Teoria dos Grafos
TP2

Fernando Bruzzi – DRE: 123360519
Pedro Lima – DRE: 123492138

Outubro 2024

1 Introdução

A fim de aplicar os conhecimentos desenvolvidos em aula e aprender de forma mais profunda sobre como funcionam os algoritmos estudados, foram adicionados novos métodos as classes desenvolvidas na etapa anterior do trabalho para a representação de grafos. A linguagem de programação usada foi C++, devido a possibilidade de processar altos volumes de dado sem ter muitos problemas com a memória, além do fato dela ser uma linguagem orientada a objetos, o que permite um encapsulamento dos métodos implementados. Qualquer dúvida relacionada a implementação e ao algoritmo foi resolvida a partir da consulta dos livros da disciplina e de debates com os colegas e com o monitor.

2 Decisões de Implementação

- Para a representação de pesos na matriz foi feita a troca de uma matriz de valores booleanos para uma matriz de floats, já que o peso das arestas que ligava os vértices pode ser qualquer número real. Sendo que, o valor da aresta que liga u até v ocupava a posição M[u-1][v-1]. Caso não existisse uma aresta que ligasse os vértices u e v, a posição M[u-1][v-1] assumia o valor INFINITY de um float, um macro do C++.
- Para a representação de pesos no vetor de adjacência, agora as posições V[u-1] que correspondem aos vértices que se ligam ao vértice u não guardam inteiros, mas sim um tipo pair¡int, float¿, onde o primeiro elemento do pair corresponde ao vértice que se liga a u, e o segundo elemento corresponde ao peso dessa ligação.
- É importante destacar que os métodos correspondentes a BFS e DFS não foram alterados, e têm um caráter somente de exploração, o que é necessário para descobrir componentes conexas. Assim, a sua utilização deve ser feita com cuidado para trazer resultados corretos.
- Métodos como o diâmetro do grafo, bem como a distância entre dois vértices foram alterados para fornecer os resultados com base no algoritmo de Dijkstra. Sendo que, devido a maior velocidade do Dijkstra com a utilização de um heap, esse foi o utilizados em ambos métodos mencionados.
- Para a implementação do algoritmo de Dijkstra utilizando um vetor, foi realizada a tradução do código presente no livro [1, p. 301] para C++.
- O heap utilizado no trabalho foi feito a partir da alteração da priority queue padrão do C++ para ela funcionar como um heap mínimo onde a sua chave era o custo de um vértice e o elemento junto a chave era o vértice.
- Devido a implementação da matriz de adjacência e do vetor de adjacência separadamente, foram definidos os operadores "Compare"duas vezes para garantir o funcionamento correto do código de cada uma das classes, mesmo que a outra não fosse importada junto.
- Para a implementação do algoritmo de Dijkstra utilizando um heap mínimo foi feita a adaptação do pseudocódigo apresentado em aula. Optamos por não arcar com a complexidade de elaborar um heap que pudesse ter o conteúdo da sua chave(custo) atualizado a partir do acesso ao elemento que ela guarda(vértice), o que seria necessário quando o algoritmo de Dijkstra encontrasse um novo caminho(mais barato) a um vértice que já estivesse no heap. Ao invés disso, adicionamos um vértice sempre que ele tem um custo novo mais barato descoberto e garantimos a corretude do algoritmo porque esse vértice será obrigatoriamente explorado com o custo menor, devido a estrutura do heap mínimo. Isso só foi possível devido a

utilização de um vetor secundário de explorados que indicava que só deveríamos permitir mudanças ou operar sobre um vértice que saia do topo do heap, quando esse ainda não tivesse sido explorado. Ainda que esse processo gastasse mais memória, a redução de tempo combinada com a representação por vetor de adjacência(que gasta menos memória) compensava. Mais detalhes sobre o tempo de execução estão descritos no estudo de caso.

