INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO
Campus Campos do Jordão

ESTRUTURA DE DADOS

Grafos

Algoritmo de Dijkstra

Professor Mestre Igor de Moraes Sampaio
igor.sampaio@ifsp.edu.br

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - **Campus Campos do Jordão**

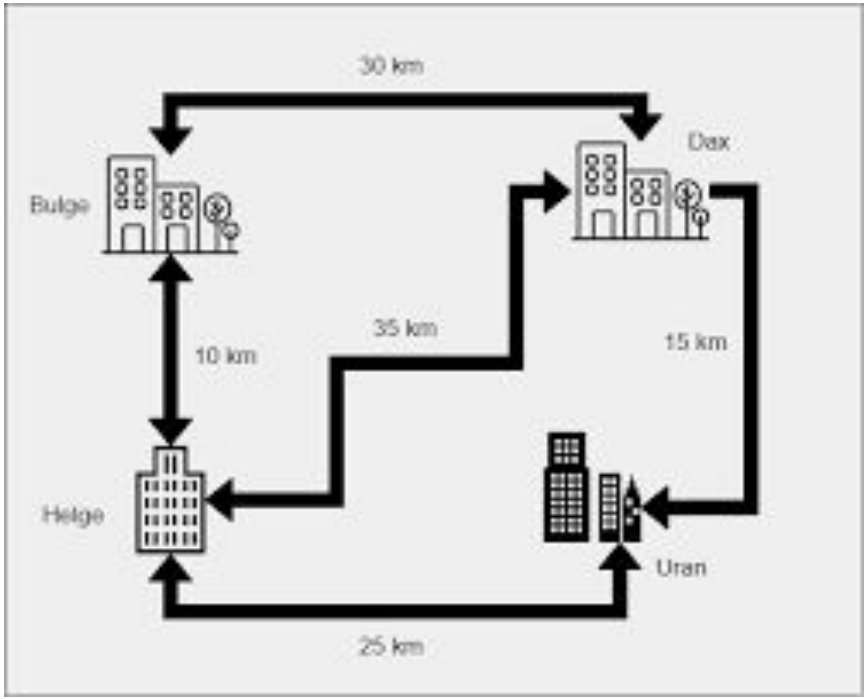
Contexto

Problema do Caixeiro Viajante

O problema do caixeiro viajante (ou Travelling Salesman Problem – TSP) é um clássico da matemática, ciência da computação e otimização.

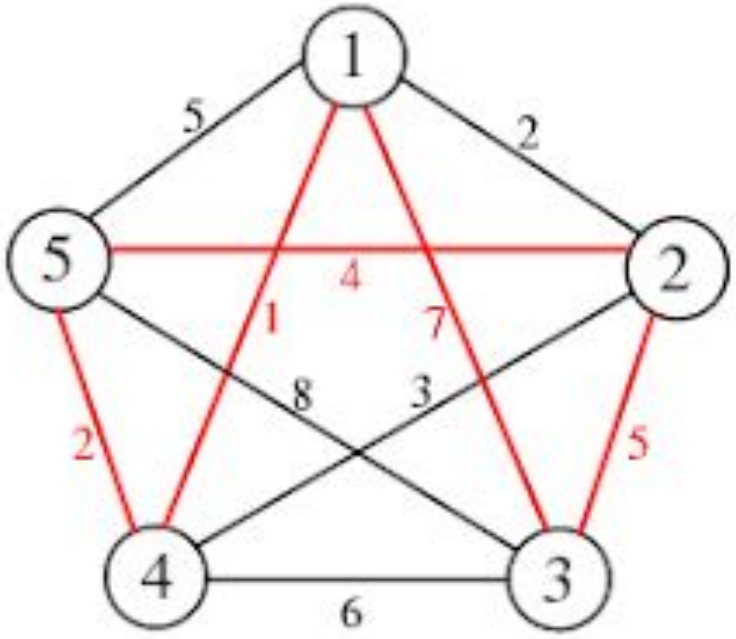
Enunciado:
Dado um conjunto de cidades e as distâncias entre cada par delas, qual é o caminho mais curto que permite ao caixeiro:

- Visitar cada cidade exatamente uma vez, e
- Retornar à cidade de origem?



A Contribuição de Euler

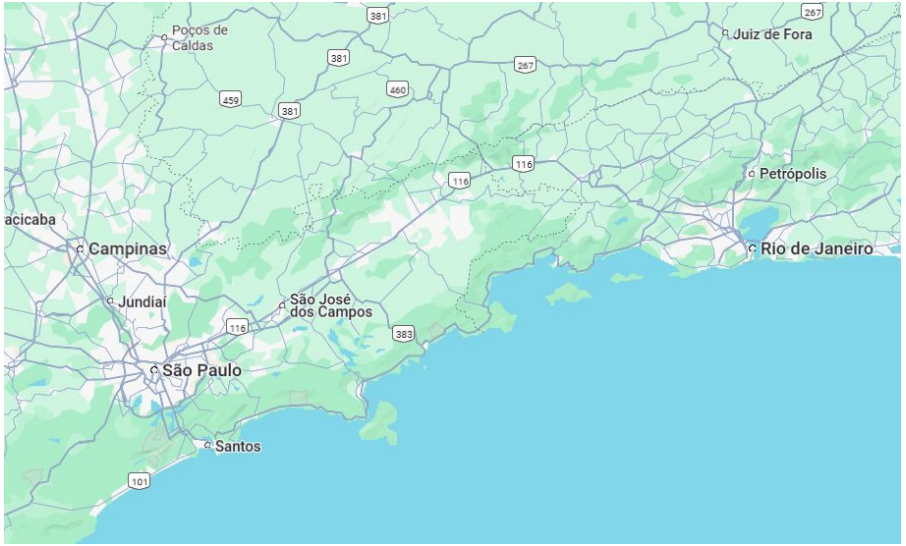
- Representado por um grafo ponderado completo, onde:
  - Os vértices são as cidades,
  - As arestas têm pesos representando as distâncias ou custos de viagem.
- É um exemplo típico de problema NP-difícil: não há solução eficiente conhecida para casos grandes.
- Aplicações práticas:
  - Logística e roteamento,
  - Planejamento de entregas,
  - Bioinformática (ex: montagem de genomas),
  - Impressão 3D, entre outros.



Caminho de Peso Mínimo

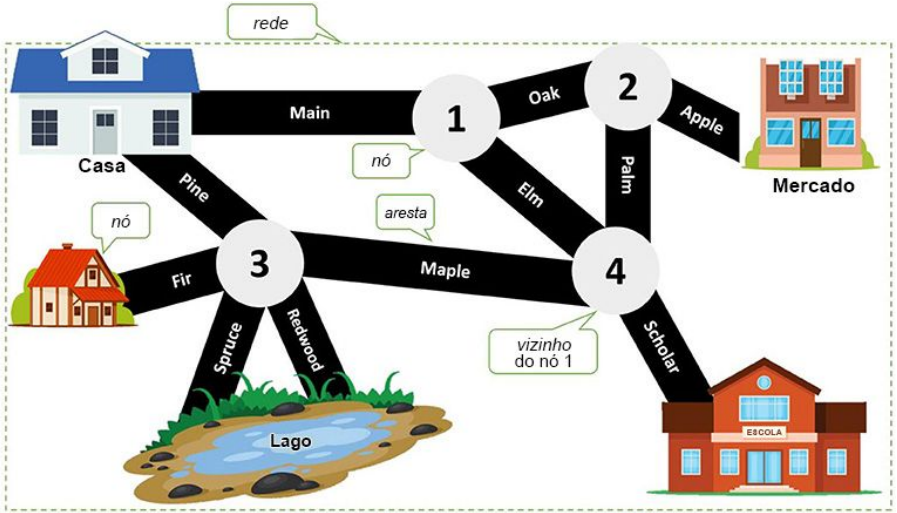
- Suponha que você deseja encontrar o caminho mais curto possível do Rio de Janeiro a São Paulo.

- Como determinar a rota mais curta?



O Problema do Caminho Mais Curto

- **Objetivo:**
  - Encontrar o caminho de menor custo total entre dois vértices em um grafo ponderado.
- **Entrada:**
  - Um grafo G = (V, E) com pesos não-negativos nas arestas, um vértice de origem s.
- **Saída:**
  - Para cada vértice v ∈ V, o caminho mais curto de s até v.



Versões

- **Origem Única:**
  - Encontrar caminhos mais curtos de um vértice para todos os outros.
- **Destino Único:**
  - Encontrar caminhos mais curtos de todos os vértices para um vértice específico.
- **Par Único:**
  - Encontrar o caminho mais curto entre dois vértices específicos.
- **Todos os Pares:**
  - Encontrar caminhos mais curtos entre todos os pares de vértices.

Algoritmo de Dijkstra

Algoritmo de Dijkstra

Quem foi Edsger W. Dijkstra?

