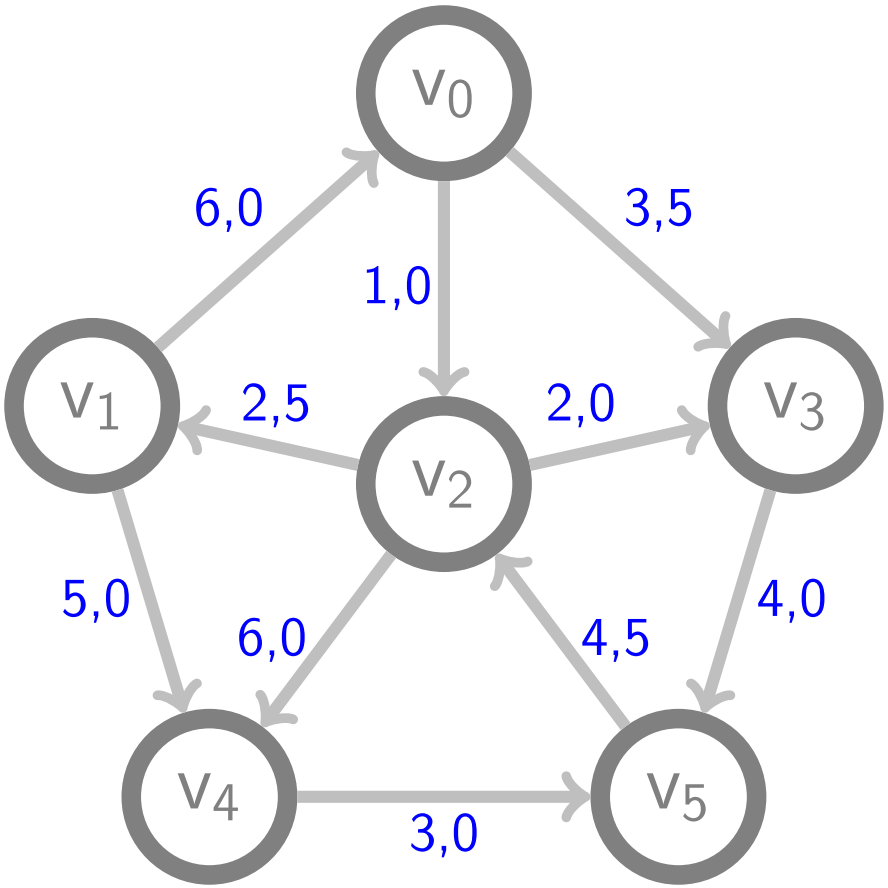
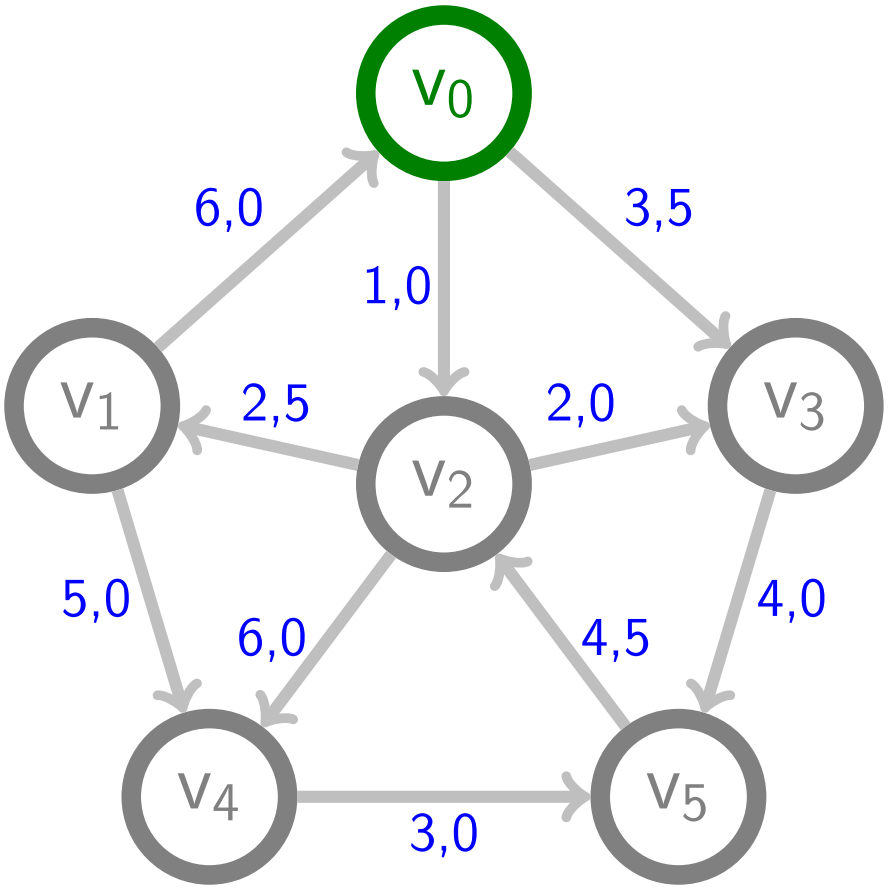
Algoritmos e Estruturas de Dados II Aula 13 - Caminhos de Custo Mínimo - Bellman-Ford

