UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Algoritmos e Estruturas de Dados

2021-2022

Projecto 4 – Grafos

Introdução

No 4.º Projecto da cadeira de AED pretende-se explorar a implementação e a utilização de algoritmos baseados em Grafos.

Problema A: Implementação de MaxCycleMST (14 valores)

Para além do algoritmo de Prim, existem vários outros algoritmos que determinam uma Árvore Mínima Abrangente (Mininum Spanning Tree – MST). Um deles é baseado no teorema de que o arco de maior valor num ciclo nunca poderá fazer parte de uma Árvore Mínima Abrangente.

Este algoritmo, chamemos-lhe MaxCycleMST[1] funciona do seguinte modo:

Dado um grafo pesado não dirigido totalmente ligado G,

1) Criar um novo grafo que irá corresponder à MST
2) Adicionar um arco de cada vez à MST
    a. Ao adicionarmos um arco, testamos se a adição desse arco criou um ciclo
    b. Caso não exista um ciclo, voltamos a repetir o processo a partir de 2)
    c. Caso exista um ciclo, determinar o arco de maior peso nesse ciclo
    d. Apagar da MST o arco de maior peso do ciclo detetado
    e. Repetir o processo a partir de 2)

Poderão usar o algoritmo de deteção de ciclos lecionado nas teóricas, no entanto terão que fazer algumas alterações. O algoritmo lecionado foi desenhado para grafos dirigidos, e num grafo não dirigido um ciclo não conta se usarmos o arco por onde chegámos para voltar para trás (porque não podemos repetir arcos) e temos que fazer esse teste. Outra alteração necessária é guardar os vértices e arcos que fazem parte do caminho, ou o arco de maior peso que faz parte do caminho para se conseguir determinar qual o arco de maior peso.

Existem algumas otimizações que podem ser feitas ao algoritmo tendo em conta as propriedades de Grafos e Árvores Mínimas Abrangentes, e tendo em conta a necessidade de obter o arco de maior peso de um ciclo. Cabe a cada aluno determinar qual a melhor forma de implementar o MaxCycleMST, desde que cumpram a ideia principal de adicionar arcos, e apagar o arco de maior valor de um ciclo, caso apareçam ciclos.

Pode usar, e assumir que irão estar definidos no *Mooshak*, as classes *UndirectedEdge* e *UndirectedWeightedGraph*.

[1] Este algoritmo não é tão eficiente como o algoritmo de Prim. No entanto os algoritmos para cálculo da MST que são mais eficiente ou já estão implementados no Livro, ou então são muito complicados (ex: algoritmo de Chazelle).

Implemente a classe *MaxCycleMST* dentro da package *aed.graphs*. Poderá implementar outros métodos privados auxiliares caso assim o entenda, mas deverá implementar obrigatoriamente os seguintes métodos públicos:

MaxCycleMST – Implementa um algoritmo para calcular uma MST (Mininum Spanning Tree) a partir de um grafo não dirigido pesado totalmente ligado.		
	`MaxCycleMST(UndirectedWeightedGraph g)`	Cria uma instância do algoritmo para determinar a MST para o grafo recebido. Este grafo não poderá ser alterado pelo algoritmo.
`UndirectedEdge`	`determineMaxInCycle(UndirectedWeightedGraph g)`	Este método deteta se existe um ciclo no grafo g recebido. Caso exista um ciclo retorna o arco com maior peso nesse ciclo. Caso não exista ciclo retorna *null*.
`UndirectedWeightedGraph`	`buildMST()`	Este método calcula e retorna um grafo pesado não dirigido que corresponde a uma árvore mínima abrangente para o grafo originalmente recebido no construtor.
`UndirectedWeightedGraph`	`getMST()`	Este método devolve a árvore mínima abrangente construída no passo anterior. Caso ainda não tenha sido construída (porque o método acima não foi chamado) retorna *null*.
`void`	`main(string[] args)`	Método main, que deverá ser usado para testar os métodos acima.

Objectivos/Requisitos Técnicos

Poderão fazer alterações e otimizações ao algoritmo apresentado, no entanto o vosso algoritmo terá de consistir numa série de iterações em que cada iteração corresponde a apagar o arco de maior peso de um ciclo existente no grafo.

Testes à eficiência

Determine de forma analítica qual a complexidade temporal do método buildMST tendo em conta o número de Vértices V e Edges E do grafo. Escreva esta análise como comentário no método main do código entregue.

Utilize métodos empíricos, e ensaios de razão dobrada para determinar a eficiência do método buildMST e compare com a análise teórica efetuada.

Escreva um sumário dos testes e resultados, e uma breve descrição como comentários no ficheiro Java a submeter.

Método main

Este método deverá implementar os testes usados para testar e comparar os métodos acima. Embora este método não seja validado de forma automática pelo Mooshak será tido em consideração na validação do projeto, e contará para a nota final do mesmo.

Implemente a classe *MaxCycleMST* no ficheiro *MaxCycleMST.java*. Submeta **apenas o ficheiro *MaxCycleMST.java*** no Problema A.