Cientista da computação holandês (1930-2002)

Pioneiro em várias áreas da ciência da computação

Prêmio Turing em 1972

Publicou o algoritmo em 1959

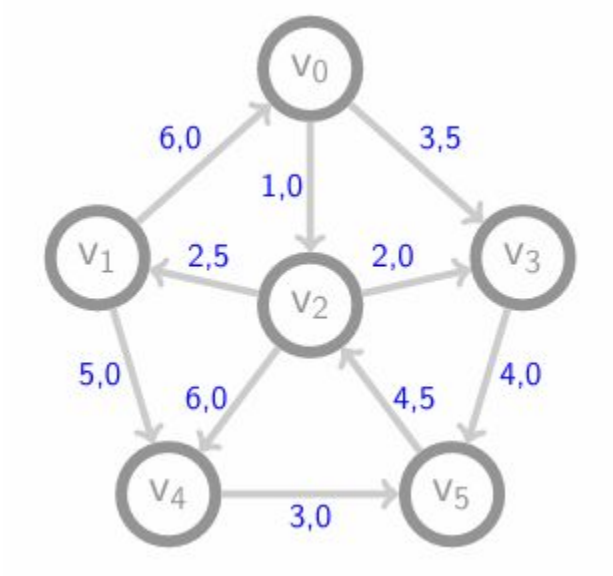
"O algoritmo de Dijkstra resolve o problema do caminho mais curto de origem única em um grafo ponderado com pesos não-negativos."

- **Abordagem Gulosa:**
  - A cada passo, escolhe o vértice não visitado com a menor distância conhecida da origem.
- **Relaxamento:**
  - Atualiza as distâncias dos vizinhos se um caminho mais curto for encontrado através do vértice atual.
- **Garantia:**
  - Ao final, encontra o caminho mais curto da origem para todos os vértices alcançáveis.

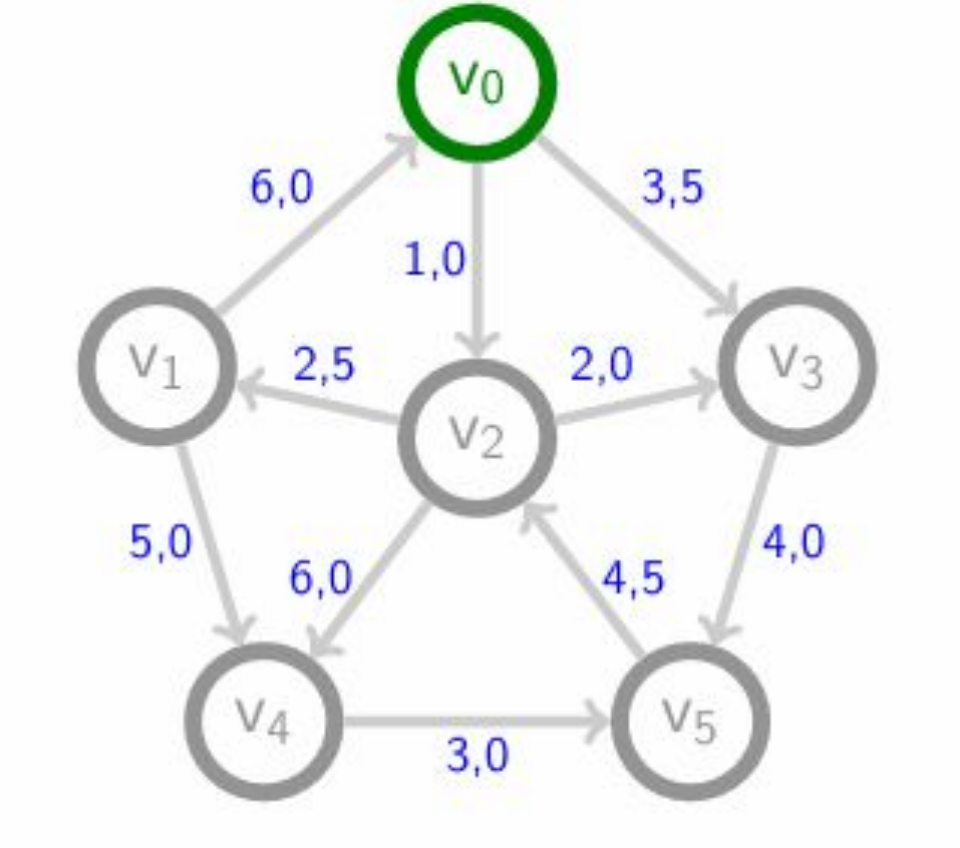
Árvore de Caminhos de Peso Mínimo

Árvore de Caminhos de Peso Mínimo

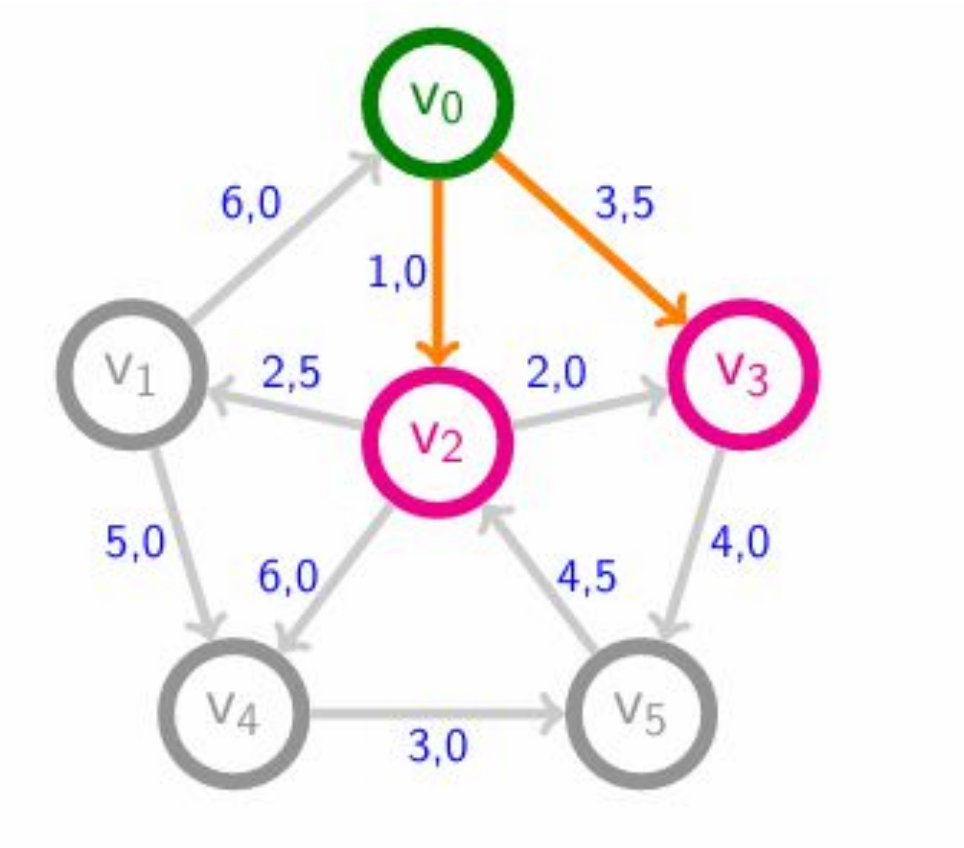
- É uma árvore enraizada que contém um caminho mais curto desde a origem s até todo vértice acessível a partir de s.