Prof. Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri

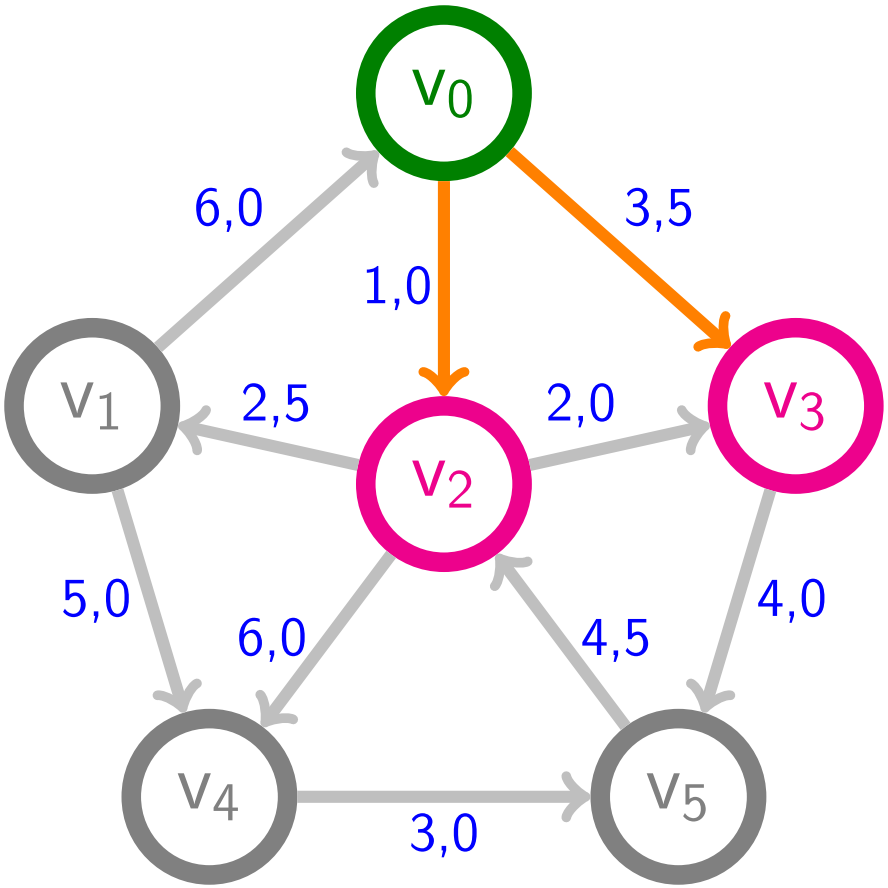
Caminhos de Peso Mínimo



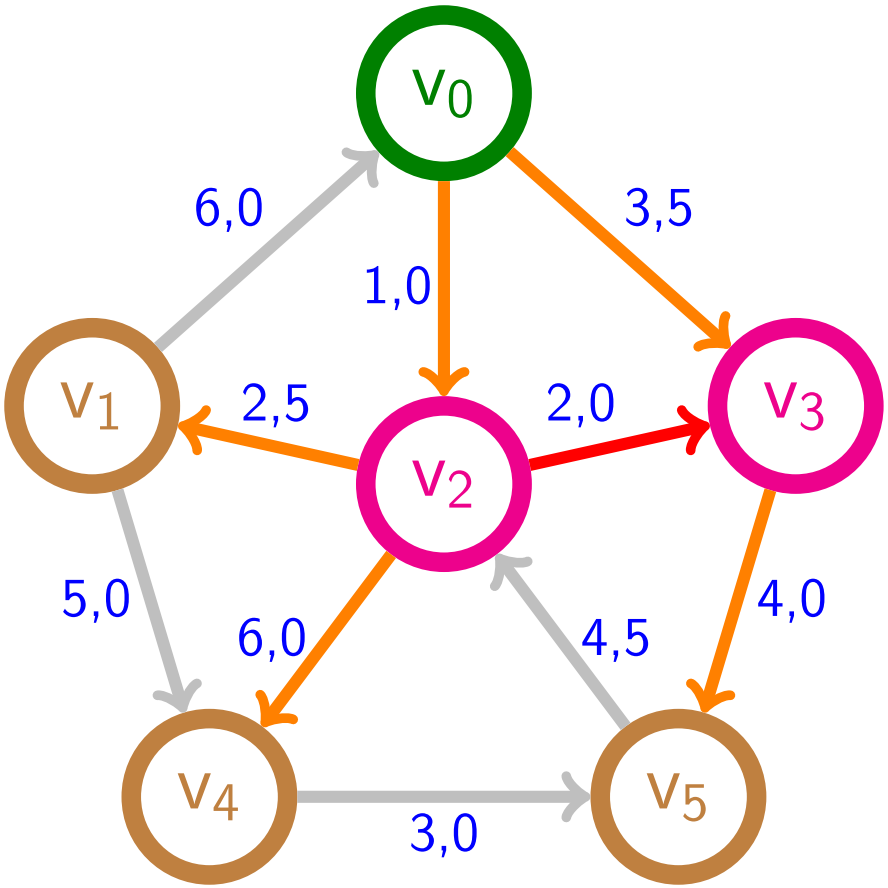
Caminhos de Peso Mínimo



Caminhos de Peso Mínimo

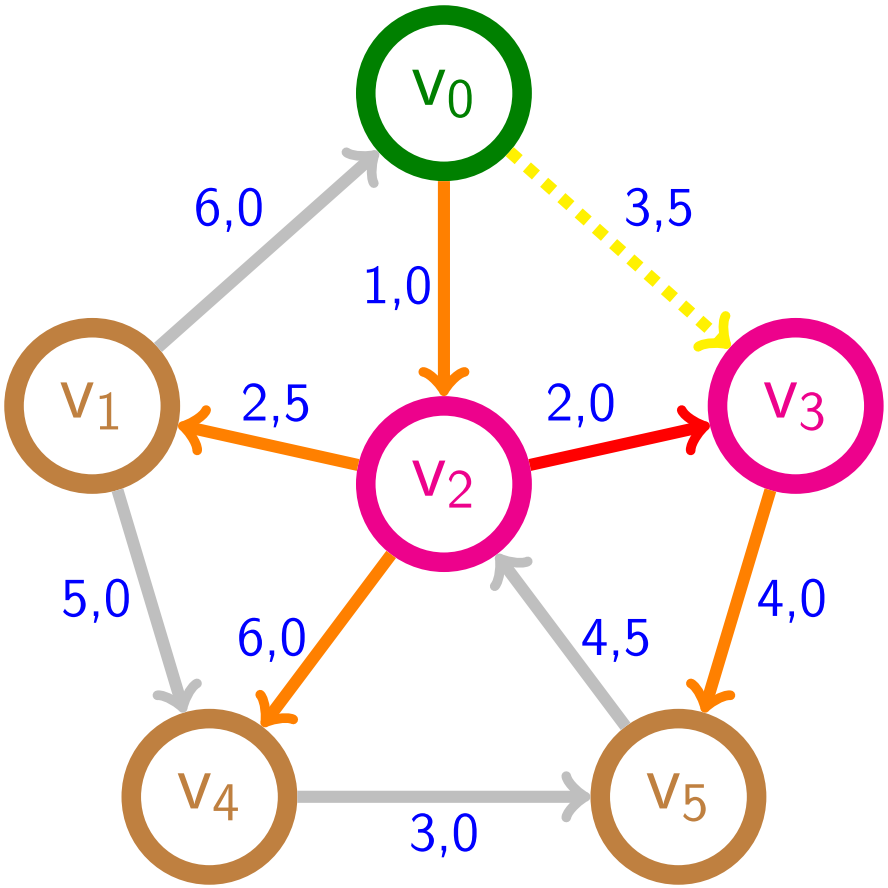


Caminhos de Peso Mínimo



Existe **um caminho de menor custo** ('atalho') de v0 para v3, passando por v2.

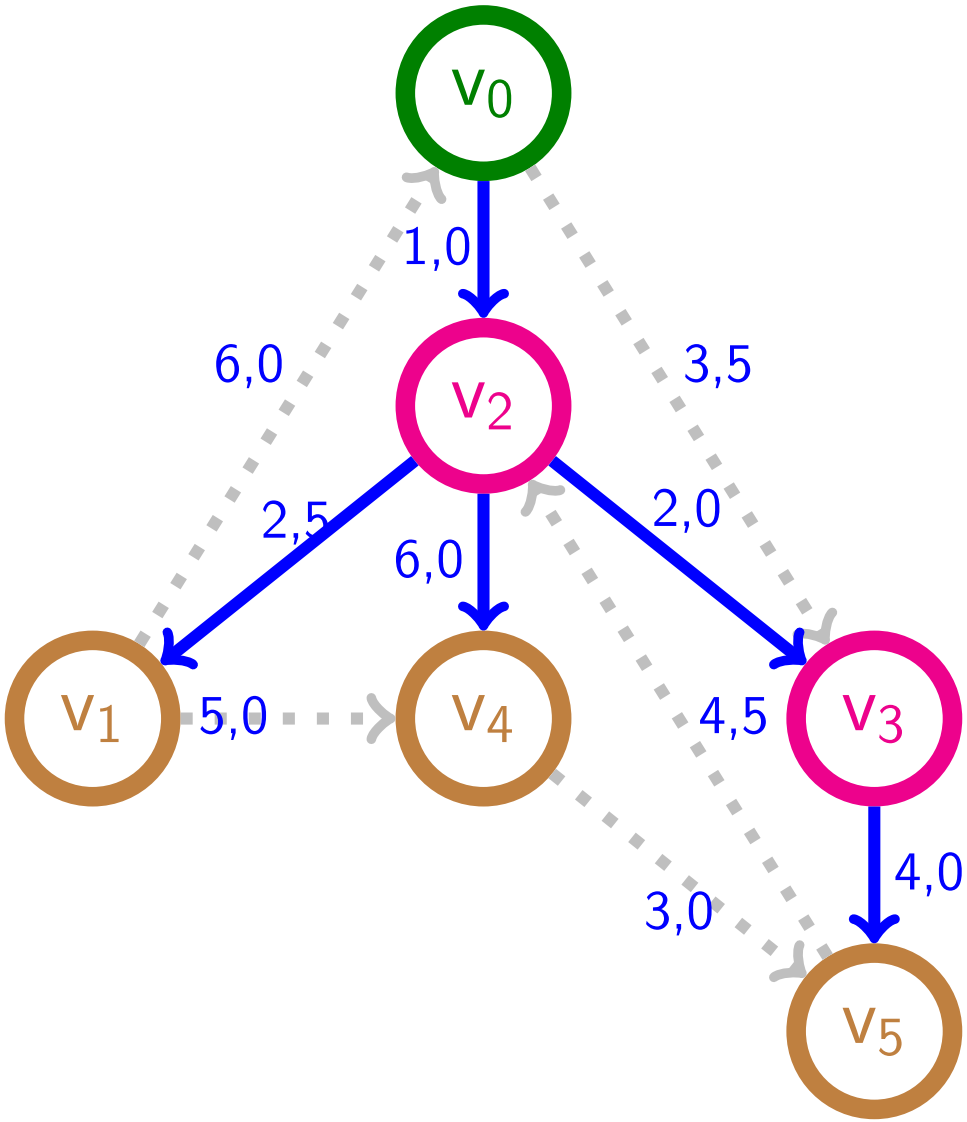
Caminhos de Peso Mínimo



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



Caminhos de Peso Mínimo

Mas como encontramos os caminhos de custo (ou peso) mínimo?

