

Caminhos de Peso Mínimo

Prof. Andrei Braga



Conteúdo

- Caminhos de peso mínimo
- Algoritmo de Dijkstra
- Algoritmo de Bellman-Ford
- Exercícios
- Referências

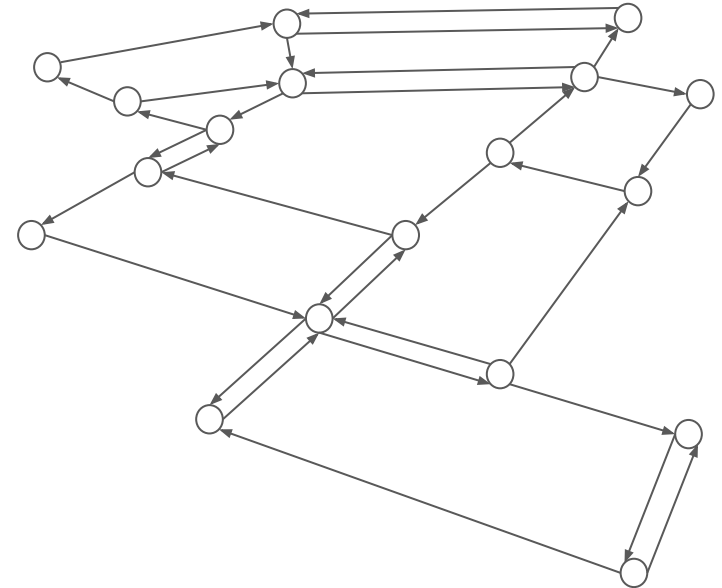
Grafos dirigidos (Digrafos)

- Vimos situações que podem ser modeladas com grafos dirigidos (digrafos) – grafos cujas arestas têm uma **direção** (ou **orientação** ou **sentido**)
- Exemplo:
 - Temos um mapa de vias (ruas ou rodovias) e estamos interessados nos caminhos que podemos percorrer neste mapa
 - Uma via que conecta um ponto x a um ponto y pode ter apenas a mão de x para y, apenas a mão de y para x ou ambas as mãos
 - Podemos representar este mapa como um grafo onde cada **aresta** tem uma direção e representa **uma mão de uma via**



Grafos dirigidos (Digrafos)

- Vimos situações que podem ser modeladas com grafos dirigidos (digrafos) – grafos cujas arestas têm uma **direção** (ou **orientação** ou **sentido**)
- Exemplo:
 - Temos um mapa de vias (ruas ou rodovias) e estamos interessados nos caminhos que podemos percorrer neste mapa
 - Uma via que conecta um ponto x a um ponto y pode ter apenas a mão de x para y, apenas a mão de y para x ou ambas as mãos
 - Podemos representar este mapa como um grafo onde cada **aresta** tem uma direção e representa **uma mão de uma via**



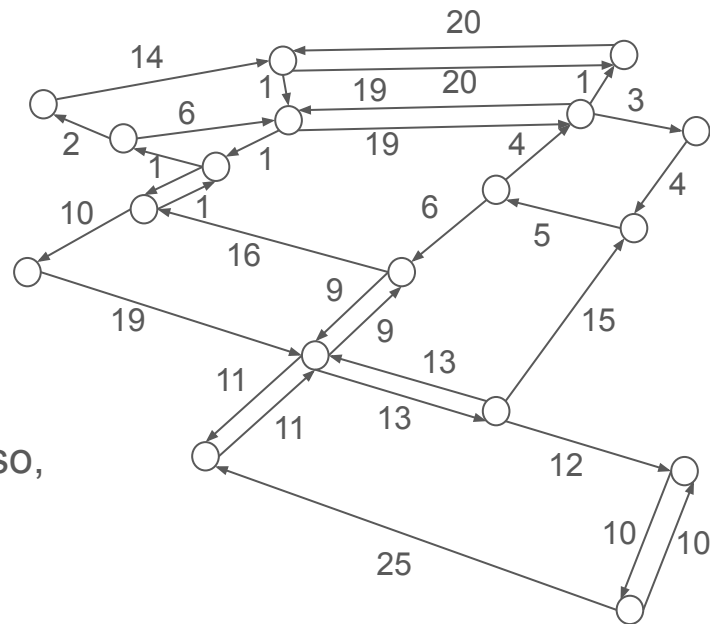
Digrafos com pesos nas arestas

- Em situações como estas, também pode fazer sentido considerarmos pesos nas arestas do digrafo
- Exemplo:
 - Temos um mapa de vias (ruas ou rodovias) e estamos interessados em caminhos **curtos** (considerando as **distâncias no mapa**) que podemos percorrer neste mapa
 - Podemos representar este mapa como um grafo onde cada aresta tem uma direção, que representa uma mão de uma via, e tem um peso, que representa a distância no mapa entre os pontos conectados



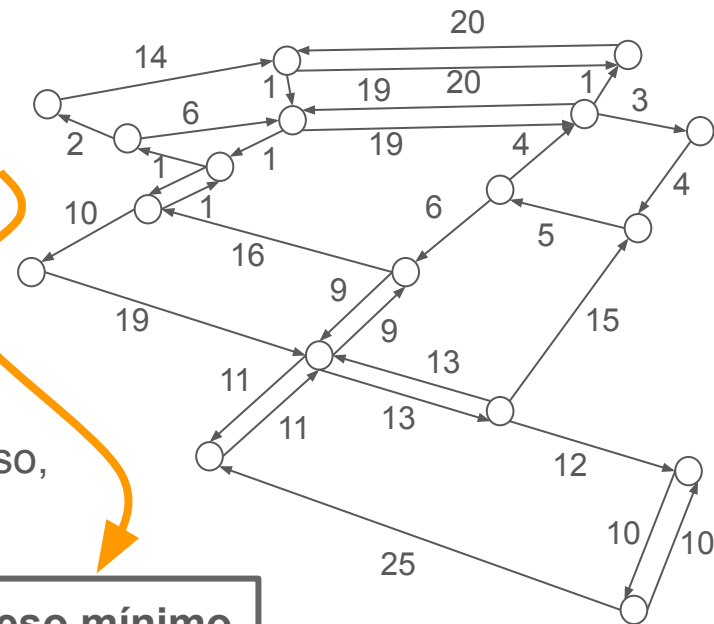
Digrafos com pesos nas arestas

- Em situações como estas, também pode fazer sentido considerarmos pesos nas arestas do digrafo
- Exemplo:
 - Temos um mapa de vias (ruas ou rodovias) e estamos interessados em caminhos **curtos** (considerando as **distâncias no mapa**) que podemos percorrer neste mapa
 - Podemos representar este mapa como um grafo onde cada aresta tem uma direção, que representa uma mão de uma via, e tem um peso, que representa a distância no mapa entre os pontos conectados



Digrafos com pesos nas arestas

- Em situações como estas, também pode fazer sentido considerarmos pesos nas arestas do digrafo
- Exemplo:
 - Temos um mapa de vias (ruas ou rodovias) e estamos interessados em caminhos **curtos** (considerando as **distâncias no mapa**) que podemos percorrer neste mapa
 - Podemos representar este mapa como um grafo onde cada aresta tem uma direção, que representa uma mão de uma via, e tem um peso, que representa a distância no mapa entre os pontos conectados



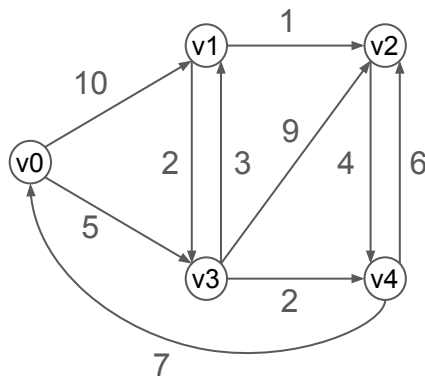
caminhos de **peso mínimo**

Peso de um caminho

- O **peso** de um caminho em um digrafo G é a soma dos pesos das arestas do caminho (o mesmo vale para um passeio, uma trilha e um ciclo)

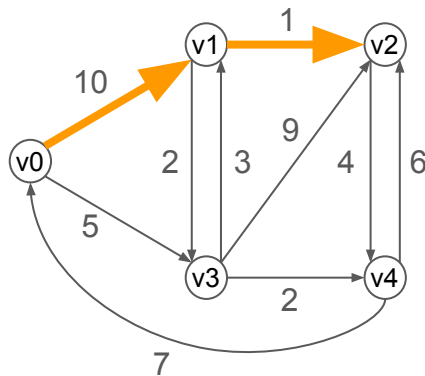
- Exemplo:

- No digrafo ao lado,
 - O peso do caminho $v_0 v_1 v_2$



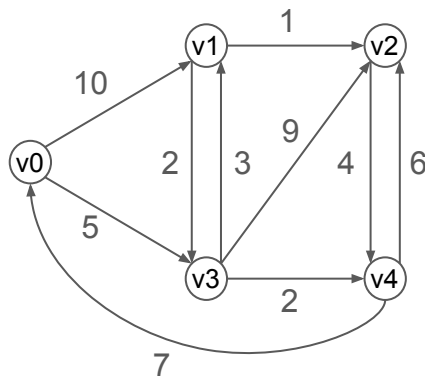
Peso de um caminho

- O **peso** de um caminho em um digrafo G é a soma dos pesos das arestas do caminho (o mesmo vale para um passeio, uma trilha e um ciclo)
- Exemplo:
 - No digrafo ao lado,
 - O peso do caminho $v_0 v_1 v_2$ é 11