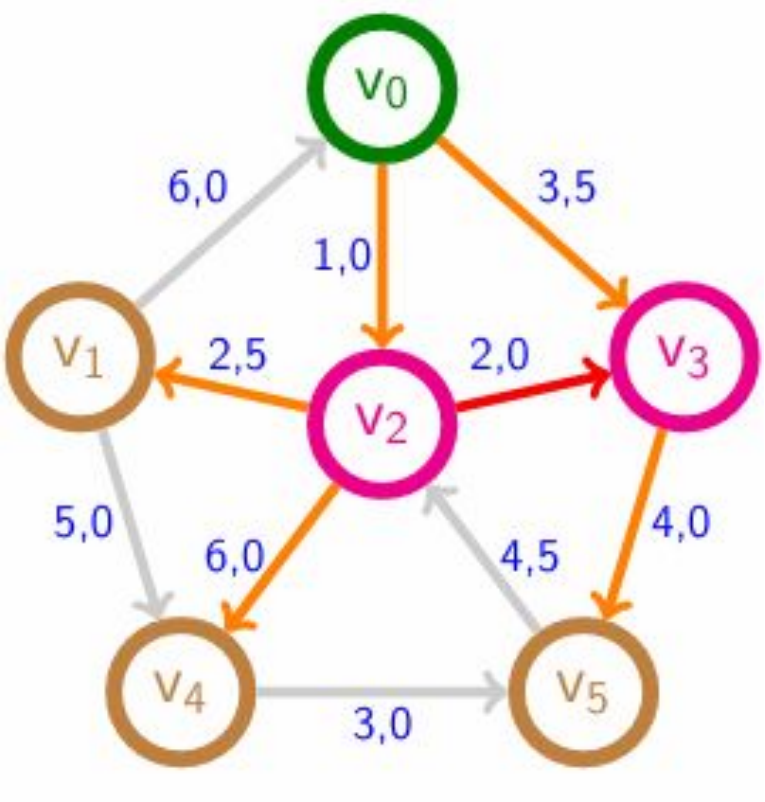
Árvore de Caminhos de Peso Mínimo



Árvore de Caminhos de Peso Mínimo



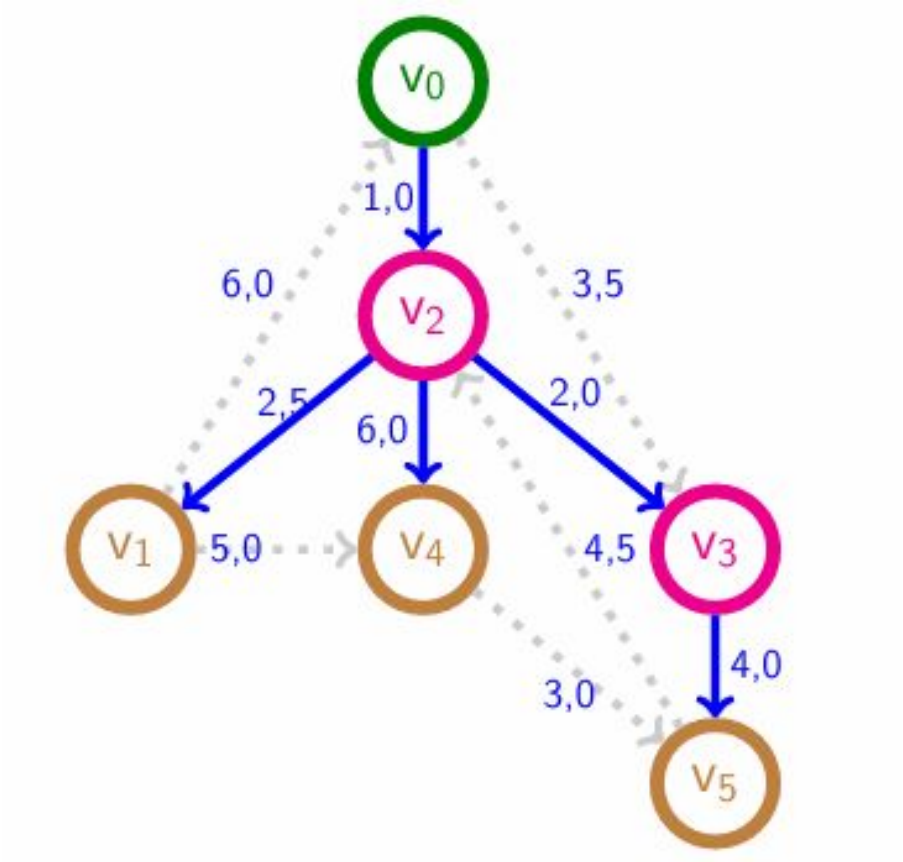
Árvore de Caminhos de Peso Mínimo



- Existe um caminho de menor custo (‘atalho’) de v0 para v3, passando por v2.

- A operação de atualizar a distância entre dois vértices, passando por outro(s) vértice(s) é chamada de relaxamento.

Árvore de Caminhos de Peso Mínimo



Processo

Início

- Objetivo:
  - Encontrar o caminho mais curto de um vértice (s) para todos os outros vértices (v).

- Para cada vértice v do grafo, mantemos um atributo d[v] que é um limite superior para o peso do caminho mais curto do nó inicial s a v.

- Dizemos que d[v] é uma estimativa de caminho mais curto, inicialmente feito ∞.

- Também armazenamos o vértice que precede v (p[v] - precedente de v) no caminho mais curto de s a v.

Início

- **Faça a estimativa de distância de s a qualquer vértice ser infinita:**
  - Exceto, claro, a distância de s a s que é 0.
  - Ou seja, d[s] = 0 e d[v] = ∞ para todo v ≠ s

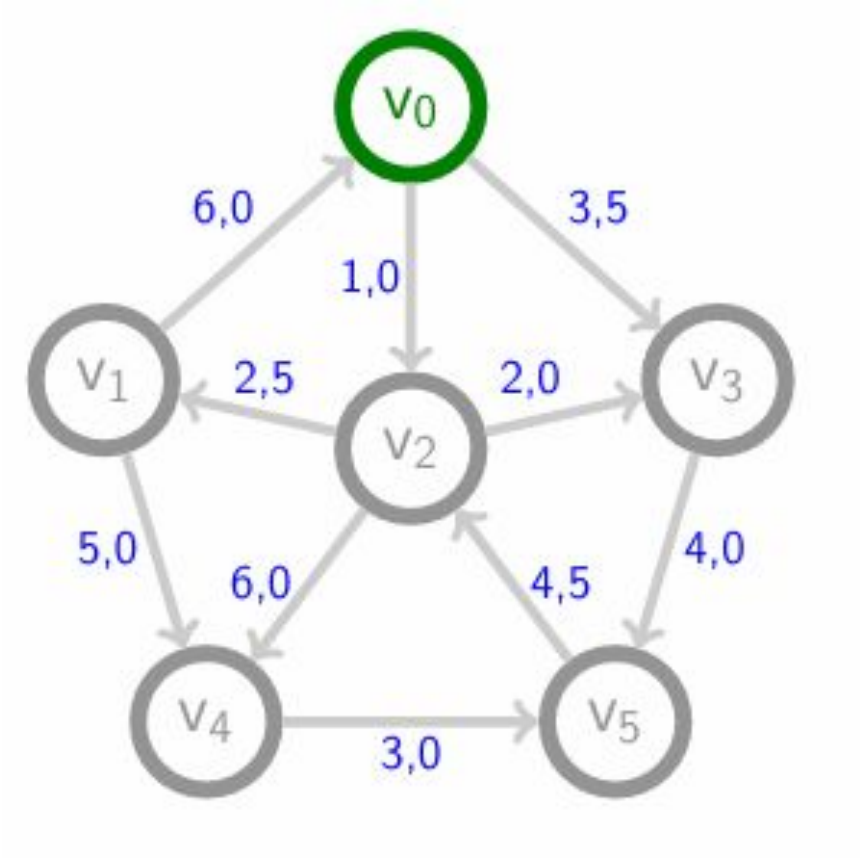
- **Faça os precedentes dos nós serem um valor qualquer.**
  - Na prática, podemos fazer p[v] = -1, já que não temos vértices de índice -1 na estrutura Grafo.

- **Marco todos os vértices como "não descobertos"**

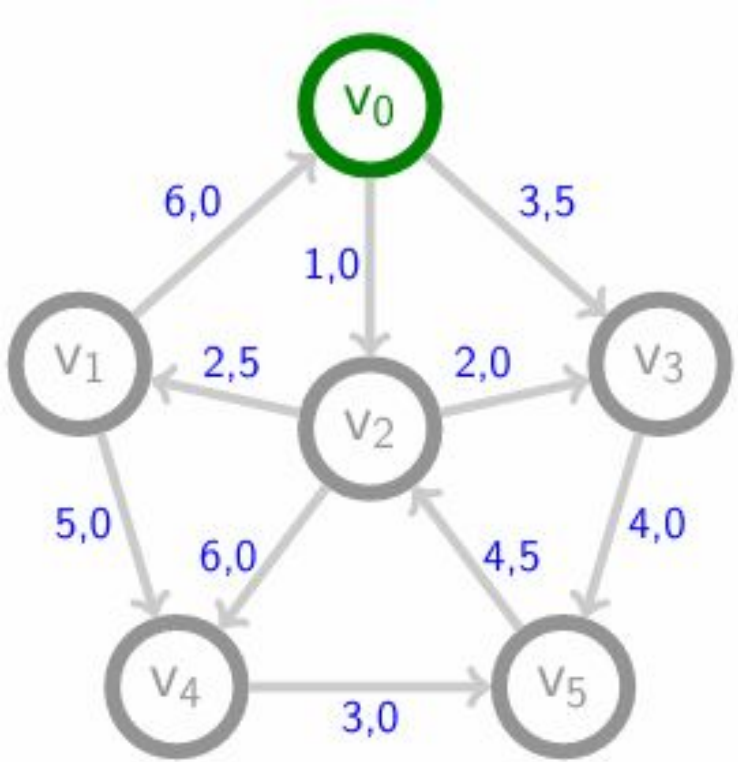
Processo

- **Enquanto houver vértice não descoberto:**
  - Escolha o vértice não descoberto u cuja estimativa seja a menor dentre os demais abertos.
  - Marque como descoberto o vértice u.
  - Para todo nó não descoberto v na adjacência de u:
    - Some d[u] ao peso da aresta (u,v)
    - Caso a soma seja menor que d[v], atualize d[v] e faça ρ[v] = u
    - Procedimento chamado de relaxamento da aresta (u, v)

Dijkstra - Exemplo



Dijkstra - Exemplo



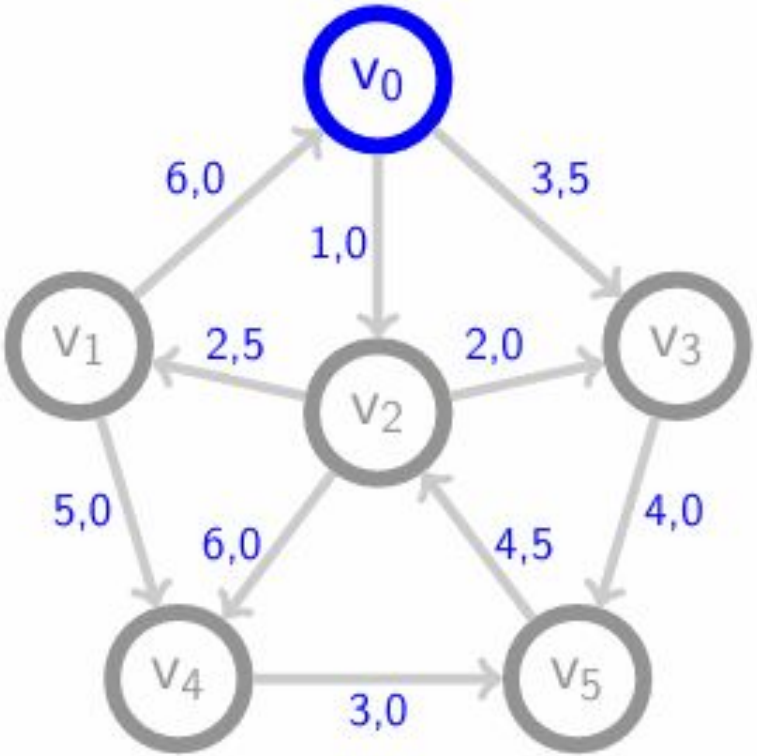
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5

