Relatório Final
A3 - Pesquisa Implementada a um Guia Turístico

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Gonçalo Lobo - 201101834

João Soares - 201206052

Mafalda Falcão - 201204016

31 de Maio de 2015

Índice

Objetivo

O tema do projeto desenvolvido é *Pesquisa na Implementação de um Guia Turístico*. O objetivo é determinar um percurso turístico personalizado de acordo com os requisitos e preferências do utilizador.

Consideramos que temos um turista numa cidade desconhecida, com um determinado tempo limite, que pretende visitar o maior número de monumentos e locais que lhe despertam maior interesse (grau de interesse/importância), dispendendo o menos tempo possível.

Inicialmente temos várias cidades em que é conhecido os seus monumentos/locais de interesse e Hóteis, a distância entre cada um dos pontos no mapa e o tempo de visita aconselhado para cada um. Após a escolha da cidade a explorar, o utilizador determina qual o seu hotel e os pontos que está interessado em visitar e atribui uma importância a cada um deles.

Assim, o nosso objetivo é determinar um percurso turístico (percurso proposto) determinado quer por minimização do custo, quer por maximização do número e importância dos monumentos/locais a visitar.

Especificação

❖ Análise

Com o projeto desenvolvido, foi criada uma ferramenta que permite encontrar soluções para problemas do tipo TSP, adaptados a guias turísticos, usando técnicas de pesquisa. Este tipo de problema consiste em encontrar um itinerário ótimo entre vários pontos de interesse, de uma localidade, tendo em conta restrições temporais e as prioridades atribuídas a cada ponto.

A nível de restrições temporais o nosso problema tem em consideração o tempo total que dispõe para visitar a localidade e tenta maximizar o número de pontos de interesse visitados. A duração da visita e a localização de cada ponto de interesse são fornecidas. Antes de iniciar o percurso é solicitado ao utilizador que atribua prioridades de 1 a 10 (sendo que 1 corresponde à maior prioridade) a cada ponto de interesse.

Adicionalmente, a ferramenta desenvolvida possui uma interface gráfica que permite a um utilizador especificar a cidade que quer visitar, definir com que grau de prioridade e quais os pontos de interesse por onde deseja passar, a velocidade de deslocamento e o tempo que dispõe para viajar.

❖ Abordagem

De modo a resolver o problema proposto, primeiramente foi criada uma estrutura de dados representativa da cidade (City).

Esta estrutura de dados consiste num grafo fortemente conexo em que os nós correspondem a hóteis e a pontos de interesse. A inicialização das cidades e o carregamento da informação é efetuada através de ficheiros .csv com a seguinte estrutura:

Célula (0, 0) - Nome da cidade;

Célula (x, x) - Dados relativos ao ponto de interesse X (nomeadamente duração e prioridade);

Célula (x, y) - Distância entre o ponto de interesse X e Y.

Após geração do grafo da cidade foi implementado um método de pesquisa para retorno de soluções para caminhos possíveis. Este percurso é influenciado pelo tempo que o utilizador define para visitar a cidade e pela sua velocidade de deslocamento. Para alcançar este objetivo implementou-se um método de pesquisa informado, A*.

O algoritmo usado exige o uso de uma função heurística. Assim, foi criada uma função para estimar o custo heurístico de cada estado. Este custo é calculado somando o custo do ponto de menor custo ainda não visitado, mais o custo de viajar por todos os pontos ainda não visitadas (é usada uma heurística de minimum spanning tree), mais a distância de um ponto, ainda não visitada, ao hotel inicial.

Para cada ponto, partindo de um outro nó, o cálculo do custo tem em conta a distância a que este encontra a dividir pela velocidade de deslocamento, mais a duração da visita do nó, multiplicando tudo pelo valor atribuído à prioridade.