3 Estudo de Caso

Para cada questão levantada no estudo de caso, apresentaremos uma tabela que será comentada conforme for necessário. Vale destacar que, devido ao custo elevado de memória para representar os grafos com uma matriz, muitas linhas se encontram com um "X"marcado para indicar que esses dados não foram obtidos. Além disso, destacamos que os dados utilizados para o estudo de caso encontram-se disponíveis em um arquivo em CSV que busca juntar informações sobre todos os grafos nas diferentes implementações. Sendo que, até o arquivo referente ao grafo 2 o cálculo das distâncias foi feito tanto com o Dijkstra com heap, quanto com o Dijkstra com vetor. Infelizmente, o tempo de execução do Dijkstra com vetor era inviável a partir do grafo 3.

3.1 Questão 1

Tempo Médio Dijkstra [em nanosegundos]				
	Matriz		Vetor de Adjacência	
	Heap	Sem Heap	Heap	Sem Heap
Grafo 1	$6.42e^{+6}$	$5.35e^{+8}$	$1.12e^{+5}$	$2.46e^{+8}$
Grafo 2	$4.66e^{+7}$	$4.10e^{+9}$	$4.33e^{+}5$	$1.92e^{+9}$
Grafo 3	X	X	$3.64e^{+6}$	$2.80e^{+10}$
Grafo 4	X	X	$6.27e^{+7}$	∞
Grafo 5	X	X	$6.22e^{+8}$	∞

Tabela 1: Tabela de Tempos Médios de Dijkstra para Matriz e Vetor de Adjacência com implementação de Heap e sem Heap

Assim como observado no trabalho anterior, a maior parte da coleta dos dados foi inviável utilizando a representação em matriz, devido ao seu alto custo de memória. Todavia, com a utilização da representação em vetor de adjacência a execução dos algoritmos ocorreu sem grandes problemas. É importante destacar que, como esperado, o tempo de execução do algoritmo quando utilizávamos a estrutura de heap para guardar o custo ocorreu foi muito mais rápido. Podemos entender esse resultado analisando a complexidade do algoritmo de Dijkstra com heap e sem heap. Sabemos que para o caso sem heap teremos uma complexidade de $O(n^2)$ já que precisamos passar por todos os vértices pelo menos uma vez e olhar seus vizinhos, no caso de um grafo conexo. Já no caso da implementação a partir do heap, teremos uma complexidade de $O((n+m)log(n)$.

3.2 Questão 2

Distância entre o vértice 10 e 20		
	Com Vetor	Com Heap
	Distância	Distância
Grafo 1	2.38	2.38
Grafo 2	1.81	1.81
Grafo 3	X	0.8
Grafo 4	X	2.66
Grafo 5	X	17.34

Caminho mínimo 10-20 Grafo 1:
10 → 2038 → 5094 → 2875 → 4746 → 8892 → 3578 → 2502 → 4627 → 4252 → 9853 → 9285 → 1034 → 20

Caminho mínimo 10-20 Grafo 2: 10 → 20401 → 10525 → 15491 → 15369 → 18571 → 1573 → 3448 → 8238 → 3358 → 13344 → 9448 → 23467 → 3907 → 20

Caminho mínimo 10-20 Grafo 3: 10 → 2858 → 35293 → 3905 → 24445 → 39345 → 61279 → 41321 → 70540 → 26049 → 84450 → 68904 → 87862 → 86778 → 27694 → 16608 → 86355 → 15760 → 20

Caminho mínimo 10-20 Grafo 4: 10 → 532245 → 877016 → 554617 → 430107 → 437166 → 805646 → 841402 → 663364 → 655748 → 634947 → 318070 → 807260 → 20

Caminho mínimo 10-20 Grafo 5: 10 → 7622850 → 4615833 → 6528266 → 1295458 → 7103730 → 8790605 → 7788327 → 6766415 → 7155379 → 2031914 → 3908338 → 8885874 → 20