Dijkstra - Exemplo



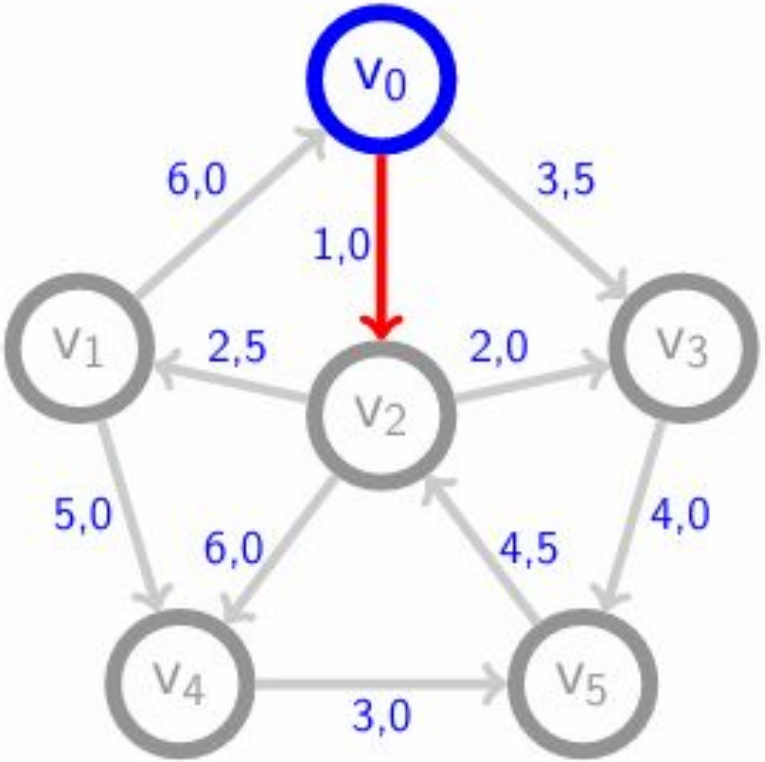
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	-	-	-	-	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	-1	-1	-1	-1	-1

Dijkstra - Exemplo



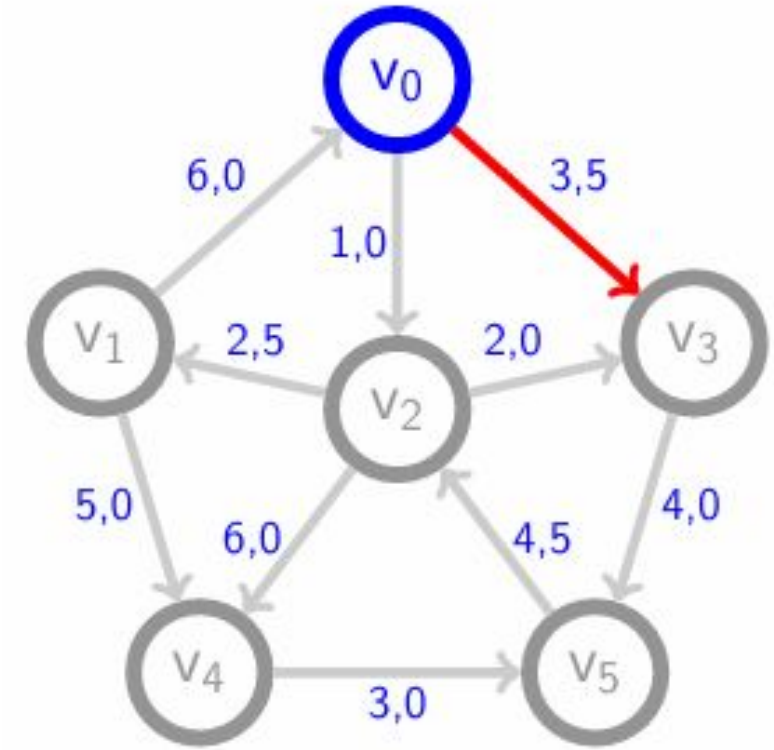
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	-	1,0	-	-	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	-1	0	-1	-1	-1

Dijkstra - Exemplo



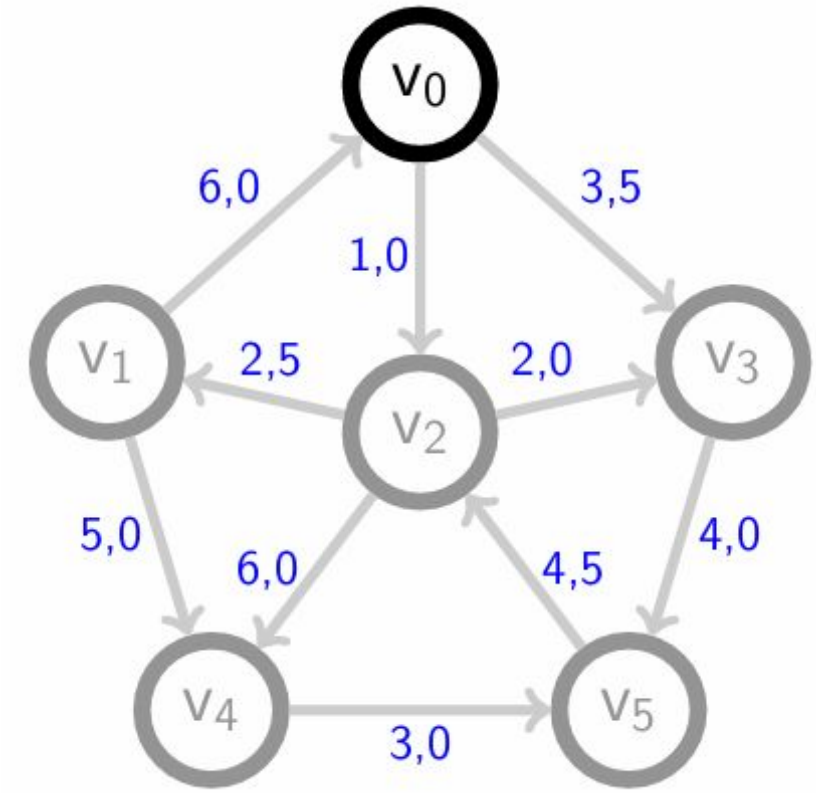
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	-	1,0	3,5	-	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	-1	0	0	-1	-1

Dijkstra - Exemplo



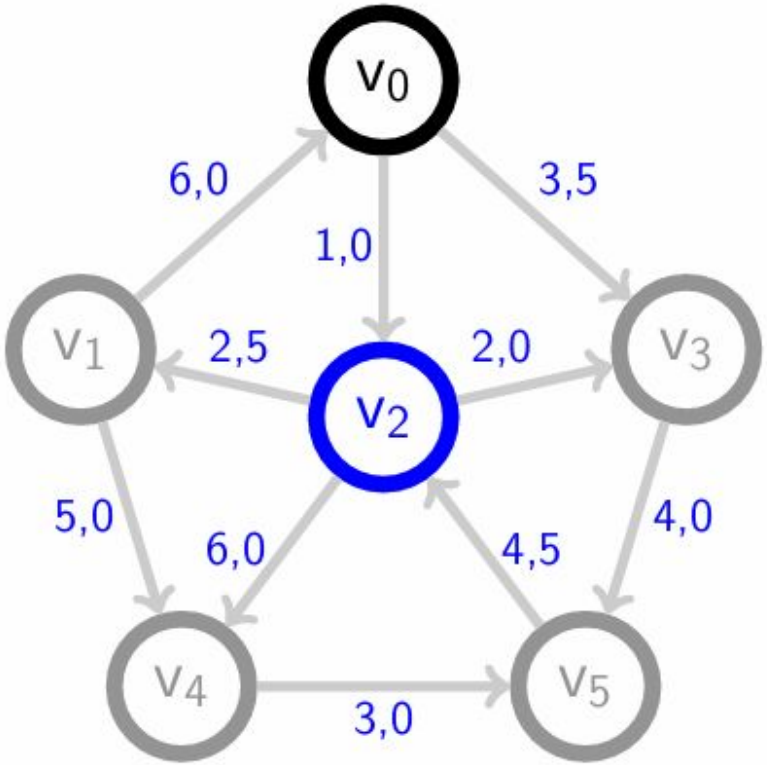
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	-	1,0	3,5	-	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	-1	0	0	-1	-1