A implementação parte de duas listas, uma lista aberta com os nós ainda não visitados e uma lista fechada que é inicializada com o nó inicial (Hotel). Enquanto houver elementos na lista aberta ou o enquanto tempo limite não tenha sido ultrapassado, o algoritmo calcula, dentro dos nós abertos, aquele com menor custo partindo do último nó da lista fechada. Quando o encontra, desconta do tempo limite o tempo de percurso, adiciona o nó à lista fechada e remove-o da aberta. Inicia-se assim uma nova iteração usando o último nó encontrado como referência.

Desenvolvimento

Durante o desenvolvimento deste o projeto os elementos do grupo utilizaram o Sistema Operativo *Windows* e o IDE *IntelliJ*. A Linguagem de Programação escolhida foi *Java* pois é a linguagem com que estamos mais familiarizados e também pelas funcionalidades disponíveis tanto a nível de representação como a nível de estruturas de dados utilizados no algoritmo. Para a *GUI (Graphical User Interface)* servimo-nos do *SWING* do Java utilizando também uma biblioteca externa *GraphStream* para representação dos dados ao utilizador de um forma fácil de perceber. *GraphStream* é uma *Java library* para análise e modelação de grafos dinâmicos que permite gerar grafos, visuzalizá-los e alterar a sua representação. Utilizamos também a tecnologia colaborativa *GIT* como repositório.

Mais Informação sobre *GraphStream:* http://graphstream-project.org/

Estrutura da Aplicação

O programa desenvolvido dispõe de três módulos principais: um para a representação da GUI, um para a execução do algoritmo e outro com estruturas de dados adicionais.

Package GUI

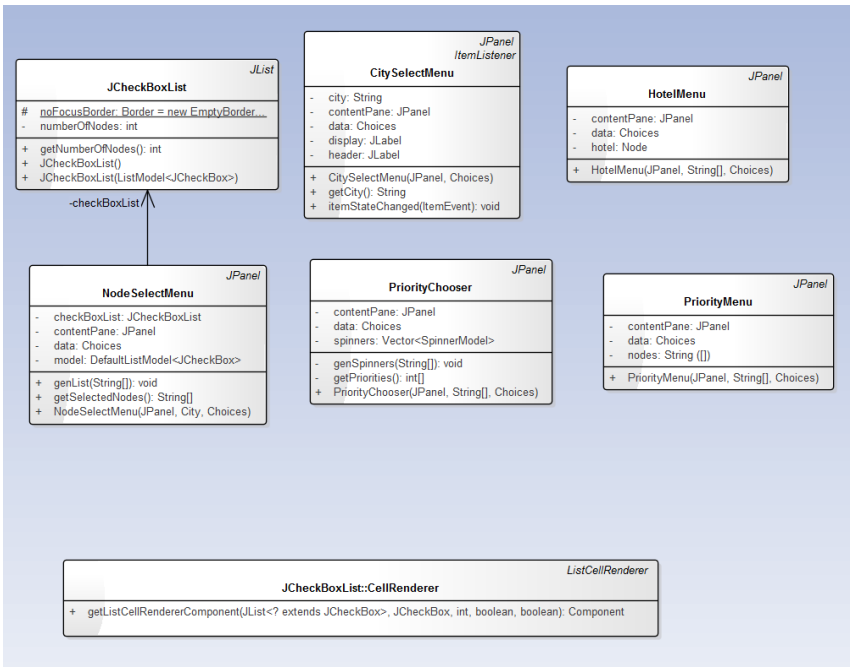
Este package é responsável pela implementação de parte gráfica - criação de menus e armazenamento de dados para utilização e interação com o package de iniciação/lançamento da aplicação (*package Main)*.

Este package é composto pelos seguintes ficheiros:

- ➔ JCheckBoxList.java: criação da estrutura de uma lista com *check boxes* de seleção para o menu de seleção dos locais a visitar;
- ➔ CitySelectMenu.java: menu inicial, que permite que o utilizador selecione a cidade que pretende visitar;
- ➔ NodeSelectMenu.java: menu que permite que o utilizador escolha quais os locais da cidade quer visitar;
- ➔ HotelMenu.java: menu que permite que o utilizador selecione o hotel onde está hospedado;

➔ PriorityMenu.java:menu que permite que o utilizador escolha se quer definir prioridades aos locais escolhidos, anteriormente.

