UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA

ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO

TEORIA DOS GRAFOS - COS242

Trabalho 2

2019.2

Alunos:

Ana Paula Falcão
Andrew Cruz

Professor:

Daniel Ratton

15 de outubro de 2019

1 Decisões de projeto

Neste trabalho, a dupla precisou estender o que havia sido implementado para suportar grafos com pesos e oferecer novas funcionalidades como encontrar as menores distências entre dois vértices - utilizando o algoritmo de Dijkstra -, encontrar a árvore geradora mínima de um grafo e calcular a excentricidade do mesmo. O ponto inicial de mudança foi a forma como lemos o arquivo de entrada e montamos o grafo: agora é necessário verificar se todos as arestas possuem pesos e se eles são positivos, uma vez que o algoritmo de Dijkstra só funciona corretamente para esse caso. Se estiver tudo correto, a lista ou matriz é criada. A matriz armazena o peso da aresta na posição correspondente e a lista de adjacências não salva apenas os vizinhos do vértice, mas também o peso da aresta entre eles. Outro ponto importante foi a criação de uma estrutura heap, em um arquivo separado, para apoiar a implementação otimizada dos algoritmos que necessitam dessa estrutura. Além disso, vale ressaltar que o conjunto de vértices explorados foi implementado sobre a estrutura de dados Set do Python. Essa estrutura permite que a existência de um elemento dentro dela seja verificada com complexidade O(1), enquanto a mesma operação na lista possui complexidade O(n), ambas no caso médio. O preço por esse aumento é o consumo de memória: o Set ocupou mais de três vezes o espaço ocupado por uma lista.

2 Implementação

2.1 Dijkstra

A implementação utiliza uma estrutura de dados que foi implementada no arquivo heap.py. Inicialmente, a estrutura é criada vazia e o método buildHeap atribui peso infinito a todas os vértices do grafo. Em seguida o vértice inicial é colocado no heap com distância 0 (atualmente o algoritmo só sabe que a distância do vértice inicial até ele mesmo é igual a zero) e todos os outros vértices recebem distância infinita. O Set de explorados é criado, e o primeiro elemento é adicionado a ele. Enquanto houver elementos no heap, o algoritmo pega um e verifica se ele já foi explorado - operação O(1) com Set. Se o elemento não tiver sido explorado, seus vizinhos são analisados. Aqui, verifica-se se o peso total do vértice inicial até o atual é maior que o encontrado agora. Se sim, atualiza-se o caminho mínimo e o peso total.

2.2 Prim

A implementação do algoritmo de Prim utiliza basicamente as mesmas estruturas de dados, no entanto, o objetivo agora é encontrar MSTs, verificando os menores custos para chegar a cada vértice do grafo. Para isso, criamos uma lista contendo as distâncias mínimas e outra contendo os pais de cada vértice, e ambas são atualizadas a medida que o algoritmo roda. Um elemento com custo infinito no heap significa que não há caminho até ele, ou seja, o grafo é desconexo. Assim, o algoritmo é encerrado com os pesos que foram encontrados até o momento. A verificação de menores custos e se o vértice encontra-se no conjunto de explorados é similar ao Dijkstra.

2.3 Excentricidade

A excentricidade foi calculada rodando o algoritmo de Dijkstra para cada vértice e pegando o máximo dentre os valores encontrados.

2.4 Maiores Graus Prim

Nesse caso, nós percorremos uma lista que contém os pais de cada vértice na árvore geradora. A cada vez que um vértice é identificado como pai de outro nessa lista, uma lista auxiliar é atualizada acrescentando um à posição correspondente do vértice pai. Esse processo é repetido três vezes.