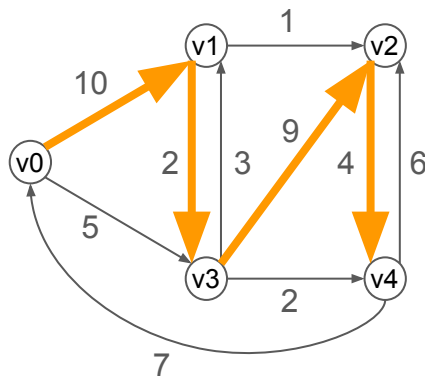
Peso de um caminho

- O **peso** de um caminho em um digrafo G é a soma dos pesos das arestas do caminho (o mesmo vale para um passeio, uma trilha e um ciclo)
- Exemplo:
 - No digrafo ao lado,
 - O peso do caminho $v_0 v_1 v_2$ é 11
 - O peso do caminho $v_0 v_1 v_3 v_2 v_4$



Peso de um caminho

- O **peso** de um caminho em um digrafo G é a soma dos pesos das arestas do caminho (o mesmo vale para um passeio, uma trilha e um ciclo)
- Exemplo:
 - No digrafo ao lado,
 - O peso do caminho $v_0 v_1 v_2$ é 11
 - O peso do caminho $v_0 v_1 v_3 v_2 v_4$ é 25

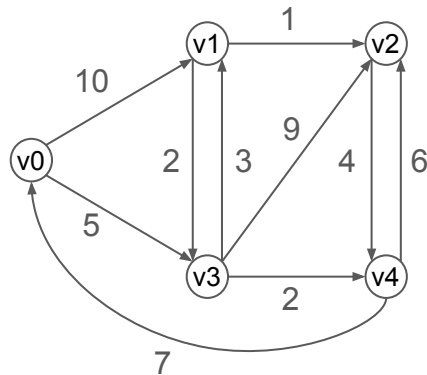


Distância ponderada

- A **distância ponderada** de um vértice v_i para um vértice v_j em um digrafo G , denotada por $dp(v_i, v_j)$, é
 - o menor peso de um $v_i v_j$ -caminho em G ou
 - ∞ (infinita) caso não exista um $v_i v_j$ -caminho em G
- Note que, em geral, $dp(v_i, v_j) \neq dp(v_j, v_i)$

- Exemplo:

- No digrafo ao lado,
 - $dp(v_0, v_1) =$ e $dp(v_1, v_0) =$,
 - $dp(v_3, v_2) =$ e $dp(v_2, v_3) =$ e
 - $dp(v_4, v_4) =$

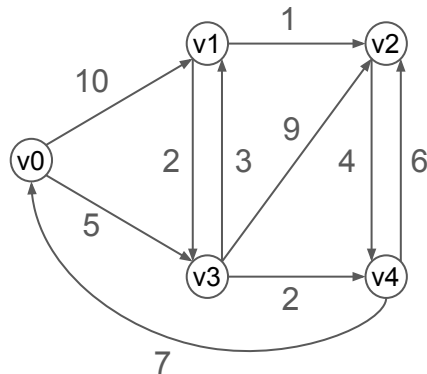


Distância ponderada

- A **distância ponderada** de um vértice v_i para um vértice v_j em um digrafo G , denotada por $dp(v_i, v_j)$, é
 - o menor peso de um $v_i v_j$ -caminho em G ou
 - ∞ (infinita) caso não exista um $v_i v_j$ -caminho em G
- Note que, em geral, $dp(v_i, v_j) \neq dp(v_j, v_i)$

- Exemplo:

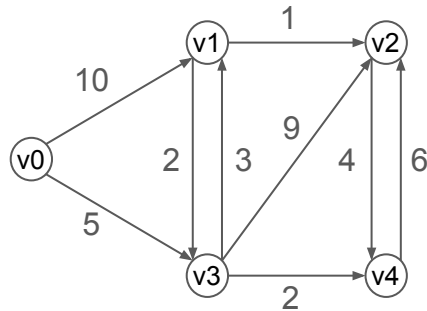
- No digrafo ao lado,
 - $dp(v_0, v_1) = 8$ e $dp(v_1, v_0) = 11$,
 - $dp(v_3, v_2) = 4$ e $dp(v_2, v_3) = 16$ e
 - $dp(v_4, v_4) = 0$



Distância ponderada

- A **distância ponderada** de um vértice v_i para um vértice v_j em um digrafo G , denotada por $dp(v_i, v_j)$, é
 - o menor peso de um $v_i v_j$ -caminho em G ou
 - ∞ (infinita) caso não exista um $v_i v_j$ -caminho em G
- Note que, em geral, $dp(v_i, v_j) \neq dp(v_j, v_i)$

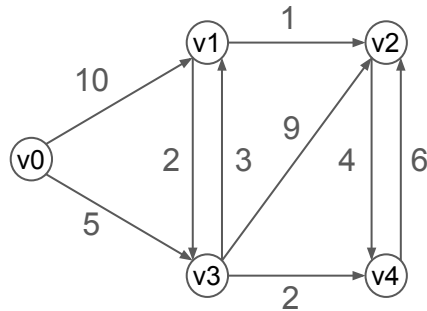
- Exemplo:
 - No digrafo ao lado,
 - $dp(v_1, v_0) =$



Distância ponderada

- A **distância ponderada** de um vértice v_i para um vértice v_j em um digrafo G , denotada por $dp(v_i, v_j)$, é
 - o menor peso de um $v_i v_j$ -caminho em G ou
 - ∞ (infinita) caso não exista um $v_i v_j$ -caminho em G
- Note que, em geral, $dp(v_i, v_j) \neq dp(v_j, v_i)$

- Exemplo:
 - No digrafo ao lado,
 - $dp(v_1, v_0) = \infty$



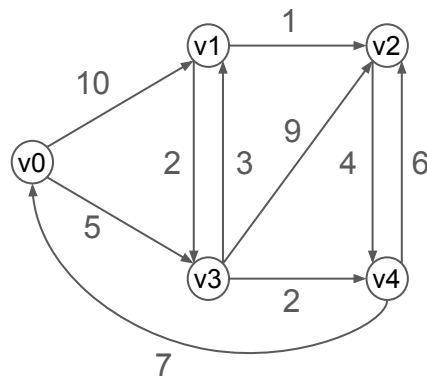
Problema dos caminhos de peso mínimo

- **Problema:** Dado um digrafo G e um vértice s de G , encontre, para cada vértice v de G , um sv -caminho de peso mínimo em G

- Exemplo:

- Sendo G dado pelo digrafo ao lado e s igual a v_0 , uma solução para o problema é

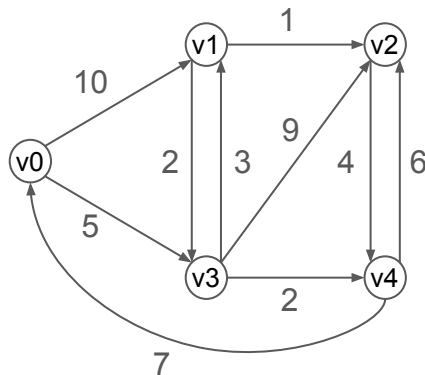
- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$ e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$



Problema dos caminhos de peso mínimo

- **Problema:** Dado um digrafo G e um vértice s de G , encontre, para cada vértice v de G , um sv -caminho de peso mínimo em G
- Existem algumas variações interessantes deste problema

- Exemplo:
 - Sendo G dado pelo digrafo ao lado e s igual a v_0 , uma solução para o problema é
 - $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
 - $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
 - $v_0 v_3$ — $dp(v_0, v_3) = 5$ e
 - $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

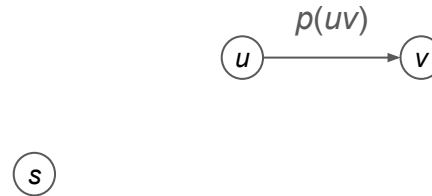


Relaxação de uma aresta

- Os algoritmos que veremos para resolver o problema dos caminhos de peso mínimo se baseiam em uma operação chamada de **relaxação de uma aresta**
- Antes de ver os algoritmos, vamos entender esta operação

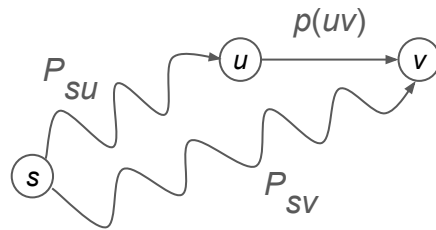
Relaxação de uma aresta

- Considere um digrafo G , um vértice s de G , e uma aresta uv de G com peso $p(uv)$



Relaxação de uma aresta

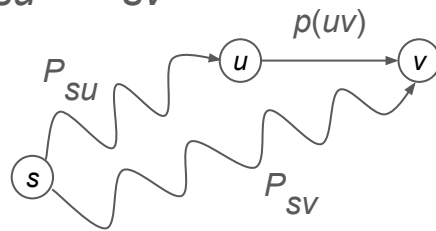
- Considere um digrafo G , um vértice s de G , e uma aresta uv de G com peso $p(uv)$
- Seja P_{su} um su -caminho de peso mínimo em G e P_{sv} um sv -caminho de peso mínimo em G



Relaxação de uma aresta

- Considere um digrafo G , um vértice s de G , e uma aresta uv de G com peso $p(uv)$
- Seja P_{su} um su -caminho de peso mínimo em G e P_{sv} um sv -caminho de peso mínimo em G
- O que podemos dizer sobre os pesos de P_{su} e P_{sv} ?