➔ PriorityChooser.java: menu que permite que o utilizador defina as prioridades nos locais escolhidos, anteriormente.



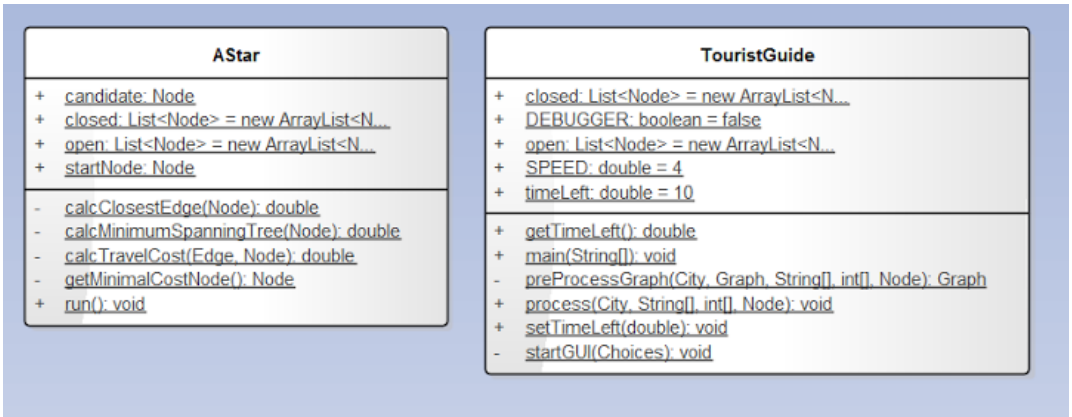
Package Main

Este package é responsável por iniciar a aplicação. Contém a implementação do algoritmo A* assim como as estruturas de dados necessárias para a execução deste.

Este package é composto pelos seguintes ficheiros:

- AStar.java: Classe que implementa o algoritmo.

- TouristGuide.java: Responsável por inicializar a aplicação, recebe as escolhas do utilizador na GUI e adapta a execução do algoritmo conforme. Contém também as variáveis globais Speed e timeLeft que representam a velocidade média a que o turista se desloca no percurso (em Km/hora) e a restrição temporal para o percurso (10 Horas). O método *preProcessGraph* recebe os nós escolhidos pelo utilizador e retira do grafo os os nós que não vão ser utilizados.

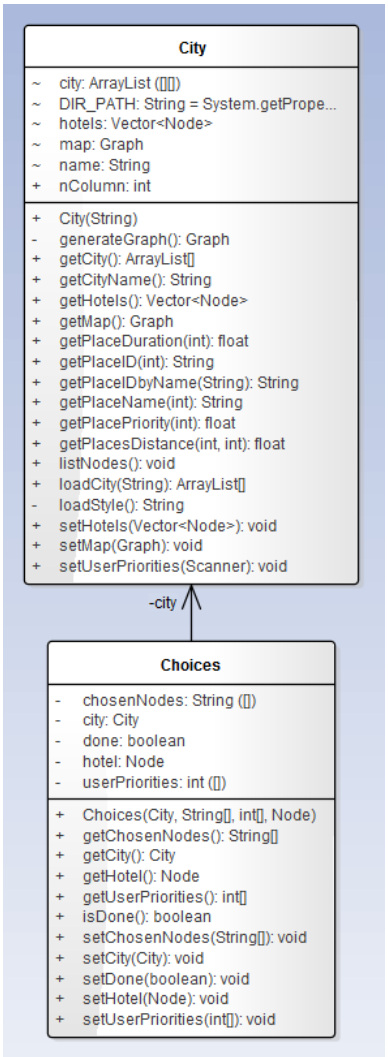


Package Logic

Este package complementa a execução do módulo *Main* com estruturas de dados adicionais.
Este package é composto pelos seguintes ficheiros:

- ➔ City.java: Classe que efetua o *parsing* dos ficheiros CSV's e cria o grafo representante da cidade.