Dijkstra - Exemplo



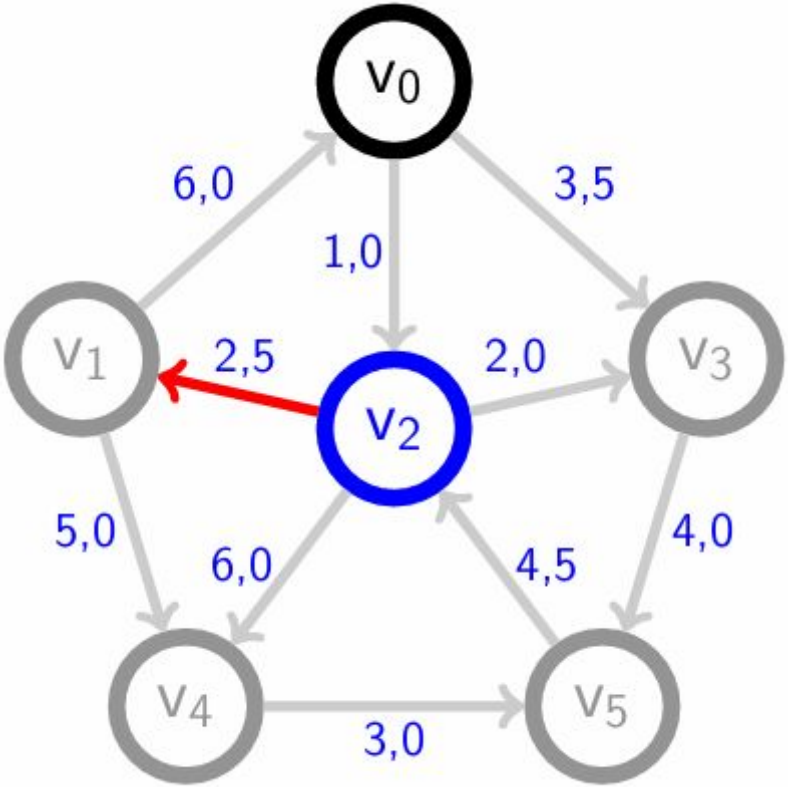
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	-	1,0	3,5	-	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	-1	0	0	-1	-1

Dijkstra - Exemplo



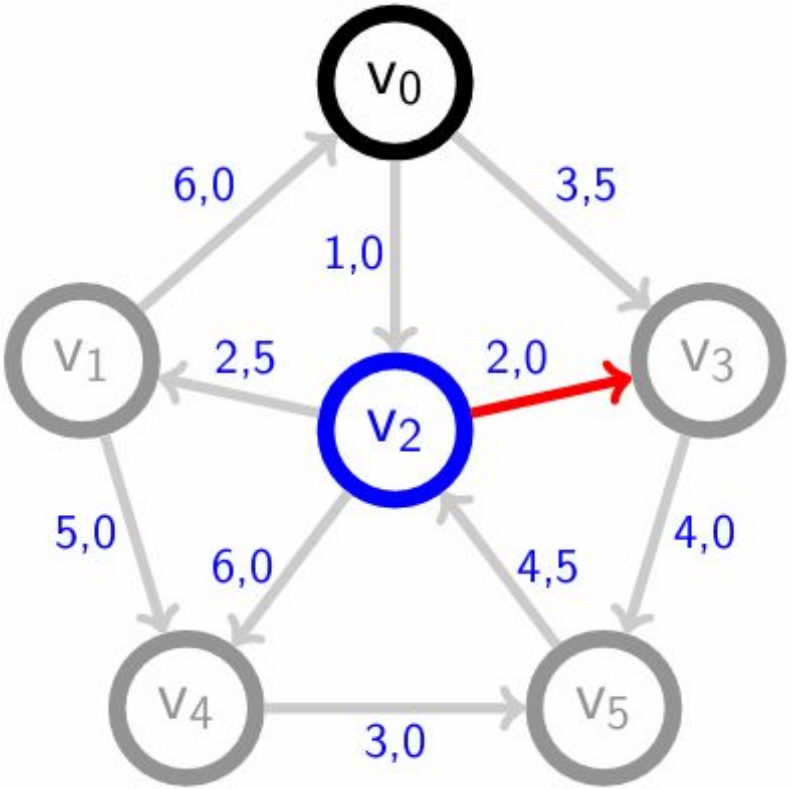
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,5	-	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	0	-1	-1

Dijkstra - Exemplo



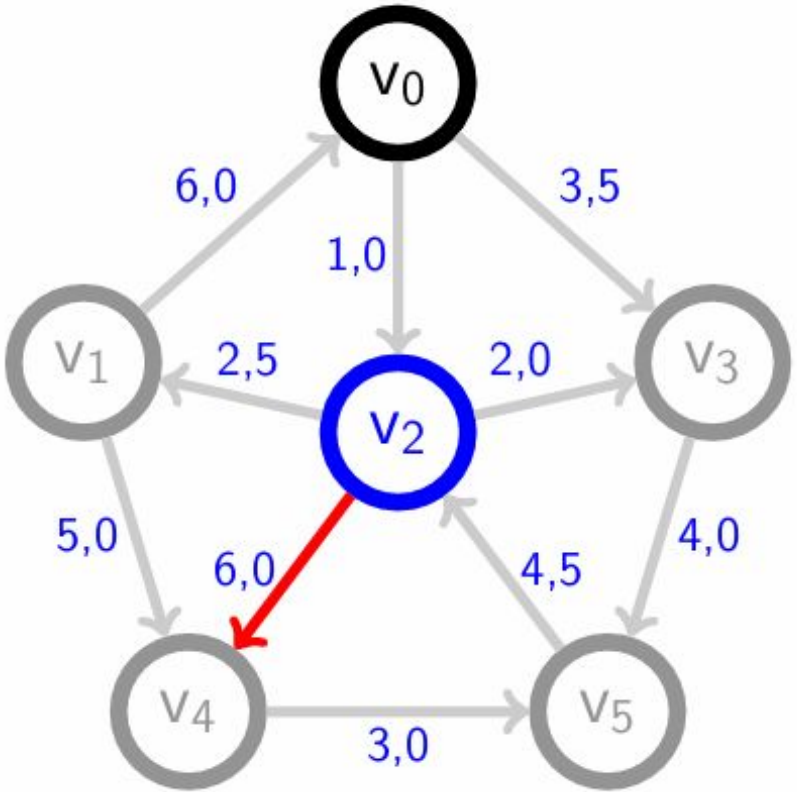
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	-	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	-1	-1

Dijkstra - Exemplo



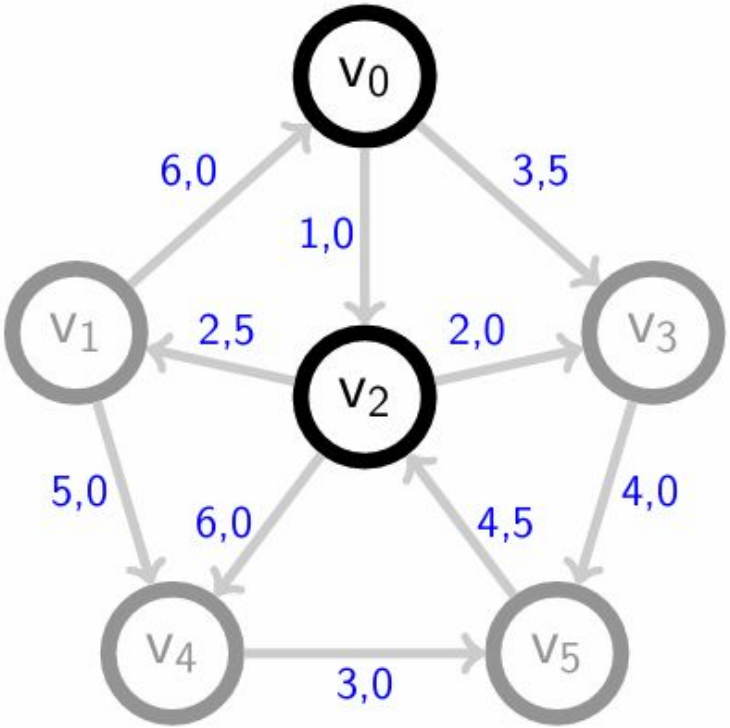
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	-1

Dijkstra - Exemplo



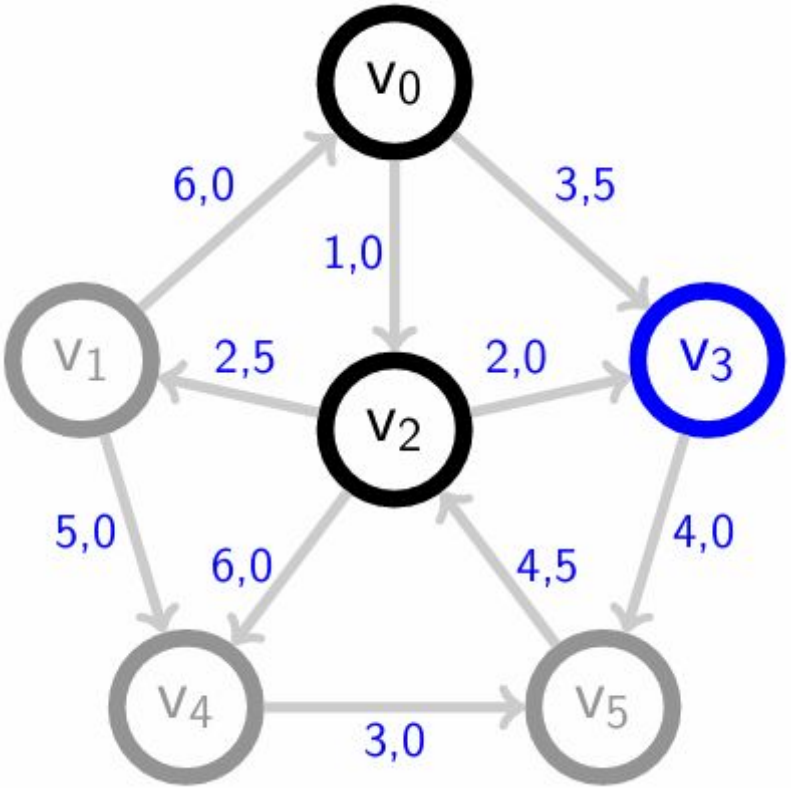
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	-1