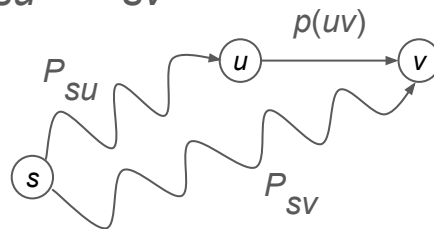
peso de P_{sv} ? peso de P_{su} + $p(uv)$
 $\leq ?$
 $\geq ?$



Relaxação de uma aresta

- Considere um digrafo G , um vértice s de G , e uma aresta uv de G com peso $p(uv)$
- Seja P_{su} um su -caminho de peso mínimo em G e P_{sv} um sv -caminho de peso mínimo em G
- O que podemos dizer sobre os pesos de P_{su} e P_{sv} ?

$$\text{peso de } P_{sv} \leq \text{peso de } P_{su} + p(uv)$$



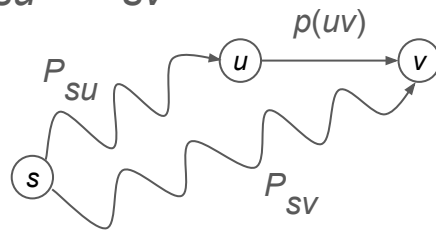
Relaxação de uma aresta

- Considere um digrafo G , um vértice s de G , e uma aresta uv de G com peso $p(uv)$
- Seja P_{su} um su -caminho de peso mínimo em G e P_{sv} um sv -caminho de peso mínimo em G
- O que podemos dizer sobre os pesos de P_{su} e P_{sv} ?

$$\text{peso de } P_{sv} \leq \text{peso de } P_{su} + p(uv)$$

- Consequentemente,

$$dp(s,v) \leq dp(s,u) + p(uv)$$

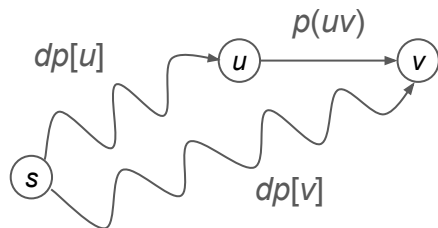


Relaxação de uma aresta

- Os algoritmos que veremos a seguir usam um vetor dp sobre o qual podemos falar o seguinte:
 - Durante a execução do algoritmo,
 - $dp[u]$ contém o menor peso de um su -caminho encontrado até o momento e
 - $dp[v]$ contém o menor peso de um sv -caminho encontrado até o momento
 - Ao fim da execução do algoritmo,
 - $dp[u]$ contém o peso mínimo de um su -caminho, ou seja, $dp[u] = dp(s,u)$ e
 - $dp[v]$ contém o peso mínimo de um sv -caminho, ou seja, $dp[v] = dp(s,v)$
- O que podemos dizer sobre $dp[u]$ e $dp[v]$?

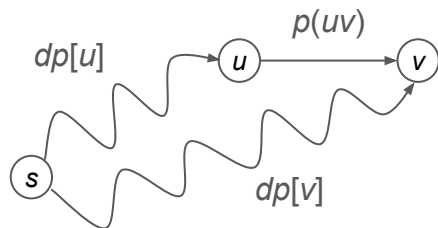
Relaxação de uma aresta

- Os algoritmos que veremos a seguir usam um vetor dp sobre o qual podemos falar o seguinte:
 - Durante a execução do algoritmo,
 - $dp[u]$ contém o menor peso de um su -caminho encontrado até o momento e
 - $dp[v]$ contém o menor peso de um sv -caminho encontrado até o momento
 - Ao fim da execução do algoritmo,
 - $dp[u]$ contém o peso mínimo de um su -caminho, ou seja, $dp[u] = dp(s,u)$ e
 - $dp[v]$ contém o peso mínimo de um sv -caminho, ou seja, $dp[v] = dp(s,v)$
- O que podemos dizer sobre $dp[u]$ e $dp[v]$?



Relaxação de uma aresta

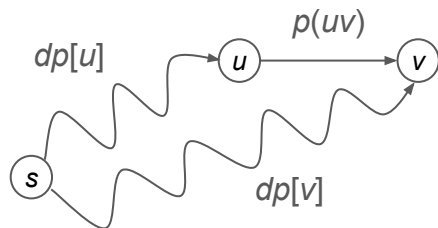
- Os algoritmos que veremos a seguir usam um vetor dp sobre o qual podemos falar o seguinte:
 - Durante a execução do algoritmo,
 - $dp[u]$ contém o menor peso de um su -caminho encontrado até o momento e
 - $dp[v]$ contém o menor peso de um sv -caminho encontrado até o momento
 - Ao fim da execução do algoritmo,
 - $dp[u]$ contém o peso mínimo de um su -caminho, ou seja, $dp[u] = dp(s,u)$ e
 - $dp[v]$ contém o peso mínimo de um sv -caminho, ou seja, $dp[v] = dp(s,v)$
 - O que podemos dizer sobre $dp[u]$ e $dp[v]$?
Se $dp[v] > dp[u] + p(uv)$, então $dp[v]$ ainda não contém o peso mínimo de um sv -caminho



Relaxação de uma aresta

- Os algoritmos que veremos a seguir usam um vetor dp sobre o qual podemos falar o seguinte:
 - Durante a execução do algoritmo,
 - $dp[u]$ contém o menor peso de um su -caminho encontrado até o momento e
 - $dp[v]$ contém o menor peso de um sv -caminho encontrado até o momento
 - Ao fim da execução do algoritmo,
 - $dp[u]$ contém o peso mínimo de um su -caminho, ou seja, $dp[u] = dp(s,u)$ e
 - $dp[v]$ contém o peso mínimo de um sv -caminho, ou seja, $dp[v] = dp(s,v)$
 - O que podemos dizer sobre $dp[u]$ e $dp[v]$?

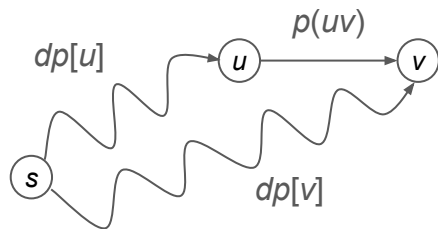
Se $dp[v] > dp[u] + p(uv)$, então $dp[v]$ ainda não contém o peso mínimo de um sv -caminho e podemos fazer $dp[v] = dp[u] + p(uv)$



Relaxação de uma aresta

- Os algoritmos que veremos a seguir usam um vetor dp sobre o qual podemos falar o seguinte:
 - Durante a execução do algoritmo,
 - $dp[u]$ contém o menor peso de um su -caminho encontrado até o momento e
 - $dp[v]$ contém o menor peso de um sv -caminho encontrado até o momento
 - Ao fim da execução do algoritmo,
 - $dp[u]$ contém o peso mínimo de um su -caminho, ou seja, $dp[u] = dp(s,u)$ e
 - $dp[v]$ contém o peso mínimo de um sv -caminho, ou seja, $dp[v] = dp(s,v)$
 - O que podemos dizer sobre $dp[u]$ e $dp[v]$?

Ao fazer com que $dp[v] \leq dp[u] + p(uv)$, podemos dizer que esta **restrição** está satisfeita ou **relaxada**, que **relaxamos** esta **restrição**, ou ainda que **relaxamos** esta **aresta**



Relaxação de uma aresta

- Assim, a operação de relaxação da aresta uv consiste no seguinte:
 1. Se $dp[v] > dp[u] + p(uv)$:
 2. $dp[v] = dp[u] + p(uv)$

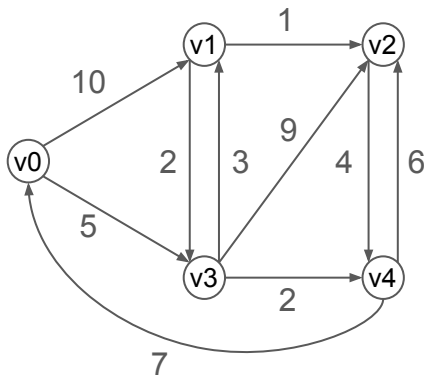
Representação dos caminhos de peso mínimo

- Vamos representar os caminhos de peso mínimo de maneira semelhante ao que fizemos em algoritmos vistos anteriormente: vamos representar estes caminhos através de uma árvore
- Esta árvore terá as seguintes propriedades:
 - O vértice s será a raiz da árvore
 - Para todo vértice v , o caminho entre s e v na árvore no sentido de s para v corresponderá a um sv -caminho de peso mínimo no digrafo G

Representação dos caminhos de peso mínimo

- Esta árvore terá as seguintes propriedades:
 - O vértice s será a raiz da árvore
 - Para todo vértice v , o caminho entre s e v na árvore no sentido de s para v corresponderá a um sv -caminho de peso mínimo no digrafo G

- Exemplo:

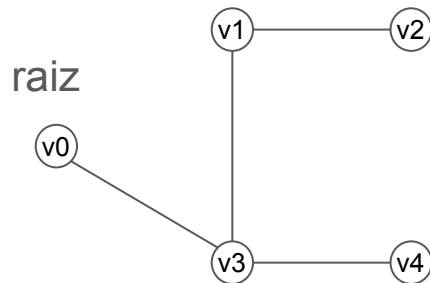


Caminhos de peso mínimo

s é igual a v_0

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$ e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