- Existem diversas abordagens, aprenderemos três que usam o relaxamento.
  - **Algoritmo de Floyd-Warshall:** usado para encontrar os caminhos de peso mínimo entre todos os pares de vértices.
  - **Bellman-Ford:** usado para encontrar os caminhos de peso mínimo entre um vértice de origem e os demais.
  - **Dijkstra:** usado para encontrar os caminhos de peso mínimo entre um vértice de origem e os demais.

Algoritmo de Bellman-Ford

Publicado de forma independente em 1958 e 1956[1], este algoritmo tem as seguintes características:

[1] E proposto em 1955 por Alfonso Shimbel.

Algoritmo de Bellman-Ford

Publicado de forma independente em 1958 e 1956[1], este algoritmo tem as seguintes características:

- **Objetivo:** Encontrar os caminhos de custo mínimo entre um vértice de origem e os demais vértices;

[1] E proposto em 1955 por Alfonso Shimbel.

Algoritmo de Bellman-Ford

Publicado de forma independente em 1958 e 1956[1], este algoritmo tem as seguintes características:

- **Objetivo:** Encontrar os caminhos de custo mínimo entre um vértice de origem e os demais vértices;
- **Aplicação:** Grafos direcionados e ponderados (arestas podem ter pesos negativos, mas não pode haver um ciclo negativo);

[1] E proposto em 1955 por Alfonso Shimbel.

Algoritmo de Bellman-Ford

Publicado de forma independente em 1958 e 1956[1], este algoritmo tem as seguintes características:

- **Objetivo:** Encontrar os caminhos de custo mínimo entre um vértice de origem e os demais vértices;
- **Aplicação:** Grafos direcionados e ponderados (arestas podem ter pesos negativos, mas não pode haver um ciclo negativo);
- **Complexidade de tempo:** $O(V * E)$;

[1] E proposto em 1955 por Alfonso Shimbel.

Algoritmo de Bellman-Ford

Publicado de forma independente em 1958 e 1956[1], este algoritmo tem as seguintes características:

- **Objetivo:** Encontrar os caminhos de custo mínimo entre um vértice de origem e os demais vértices;
- **Aplicação:** Grafos direcionados e ponderados (arestas podem ter pesos negativos, mas não pode haver um ciclo negativo);
- **Complexidade de tempo:** $O(V * E)$;
- Abordagem utiliza Programação Dinâmica.

[1] E proposto em 1955 por Alfonso Shimbel.

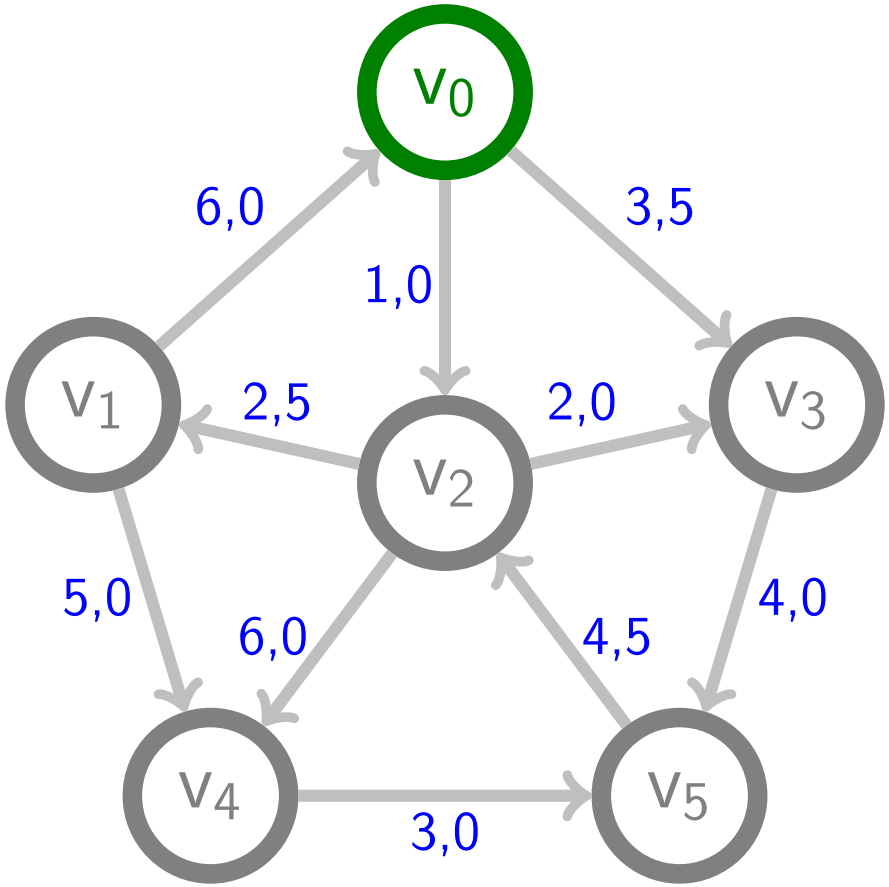
Algoritmo de Bellman-Ford

Pseudocódigo:

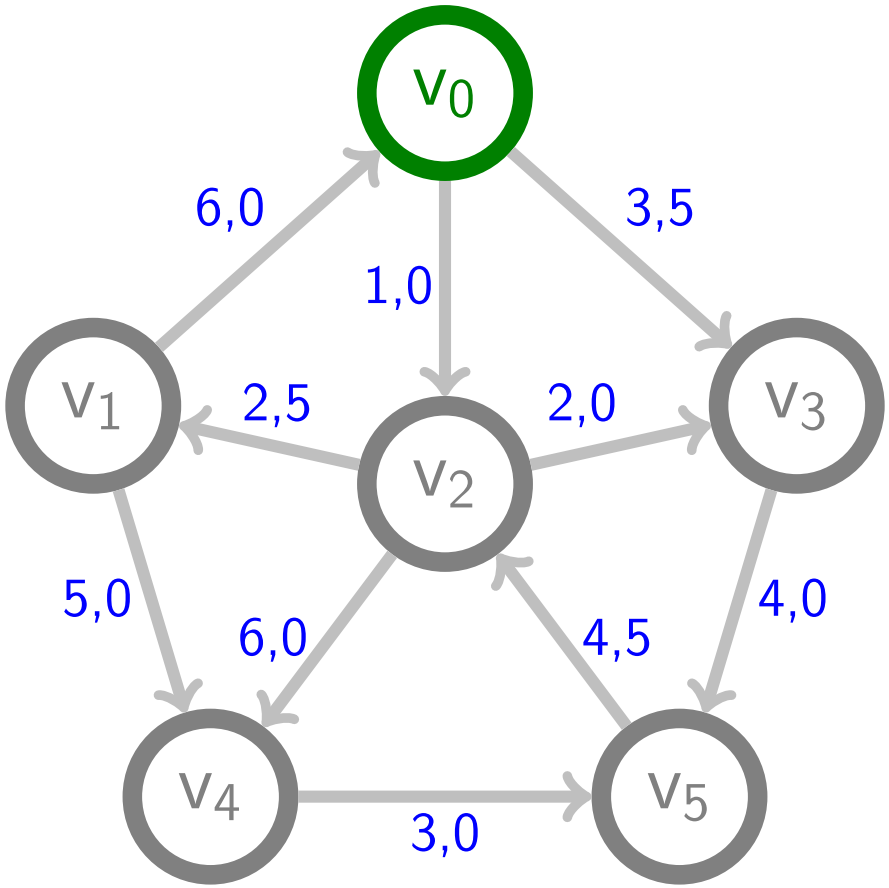
```
REPITA n-1 VEZES:
    PARA CADA ARESTA (u-v):
        SE dist[v] > dist[u] + peso(u-v) ENTÃO
            dist[v] = dist[u] + peso(u-v)
            pred[v] = u
```

Sendo *n* igual ao número de vértices, *dist* um arranjo de distância de cada vértice ao vértice de origem, iniciada com zero para o vértice de origem e 'infinito' para os demais, *pred* o arranjo de predecessores e peso(u-v) o peso da aresta de *u* para *v*.

