# Implementação dos algoritmos *Dijkstra* e DFS sobre grafos

Paulo Ricardo & Vinícios Bidin

16 de maio de 2023

## Sumário



# 1 Introdução

Trata-se de um projeto que tem como proposta a implementação dos algoritmos *Dijkstra* e DFS ou (*Depth First Search)* sobre grafos. Sendo o primeiro para busca do caminho mínimo entre dois vértices, especificamente para grafos ponderados, e o segundo uma busca em produndidade.

O grafo em que estes algoritmos é aplicado se trata de um grafo pequeno com somente oito vértices que representam paradas de ônibus e as arestas com seus pesos, as distâncias entre esses.

# 2 Compilação e Execução

## 2.1 Dependências

São dependências do projeto, a biblioteca *Graphviz* sendo utilizada a linguagem *Dot* para a visualização do grafo gerado, utiliza-se ainda o *Makefile* para compilação de todo o projeto, para que não seja necessário ficar compilando os arquivos de forma individual.

### 2.2 Compilação

Uma vez no diretório do projeto, compile, utilizando `make`. Outra maneira, é utilizando o comando `make run`. Desta forma, além de compilar, o binário `main` será executado, gerando os arquivos "data/dfs.dot" e "data/dijkstra.dot". Ainda, utilizando o comando `make plot`, além de compilado e executado, os vetores *(Scalable Vector Graphics)* ou *(svg)* serão gerados, seus caminhos serão "data/dfs.svg"e "data/dijkstra.svg".

### 2.3 Execução

Para executar, basta executar o arquivo binário de saída `main`, utilizando o comando `./main`. Após feito, assim como rodando o comando `make run`, os arquivos *.dot* serão gerados, precisando então gerar os vetores a partir destes.

## 3 Resultados

### 3.1 Dijkstra

O algoritmo de *Dijkstra*, dado um grafo ponderado e os vértices de origem e destino, encontrará o melhor caminho tendo este, o menor custo possível. Isto é feito com o auxílio de uma lista de prioridades de vértices e suas distâncias ao vértice de origem. Inicialmente com exceção do vértice de origem que possui 0, todos os demais vértices são marcados com uma distância inifinita. Repeditamente, o algoritmo seleciona o vértice com a menor distância e verifica todos os vizinhos (vértices alcançáveis com um passo deste), e atualizando suas distâncias caso uma distância menor é encontrada. Este processo é feito até que o vértice de destino seja alcançado. Sua complexidade temporal pode ser limitada tanto inferior quanto superiormente pela função $\Theta(|E|+|V|lg|V|)$, sendo $|E|$ o número de arestas no grafo e $|V|$ o número de vértices.

Segue a representação do grafo descrito feito com o auxílio do *Graphviz*, como mencionado anteriormente. Neste grafo, o caminho mínimo do vértice de origem *0* ao vértice de destino *7* está destacado.

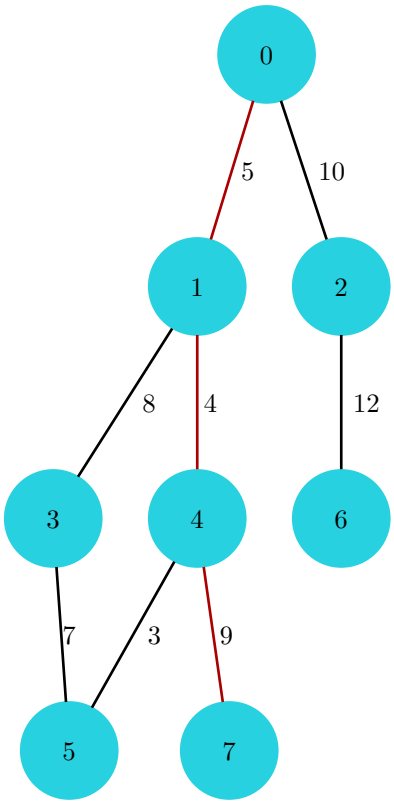


Figura 1: Caminho mínimo de *Dijkstra*

### 3.2 DFS

A busca em profundidade (DFS) é um algoritmo de travessia de grafos que explora o máximo possível ao longo de cada ramificação antes de retroceder. Ele começa em um nó escolhido e visita todos os seus vizinhos, em seguida, recursivamente, explora os vizinhos ainda não visitados até que não os haja mais. Utiliza um vetor para manter o controle dos nós a serem visitados. Inicialmente, o nó de partida é inserido no vetor. Em cada passo, um nó é removido do vetor e marcado como visitado. Em seguida, todos os seus vizinhos não visitados são inseridos na vetor. Esse processo continua até que o vetor esteja vazio, indicando que todos os nós foram visitados. Para tal, a complexidade temporal pode ser definida por $O(|V| + |E|)$, onde $|V|$ é o número de vértices no grafo e $|E|$ o número de arestas deste.

Na figura a seguir, tem-se representado o mesmo grafo apresentado anteriormente, porém desta vez sem peso nas arestas, e ainda, destacado da cor vermelha tem-se os vértices visitados para fazer uma busca em profundidade do nó *0* até novamente o nó *7*.

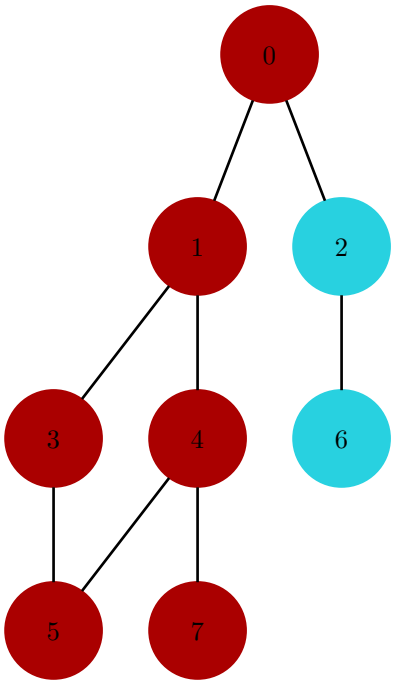


Figura 2: Busca em profundidade