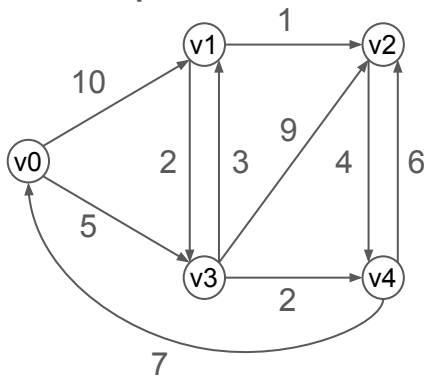
Árvore



Representação dos caminhos de peso mínimo

- Além disso, vamos usar a mesma representação de árvores utilizada em algoritmos vistos anteriormente: vamos representar a árvore através de um vetor *pai*

- Exemplo:

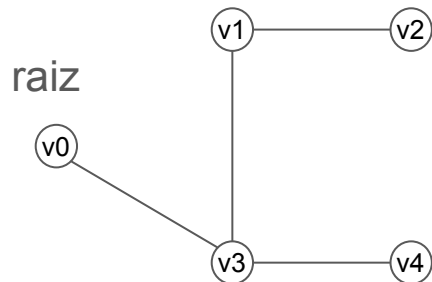


Caminhos de peso mínimo

s é igual a v_0

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$ e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

Árvore

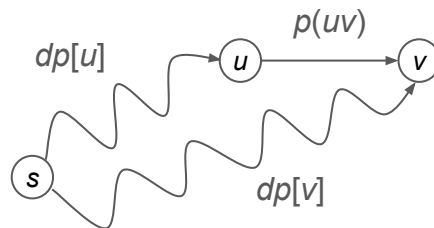


pai

-1	3	1	0	3
0	1	2	3	4

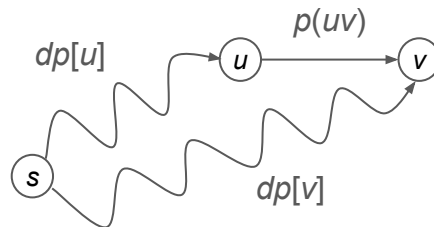
Representação dos caminhos de peso mínimo

- Quando fazemos a operação de relaxar uma aresta uv , podemos encontrar um sv -caminho de menor peso
- Relaxação da aresta uv :
 1. Se $dp[v] > dp[u] + p(uv)$:
 2. $dp[v] = dp[u] + p(uv)$



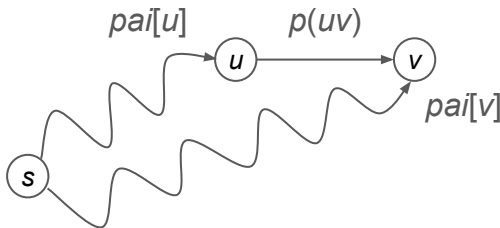
Representação dos caminhos de peso mínimo

- Quando fazemos a operação de relaxar uma aresta uv , podemos encontrar um sv -caminho de menor peso
- Relaxação da aresta uv :
 - Se $dp[v] > dp[u] + p(uv)$:
 - $dp[v] = dp[u] + p(uv)$
 - $pai[v] = u$
- Se encontrarmos um sv -caminho de menor peso, vamos armazenar este caminho usando o vetor pai



Representação dos caminhos de peso mínimo

- Sendo assim, os algoritmos que veremos a seguir usam o vetor *pai* da seguinte maneira:
 - Durante a execução do algoritmo,
 - $pai[u]$ contém o predecessor (o vértice que vem antes) de u no su -caminho de menor peso encontrado até o momento e
 - $pai[v]$ contém o predecessor (o vértice que vem antes) de v no sv -caminho de menor peso encontrado até o momento
 - Ao fim da execução do algoritmo,
 - $pai[u]$ contém o predecessor de u em um su -caminho de peso mínimo e
 - $pai[v]$ contém o predecessor de v em um sv -caminho de peso mínimo



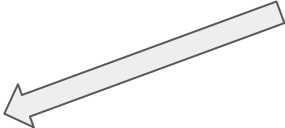
Arestas com pesos não-negativos

- Vamos considerar primeiro o problema dos caminhos de peso mínimo no caso em que **todas as arestas** do digrafo possuem **peso não-negativo**
- Neste caso, o problema pode ser resolvido usando o Algoritmo de Dijkstra
- O Algoritmo de Dijkstra pode ser descrito de maneira semelhante ao Algoritmo de Prim

Algoritmo de Dijkstra

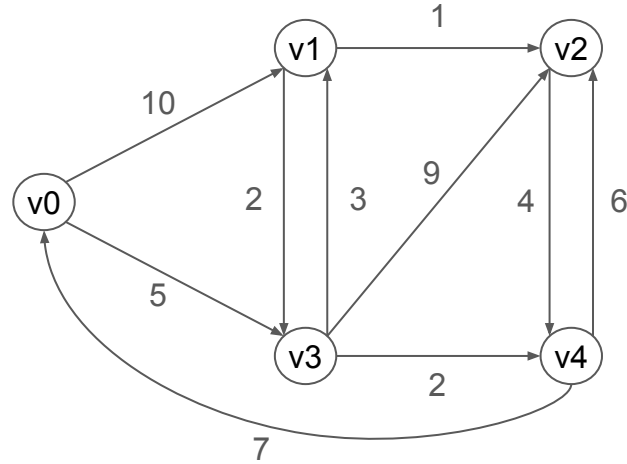
Dijkstra(G, s)

Inicialmente, T é uma árvore que consiste apenas no vértice s de G



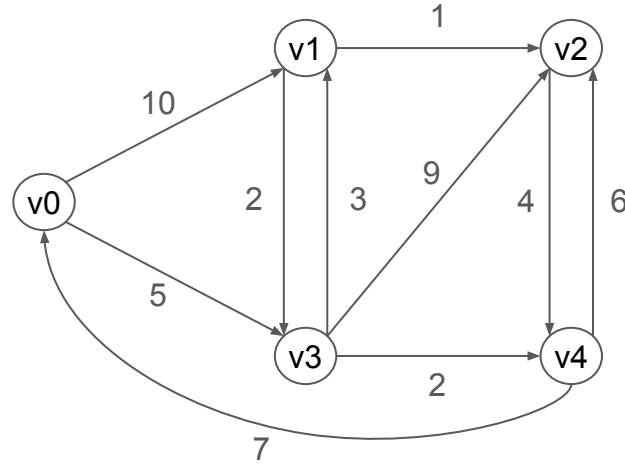
1. $T = (\{s\}, \emptyset)$
2. Enquanto é possível aumentar T :
3. Encontre um vértice w de G tal que w não está em T e a distância ponderada de s para w é mínima
4. Encontre uma aresta xw de G tal que x está em T e xw completa um caminho de peso mínimo de s para w
5. Adicione xw a T
6. Retorne T

Algoritmo de Dijkstra



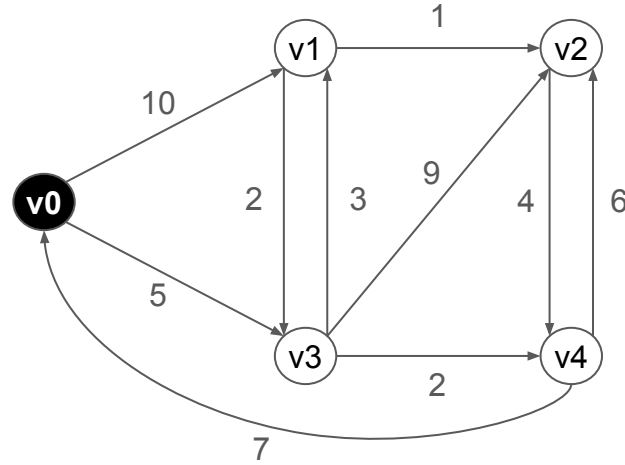
Algoritmo de Dijkstra

$S = v_0$



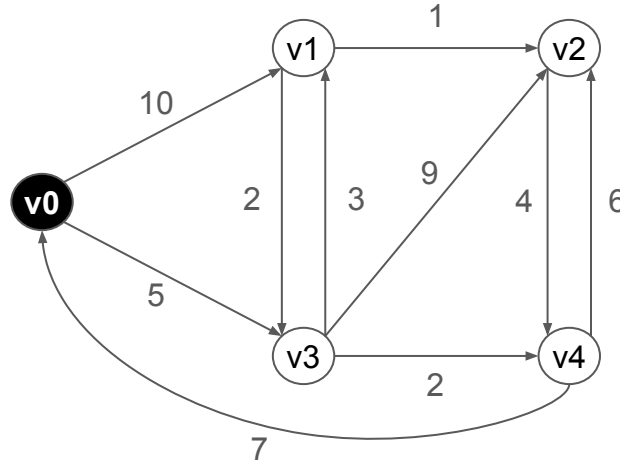
Algoritmo de Dijkstra

$S = v_0$



Algoritmo de Dijkstra

$S = v_0$



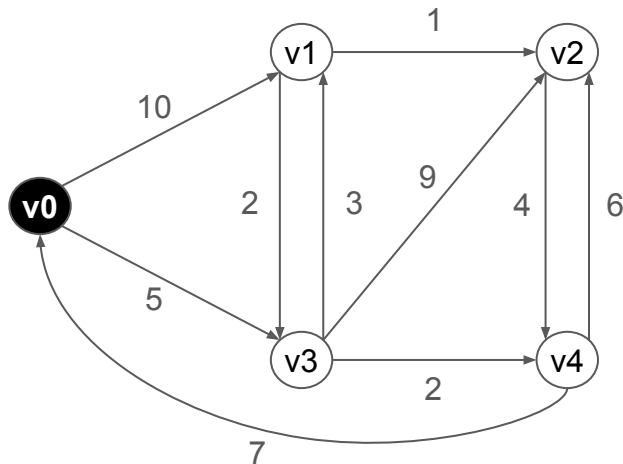
Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$
e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

