TP1 – Algoritmos I

Aluno: Robert Garcia da Silva, Matrícula: 2019436676

Documentação TP1

Pontos-chave observados para a solução do problema

Para o problema apresentado no trabalho observou-se dois pontos essenciais para sua solução:

- Conseguir identificar a distância de cada posto de vacinação (PV) a partir de cada centro de distribuição (CD). A palavra “distância”, nesse caso, se refere a quantidade de PVs necessário se atravessar para ser alcançado um determinado PV partindo-se de um CD.

- Conseguir identificar a existência ou não de alguma rota, dentre todas possíveis para cada CD, em que é atravessado um mesmo PV mais de uma vez para uma mesma rota.

Modelagem Matemática e Computacional

Como visto nas aulas da disciplina e analisando o problema apresentado, a modelagem matemática para o problema foi utilizando Grafos direcionados devido a forte ligação e intuição do problema e sua representação com essa estrutura matemática.

Para a modelagem escolhida foi adotado a representação:

- Todos os PVs e CDs serão representados como vértices de um mesmo grafo, suas ligações direcionadas serão representados como arestas desse grafo.

- O PV01, PV02 e assim por diante foram representados com os vértices, respectivamente, 01, 02 e assim por diante. Seja **M** a quantidade de PVs, o CD01, CD02 e assim por diante foram representados com os vértices, respectivamente, **M**+1, **M**+2 e assim por diante. A Figura abaixo mostra um exemplo de um grafo modelado representando a Figura 1 do roteiro do trabalho.

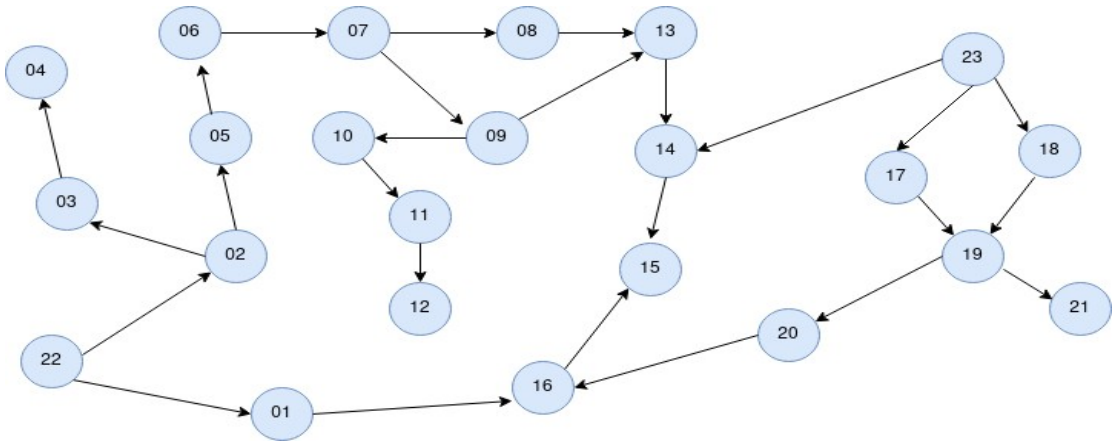

Figure 1: Grafo representativo da Figura 1 do roteiro do trabalho.

Para a solução dos dois pontos-chave colocados acima, dado o que foi apresentado na disciplina e a modelagem escolhida para o problema, usando grafos, foram escolhidos dois algoritmos para a solução dos pontos-chave observados. Para o primeiro ponto-chave apresentado, relacionado a distância entre os vértices foi escolhido a busca em largura (**Breadth-First Search – BFS**) e para o segundo ponto-chave, referente a passagem por um mesmo vértice mais de uma vez por uma mesma rota, foi escolhido a busca em profundidade (**Depth-First Search – DFS**).

O BFS foi escolhido pois com ele é possível gerar uma árvore contendo a distância em que cada vértice se encontra em relação ao um outro dado vértice escolhido, o que soluciona a primeira parte do problema. O DFS foi escolhido pois com ele é possível verificar a existência de ciclos em um grafo.

Para a representação computacional de um Grafo foi adotado inicialmente a lista de adjacência pois preferiu-se trabalhar com uma mesma estrutura de dados tanto para modelagem do grafo quanto para a saída da árvore de distância dos vértices a um vértice root usando BFS, como mencionado acima, e por ter observado, pelos testes de caso apresentados, que em geral a quantidade de vértices e arestas, dos grafos modelados, é aproximada, logo, a quantidade de vértices ao quadrado e o produto do número de vértices pelo número de arestas é aproximado (V^2 é aproximadamente igual a VA em que V representa o número de vértices e A o número de arestas) tornando o espaço de memória ocupado pela representação com uma matriz de adjacência próximo ao espaço ocupado por sua representação com em lita de adjacência.

Para implementação do BFS foi usado a estrutura de dados **fila** para realizar o controle dos vértices já visitados e os próximos a serem visitados e **map** para representar a árvore BFS com cada vértice do grafo e sua respectiva distância a partir de um vértice root escolhido.

Para implementação do DFS foi usado a estrutura de dados **pilha** para realizar o controle dos vértices já visitados e a escolha dos próximos a serem visitados.