Algoritmo de Bellman-Ford



Algoritmo de Bellman-Ford



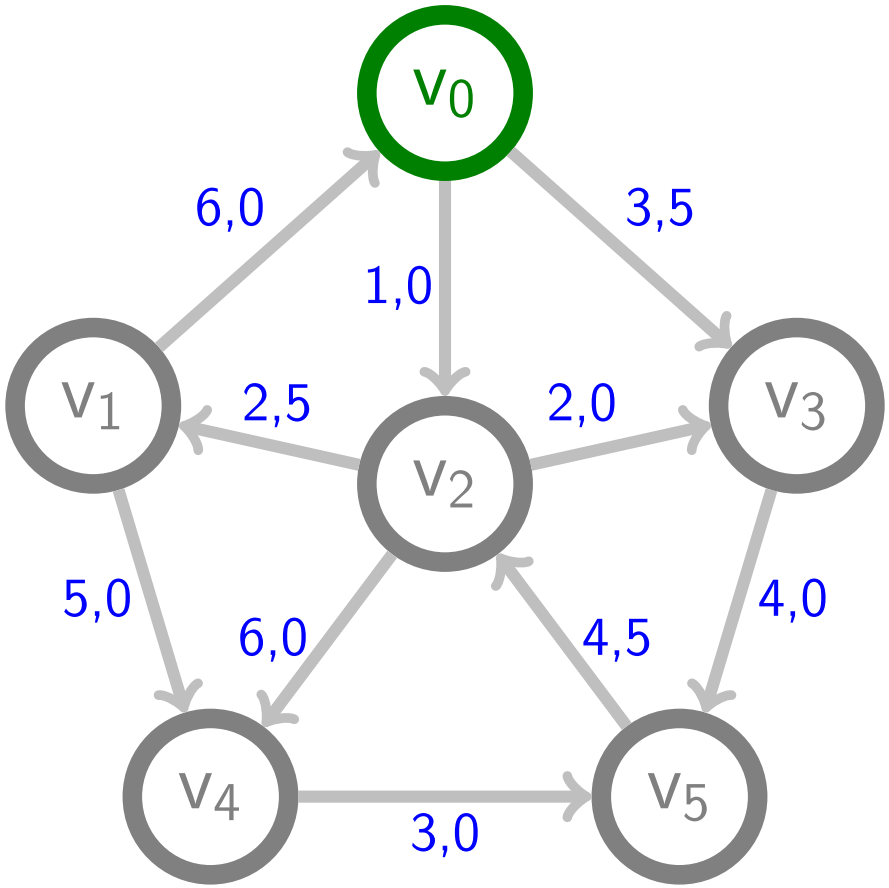
Arranjo de Distâncias

v0	v1	v2	v3	v4	v5

Arranjo de Predecessores

v0	v1	v2	v3	v4	v5

Algoritmo de Bellman-Ford



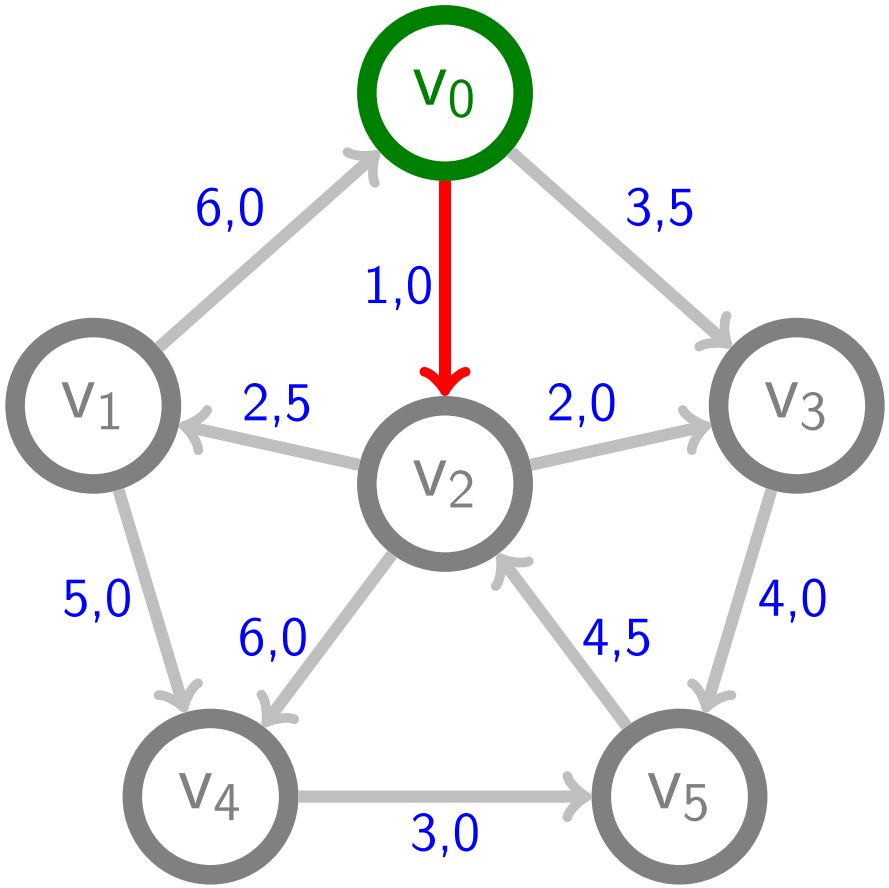
Arranjo de Distâncias

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

Arranjo de Predecessores

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

Algoritmo de Bellman-Ford



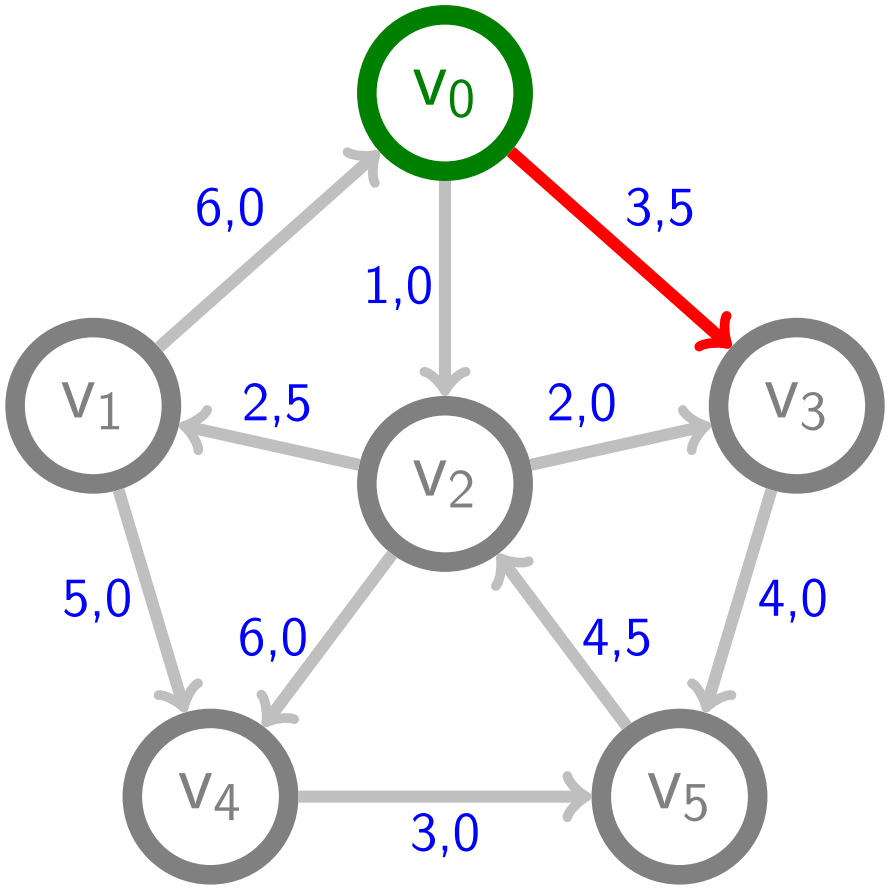
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