Dijkstra - Exemplo



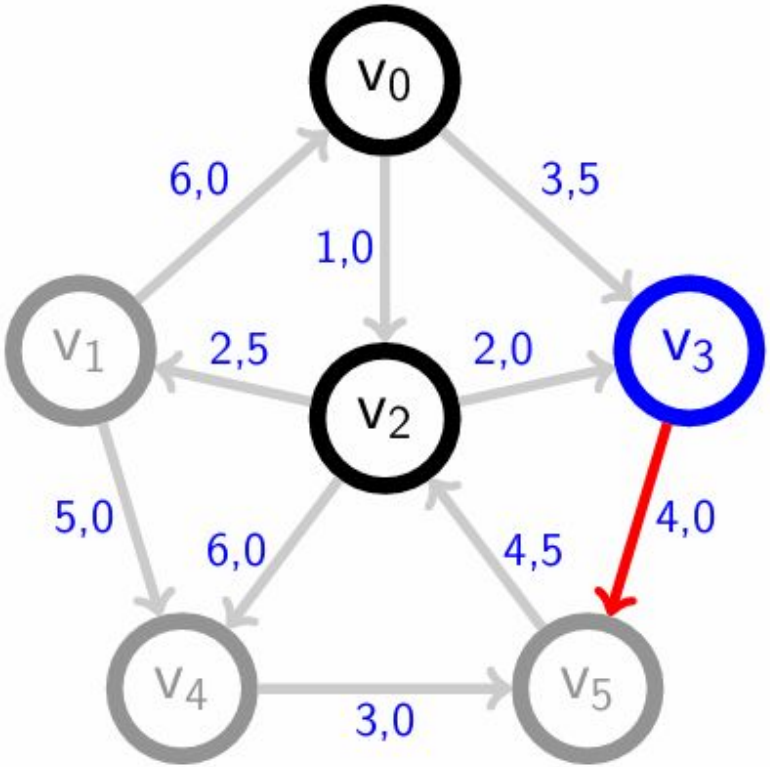
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	-

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	-1

Dijkstra - Exemplo



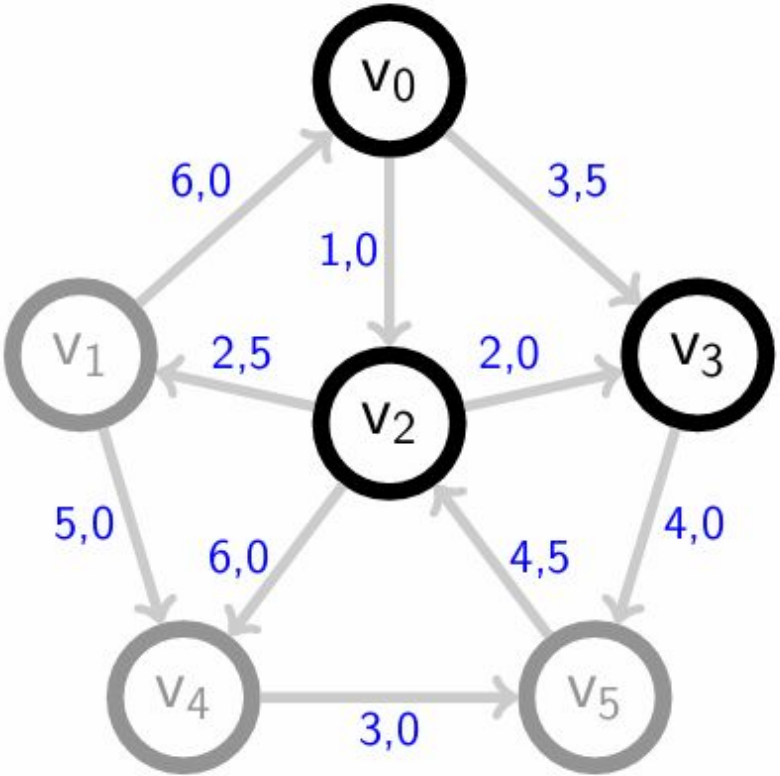
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



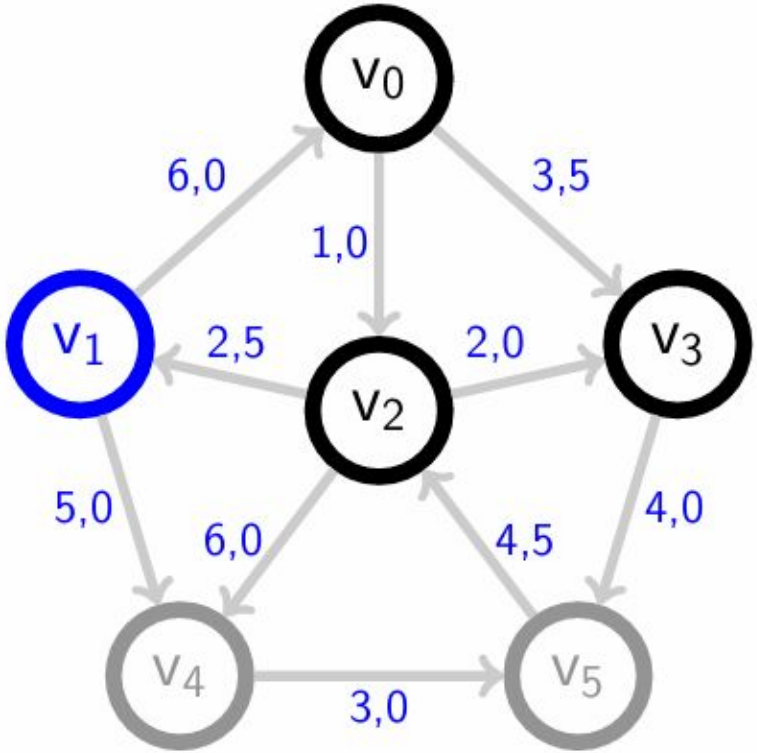
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



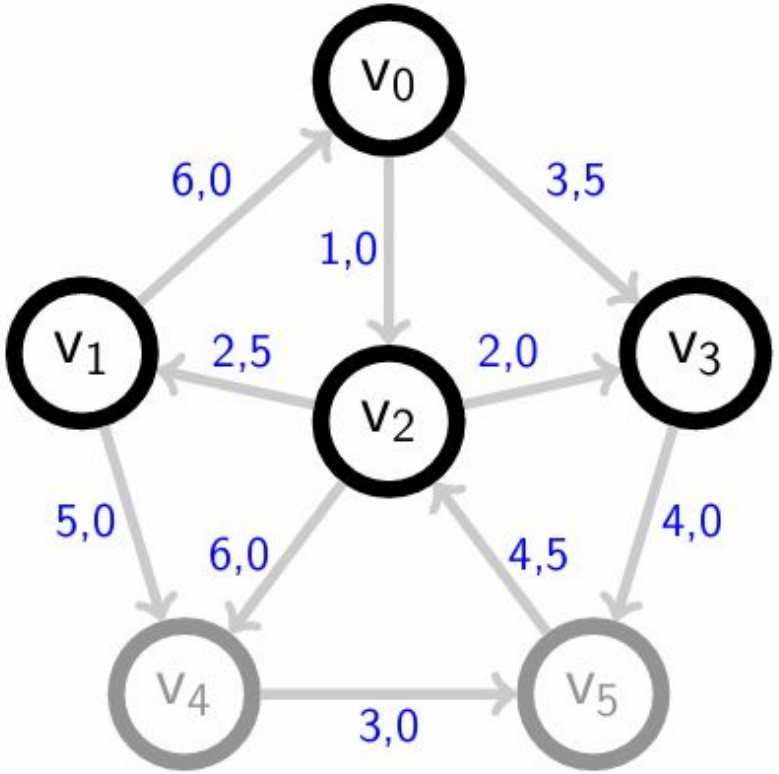
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



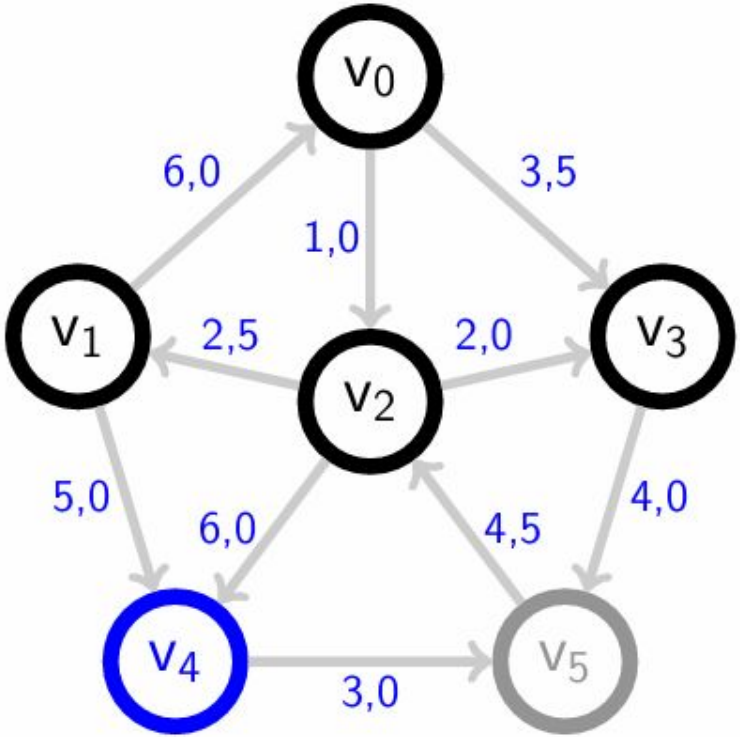
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