- ➔ Choices.java: Estrutura utilizada para guardar as escolhas do utilizador na GUI para personalizar a execução do algoritmo.



Detalhes de Implementação

A aplicação está pronta a receber bases de dados para carregamento de informação para os grafos. Associadas a estas, são carregadas, para o visualizador, folhas de estilo personalizáveis e *sprites* representativas dos vários pontos de interesse a visitar.

No caso da cidade de Paris e de Londres, temos em conta dados reais (as distâncias entre os vários locais da cidade são reais - calculadas no *website* http://www.freemaptools.com/how-far-is-it-between.htm, que implementa a *API* do *Google Maps*. O tempo gasto relativo a cada local recebeu um tipo de tratamento análogo, visto que foram colocados os tempos médios de visita, com base nas pesquisas efetuadas).

Experiências

Experiência 1

- Tempo para visitar todos os nós sem prioridades definidas

Output:

Time left: 10.0 OPEN: [Clerigos, Piolho, SBento, Palacio] CLOSED: [Hotel] REF: Hotel Min Cost from Hotel to Clerigos with a cost of 3.0 Min Cost from Hotel to Piolho with a cost of 3.0 Min Cost from Hotel to SBento with a cost of 1.0 Min Cost from Hotel to Palacio with a cost of 6.0 Min ST from Hotel is 13.0 Minimal cost node is SBento with a cost of 14.0 --- Time left: 9.375 OPEN: [Clerigos, Piolho, Palacio] CLOSED: [Hotel, SBento] REF: SBento Min Cost from SBento to Clerigos with a cost of 7.0 Min Cost from SBento to Piolho with a cost of 2.0 Min Cost from SBento to Palacio with a cost of 3.0 Min ST from SBento is 12.0 Minimal cost node is Piolho with a cost of 14.0 --- Time left: 8.125 OPEN: [Clerigos, Palacio] CLOSED: [Hotel, SBento, Piolho] REF: Piolho Min Cost from Piolho to Clerigos with a cost of 7.0 Min Cost from Piolho to Palacio with a cost of 7.0 Min ST from Piolho is 14.0 Minimal cost node is Clerigos with a cost of 21.0 --- Time left: 4.875 OPEN: [Palacio] CLOSED: [Hotel, SBento, Piolho, Clerigos] REF: Clerigos Min Cost from Clerigos to Palacio with a cost of 3.0 Min ST from Clerigos is 3.0 Minimal cost node is Palacio with a cost of 6.0 --- Time left: 2.625 OPEN: [] CLOSED: [Hotel, SBento, Piolho, Clerigos, Palacio] REF: Palacio All nodes were visited. Going back to the hotel. Solution: (time remaining: 1.625 Hours) [Hotel, SBento, Piolho, Clerigos, Palacio, Hotel]	**Prioridades:** 1 - Palácio 1 - Piolho 1 - Clérigos 1 - São Bento **Análise Resultados:** Algoritmo segue sempre caminho com menor custo conforme previsto. Como não há prioridades diferentes a execução assemelha-se ao algoritmo de Djikstra para determinar o caminho mais curto.

Experiência 2

- Tempo para visitar todos os nós com prioridades definidas

Output:

Time left: 10.0 OPEN: [Clerigos, Piolho, SBento, Palacio] CLOSED: [Hotel] REF: Hotel Min Cost from Hotel to Clerigos with a cost of 15.0 Min Cost from Hotel to Piolho with a cost of 6.0 Min Cost from Hotel to SBento with a cost of 10.0 Min Cost from Hotel to Palacio with a cost of 6.0 Min ST from Hotel is 37.0 Minimal cost node is Piolho with a cost of 43.0 --- Time left: 8.5 OPEN: [Clerigos, SBento, Palacio] CLOSED: [Hotel, Piolho] REF: Piolho Min Cost from Piolho to Clerigos with a cost of 35.0 Min Cost from Piolho to SBento with a cost of 15.0 Min Cost from Piolho to Palacio with a cost of 7.0 Min ST from Piolho is 57.0 Minimal cost node is Palacio with a cost of 64.0 --- Time left: 5.25 OPEN: [Clerigos, SBento] CLOSED: [Hotel, Piolho, Palacio] REF: Palacio Min Cost from Palacio to Clerigos with a cost of 15.0 Min Cost from Palacio to SBento with a cost of 15.0 Min ST from Palacio is 30.0 Minimal cost node is Clerigos with a cost of 45.0 --- Time left: 3.0 OPEN: [SBento] CLOSED: [Hotel, Piolho, Palacio, Clerigos] REF: Clerigos Min Cost from Clerigos to SBento with a cost of 55.0 Min ST from Clerigos is 55.0 Minimal cost node is SBento with a cost of 110.0 --- Time left: 1.25 OPEN: [] CLOSED: [Hotel, Piolho, Palacio, Clerigos, SBento] REF: SBento All nodes were visited. Going back to the hotel. Solution: (time remaining: 1.125 Hours) [Hotel, Piolho, Palacio, Clerigos, SBento, Hotel]	**Prioridades:** 1 - Palácio 2 - Piolho 5 - Clérigos 10 - São Bento **Análise Resultados:** Algoritmo dá prioridade ao Palácio e a Piolho. Apesar de São Bento estar mais próximo, a sua baixa prioridade coloca-o como último ponto a visitar.

Experiência 3

- Tempo insuficiente para visitar todos os nós com prioridades definidas

Output:

Time left: 6.0 OPEN: [Clerigos, Piolho, SBento, Palacio] CLOSED: [Hotel] REF: Hotel Min Cost from Hotel to Clerigos with a cost of 15.0 Min Cost from Hotel to Piolho with a cost of 6.0 Min Cost from Hotel to SBento with a cost of 10.0 Min Cost from Hotel to Palacio with a cost of 6.0 Min ST from Hotel is 37.0 Minimal cost node is Piolho with a cost of 43.0 --- Time left: 4.5 OPEN: [Clerigos, SBento, Palacio] CLOSED: [Hotel, Piolho] REF: Piolho Min Cost from Piolho to Clerigos with a cost of 35.0 Min Cost from Piolho to SBento with a cost of 15.0 Min Cost from Piolho to Palacio with a cost of 7.0 Min ST from Piolho is 57.0 Minimal cost node is Palacio with a cost of 64.0 --- Time left: 1.25 OPEN: [Clerigos, SBento] CLOSED: [Hotel, Piolho, Palacio] REF: Palacio Min Cost from Palacio to Clerigos with a cost of 15.0 Min Cost from Palacio to SBento with a cost of 15.0 Min ST from Palacio is 30.0 Minimal cost node is Clerigos with a cost of 45.0 --- No more time to visit other nodes. Going back to the hotel. Time left: 0.25 Solution: (time remaining: 0.25 Hours) [Hotel, Piolho, Palacio, Hotel]	**Prioridades:** 1 - Palácio 2 - Piolho 5 - Clérigos 10 - São Bento **Análise Resultados:** Algoritmo verifica que não tem tempo para visitar mais pontos, retorna ao Hotel antes do tempo limite terminar.

Experiência 4

- Tempo insuficiente para visitar pontos

Output:

Time left: 2.0 OPEN: [Clerigos, Piolho, SBento, Palacio] CLOSED: [Hotel] REF: Hotel No time to visit any places. Solution: (time remaining: 2.0 Hours) [Hotel]	**Prioridades:** 1 - Palácio 2 - Piolho 5 - Clérigos 10 - São Bento **Análise Resultados:** Algoritmo não sai do Hotel.

Conclusões

Relativamente às experiências levadas a cabo, o grupo conclui que a solução apresentada neste relatório, apesar de não ser ótima, vai ao encontro de um bom equilíbrio entre a visita de um maior número de nós e a prioridade dos mesmos.

É opinião geral do grupo que as metas estabelecidas no enunciado foram cumpridas. Implementou-se uma interface gráfica para simplificar a utilização da aplicação e a visualização dos resultados.