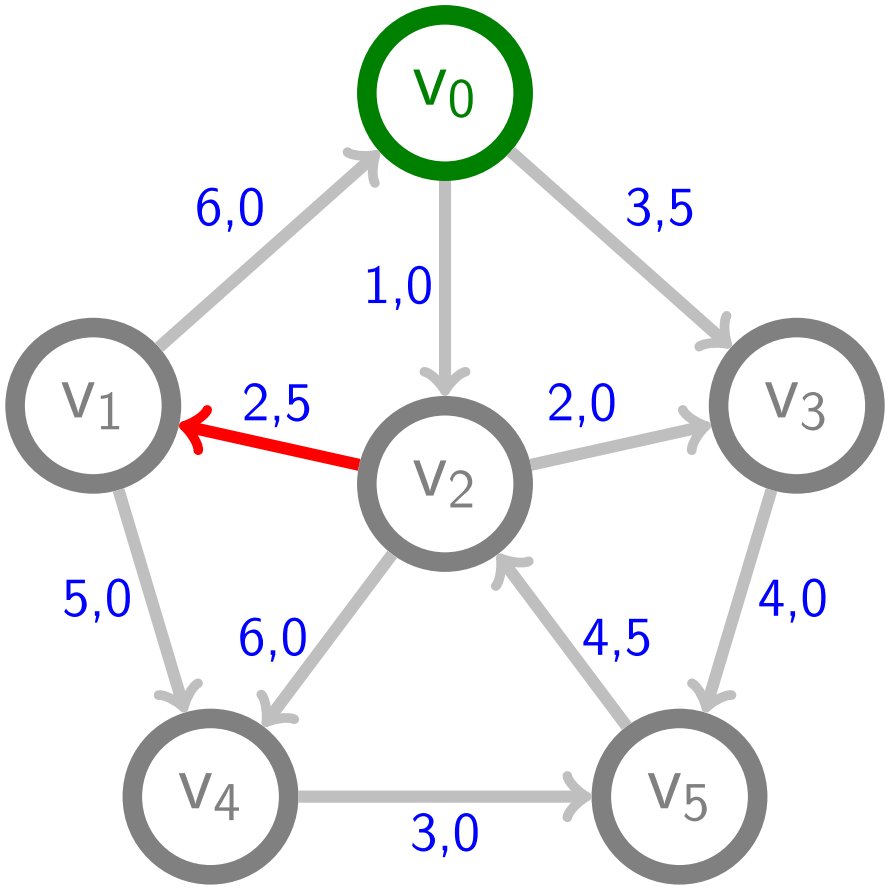
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



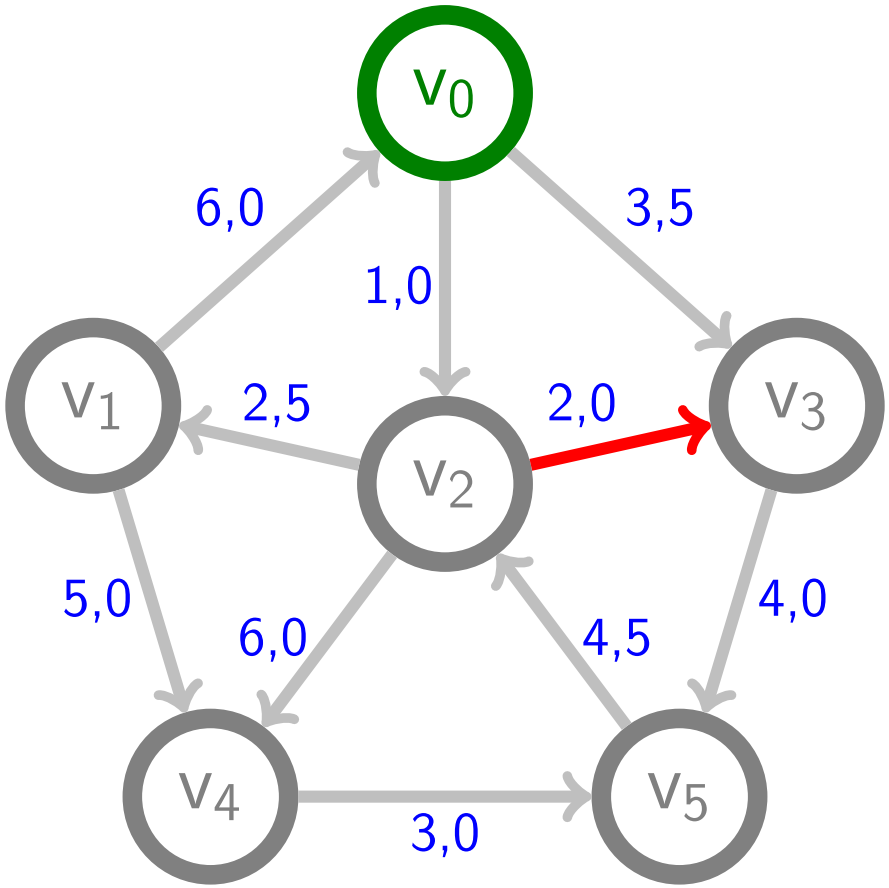
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



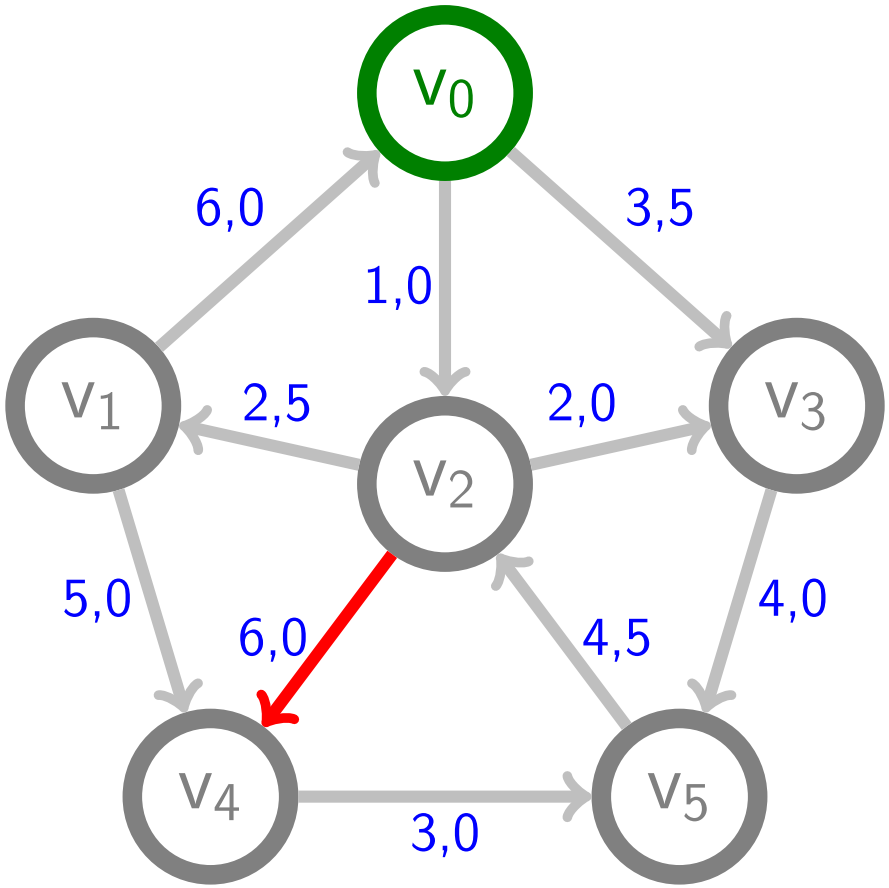
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