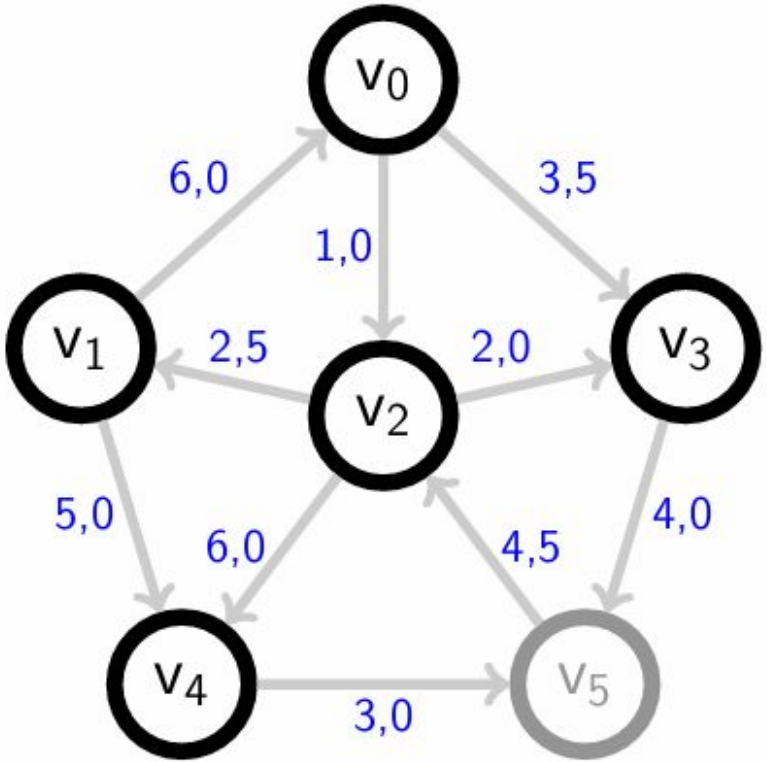
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



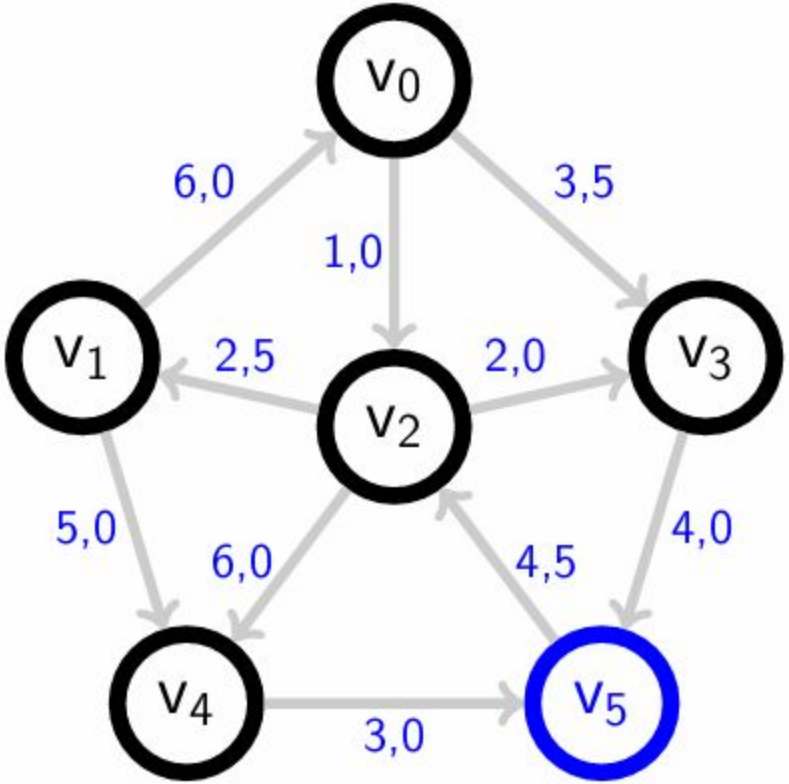
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



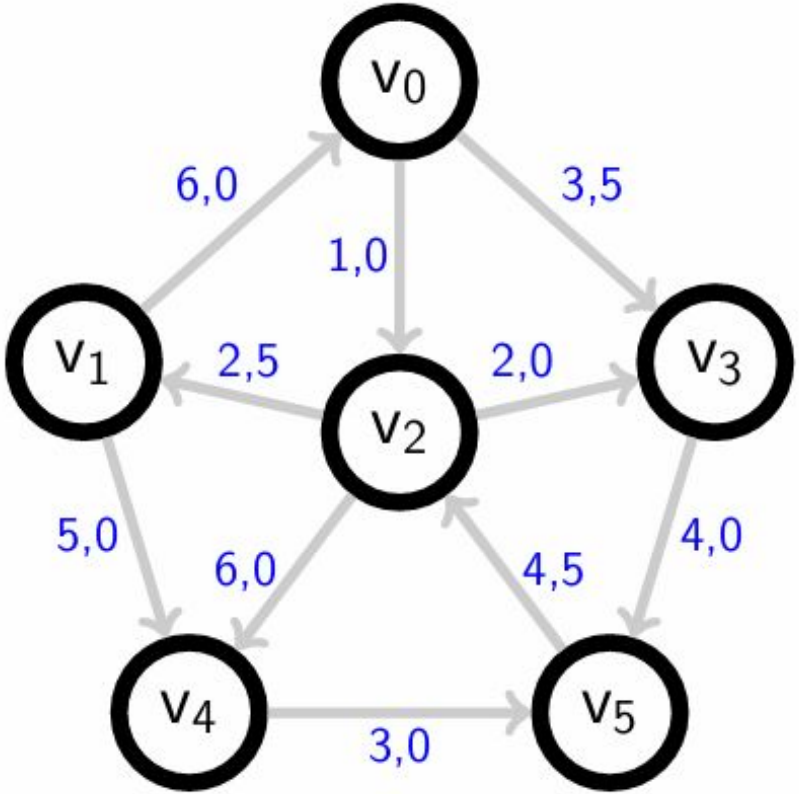
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



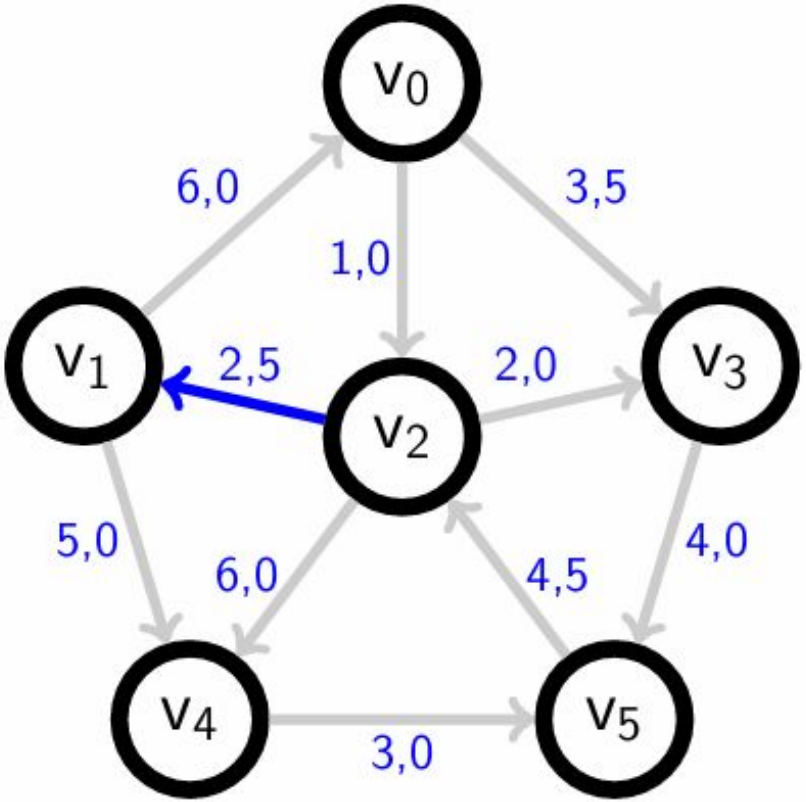
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