Tabela 2: Distâncias e caminhos mínimos entre os vértices 10 e 20

Distância entre o vértice 10 e 30		
	Com Vetor	Com Heap
	Distância	Distância
Grafo 1	1.72	1.72
Grafo 2	1.72	1.72
Grafo 3	X	0.89
Grafo 4	X	2.02
Grafo 5	X	17.24

Caminho mínimo 10-30 Grafo 1: 10 → 2038 → 2658 → 682 → 4045 → 30

Caminho mínimo 10-30 Grafo 2:
10 → 22754 → 23713 → 16217 → 8863 → 12125 → 17348 → 22569 → 8688 → 30

Caminho mínimo 10-30 Grafo 3: 10 → 2858 → 35293 → 96094 → 85187 → 53177 → 64898 → 87087 → 77935 → 42304 → 78071 → 93985 → 27642 → 26151 → 30

Caminho mínimo 10-30 Grafo 4: 10 → 873980 → 182382 → 261305 → 81769 → 694525 → 219466 → 207765 → 44636 → 183747 → 920911 → 35338 → 239369 → 417194 → 132461 → 740528 → 707581 → 30

Caminho mínimo 10-30 Grafo 5:
10 → 2846336 → 5413988 → 202621 → 9270546 → 6836812 → 2983314 → 8775277 → 5549785 → 3485759 → 4341576 → 8868678 → 3166991 → 5117196 → 7291082 → 1816279 → 5664093 → 8947758 → 30

Tabela 3: Distâncias e caminhos mínimos entre os vértices 10 e 30

Distância entre o vértice 10 e 40		
	Com Vetor	Com Heap
	Distância	Distância
Grafo 1	2.05	2.05
Grafo 2	1.92	1.92
Grafo 3	X	0.84
Grafo 4	X	2.52
Grafo 5	X	17.37

Caminho mínimo 10-40 Grafo 1: 10 → 2038 → 2658 → 682 → 5941 → 6759 → 2195 → 1223 → 40

Caminho mínimo 10-40 Grafo 2:
10 → 22754 → 23713 → 3461 → 23673 → 23726 → 18966 → 12308 → 6984 → 11469 → 40

Caminho mínimo 10-40 Grafo 3:
10 → 2858 → 35293 → 3905 → 24445 → 86286 → 14327 → 19697 → 15930 → 46307 → 40

Caminho mínimo 10-40 Grafo 4: 10 → 873980 → 182382 → 674971 → 557022 → 188455 → 418550 → 275098 → 277633 → 801708 → 730863 → 274827 → 386086 → 188007 → 477496 → 40

Caminho mínimo 10-40 Grafo 5: 10 → 1299954 → 1419794 → 9885733 → 5641294 → 409605 → 4105715 → 660832 → 5850281 → 3215052 → 7676193 → 9008138 → 4846487 → 9798152 → 40

Tabela 4: Distâncias e caminhos mínimos entre os vértices 10 e 40

Distância entre o vértice 10 e 50		
	Com Vetor	Com Heap
	Distância	Distância
Grafo 1	1.2	1.2
Grafo 2	1.57	1.57
Grafo 3	X	0.87
Grafo 4	X	2.69
Grafo 5	X	15.43

Caminho mínimo 10-50 Grafo 1: 10 → 2038 → 7521 → 2258 → 3235 → 8078 → 2532 → 2527 → 50

Caminho mínimo 10-50 Grafo 2:
10 → 22754 → 23713 → 16217 → 8863 → 627 → 23978 →→ 14272 → 3594 → 24811 → 24612 → 14607 → 50

Caminho mínimo 10-50 Grafo 3:
10 → 56663 → 84437 → 19040 → 82510 → 17622 → 94992 → 70948 → 19878 → 50