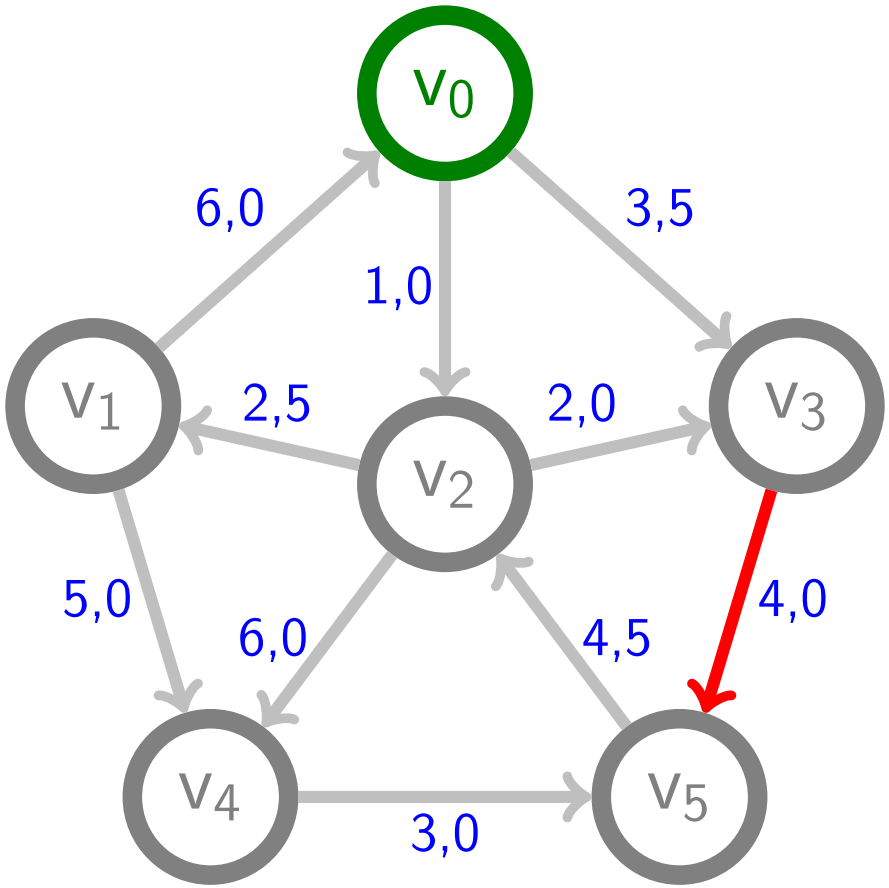
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



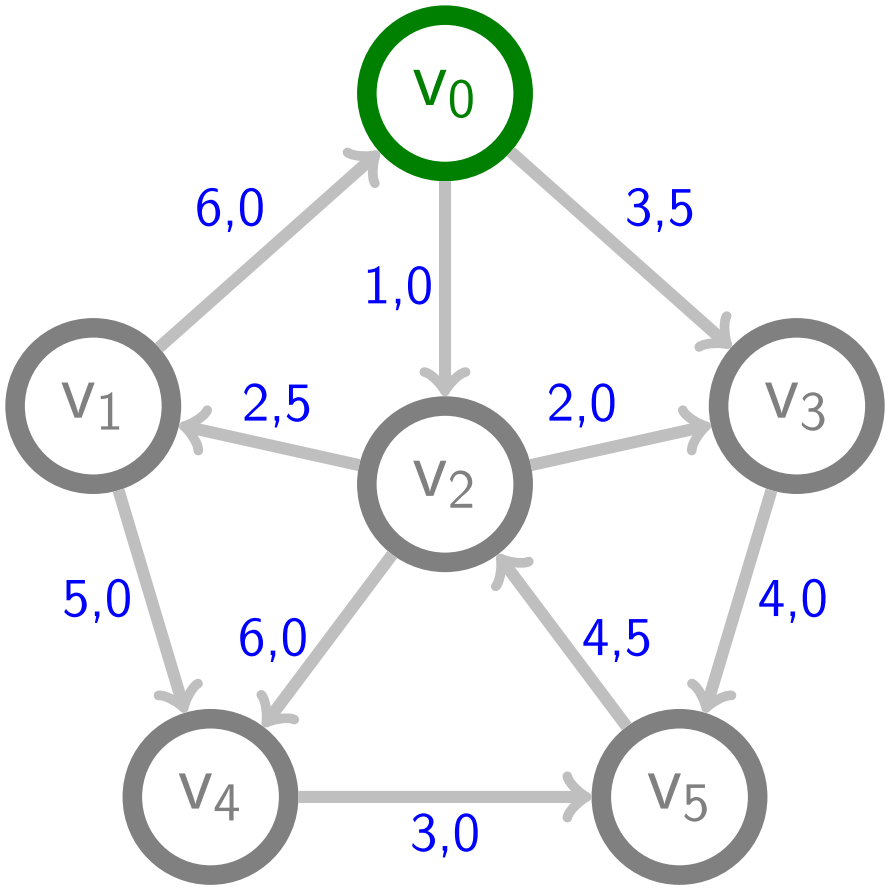
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



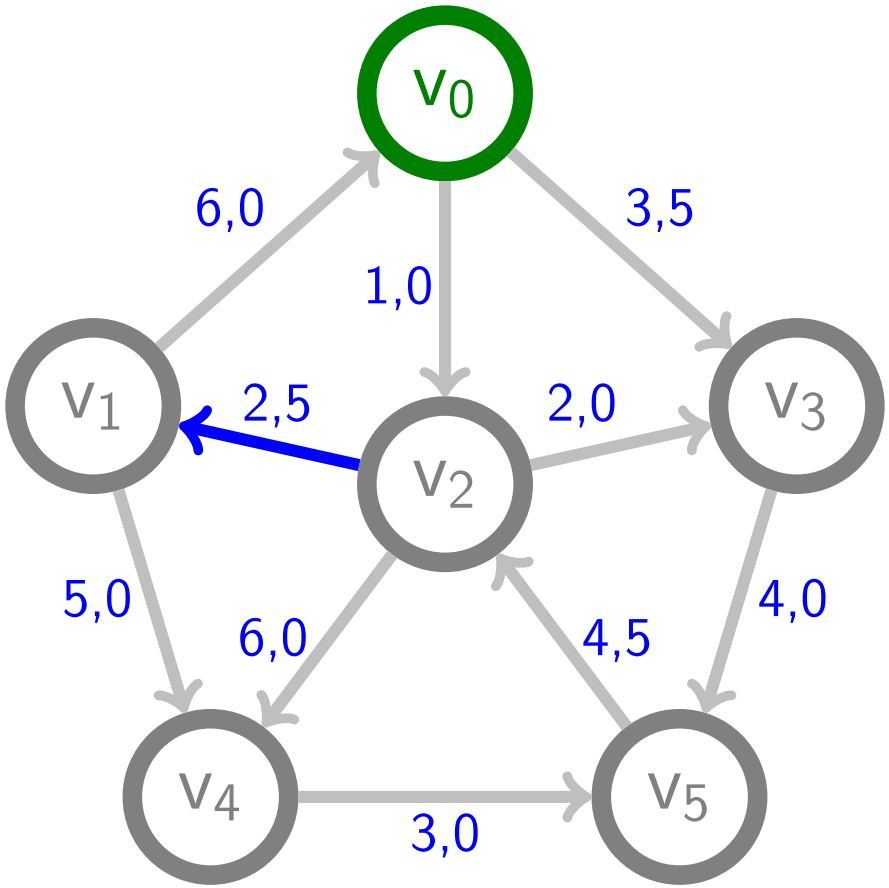
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