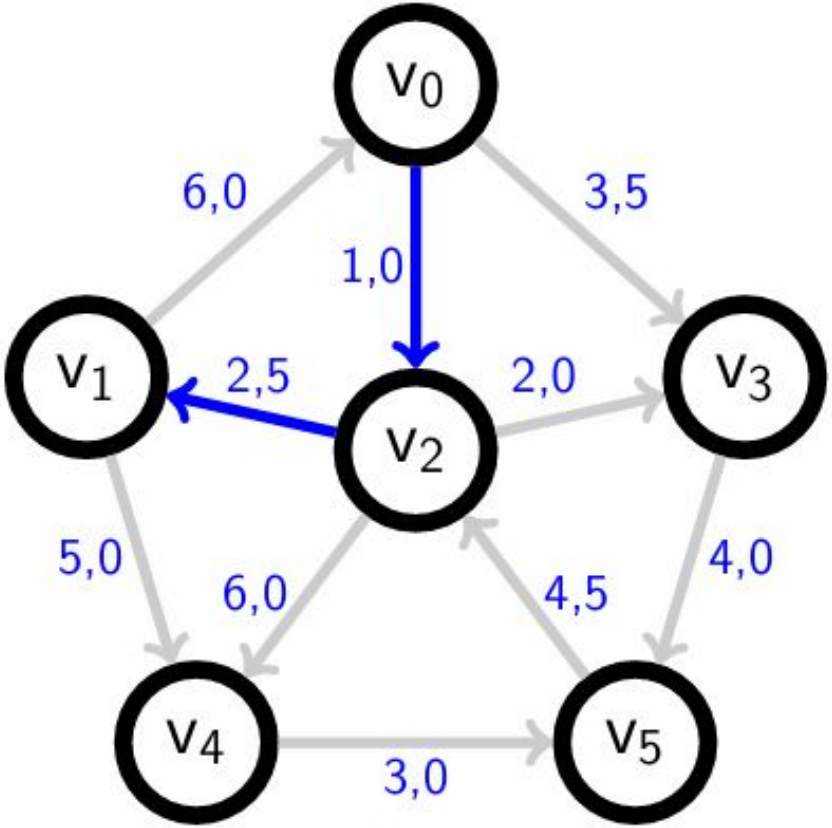
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



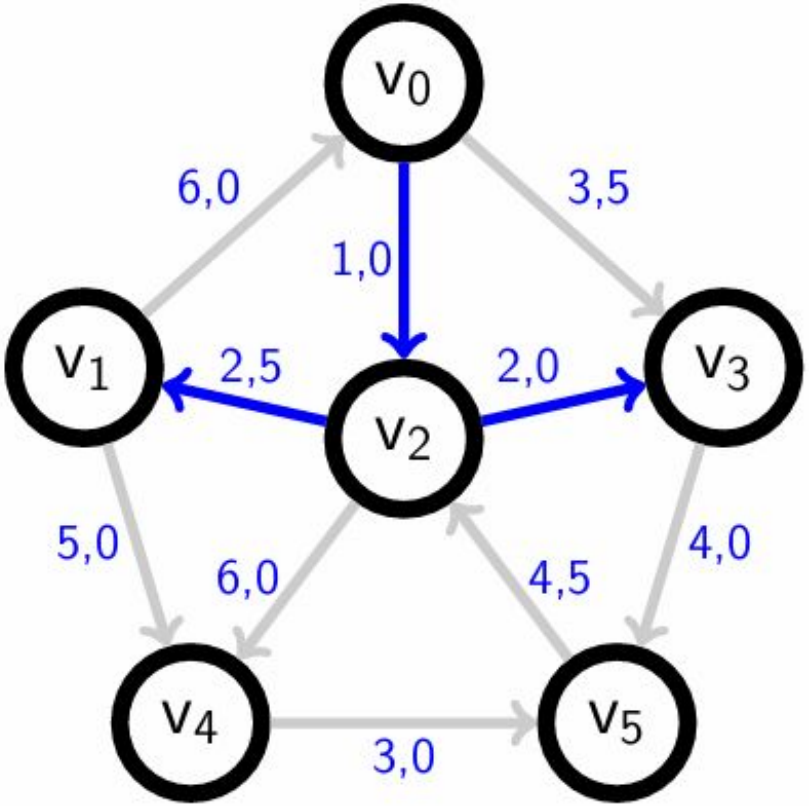
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



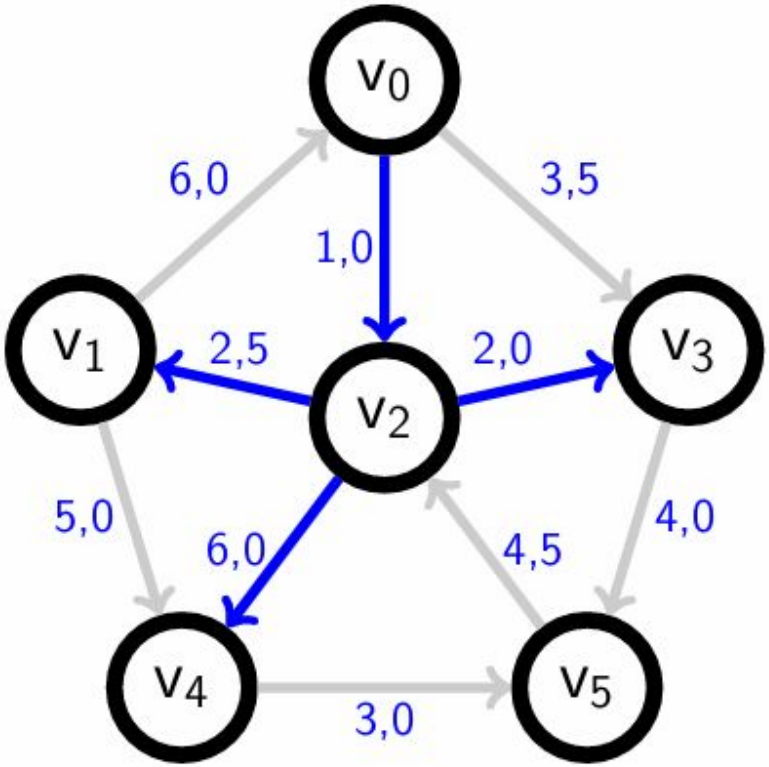
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



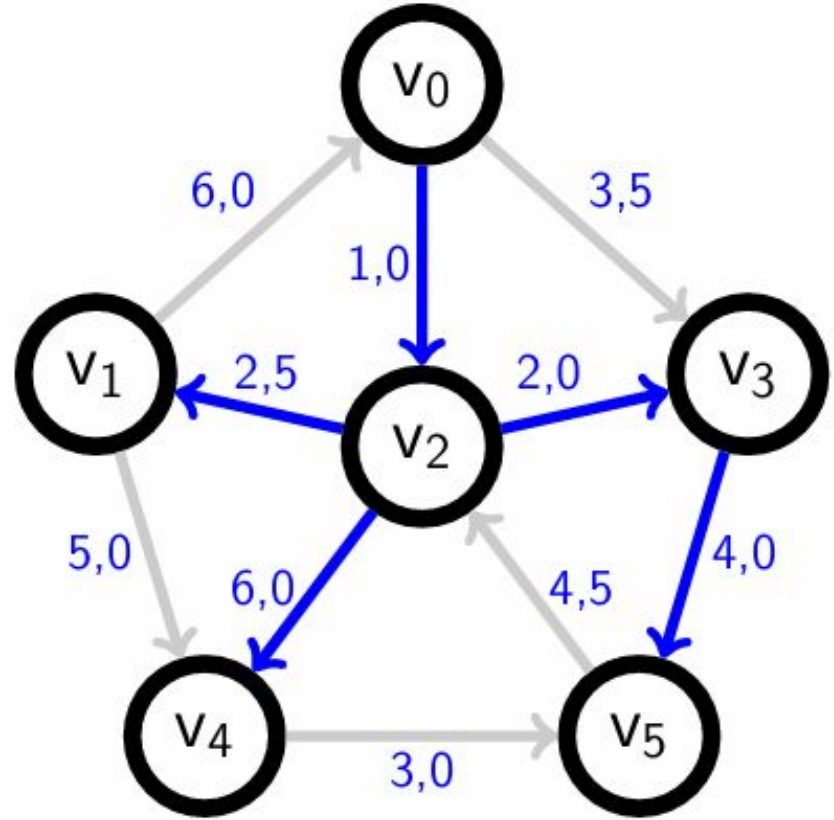
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



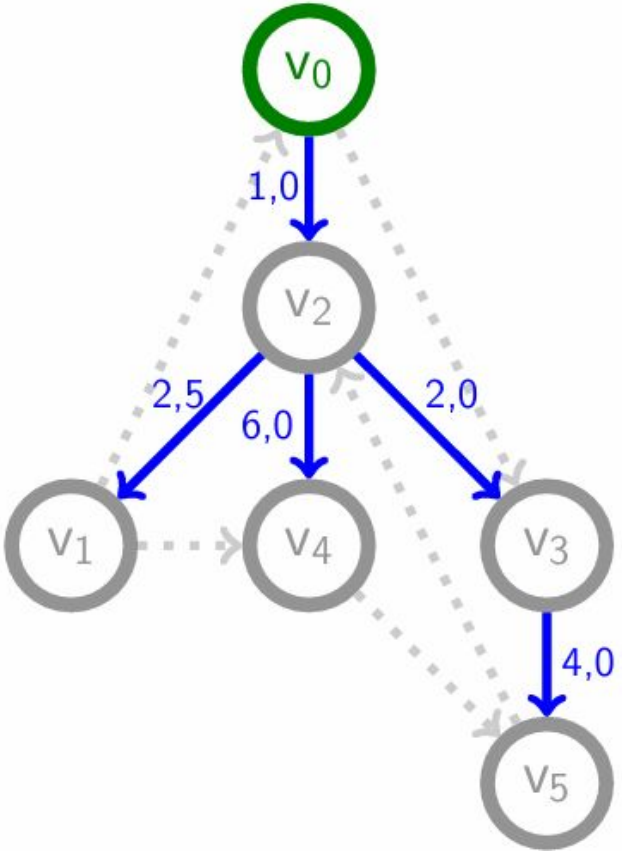
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Dijkstra - Exemplo



Arranjo de Distâncias

v0	v1	v2	v3	v4	v5
0,0	3,5	1,0	3,0	7,0	7,0

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5
0	2	0	2	2	3

Desafios

Desafio

- Implemente o algoritmo de Dijkstra.