O BFS e o DFS são os algoritmos chave para a solução do problema como um todo ,são os algoritmos responsáveis para a solução. Na solução usada no presente trabalho o BFS e DFS são executados uma vez, cada, para cada CD. A análise da complexidade temporal, número de execuções, foi analisada, entendida como sendo proporcional a $\mathbf{V^2 + AV}$ para uma execução do DFS e BFS no cenário em que o grafo é direcionado conexo e não apresenta ciclos, em que V representa o número de vértices do grafo e A seu número de arestas. Seja NCD o número de centros de distribuição a execução do algoritmo é proporcional a $\mathbf{NCD(V^2 + AV)}$ número de execuções.

O Pseudocódigo de uma possível implementação, implementação usada no presente trabalho, implementação simples e não ótima, para BFS e DFS, bem como a análise feita, entendida são apresentados abaixo.

```
Pseudocódigo - DFS

Seja v um vértice de um grafo direcionado G.
seja P uma stack;
Marque todos os vértices v de G como não visitados.
Escolhe um vértice v, chamado de root, para iniciar a busca em profundidade. A busca em profundidade irá retornar True
se uma rota partindo-se de v, root, passar por um outro vértice do grafo mais de uma ves e False caso contrário.


Enquanto True: //Executado V vezes
	if(v não foi visitado): // Esse laço if é verificado V vezes. Para cada V é executado três instruções.
		Marca v como visitado;
		Adicione v a P;
		Marca v como pertencente a P;
	FimIf

	Para(i =0; i < número de vértices vizinhos de v, i++): // Executado V + A vezes.
		//seja v[i] um vizinho de v
		if(v[i] pertence a P):
			return True; // Já foi visitado
		else if(v[i] não foi visitado):
			Acione um flag como vértice achado;
			break
	FimPara

	if(flag vértice achado é True):
		v = v[i];
	else:
		Marca v como não pertencente a P.
		Retira v de P.
		Se P está vazia break em EnquantoTrue.
		v = Topo de P.
	FimIf

FimEnquanto
	return false;
```

Figure 2: Pseudocódigo DFS.

```
Pseudocódigo - BFS

Seja v um vértice de um grafo direcionado G.
Seja L uma lista.
Marque todos os vértices v de G como não visitados.
Escolhe um vértice v, chamado de root, para iniciar a busca em largura.
Seja M um map. //map<vértice, distância do vértice root>.
Faça map<root, 0>; // distância do vértice root.
Marque root como visitado.


Enquanto True:
	if(v não foi visitado):
		Marque v como visitado;
		Coloque v na lista.
	FimIf

	Para(i =0; i < número de vértices vizinhos de v, i++):
		//seja v[i] um vizinho de v
		if(v[i] não foi visitado):
			Marque V[i] como visitado;
			Faça M<v[i], M[v]+1>; //distancia de v[i] é igual a distância do vértice pai, v, somado um.
			Coloque v[i] na lista.
	FimPara

	if(lista está vazia):
		break EnquantoTrue;
	else:
		Retira o primeiro elemento de L.
		v = Primeiro elemento de L.
	FimIf;
return M;


/*
- Análise feita, entendida


Tanto o DFS quando o BFS são executados uma vez, cada, para cada CD. Seja V e A o número de vértices e arestas,
respectivamente de G. Seja C o número de centros de distribuição. Tanto o DFS e o BFS apresentam complexidade
proporcional a V(V + A), para o número de centros de distribuição temos C(V² + VA) para o caso em que G é direcionado
conexo e é não ciclico, não atravessa um mesmo vértice mas de uma vez.

*/
```

Figure 3: PseudoCódigo BFS bem e análise feita, entendida.

Referências

[1] Ime, **algoritmos para grafos.** BFS https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/bfs.html#sec:performance Acessado: Entre 12/01/2021 e 22/01/2021

[2] Ime, **algoritmos para grafos**. DFS https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dfs.html#performance Acessado: Entre 12/01/2021 e 22/01/2021

[3] **Geeks Forgeeks**, STL https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/ Acessado: Entre 12/01/2021 e 22/01/2021

[4] **Khan Academy**, BFS https://pt.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/pc/challenge-implement-breadth-first-search Acessado: Entre 12/01/2021 e 22/01/2021

[5] **Wikipédia**, STL https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library Acessado: Entre 12/01/2021 e 22/01/2021

[6] Youtube, **Algoritmos de Busca, Jussara Almedia** (DCC - UFMG) https://www.youtube.com/watch?v=aoOs3ADVH0s&feature=youtu.be Acessado: Entre 12/01/2021 e 22/01/2021

[7] Youtube, **Grafos. Marcos Castro** https://www.youtube.com/watch?v=BwZvcq5K1wU&list=PL8eBmR3QtPL13Dkn5eEfmG9TmzPpTp0cV&index=68 Acessado: Entre 12/01/2021 e 22/01/2021

[8] Youtube, **Grafos. Marcos Castro** https://www.youtube.com/watch?v=DYLfrmHHAm0&list=PL8eBmR3QtPL13Dkn5eEfmG9TmzPpTp0cV&index=70 Acessado: Entre 12/01/2021 e 22/01/2021

[9] Youtube, **Grafos. Marcos Castro** https://www.youtube.com/watch?v=1jpuy3Vizt0 Acessado: Entre 12/01/2021 e 22/01/2021

[10] A Simple Makefile Tutorial https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/ Acessado: Entre 12/01/2021 e 22/01/2021