3 Resultados

De acordo com o que foi pedido no enunciado do trabalho, seguem as tabelas com os valores pedidos referentes aos grafos disponíveis no site da disciplina:

	Relação entre os vértices				
	1 e 10	1 e 20	1 e 30	1 e 40	1 e 50
Grafo 1	31	38	48	25	30
Grafo 2	9	10	9	8	7
Grafo 3	5	10	16	8	7
Grafo 4	58	129	102	174	147

Cabe ressaltar aqui, que não conseguimos rodar o grafo 5. Primeiramente, acreditavámos ser problema de memória com o nosso computador, e portanto, colocamos para rodar em um computador

de 32GB. O resultado continuou sendo de Memory Error. Aumentamos a quantidade de memória virtual, mas o erro persistiu. Atribuimos o ocorrido ao uso do Set, pois, apesar de melhorar nosso tempo (que pode ser um ponto fraco da linguagem Python, como foi discutido no primeiro trabalho), o Set consumiu bastante memória. Mesmo assim, acreditamos que não seria o suficiente para não rodar, então algo no nosso algoritmo está consumindo bastante da nossa memória, e vamos averiguar o que pode ser.

Para o cálculo da excentricidade, foi utilizado Dijkstra:

	Excentricidade do vértice:					
	10	20	30	40	50	Tempo médio (segundos)
Grafo 1	44	55	61	42	48	0.038036
Grafo 2	54	55	53	50	52	0.962927
Grafo 3	58	58	64	57	60	79.744.283
Grafo 4	1650	1721	1628	1700	1671	99.025.101

Aqui em Prim, cabe ressaltar que o grafo de rede de colaboração é desconexo, portanto, não existe uma MST no grafo como um todo. Portanto, consideramos apenas a componente conexa de interesse.

	Prim	
	Peso	Tempo (segundos)
Grafo 1	3856	0.037574
Grafo 2	31663	0.784466
Grafo 3	312289	12.220155
Grafo 4	7668215	43.785371

Sobre a relação entre Edsger W. Dijkstra e J. B. Kruskal:

Distância: 3.48036845488905

Caminho: [’Edsger W. Dijkstra’, ’John R. Rice’, ’Dan C. Marinescu’, ’Howard Jay Siegel’, ’Edwin K. P. Chong’, ’Ness B. Shroff’, ’R. Srikant’, ’Albert G. Greenberg’, ’J. B. Kruskal’]

Sobre a relação entre Edsger W. Dijkstra e Jon M. Kleinberg:

Distância: 2.7069936175564644

Caminho: [’Edsger W. Dijkstra’, ’A. J. M. van Gasteren’, ’Gerard Tel’, ’Hans L. Bodlaender’, ’Di-

mitrios M. Thilikos', 'Prabhakar Ragde', 'Avi Wigderson', 'Eli Upfal', 'Prabhakar Raghavan', 'Jon M. Kleinberg']

Sobre a relação entre Edsger W. Dijkstra e Éva Tardos:

Distância: 2.7535141793573357

Caminho: ['Edsger W. Dijkstra', 'A. J. M. van Gasteren', 'Gerard Tel', 'Hans L. Bodlaender', 'Jan van Leeuwen', 'Mark H. Overmars', 'Micha Sharir', 'Haim Kaplan', 'Robert Endre Tarjan', 'Andrew V. Goldberg', 'Serge A. Plotkin', 'Éva Tardos']

Sobre a relação entre Edsger W. Dijkstra e Daniel R. Figueiredo:

Distância: 2.9428308695367855

Caminho: ['Edsger W. Dijkstra', 'John R. Rice', 'Dan C. Marinescu', 'Chuang Lin', 'Bo Li', 'Y. Thomas Hou', 'Zhi-Li Zhang', 'Donald F. Towsley', 'Daniel R. Figueiredo']

Os três vértices de maiores graus da MST com seus respectivos graus são:

[['Wei Li', 171], ['Wei Wang', 146], ['Wei Zhang', 143]]

Os vizinhos de Edsger W. Dijkstra na MST são:

['Carel S. Scholten']

Os vizinhos de Daniel R. Figueiredo na MST são:

['Alexandre A. Santos', 'André C. Pinho']

4 Código

O trabalho pode ser encontrado no github, com todos os demais resultados, pelo link: `https://github.com/anafalcao/Graph_Theory/`