Caminho mínimo 10-50 Grafo 4: 10 → 873980 → 182382 → 261305 → 81769 → 694525 → 382144 → 748624 → 663332 → 149827 → 349584 → 735867 → 9905 → 716970 → 472621 → 407659 → 994210 → 630598 → 50

Caminho mínimo 10-50 Grafo 5: 10 → 1299954 → 9927727 → 8451042 → 5368621 → 4276332 → 3716282 → 3724752 → 700154 → 9271362 → 6416250 → 396789 → 50

Tabela 5: Distâncias e caminhos mínimos entre os vértices 10 e 50

Distância entre o vértice 10 e 60		
	Com Vetor	Com Heap
	Distância	Distância
Grafo 1	1.66	1.66
Grafo 2	1.58	1.58
Grafo 3	X	0.97
Grafo 4	X	2.41
Grafo 5	X	17.99

Caminho mínimo 10-60 Grafo 1:
10 → 2038 → 5094 → 2875 → 3598 → 1268 → 4333 → 3383 → 8182 → 4328 → 9082 → 60

Caminho mínimo 10-60 Grafo 2:
10 → 22754 → 13268 → 9146 → 5521 → 20167 → 10684 → 1848 → 7009 → 10430 → 60

Caminho mínimo 10-60 Grafo 3: 10 → 2858 → 35293 → 3905 → 24445 → 89584 → 2874 → 99253 → 66532 → 33719 → 96654 → 75728 → 79865 → 74717 → 59399 → 8718 → 60

Caminho mínimo 10-60 Grafo 4:
10 → 873980 → 182382 → 674971 → 106317 → 27985 → 280367 → 227336 → 364950 → 883474 → 283156 → 161217 → 224210 → 188021 → 614364 → 358029 → 855615 → 822462 → 774634 → 60

Caminho mínimo 10-60 Grafo 5: 10 → 1299954 → 1419794 → 1899178 → 7524819 → 757290 → 9445351 → 9611705 → 9104671 → 8941517 → 264365 → 3945175 → 3090234 → 2071841 → 8424191 → 2860740 → 1175531 → 1925405 → 7718765 → 3193827 → 7824884 → 1454010 → 60

Tabela 6: Distâncias e caminhos mínimos entre os vértices 10 e 60

3.3 Questão 6

Distância até Edsger W. Dijkstra	
	Distância
Alan Turing	infinito
J. B. Kruskal	3.48037
Jon M.Kleinberg	2.70699
Eva Tardos	2.75351
Daniel R. Figueiredo	2.94283

Caminho de Alan Turing até Edsger W. Dijkstra: infinito

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

Caminho de Jon M.Kleinberg até Edsger W. Dijkstra:
Jon M. Kleinberg → Prabhakar Raghavan → Eli Upfal → Avi Wigderson → Prabhakar Ragde → Dimitrios M. Thilikos → Hans L. Bodlaender → Gerard Tel → A. J. M. van Gasteren → Edsger W. Dijkstra

Caminho de Eva Tardos até Edsger W. Dijkstra: Eva Tardos → Serge A. Plotkin → Andrew V. Goldberg → Robert Endre Tarjan → Haim Kaplan → Micha Sharir → Mark H. Overmars → Jan van Leeuwen → Hans L. Bodlaender → Gerard Tel → A. J. M. van Gasteren → Edsger W. Dijkstra

Caminho de Daniel R. Figueiredo até Edsger W. Dijkstra:
Daniel R. Figueiredo → Donald F. Towsley → Zhi-Li Zhang → Y. Thomas Hou → Bo Li → Chuang Lin → Dan C. Marinescu → John R. Rice → Edsger W. Dijkstra

4 Link para o repositório com o código

REPOSITÓRIO.

Referências

[1] Jayme Luiz Szwarcfiter. *Teoria computacional de grafos: Os Algoritmos*. Elsevier Brasil, 2018.

[2] Eva Tardos e Jon Kleinberg. *Algorithm Design*. Pearson, 2005.