Foram atingidos com sucesso todos os objetivos aos quais o grupo se propôs no Relatório Intercalar, e consideramos que o projeto contém todas as caraterísticas necessárias para cumprir as funções de um guia turístico automatizado.

Obviamente, é de realçar a nossa aprendizagem acerca dos algoritmos utilizados e definição de heurísticas como o principal objetivo cumprido, e aquele que nos será mais útil no futuro.

Melhoramentos

Dispondo de mais tempo para melhorar a qualidade do trabalho realizado, o grupo gostaria de implementar no projeto a possibilidade de selecionar diferentes unidades de medida para o tempo e distâncias. Adicionamelmente gostaríamos de carregar mais cidades, com mais pontos de interesse e com fotografias ilustrativas de modo a tornar a aplicação mais *user-friendly*.

Consideramos também a possibilidade de integração com a API do *Google Maps* para obter direcções GPS entre os locais de interesse obtidos, tornado o nosso projeto numa aplicação de Guia Turístico mais realista e pronta a ser lançada no mercado.

Bibliografia

- https://web.fe.up.pt/~eol/IA/1415/ia_.html
- http://docs.oracle.com/en/
- http://graphstream-project.org/doc/Tutorials/Getting-Started_1.0/

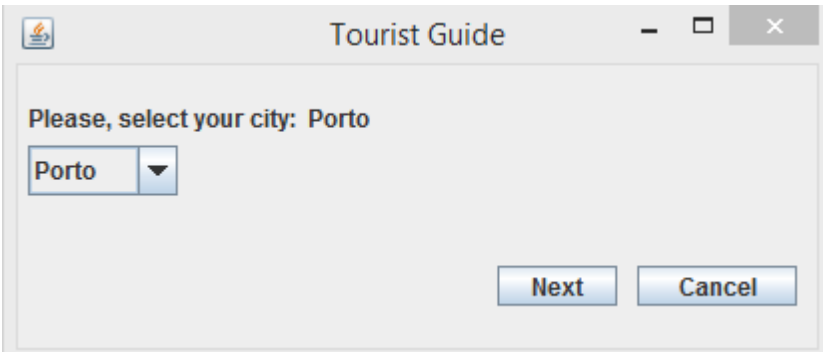
Avaliação

- Elementos do Grupo:
  - Gonçalo Lobo - ⅓ (aproximadamente 33%)
  - João Soares - ⅓ (aproximadamente 33%)
  - Mafalda Falcão - ⅓ (aproximadamente 33%)

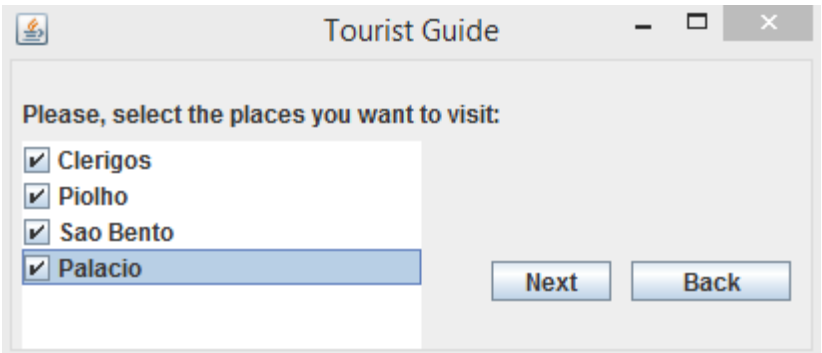
Apêndice

Em seguida, temos uma demonstração da *GUI* implementada, de modo a facilitar a utilização do programa.

