java`, ou apenas a pasta `.src`, conforme a IDE utilizada) precisam ser entregues;
- Um pequeno relatório em texto, *para cada integrante do grupo*. Ou seja, essa parte é individual – entregas iguais serão desconsideradas. O que se espera do relatório é descrito na última parte da tarefa. Ele deve estar identificado com o nome da pessoa e pode ser no formato texto puro (`.txt`) ou `.pdf`. Não entregar em `.doc` ou similares.

Você fará a entrega na forma de um único arquivo `.zip` ou `.tar.gz` contendo tudo, e com o nome identificando os integrantes da dupla. Por exemplo, `samuel-e-suzana.zip`.

Visão geral

A ideia do programa é permitir a construção de um mapa (e a interação com ele). Esse mapa é representado internamente no código como um Grafo, estrutura de dados adequada a representar relacionamentos entre diferentes entidades. No caso, as entidades (vértices) são as "regiões" que desejamos representar no mapa, e fazemos com que elas sejam adjacentes no grafo se têm acesso direto uma à outra. Lembre-se que chamamos dois vértices de *adjacentes* (ou dizemos que são *vizinhos* um do outro) quando há uma *aresta* que os liga.

Por exemplo, se estamos pensando que nosso mapa é a planta baixa de uma casa ou outra edificação qualquer, os vértices são os cômodos, e dois vértices são adjacentes se os cômodos têm acesso direto um ao outro (exemplo: porta, escada).

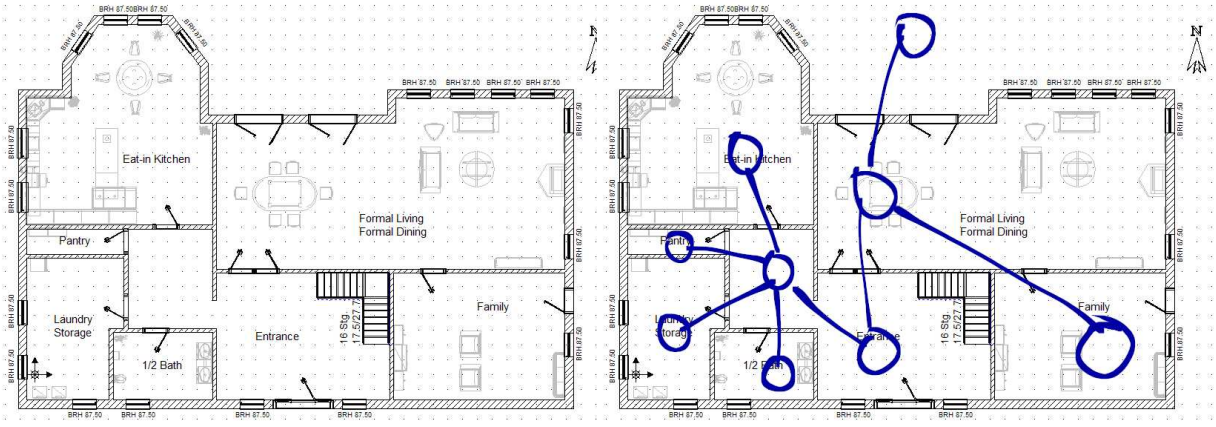


Figura 1: Um exemplo de planta, à esquerda (imagem: domínio público), e à direita o grafo obtido a partir dela.

Outro exemplo, ainda, seria o mapa do transporte metropolitano em São

Paulo: cada estação é um vértice, e dois vértices são adjacentes se uma vem em sequência da outra na mesma linha (exemplo: estações Sé e Liberdade da linha azul), ou se elas formam uma conexão entre linhas diferentes (exemplo: estações Paulista da linha amarela e Consolação da linha verde).

O programa não se importa exatamente com qual é o tipo de mapa representado: ele apenas se ocupa da representação em si, ou seja, do grafo.

A parte interativa se dá por conta de uma espécie de simulação: vamos supor que podemos colocar um personagem para caminhar pelo mapa. Podemos escolher sua posição e também pedir para que ele percorra caminhos. Além disso, deverá ser possível criar ou destruir arestas durante a execução do programa.

Sua tarefa

Sua tarefa é escrever um programa que forneça os comandos descritos a seguir e que os realize corretamente usando as estruturas de dados adequadas!

Você deverá usar as EDs vistas em sala de aula, em alguma das versões fornecidas no Blackboard, em arquivos como Grafo.java, Fila.java, etc. Você pode (e na verdade precisa) alterar algumas dessas classes para adicionar comportamentos necessários à sua aplicação. Sugere-se usar a versão mais recente de cada classe como base.

O seu programa deve ficar continuamente lendo comandos da entrada (teclado), **até que o usuário tecle `F` para finalizar**.

Parte 1: Comandos de criação do mapa

Os comandos de criação do mapa são: inicializar o mapa em si (começa sempre sem nenhuma aresta), adicionar arestas entre vértices do mapa e

adicionar item a um vértice do mapa.