Algoritmo de Dijkstra

$S = v_0$



Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$
e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

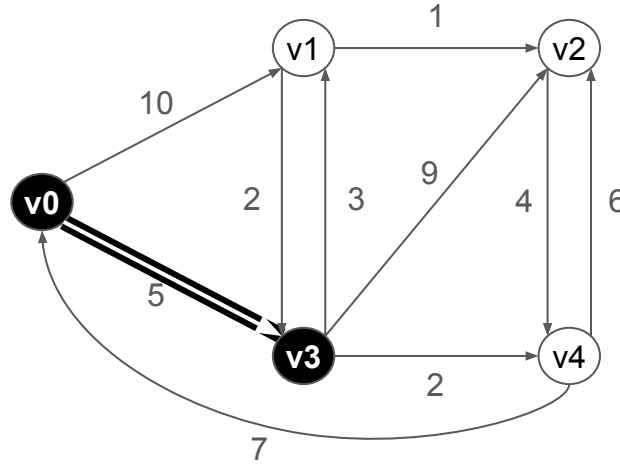
3. Encontre um vértice w de G tal que w não está em T e a distância ponderada de s para w é mínima
4. Encontre uma aresta xw de G tal que x está em T e xw completa um caminho de peso mínimo de s para w

v_3

$v_0 v_3$

Algoritmo de Dijkstra

$S = v_0$



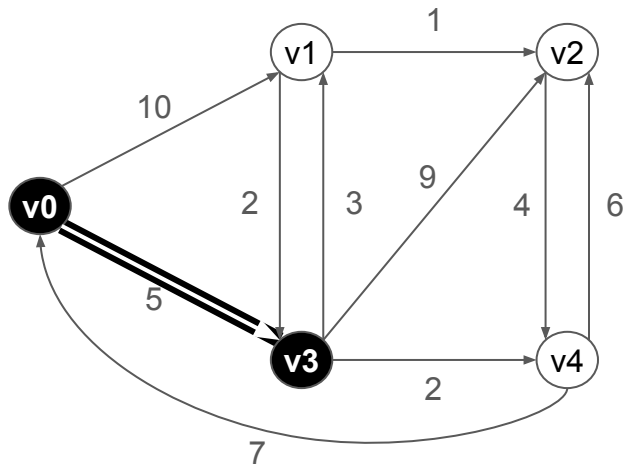
Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$
e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

Algoritmo de Dijkstra

$S = v_0$



Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$
e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

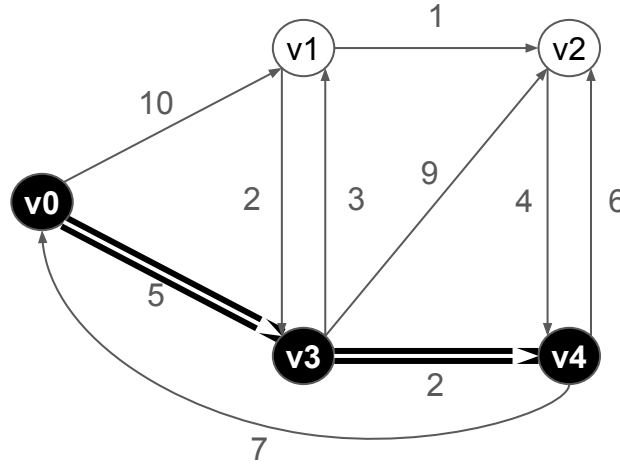
3. Encontre um vértice w de G tal que w não está em T e a distância ponderada de s para w é mínima
4. Encontre uma aresta xw de G tal que x está em T e xw completa um caminho de peso mínimo de s para w

v_4

$v_3 v_4$

Algoritmo de Dijkstra

$S = v_0$



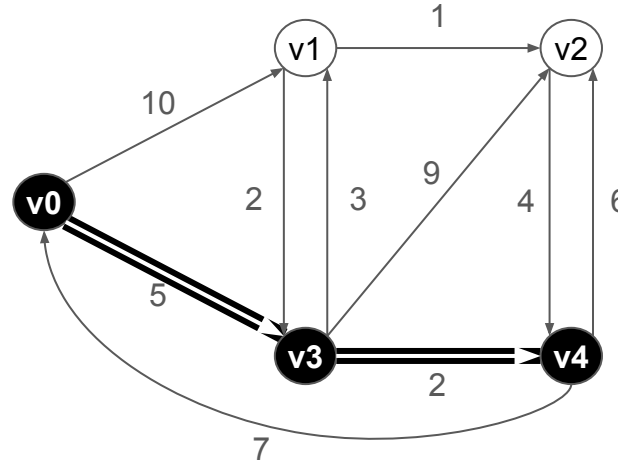
Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
 - $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
 - $v_0 v_3$ — $dp(v_0, v_3) = 5$
- e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

Algoritmo de Dijkstra

$S = v_0$



Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$
e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

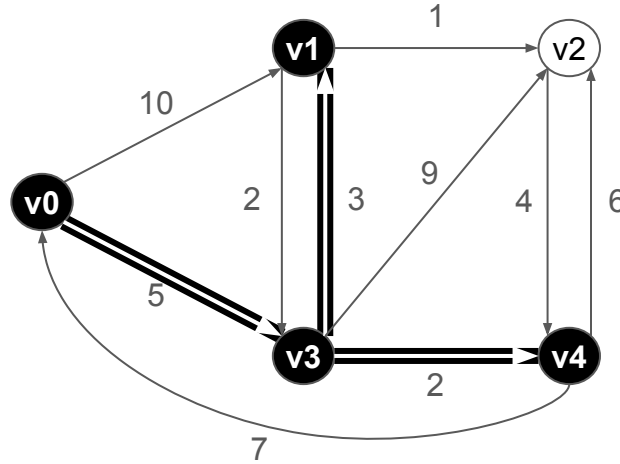
3. Encontre um vértice w de G tal que w não está em T e a distância ponderada de s para w é mínima
4. Encontre uma aresta xw de G tal que x está em T e xw completa um caminho de peso mínimo de s para w

v_1

$v_3 v_1$

Algoritmo de Dijkstra

$S = v_0$



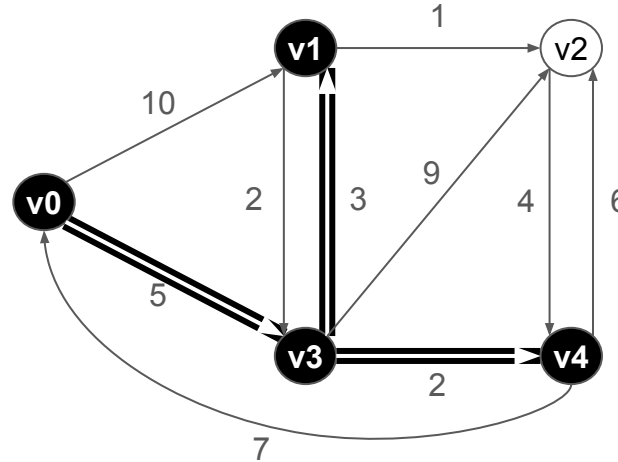
Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$
e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

Algoritmo de Dijkstra

$S = v_0$



Caminhos de peso mínimo

$S = v_0$

- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
- $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
- $v_0 v_3$ — $dp(v_0, v_3) = 5$
e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

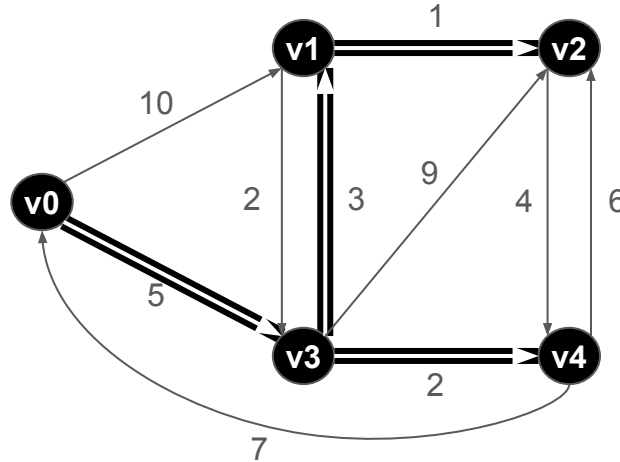
3. Encontre um vértice w de G tal que w não está em T e a distância ponderada de s para w é mínima
4. Encontre uma aresta xw de G tal que x está em T e xw completa um caminho de peso mínimo de s para w

v_2

$v_1 v_2$

Algoritmo de Dijkstra

$S = v_0$



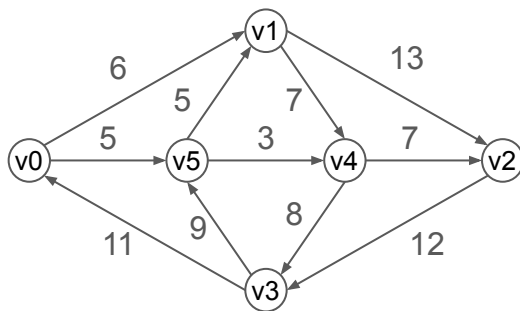
Caminhos de peso mínimo

$S = v_0$

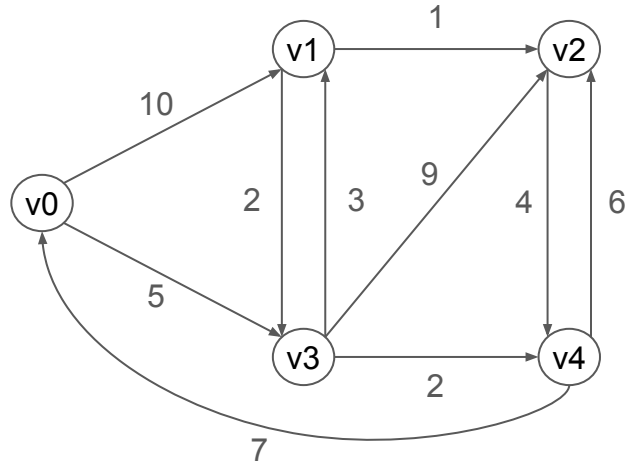
- $v_0 v_3 v_1$ — $dp(v_0, v_1) = 8$,
 - $v_0 v_3 v_1 v_2$ — $dp(v_0, v_2) = 9$,
 - $v_0 v_3$ — $dp(v_0, v_3) = 5$
- e
- $v_0 v_3 v_4$ — $dp(v_0, v_4) = 7$