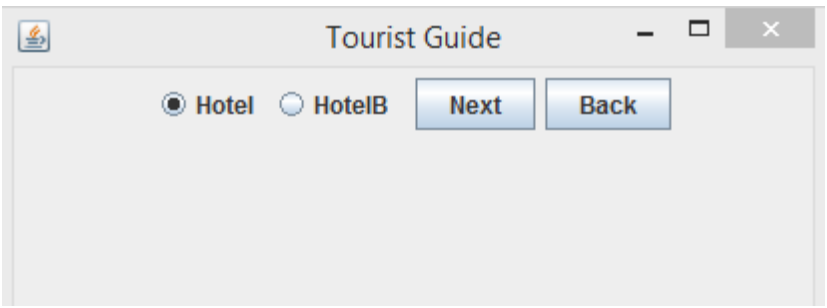
1. Selecionar a cidade pretendida;



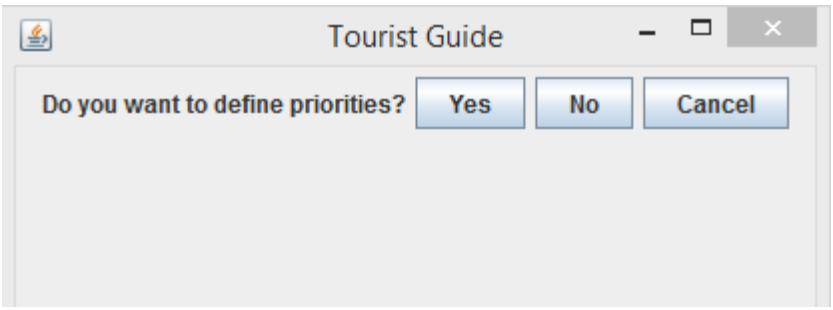
2. Selecionar os locais que quer visitar;



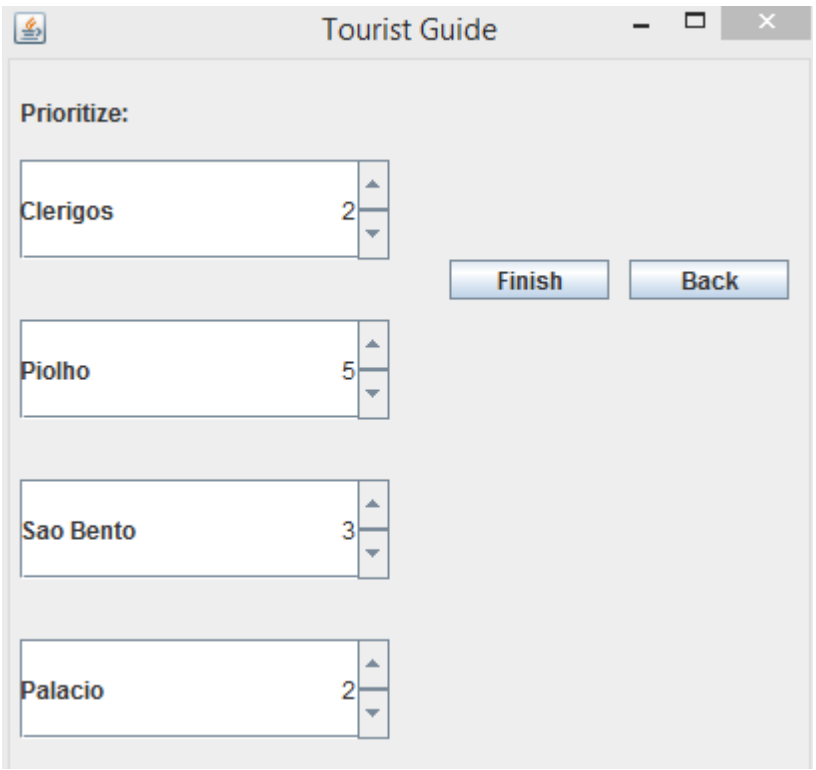
3. Selecionar qual o hotel onde está alojado;