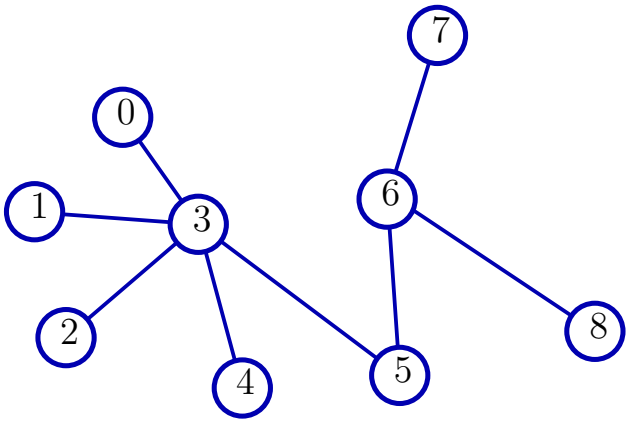
Cada vértice pode ter no máximo um item. Um item é simplesmente uma string como ‘‘`livro`’’, ‘‘`caneta`’’, ‘‘`bussola`’’. Se forem adicionados dois itens no mesmo vértice, prevalece apenas o último adicionado.

Qual solução podemos usar para guardar um atributo para cada vértice do grafo? Já fizemos algo assim em exercício anterior.

Os comandos são os seguintes:

- `G` seguido de um número inteiro positivo `N` cria um novo grafo com N vértices. Por exemplo: `G 10` cria um grafo com 10 vértices. Obs.: os vértices são rotulados como 0, 1, 2, ..., N-1.
- `A` seguido de dois rótulos `X` e `Y` cria uma aresta entre os vértices X e Y do grafo atual. Note que X e Y devem ser rótulos válidos (em um grafo com 10 vértices, o comando `A 7 10` não é válido pois o rótulo 10 não exite).
- `I` seguido de um rótulo `X` e uma string `S` adiciona o item S ao vértice X. Exemplo: `I 2 livro` adiciona o item “livro” ao vértice 2.

Vamos para um exemplo de sequência de comandos que cria um grafo. O grafo que queremos criar é aquele que representa a planta mostrada na Figura 1. Abaixo ele já conta com rótulos:



Podemos criar esse grafo com a seguinte sequência de comandos:

```
G 9
A 3 0
A 3 1
A 3 2
A 3 4
A 3 5
A 6 5
A 6 8
A 6 7
```

Para facilitar os testes do programa, você pode fazer com que, em vez de ter que carregar esses comandos apenas pela leitura do teclado, o programa também possa carregar um arquivo de texto com uma série de comandos como esse do exemplo acima.

A funcionalidade de ler a descrição do mapa a partir de um arquivo é obrigatória caso você faça o projeto em trio!

Parte 2: Comandos de interação

Nessa parte imaginamos que você pode colocar um personagem para andar no mapa (grafo) criado. Os comandos existentes são os seguintes: escolher o posicionamento do personagem, movimentar o personagem dentro do mapa, listar os itens coletados pelo personagem.

Obs.: você precisará de uma forma de saber em qual vértice o personagem está atualmente. Você também deverá ter uma estrutura de dados que guarda todos os itens que o personagem coletou até agora. Suponha que o personagem pode carregar no máximo 10 itens.

- `P` seguido de um rótulo `X`: posiciona o personagem no vértice X (deve ser um rótulo válido no grafo atual). Esse posicionamento muda imediatamente, sem que haja deslocamento dentro do mapa.

- `M` seguido de um rótulo `Y`: o personagem se desloca de sua posição atual até o vértice Y, que deve ser um rótulo válido no grafo atual. Esse deslocamento se dá **necessariamente pelo caminho de menor comprimento** da posição atual até Y. Além disso, o personagem irá coletar todos os itens que encontrar pelo caminho!

  O programa deve mostrar claramente a rota realizada pelo personagem.

- `C`. Exibe a coleção de itens do personagem. Deverá obrigatoriamente mostrar do mais recente para o mais antigo.

Lembre-se que, conforme conversamos, o array `pai` resultante de uma busca em largura consegue representar os caminhos mínimos de um vértice específico (a fonte da busca) até todos os outros.

Parte 3: Relatório

Cada pessoa da equipe deverá fazer um pequeno relatório individual, falando sobre as soluções encontradas, as dificuldades, e quais estruturas de dados foram empregadas e por quê.

Recapitulando

Abaixo, você pode conferir uma lista completa dos comandos necessários.

Lembre que em caso de trabalho em trio você deverá obrigatoriamente implementar a funcionalidade em que o programa lê um arquivo com a descrição do mapa (grafo).

É isso. Um ótimo trabalho! Boa sorte!

Comando	**Exemplo**	**Efeito do exemplo**
`G` seguido de tamanho	`G 10`	Cria grafo com 10 vértices.
`A` seguido de dois rótulos	`A 2 3`	Adiciona aresta entre vértices 2 e 3.
`I`, rótulo, string	`I 5 oculos`	Adiciona item `oculos` no vértice 5.
`P` seguido de rótulo	`P 3`	Posiciona personagem no vértice 3 (sem andar no mapa!)
`M` seguido de rótulo	`M 7`	Personagem desloca-se por caminho mínimo da posição atual até o vértice 7, coletando itens no caminho.
`C`	`C`	Mostra a coleção de itens do personagem, do mais recente até o mais antigo.
`F`	`F`	Finaliza a execução do programa.