Exercícios

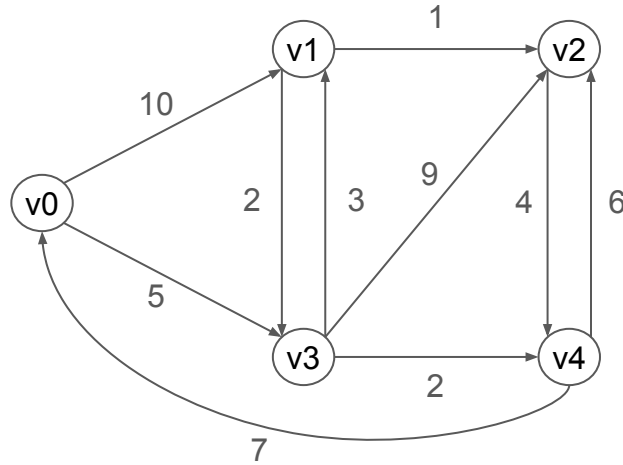
1. Execute o Algoritmo de Dijkstra para encontrar caminhos de peso mínimo do vértice v_0 para todos os outros vértices do grafo abaixo. Indique o seguinte:
 - a. A árvore de caminhos de peso mínimo obtida pelo algoritmo;
 - b. As distâncias ponderadas calculadas pelo algoritmo.



Algoritmo de Dijkstra - Implementação



Algoritmo de Dijkstra - Implementação



dp:

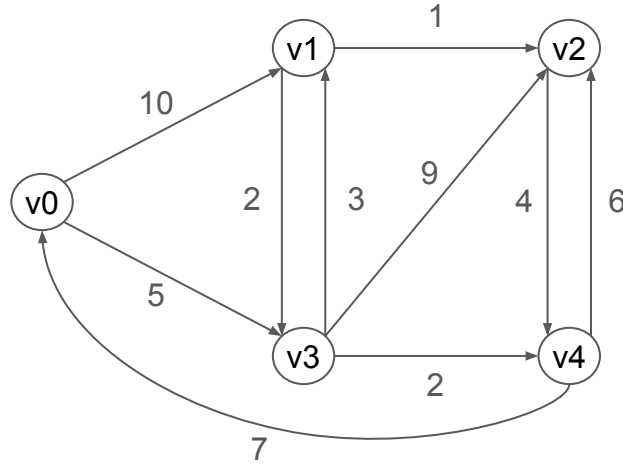
∞	∞	∞	∞	∞
0	1	2	3	4

pai:

-1	-1	-1	-1	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



dp:

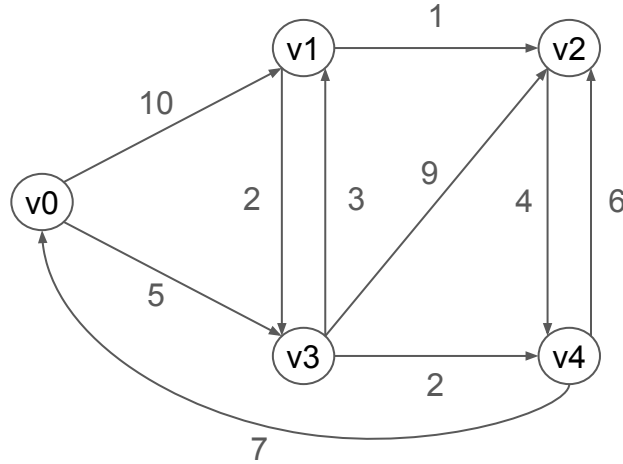
∞	∞	∞	∞	∞
0	1	2	3	4

pai:

-1	-1	-1	-1	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



dp:

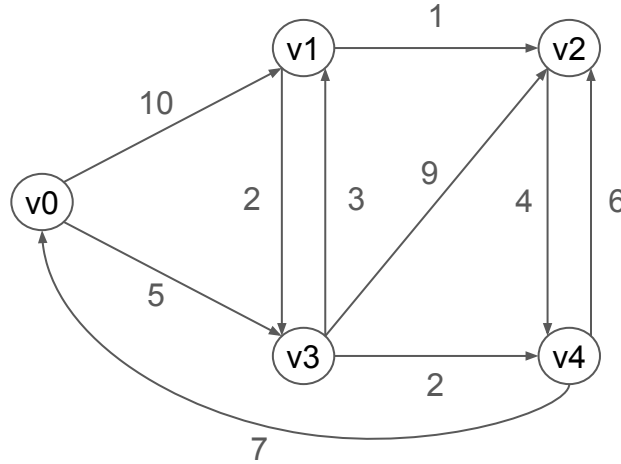
0	∞	∞	∞	∞
0	1	2	3	4

pai:

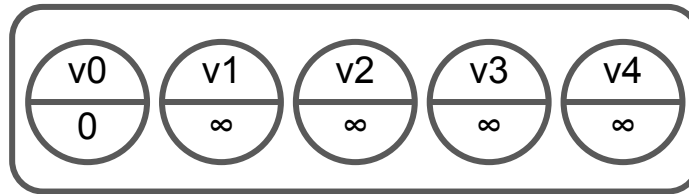
-1	-1	-1	-1	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

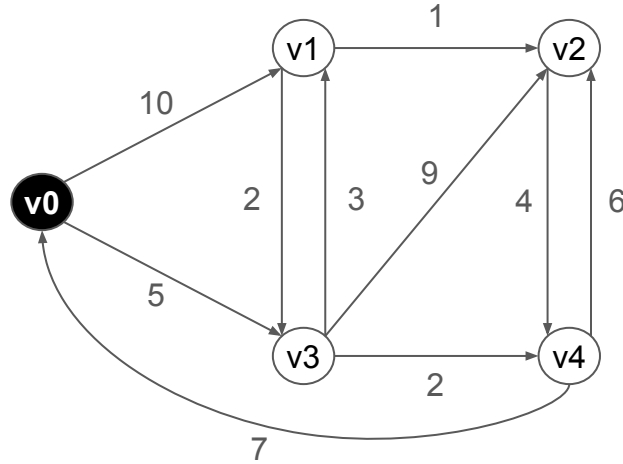
0	∞	∞	∞	∞
0	1	2	3	4

pai:

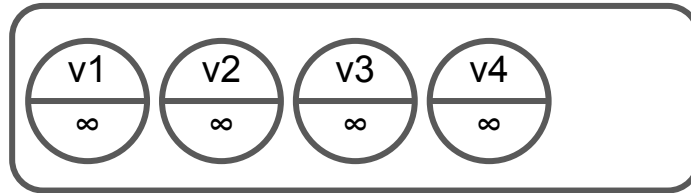
-1	-1	-1	-1	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

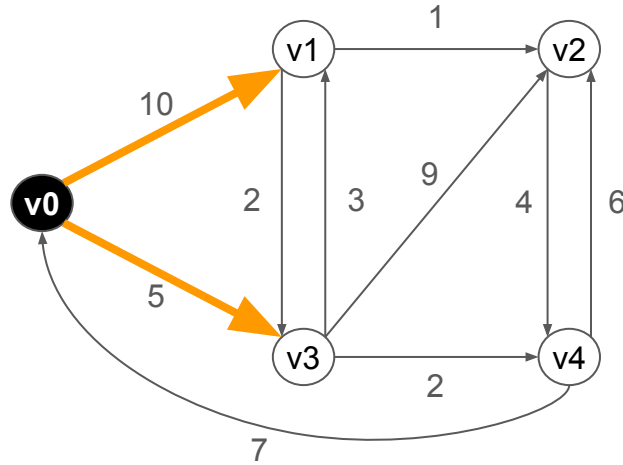
0	∞	∞	∞	∞
0	1	2	3	4

pai:

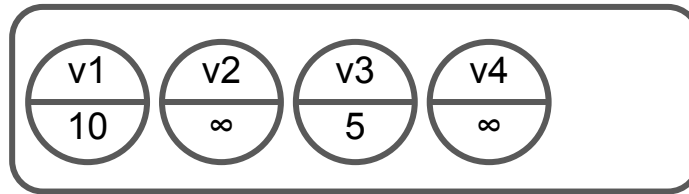
-1	-1	-1	-1	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