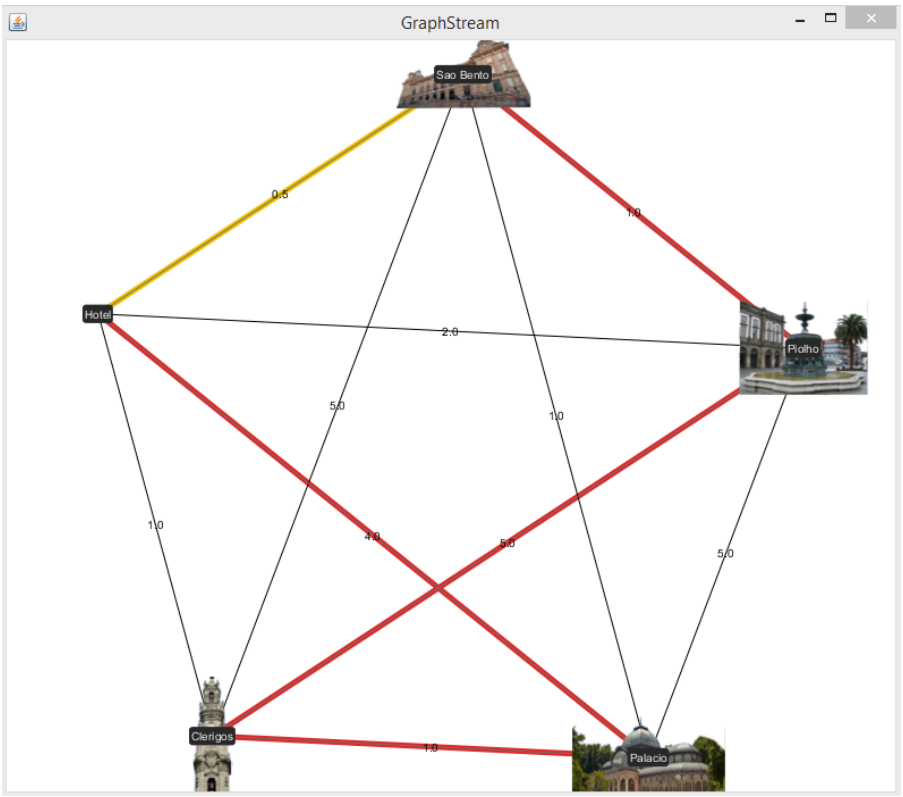
4. Escolher se quer (ou não) atribuir prioridades ao locais;

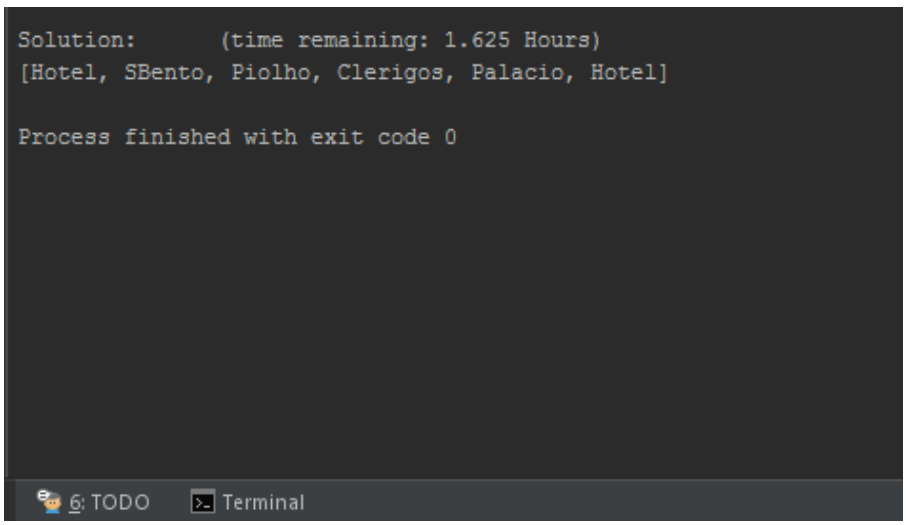


5. Caso queira definir prioridades, defina as prioridades no menu abaixo;



6. Caso tenha definido as prioridades carregue no botão *finish* e aparecerá o cenário em baixo. No caso de não ter escolhido definir prioridades, o cenário será idêntico.





(exemplo com prioridades definidas)