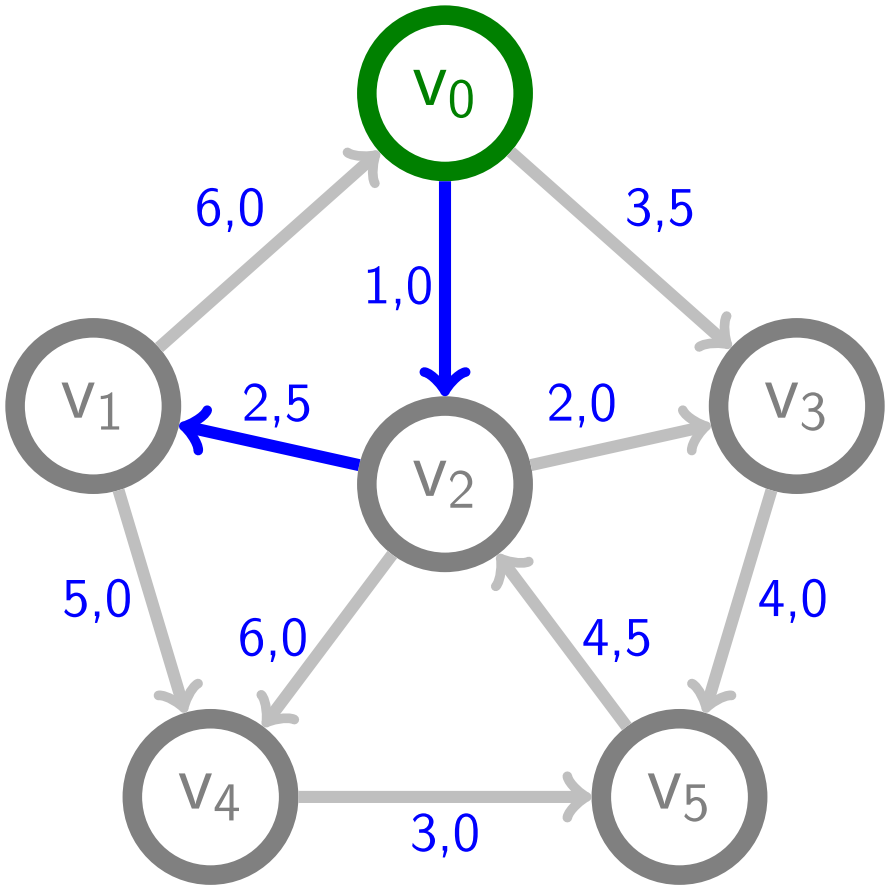
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



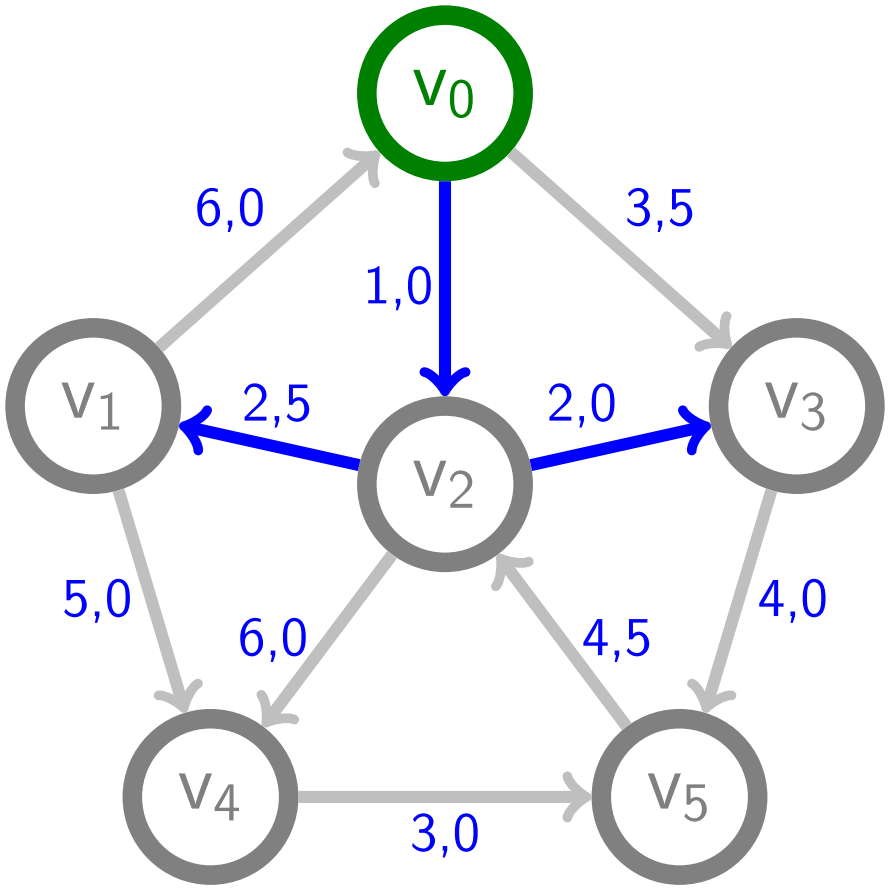
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



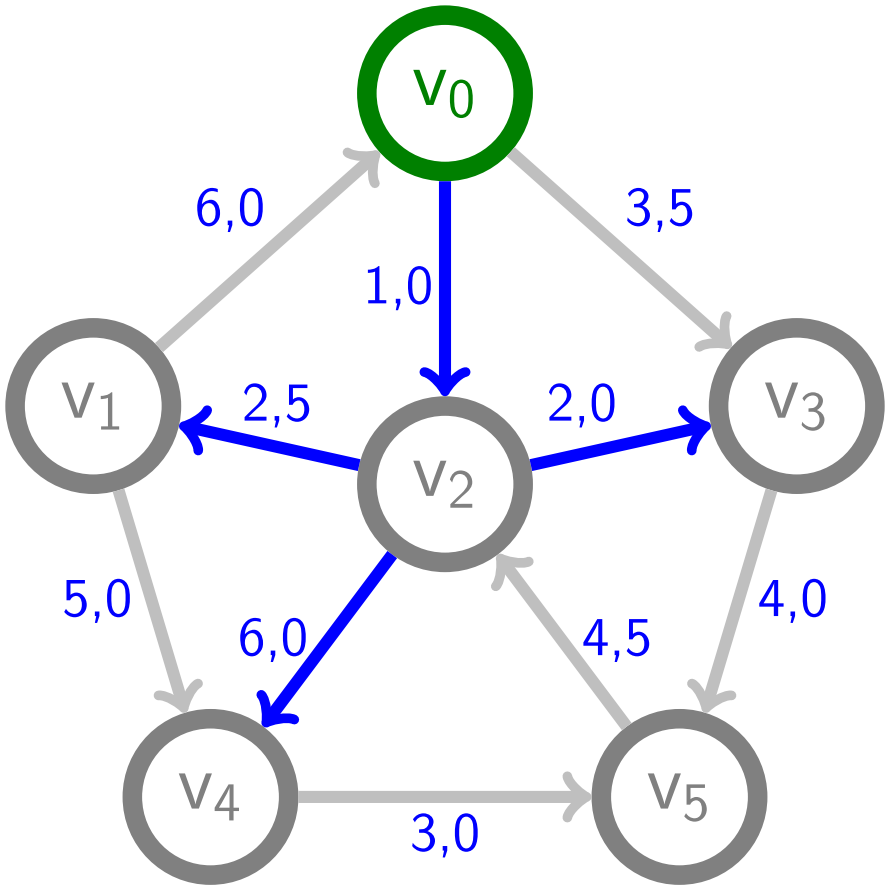
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



