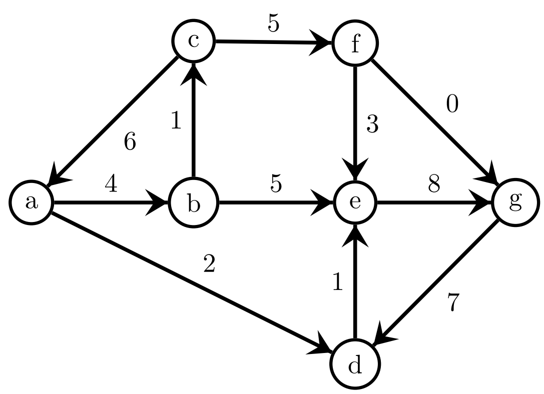
Aula Prática 4

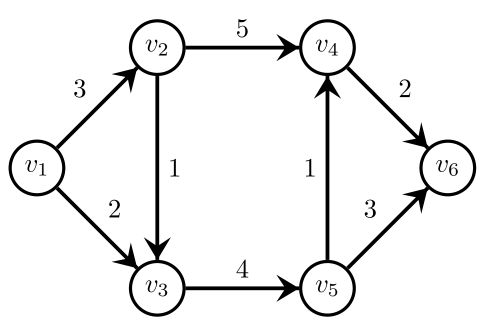
ASA 2020/2021

T1 06/07 II.1 Considere o grafo da figura.



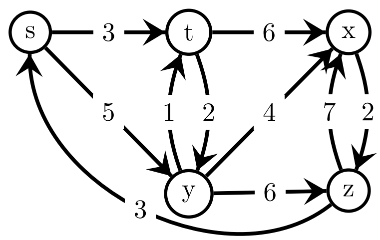
Indique os valores de d e π para cada vértice quando faltam extrair dois nós da fila de prioridade na execução do algoritmo de Dijkstra a partir do vértice c.

T1 08/09 II.3 Considere a execução do algoritmo de Dijkstra, sobre o grafo dirigido e pesado da figura abaixo, tendo como fonte o vértice v_1.



Indique os valores de d e π para os vértices $\{v_2, v_3, v_4, v_5, v_6\}$ imediatamente após a aplicação do procedimento Relax sobre todos os arcos com origem no vértice v_5, durante a execução do referido algoritmo.

Ex. 24.3-1 Run Dijkstra's algorithm on the following directed graph, first use vertex s as the source and then use vertex z. Show the d and π values and the vertices in the set S after each iteration of the **while** loop.

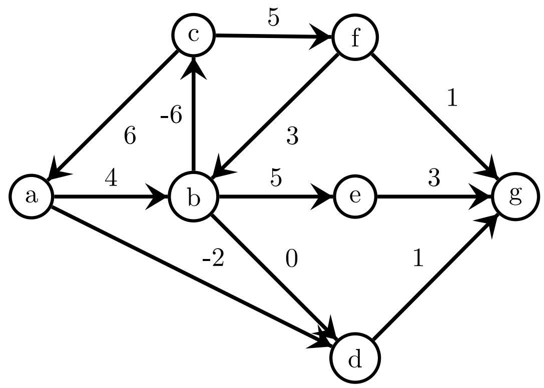


Ex. 24.3-2 Give a simple example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces incorrect answers. Why doesn't the proof of Theorem 24.6 go through when negative-weight edges are allowed?

Ex. 24.3-3 Suppose we change line 4 of Dijktra's algorithm to the following:
while $|Q| > 1$
This causes the while loop to execute $|V| - 1$ times instead of $|V|$ times. Is this proposed algorithm correct?

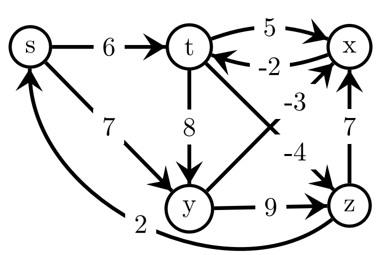
Ex. 24.3-4 Professor Gaedel has written a program that he claims implements Dijkstra's algorithm. The program produces `d[v]` and π[`v`] for each vertex $v \in V$. Give an $O(V + E)$ time algorithm to check the output of the professor's program. It should determine whether the d and π attributes match those of some shortest-paths tree. You may assume that all edge weights are nonnegative.

R1 06/07 II.1 Considere o grafo da figura.



Indique os valores de d e π para todos os vértices após duas iterações do ciclo principal do algoritmo de Bellman-Ford. Considere como fonte o vértice b e que uma ordem lexicográfica para o tratamento dos arcos (ou seja, ordem alfabética dos nós de partida e, dentre estes, ordem alfabética dos nós de chegada).

Ex. 24.1-1 Run the Bellman-Ford algorithm on the following directed graph, using vertex z as the source. In each pass, relax edge in the following order `(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)`, and show d and π values after each pass. Now, change the weight of edge (z, x) to 4 and run the algorithm again, using s as the source.



Ex. 24.1-4 Modify the Bellman-Ford algorithm so that it sets `d[v]` to $-\infty$ for all vertices v for which there is a negative-weight cycle on the path from the source to v.

R1 08/09 II.2 Considere os algoritmos para o cálculo de caminhos mais curtos. Indique se cada uma das seguintes afirmações é verdadeira (V) ou falsa (F).

1. O algoritmo de Bellman-Ford permite detectar ciclos negativos.

2. Se a relaxação dos arcos de um grafo dirigido e acíclico for efectuada de acordo com a ordenação topológica dos respectivos vértices, é possível determinar os caminhos mais curtos de fonte única em tempo $\Theta(V + E)$.

3. No algoritmo de Dijkstra, quando um vértice u é extraído da fila de prioridade, $d[u]$ e $\pi[u]$ já têm o respectivo valor final, mesmo em grafos contendo arcos com peso negativo.

4. O algoritmo de Dijkstra produz os valores finais correctos, mesmo que o ciclo principal seja executado apenas $|V| - 2$ vezes.

5. Se num grafo existir mais do que um componente fortemente ligado (SCC), têm obrigatoriamente que existir dois vértices u e v, tal que $\delta(u, v) = \infty$.

6. Os caminhos mais curtos obedecem sempre à desigualdade triangular.

7. Em grafos em que os pesos dos arcos sejam todos diferentes e inteiros positivos, existe apenas um caminho mais curto entre qualquer par de vértices.

8. O tempo de execução do algoritmo de Bellman-Ford é $O(VE^2)$.