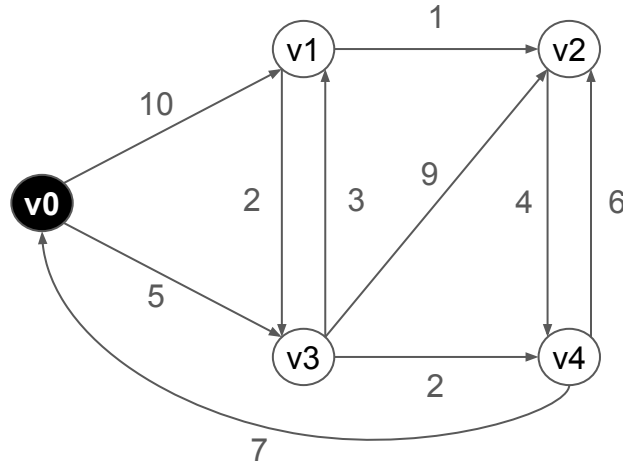
0	10	∞	5	∞
0	1	2	3	4

pai:

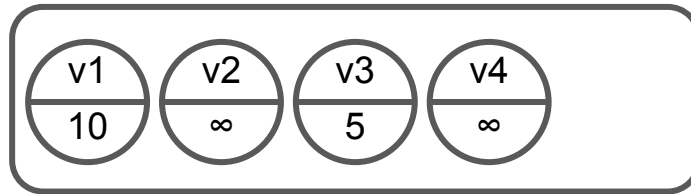
-1	0	-1	0	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

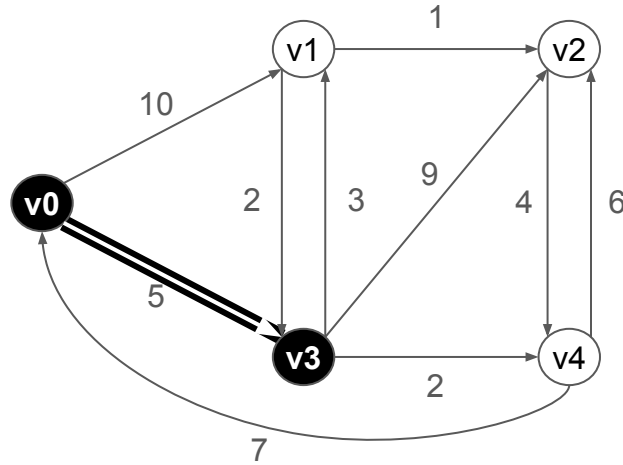
0	10	∞	5	∞
0	1	2	3	4

pai:

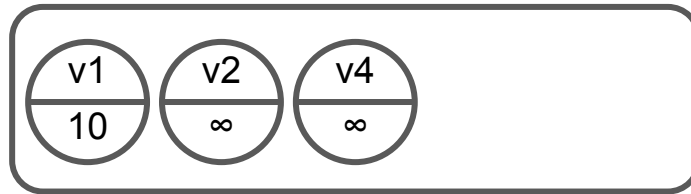
-1	0	-1	0	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

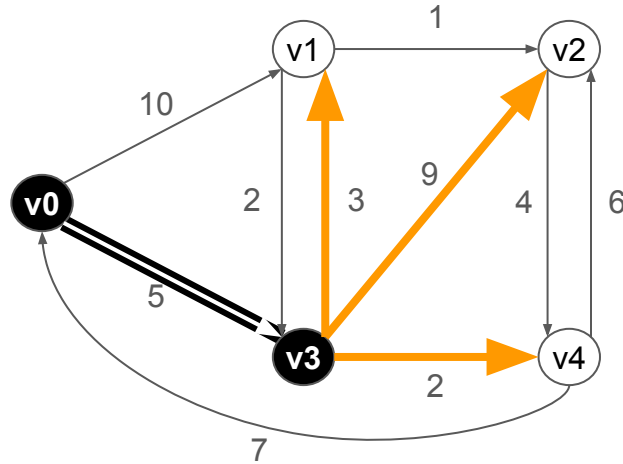
0	10	∞	5	∞
0	1	2	3	4

pai:

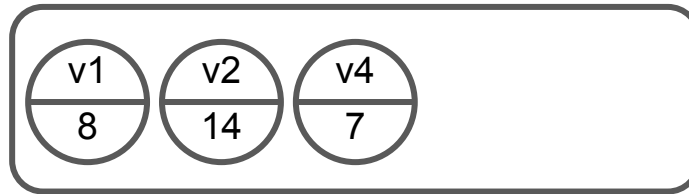
-1	0	-1	0	-1
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

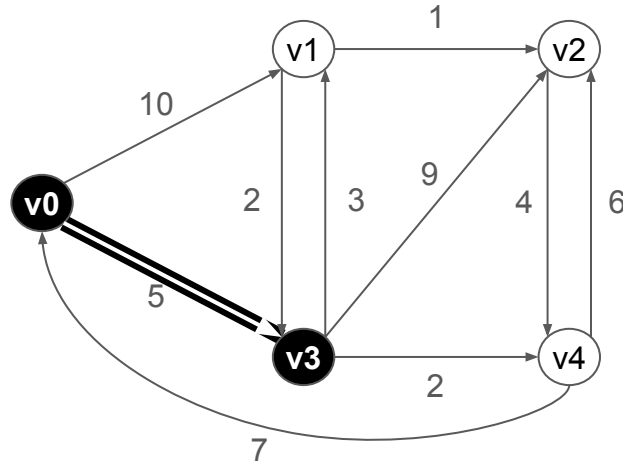
0	8	14	5	7
0	1	2	3	4

pai:

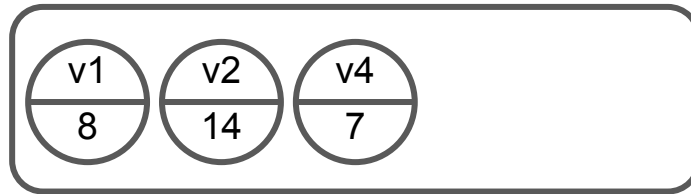
-1	3	3	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

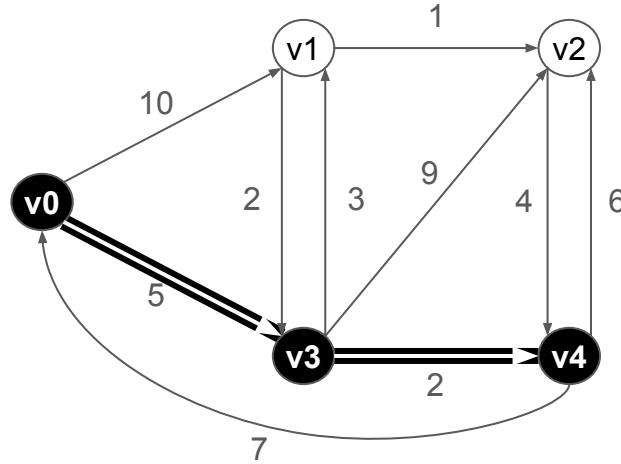
0	8	14	5	7
0	1	2	3	4

pai:

-1	3	3	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

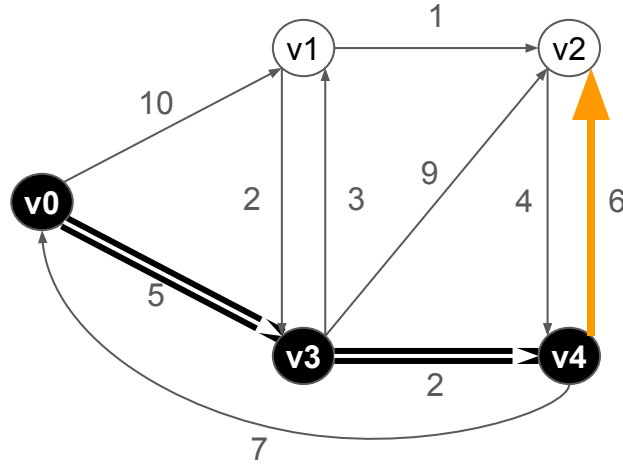
0	8	14	5	7
0	1	2	3	4

pai:

-1	3	3	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

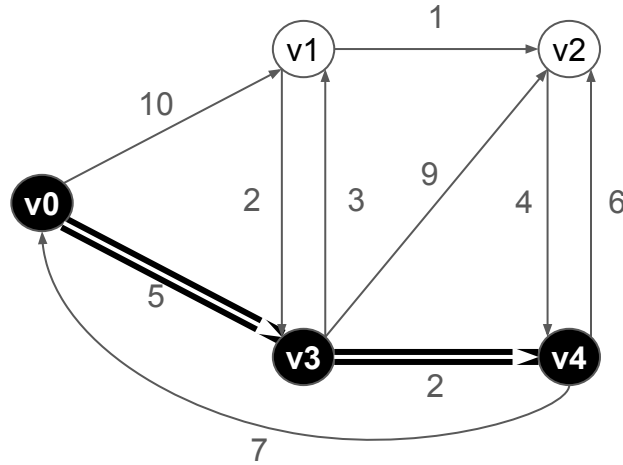
0	8	13	5	7
0	1	2	3	4

pai:

-1	3	4	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

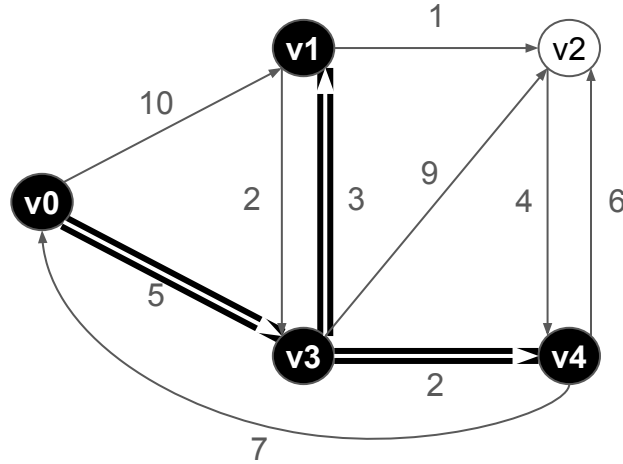
0	8	13	5	7
0	1	2	3	4

pai:

-1	3	4	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

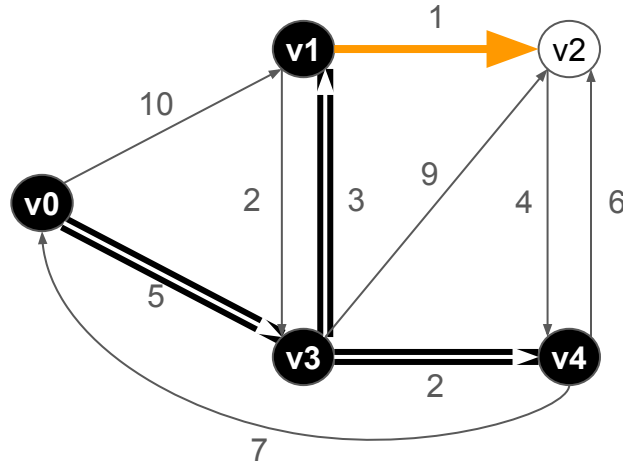
0	8	13	5	7
0	1	2	3	4

pai:

-1	3	4	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

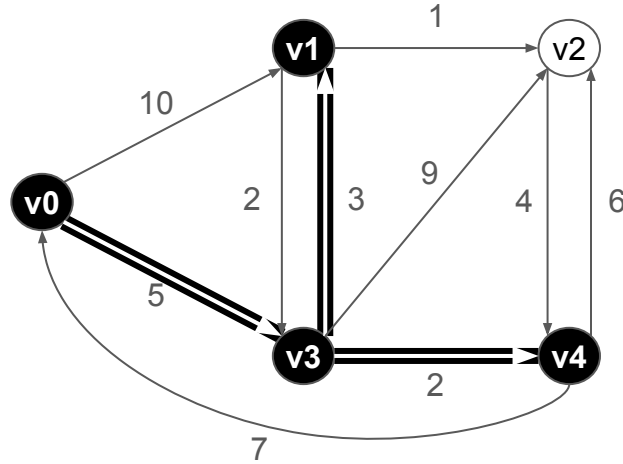
0	8	9	5	7
0	1	2	3	4

pai:

-1	3	1	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

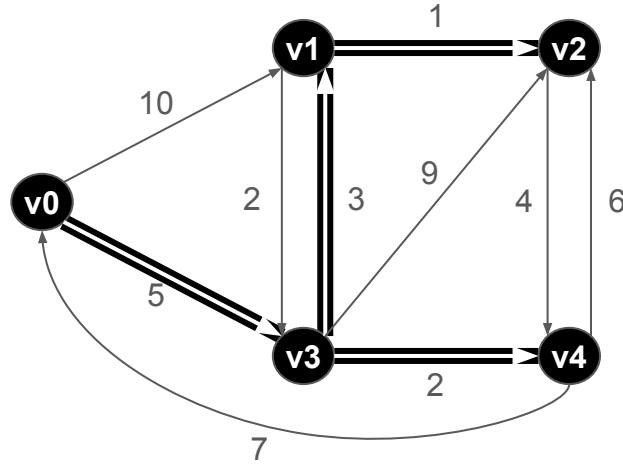
0	8	9	5	7
0	1	2	3	4

pai:

-1	3	1	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

$S = v_0$



Fila de prioridade:



dp:

0	8	9	5	7
0	1	2	3	4

pai:

-1	3	1	0	3
0	1	2	3	4

Algoritmo de Dijkstra - Implementação

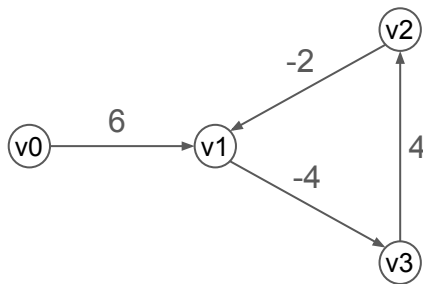
Dijkstra(G, s, pai, dp)

1. Para cada vértice w de G :
2. $dp[w] = \infty$
3. $pai[w] = -1$
4. $dp[s] = 0$
5. Crie uma fila de prioridade Q com todos os vértices de G e com a prioridade de cada vértice w sendo $dp[w]$
6. Enquanto Q não está vazia:
7. Remova o item de menor prioridade de Q ; seja u o item removido
8. Se $dp[u] \neq \infty$:
9. Para cada vizinho de saída v de u em G :
10. Se $dp[v] > dp[u] + p(uv)$: // $p(uv)$ é o peso da aresta uv
11. $dp[v] = dp[u] + p(uv)$
12. $pai[v] = u$
13. Altere a prioridade de v em Q para (o novo valor de) $dp[v]$

**Relaxação da
aresta uv**

Arestas com pesos negativos

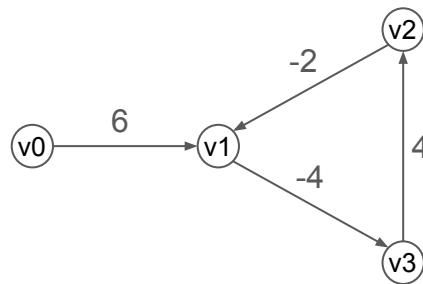
- Em algumas situações, faz sentido considerarmos arestas com pesos negativos
- Nestas situações, é possível termos **ciclos de peso negativo**
- Execute o Algoritmo de Dijkstra para o grafo abaixo. O que acontece?



Arestas com pesos negativos

- Dado um vértice v , se existe no digrafo um sv -passeio (um sv -caminho onde podem existir vértices repetidos) que contém um ciclo de peso negativo, então sempre podemos construir um sv -passeio de menor peso
- Como consequência, a noção de **distância ponderada** de s para v não tem um significado preciso

- Exemplo:
 - No digrafo ao lado,
 - $v_1 v_3 v_2 v_1$ é um ciclo de peso negativo



Arestas com pesos negativos

- Vamos considerar agora o problema dos caminhos de peso mínimo no caso em que **as arestas** do digrafo podem ter **peso negativo**
- Neste caso, o problema pode ser resolvido usando o Algoritmo de Bellman-Ford
- O Algoritmo de Bellman-Ford retorna
 - **falso** caso exista um ciclo de peso negativo alcançável a partir de s ou
 - **verdadeiro** caso contrário, retornando também os caminhos mínimos encontrados e os seus pesos

Algoritmo de Bellman-Ford

Bellman-Ford(G, s, pai, dp)

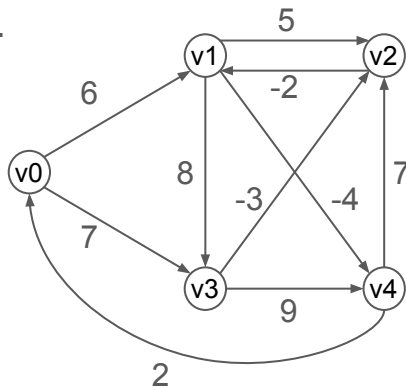
1. Para cada vértice w de G :
2. $dp[w] = \infty$
3. $pai[w] = -1$
4. $dp[s] = 0$
5. Para $i = 1$ até $|V(G)| - 1$:
6. Para cada aresta uv de G : // $p(uv)$ é o peso da aresta uv
7. Se $dp[u] \neq \infty$ e $dp[v] > dp[u] + p(uv)$:
8. $dp[v] = dp[u] + p(uv)$
9. $pai[v] = u$
10. Para cada aresta uv de G :
11. Se $dp[u] \neq \infty$ e $dp[v] > dp[u] + p(uv)$:
12. Retorne **falso**
13. Retorne **verdadeiro**

} Relaxação da
aresta uv

} Este código detecta se existe um
ciclo de peso negativo em G

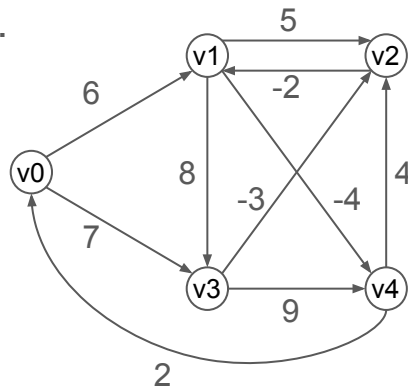
Exercícios

2. Execute o Algoritmo de Bellman-Ford para encontrar caminhos de peso mínimo do vértice v_0 para todos os outros vértices do grafo abaixo. Indique o seguinte:
- A árvore de caminhos de peso mínimo obtida pelo algoritmo;
 - As distâncias ponderadas calculadas pelo algoritmo;
 - O valor de retorno do algoritmo.



Exercícios

3. Execute o Algoritmo de Bellman-Ford para encontrar caminhos de peso mínimo do vértice v_0 para todos os outros vértices do grafo abaixo. Indique o seguinte:
- A árvore de caminhos de peso mínimo obtida pelo algoritmo;
 - As distâncias ponderadas calculadas pelo algoritmo;
 - O valor de retorno do algoritmo.



Referências

- Esta apresentação é baseada nos seguintes materiais:
 1. Capítulo 24 do livro
Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to Algorithms. 3rd. ed. MIT Press, 2009.
 2. Capítulo 21 do livro
Sedgewick, R. Algorithms in C++ – Part 5. Graph Algorithms. 3rd. ed. Addison-Wesley, 2002.