Problema B: Aplicação de árvores mínimas abrangentes (6 valores)

Existem inúmeras aplicações de árvores mínimas abrangentes. Um dos exemplos mais ilustrativos é a criação de redes de infraestrutura com custo reduzido. Suponha que numa cidade existem N casas, onde cada casa necessita de acesso a água. Para uma casa *i* ter acesso a água pode-se construir diretamente um poço na casa, com custo dado por *well[i]*, ou alternativamente pode-se construir canalização entre as casas *i* e *j* com custo dado por *c[i][j]*. Uma casa pode receber água se tiver um poço ou se houver um caminho de canalização da casa até outra casa que tenha um poço.

Neste problema iremos implementar um algoritmo para determinar qual a forma ótima de fornecer água a todas as casas, de modo a gastar o mínimo de dinheiro possível. Para resolver este problema, pode criar-se um grafo pesado não dirigido com N+1 vértices (N vértices representam as N casas, o vértice extra representa o rio subterrâneo onde todos os poços se vão ligar. Adiciona-se um arco ao grafo para cada par de casas que se possam ligar e que tenham custo c[i][j] (estes arcos representam os canos). Adiciona-se também um arco entre cada casa e o vértice extra que representa o rio com custo *well[i]* (representando a opção de se abrir um poço na casa).

Uma árvore mínima abrangente para este grafo é uma solução ótima para o nosso problema.

Implemente a classe *PipeCalculator* que implementa um algoritmo para determinar soluções para problemas deste género. A classe *PipeCalculator* deverá ser definida dentro da *package* aed.graphs. Poderá implementar outros métodos privados auxiliares caso assim o entenda, mas deverá implementar obrigatoriamente os seguintes métodos públicos:

PipeCalculator – Implementa um algoritmo para calcular a solução ótima para um problema de canalização de uma cidade.		
	`PipeCalculator(int n, float[] well, float[][] costs)`	Cria uma instância do algoritmo para resolver um problema de canalização com n casas, custos de criação de furos, e custos de canalização.
`UndirectedWeightedGraph`	`createGraph(int n, float[] well, float[][] costs)`	Cria um grafo que representa um problema de canalização.
`UndirectedWeightedGraph`	`calculateSolution(UndirectedWeightedGraph g)`	Dado um grafo que representa um problema de canalização, este método calcula a solução ótima para o problema, na forma de uma MST.
`UndirectedWeightedGraph`	`calculateSolution()`	Calcula a solução ótima para o problema recebido no construtor, na forma de uma MST.
`UndirectedWeightedGraph`	`getMST()`	Este método devolve a árvore mínima abrangente construída no passo anterior. Caso ainda não tenha sido construída (porque o método acima não foi chamado) retorna *null*.
`void`	`main(string[] args)`	Método main, que deverá ser usado para testar os métodos acima.

Objectivos/Requisitos Técnicos

Deverá usar o algoritmo de procura MaxCycleMST definido no problema A para determinar a MST para o problema recebido. Caso não tenha implementado o problema A, pode usar a

implementação do algoritmo LazyPrim (data nas teóricas) para cálculo da árvore mínima abrangente.

Método main

Este método deverá implementar os testes usados para testar e comparar os métodos acima. Embora este método não seja validado de forma automática pelo Mooshak será tido em consideração na validação do projeto, e contará para a nota final do mesmo. Não é necessária a implementação de testes de eficiência para este problema.

Implemente a classe PipeCalculator no ficheiro PipeCalculator.java. Submeta apenas o ficheiro PipeCalculator.java no Problema A.

Condições de realização e avaliação

O projecto deve ser realizado individualmente. Projectos iguais, ou muito semelhantes, originarão a reprovação na disciplina. O corpo docente da disciplina será o único juiz do que se considera ou não copiar num projecto.

Avaliação em laboratório

O 4.º Projeto tem uma componente explícita de avaliação durante o laboratório. Durante o vosso turno de laboratório a que estão inscritos, na semana de 17 a 21 de Dezembro, terão de participar de forma obrigatória. A vossa participação será avaliada, e poderei pedir-vos para implementarem (ou discutirem a implementação) de uma parte do projeto. Esta avaliação vale **3 valores da nota final** do projeto. Quem não comparecer a esta avaliação terá 0 neste componente do projecto.

Mooshak – verificação automática

O código do projecto deverá ser entregue obrigatoriamente por via electrónica, através do sistema Mooshak, **até às 23:59 do dia 21 de Janeiro.** As validações terão lugar na semana seguinte via Discord. **A avaliação e correspondente nota do projecto só terá efeito após a validação do código pelo docente.**

A avaliação da execução do código é feita automaticamente através do sistema Mooshak, a partir do início da próxima semana, usando vários testes configurados no sistema. O tempo de execução de cada teste está limitado, bem como a memória utilizada. Não é necessário o registo para quem já se registou no 1.º projecto, podendo usar o mesmo username e password. Para quem ainda não se tenha registrado, poderá fazê-lo usando o seguinte link:

http://deei-mooshak.ualg.pt/~dshak/cgi-bin/getpass-aed21

Deverão introduzir o vosso número de aluno e submeter. Irá ser gerada uma password que vos será enviada por email. Caso não recebam a password, verifiquem a vossa caixa de spam, e entrem em contacto com o corpo docente.

Uma vez criado o registo poderão fazer login no sistema Mooshak com o vosso número de aluno e a password recebida. O link para o sistema Mooshak é o seguinte:

http://deei-mooshak.ualg.pt/~dshak/