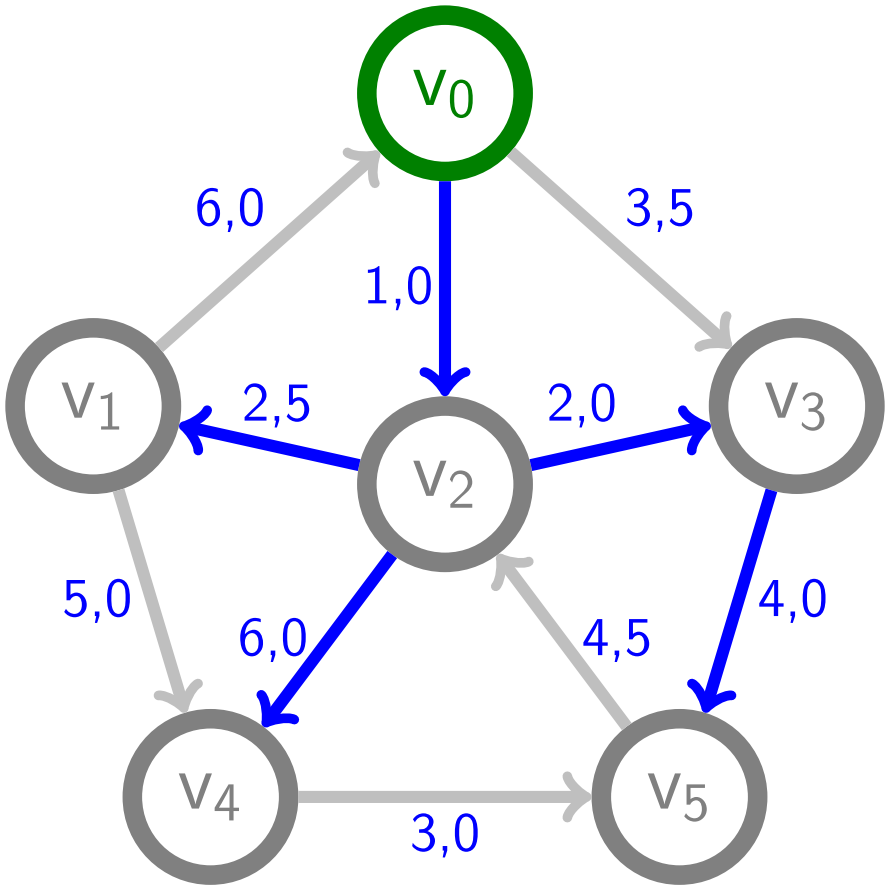
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



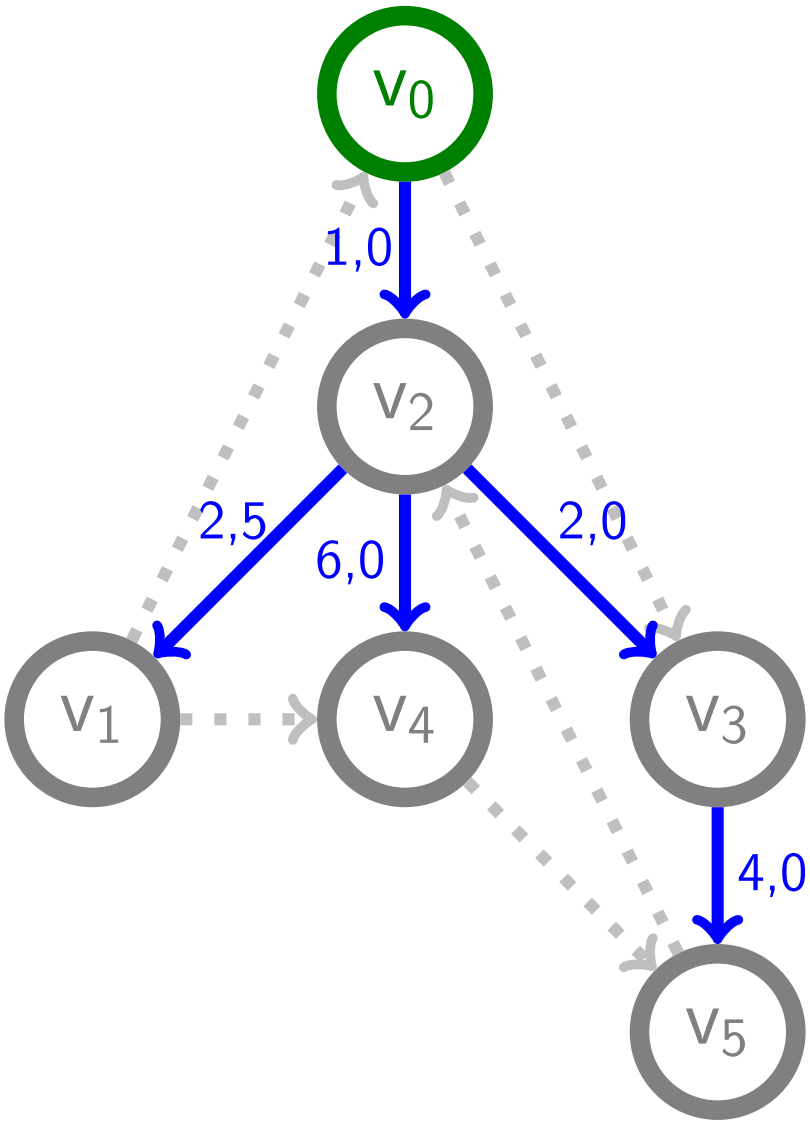
Arranjo de Distâncias

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

Algoritmo de Bellman-Ford



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

Algoritmo de Bellman-Ford - inicialização

```
void AlgoritmoDeBellmanFord(Grafo* g, int origem,
                            Peso* dist, int* pred){

  ...
```

Algoritmo de Bellman-Ford - inicialização

```
void AlgoritmoDeBellmanFord(Grafo* g, int origem,
                                      Peso* dist, int* pred){
  int i, n;
  n = g->numVertices;
  for(i=0;i<n;i++) {
    dist[i] = INFINITO;
    pred[i] = -1;
  }



  ...
```

Algoritmo de Bellman-Ford - inicialização

```
void AlgoritmoDeBellmanFord(Grafo* g, int origem,
                                      Peso* dist, int* pred){
  int i, n;
  n = g->numVertices;
  for(i=0;i<n;i++) {
    dist[i] = INFINITO;
    pred[i] = -1;
  }
  dist[origem] = 0;
  pred[origem] = origem;

  ...
```

Algoritmo de Bellman-Ford

```
    PONT2 atual, arestas = listaDeArestas(g);

}
```

Algoritmo de Bellman-Ford

```
    PONT2 atual, arestas = listaDeArestas(g);
    for(i=1;i<n;i++) {

    }

}
```

Algoritmo de Bellman-Ford

```
  PONT2 atual, arestas = listaDeArestas(g);
  for(i=1;i<n;i++) {
    atual = arestas;
    while (atual){



      atual = atual->prox;
    }
  }


}
```

Algoritmo de Bellman-Ford

```
  PONT2 atual, arestas = listaDeArestas(g);
  for(i=1;i<n;i++) {
    atual = arestas;
    while (atual){
      if(dist[atual->origem]+atual->peso < dist[atual->destino]){
        dist[atual->destino] = dist[atual->origem] + atual->peso;
        pred[atual->destino] = atual->origem;
      }
      atual = atual->prox;
    }
  }


}
```

Algoritmo de Bellman-Ford

```
  PONT2 atual, arestas = listaDeArestas(g);
  for(i=1;i<n;i++) {
    atual = arestas;
    while (atual){
      if(dist[atual->origem]+atual->peso < dist[atual->destino]){
        dist[atual->destino] = dist[atual->origem] + atual->peso;
        pred[atual->destino] = atual->origem;
      }
      atual = atual->prox;
    }
  }

  liberaListaDeArestas(arestas);
}
```

Algoritmo de Bellman-Ford

```
  PONT2 atual, arestas = listaDeArestas(g);
  for(i=1;i<n;i++) {
    atual = arestas;
    while (atual){
      if(dist[atual->origem]+atual->peso < dist[atual->destino]){
        dist[atual->destino] = dist[atual->origem] + atual->peso;
        pred[atual->destino] = atual->origem;
      }
      atual = atual->prox;
    }
  }

  liberaListaDeArestas(arestas);
}
```

$O(V * E)$

Algoritmo de Bellman-Ford

```
  PONT2 atual, arestas = listaDeArestas(g);
  for(i=1;i<n;i++) {
    atual = arestas;
    while (atual){
      if(dist[atual->origem]+atual->peso < dist[atual->destino]){
        dist[atual->destino] = dist[atual->origem] + atual->peso;
        pred[atual->destino] = atual->origem;
      }
      atual = atual->prox;
    }
  }
```

```
  atual = arestas;
  while (atual){
    if(dist[atual->origem]+atual->peso < dist[atual->destino])
        printf("### Erro: ciclo negativo encontrado! ###");
    atual = atual->prox;
  }
```

```
  liberaListaDeArestas(arestas);
}
```

$O(V * E)$

Algoritmos e Estruturas de Dados II Aula 13 - Caminhos de Custo Mínimo - Bellman-Ford

Prof. Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri