Algoritmos e Estruturas de Dados II
Aula 10 – Árvores Geradoras de Custo Mínimo - Algoritmo de Prim

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

Árvores Geradoras de Custo Mínimo

Uma **Árvore Geradora de Custo Mínimo** de um grafo G (também chamada de árvore geradora mínima ou *Minimum Spanning Tree - MST*) é uma árvore geradora do grafo G, cuja soma das arestas tem peso/custo mínimo.

Árvores Geradoras de Custo Mínimo

Uma **Árvore Geradora de Custo Mínimo** de um grafo G (também chamada de árvore geradora mínima ou *Minimum Spanning Tree - MST*) é uma árvore geradora do grafo G, cuja soma das arestas tem peso/custo mínimo.

Grafo gerador de G é um subgrafo de G que possui todos os vértices de G.

Árvores Geradoras de Custo Mínimo

Uma **Árvore Geradora de Custo Mínimo** de um grafo G (também chamada de árvore geradora mínima ou *Minimum Spanning Tree - MST*) é uma árvore geradora do grafo G, cuja soma das arestas tem peso/custo mínimo.

Grafo gerador de G é um subgrafo de G que possui todos os vértices de G.

Árvore é um tipo abstrato de dados composto por elementos conectados que possui um elemento especial chamado raiz que não possui pai e todos os demais elementos possuem um único pai. Ao olharmos a árvore como um grafo, temos um grafo conexo (ou conectado) e acíclico.

Árvores Geradoras de Custo Mínimo

Considerando que as arestas representam o **custo** para se conectar os diversos vértices...

Árvores Geradoras de Custo Mínimo

Considerando que as arestas representam o **custo** para se conectar os diversos vértices...

... as árvores geradoras de custo mínimo correspondem a **forma mais barata** de se conectar todos os vértices em um único componente.

Árvores Geradoras de Custo Mínimo

Considerando que as arestas representam o **custo** para se conectar os diversos vértices...

... as árvores geradoras de custo mínimo correspondem a **forma mais barata** de se conectar todos os vértices em um único componente.

Por exemplo, **menor custo para se conectar todos os computadores em uma rede** (os computadores correspondem aos vértices e as conexões às arestas).

Árvores Geradoras de Custo Mínimo

Tipicamente são encontradas em **grafos conexos, ponderados e não direcionados**.

Árvores Geradoras de Custo Mínimo

Tipicamente são encontradas em **grafos conexos, ponderados e não direcionados**.

Podem existir **diversas árvores** geradoras de custo mínimo de um único grafo.

Árvores Geradoras de Custo Mínimo

Tipicamente são encontradas em **grafos conexos, ponderados e não direcionados**.

Podem existir **diversas árvores** geradoras de custo mínimo de um único grafo.

Existem diferentes abordagens para identificar essas árvores.

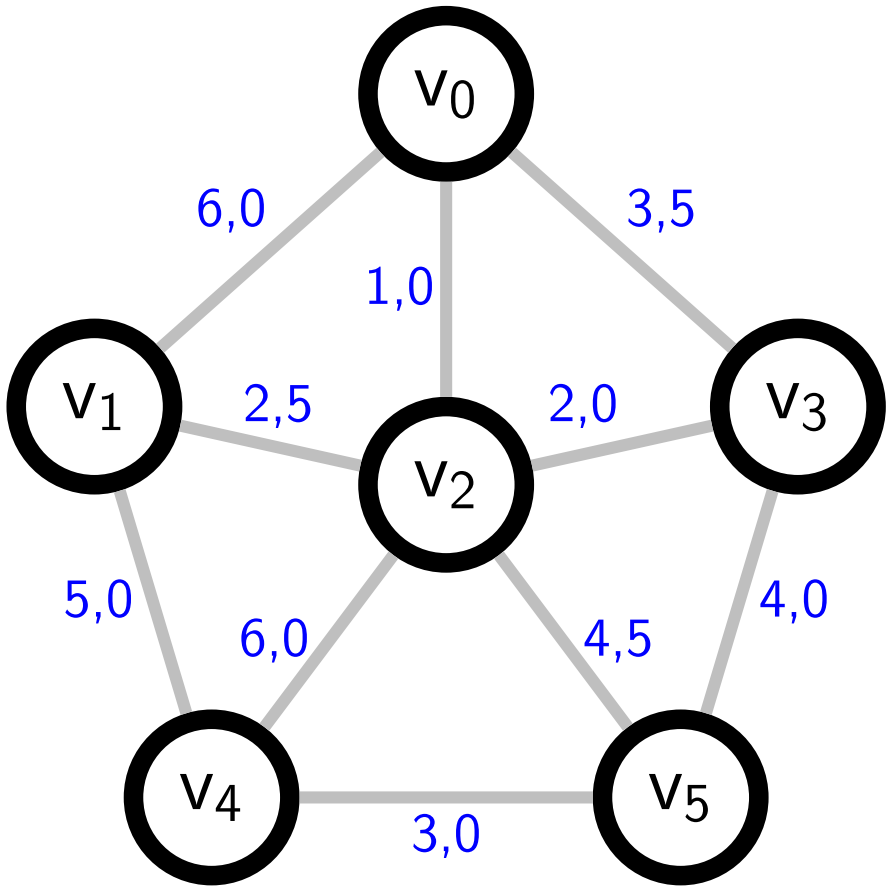
Algoritmo de Prim

Ideia geral do algoritmo:

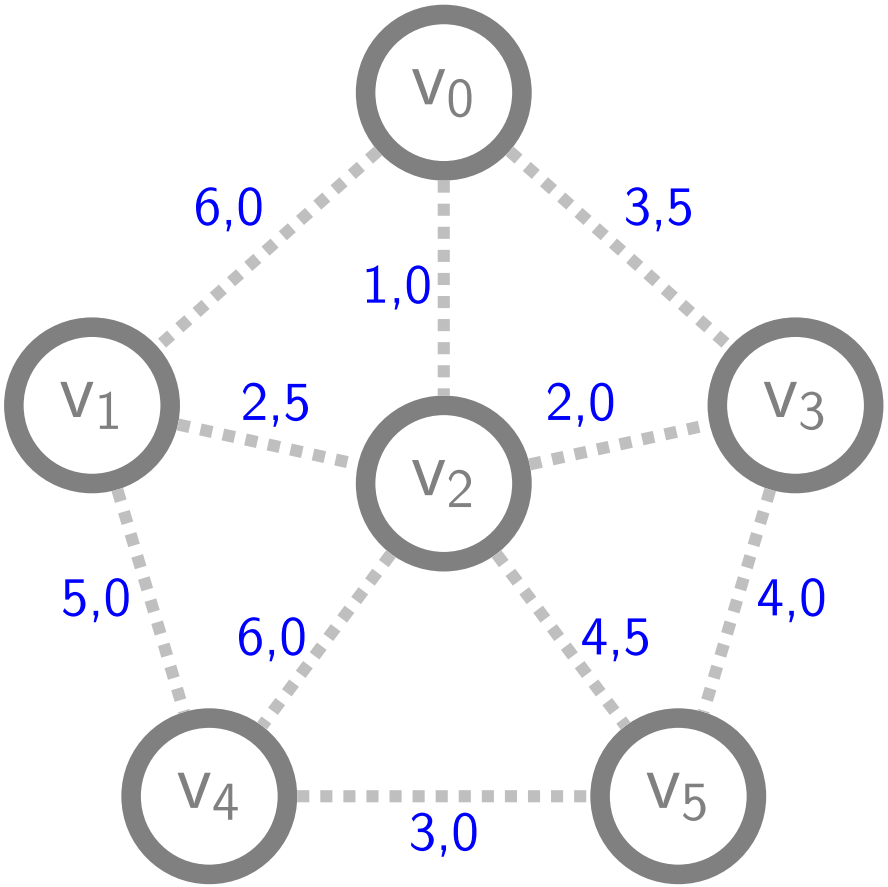
- Selecione um vértice qualquer
- Enquanto existirem vértices não selecionados
  - Encontre a aresta de menor custo[1] que liga um vértice já selecionado $v1$ a um vértice não selecionado $v2$
  - Selecione $v2$ e adicione essa aresta à árvore geradora de custo mínimo

[1] Usaremos uma fila de prioridade de pesos de arestas.

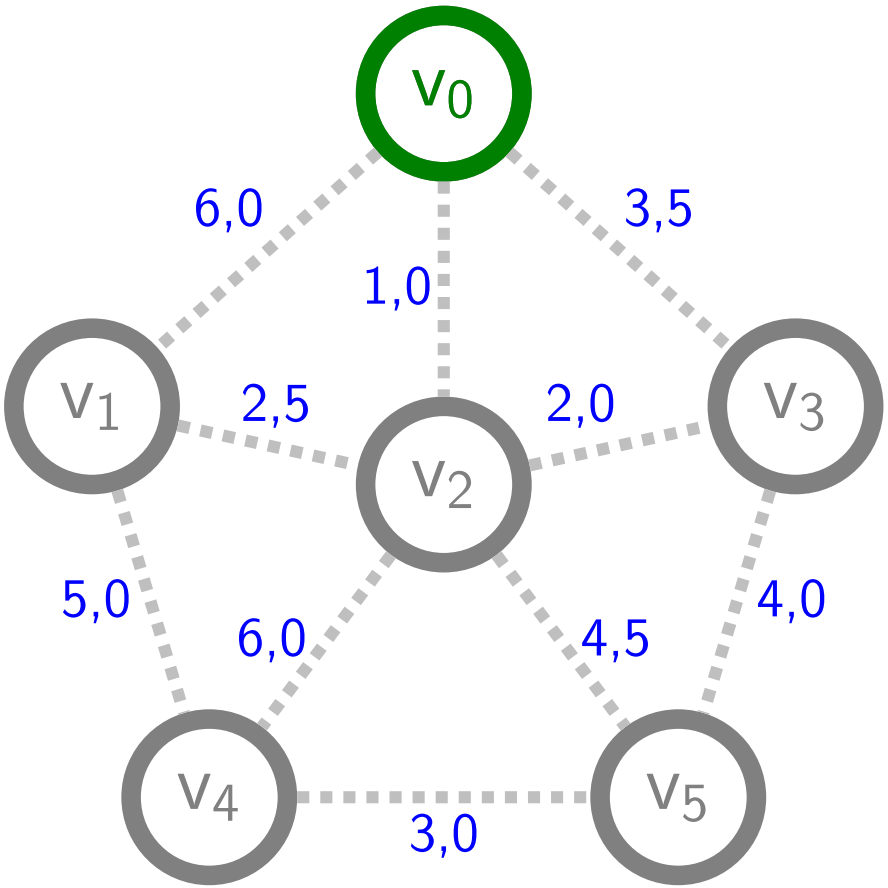
Árvore Geradora de Custo Mínimo - Prim



Árvore Geradora de Custo Mínimo - Prim



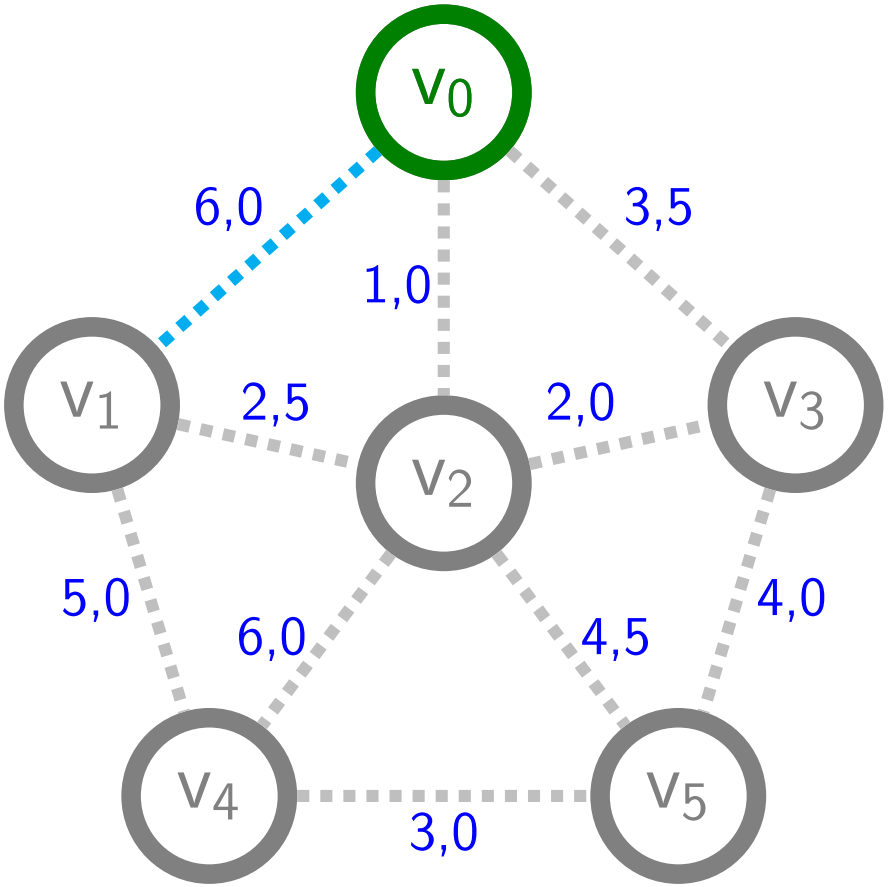
Árvore Geradora de Custo Mínimo - Prim



Custo:

Fila de Prioridade:

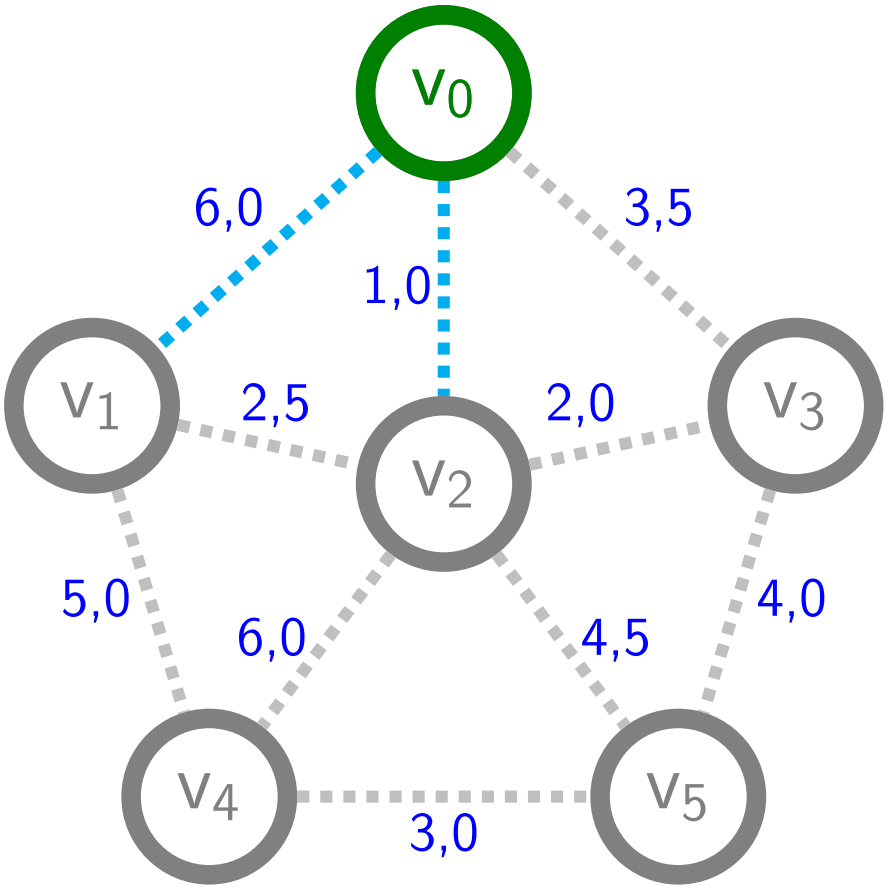
Árvore Geradora de Custo Mínimo - Prim



Custo:

Fila de Prioridade:
1 (6,0)

Árvore Geradora de Custo Mínimo - Prim



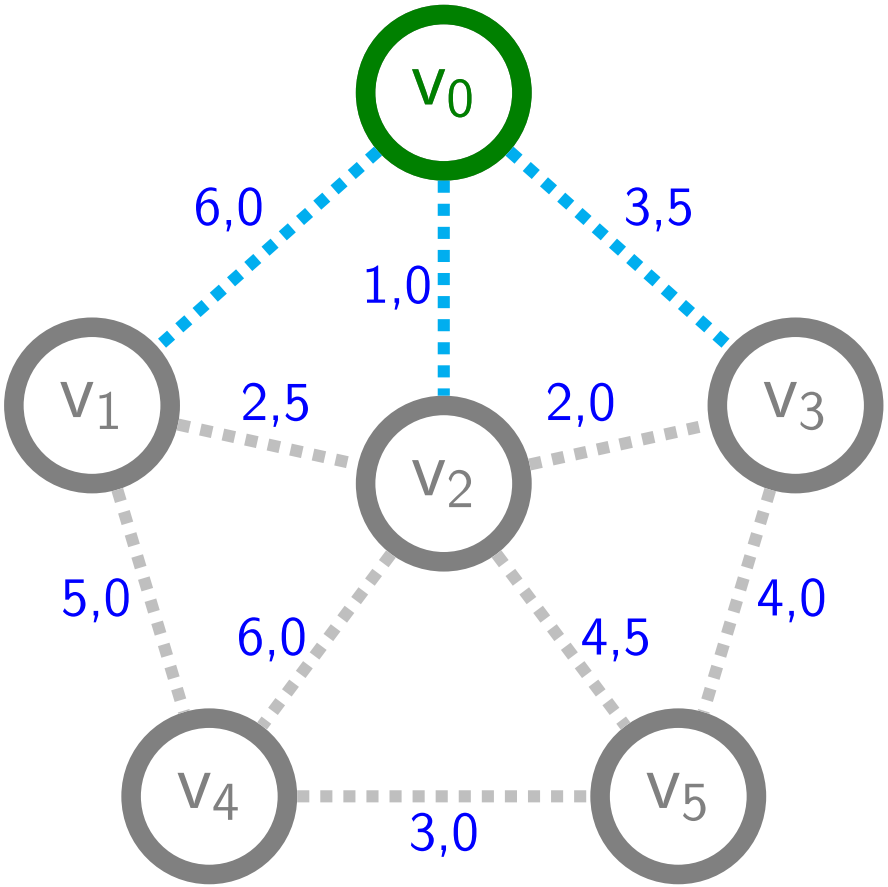
Custo:

Fila de Prioridade:

2 (1,0)

1 (6,0)

Árvore Geradora de Custo Mínimo - Prim

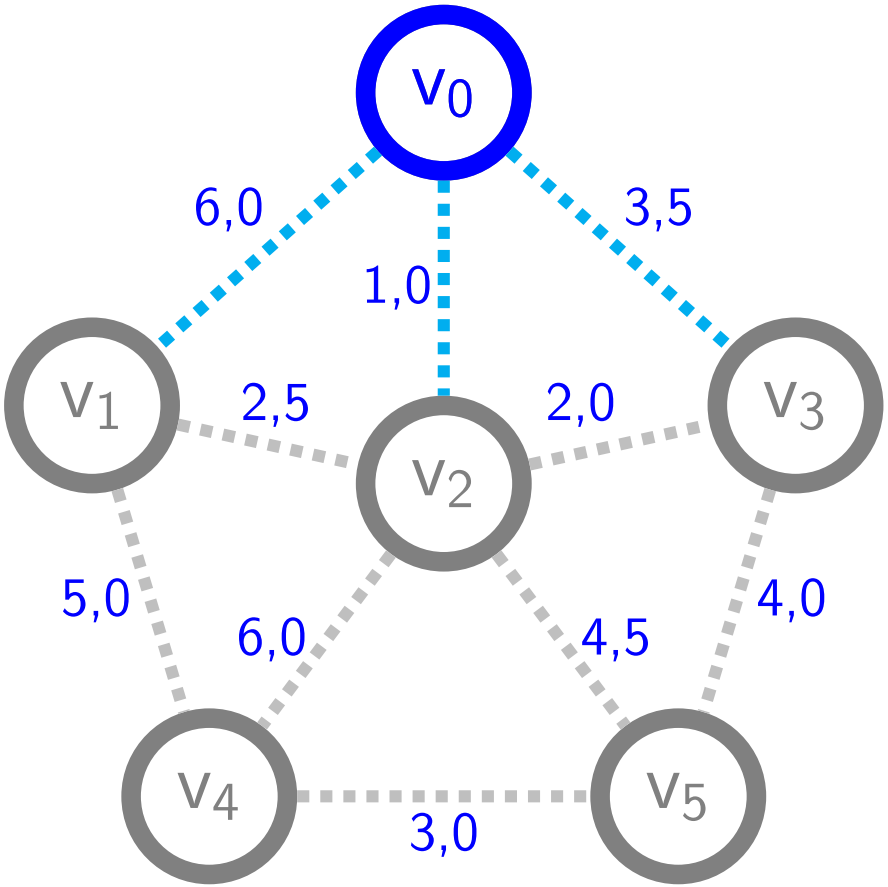


Custo:

Fila de Prioridade:

2 (1,0)
3 (3,5)
1 (6,0)

Árvore Geradora de Custo Mínimo - Prim



Custo:

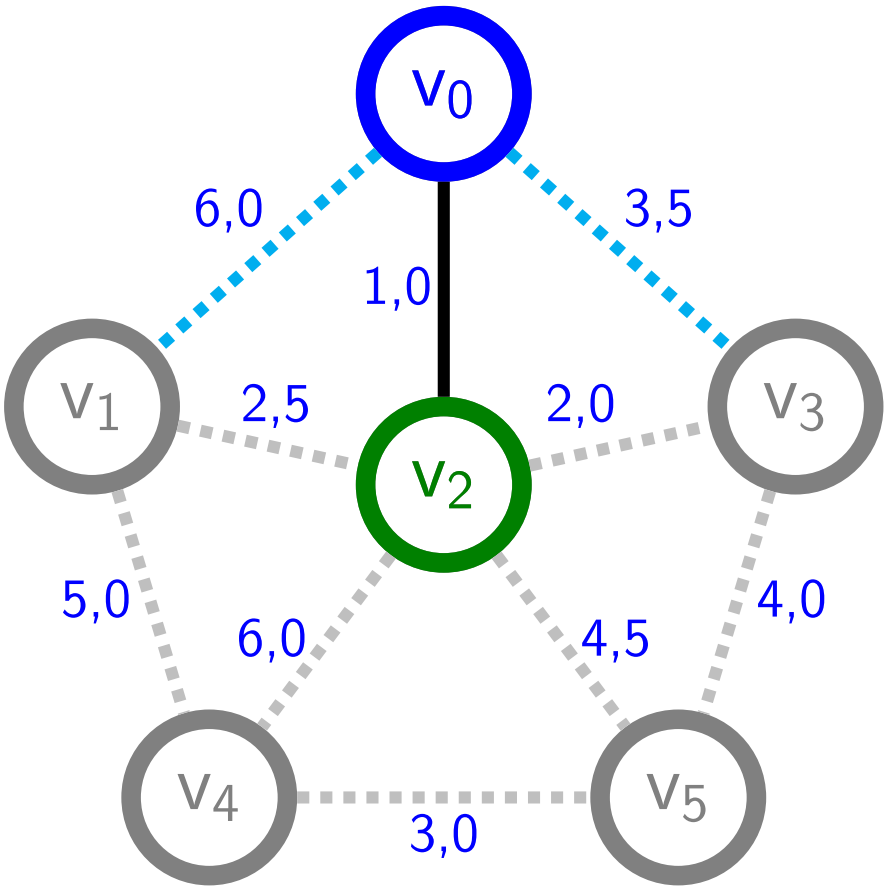
Fila de Prioridade:

2 (1,0)

3 (3,5)

1 (6,0)

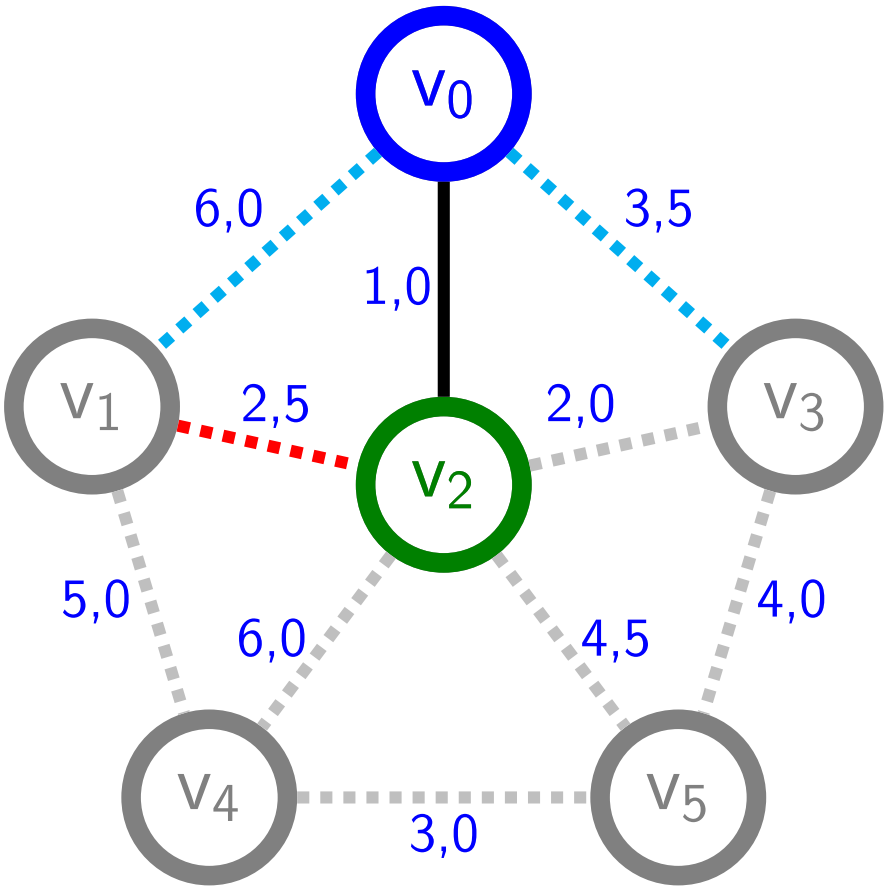
Árvore Geradora de Custo Mínimo - Prim



Custo: 1,0

Fila de Prioridade:
3 (3,5)
1 (6,0)

Árvore Geradora de Custo Mínimo - Prim



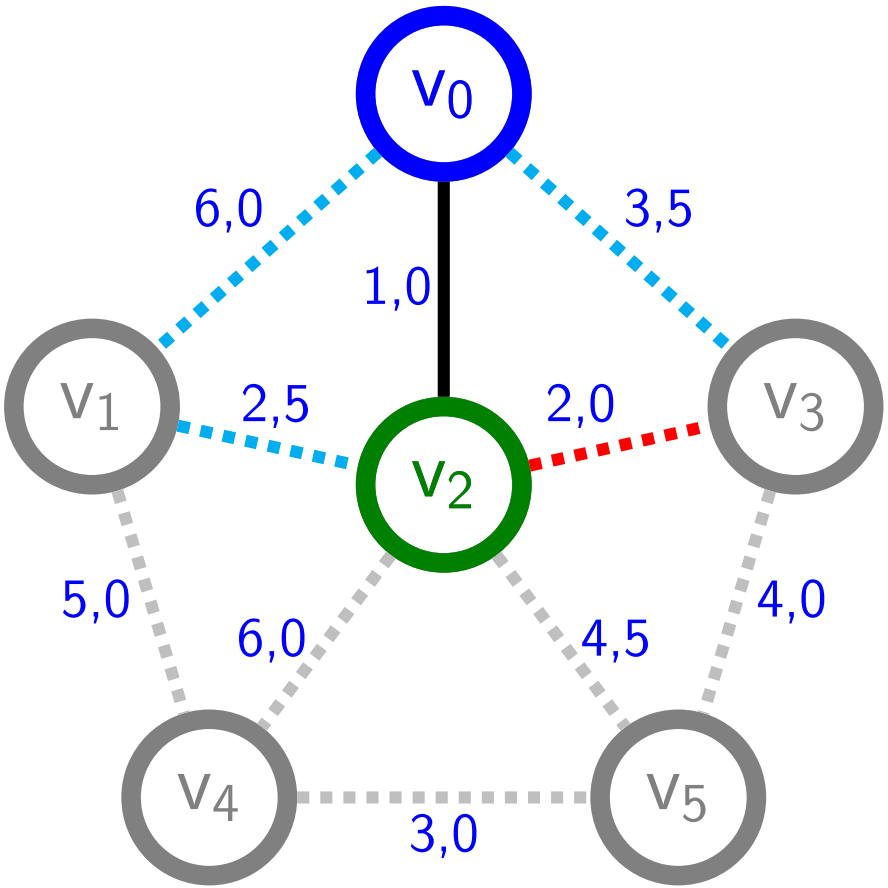
Custo: 1,0

Fila de Prioridade:

1 (2,5)

3 (3,5)

Árvore Geradora de Custo Mínimo - Prim



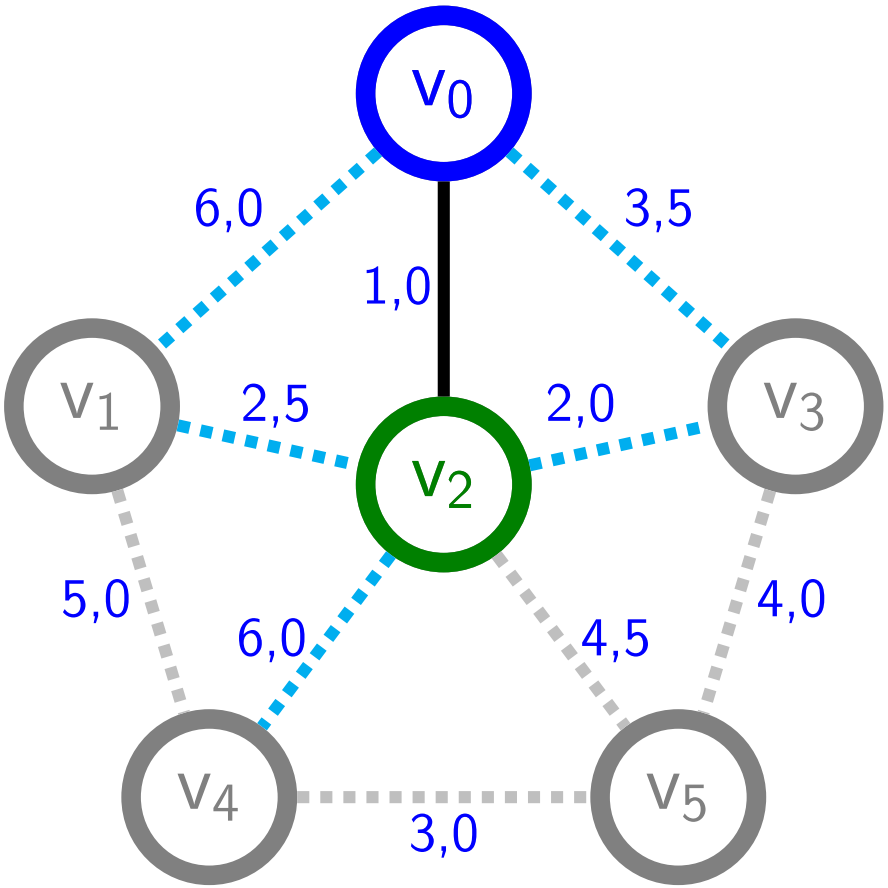
Custo: 1,0

Fila de Prioridade:

3 (2,0)

1 (2,5)

Árvore Geradora de Custo Mínimo - Prim



Custo: 1,0

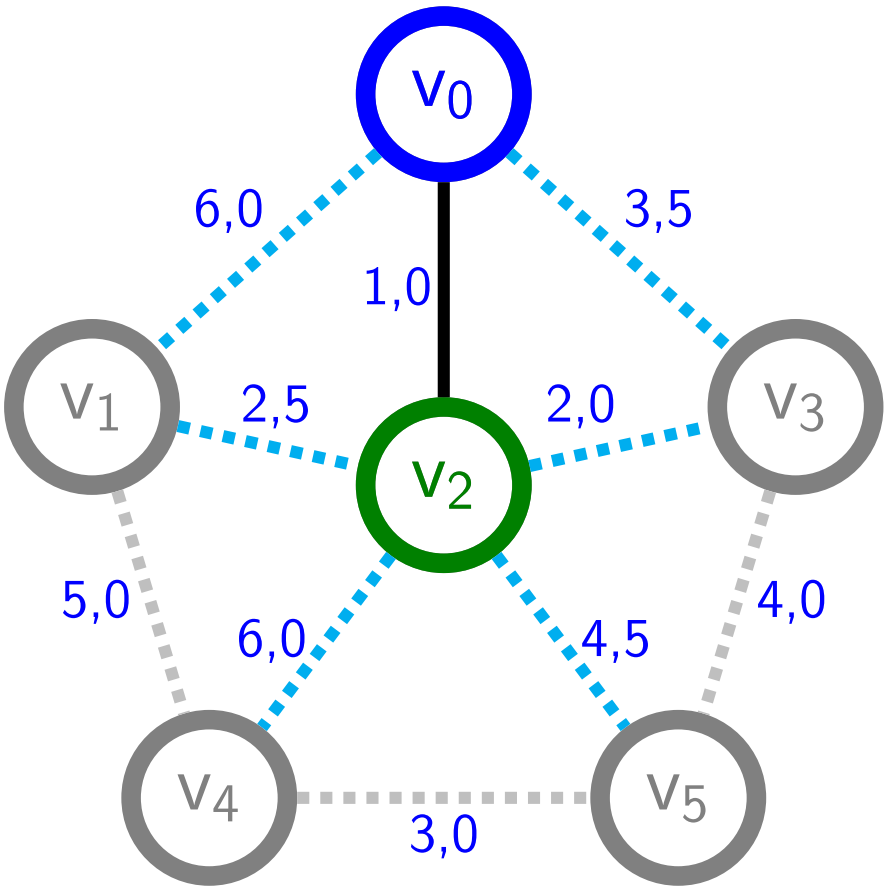
Fila de Prioridade:

3 (2,0)

1 (2,5)

4 (6,0)

Árvore Geradora de Custo Mínimo - Prim

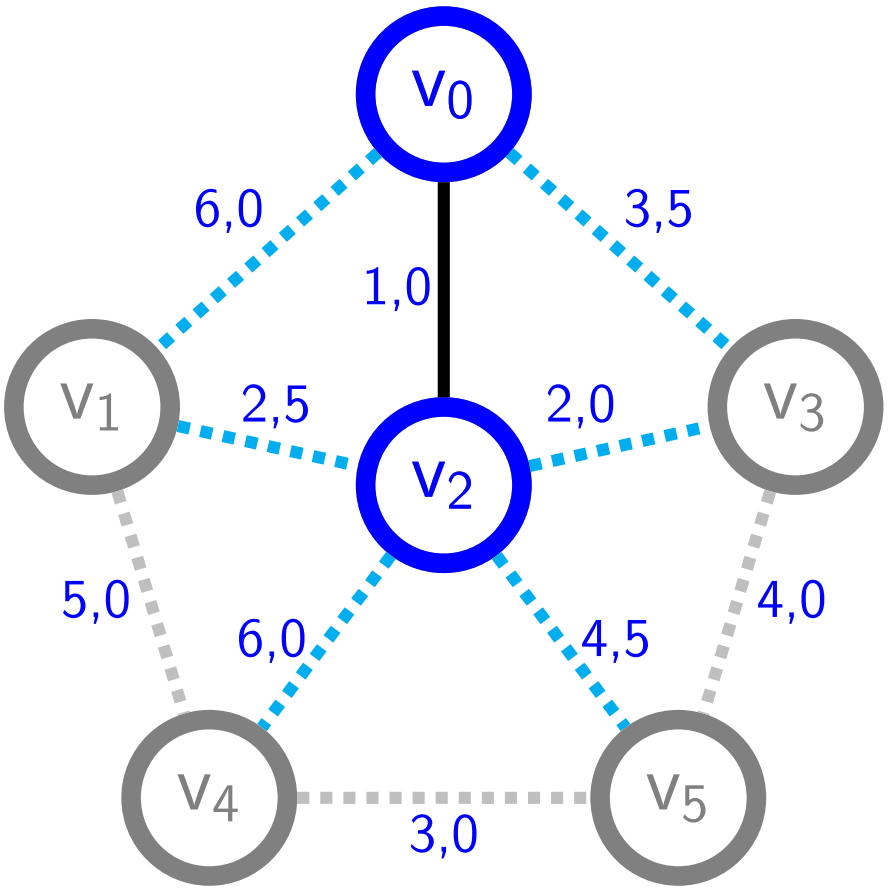


Custo: 1,0

Fila de Prioridade:

3 (2,0)
1 (2,5)
5 (4,5)
4 (6,0)

Árvore Geradora de Custo Mínimo - Prim

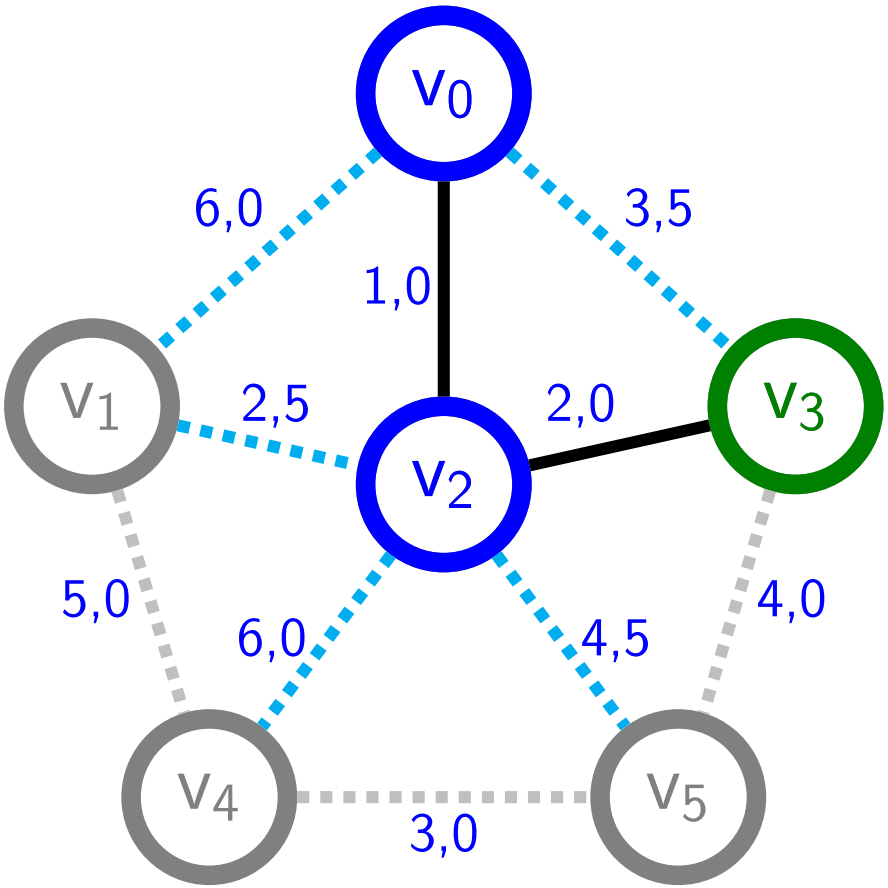


Custo: 1,0

Fila de Prioridade:

3 (2,0)
1 (2,5)
5 (4,5)
4 (6,0)

Árvore Geradora de Custo Mínimo - Prim

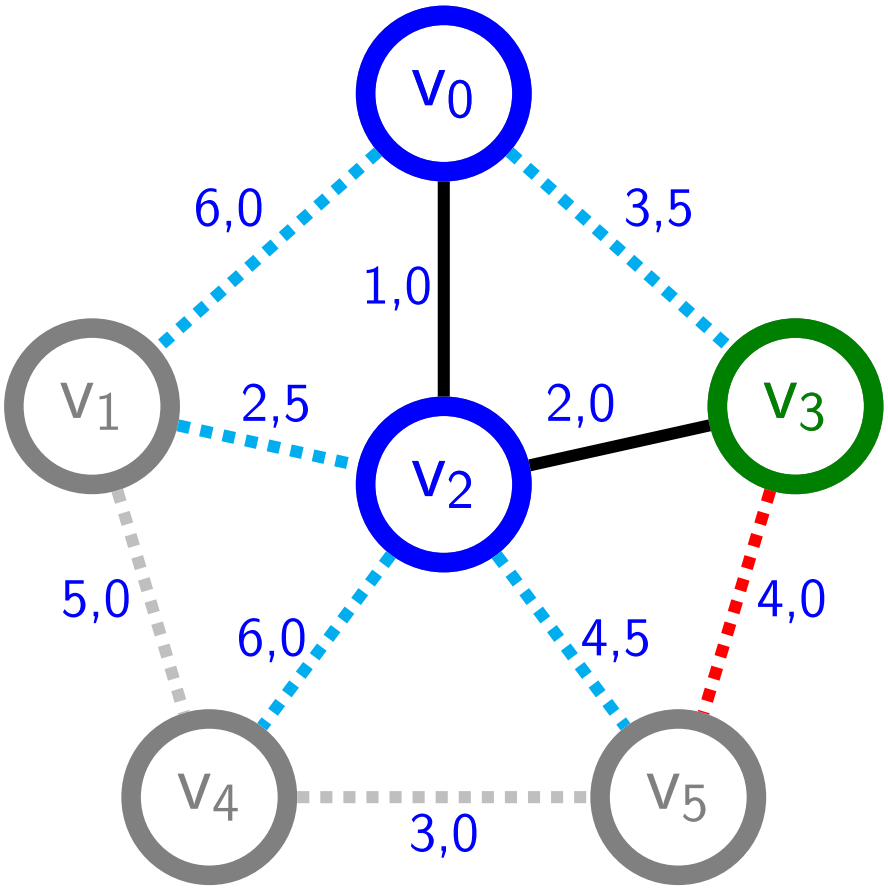


Custo: 3,0

Fila de Prioridade:

1 (2,5)
5 (4,5)
4 (6,0)

Árvore Geradora de Custo Mínimo - Prim



Custo: 3,0

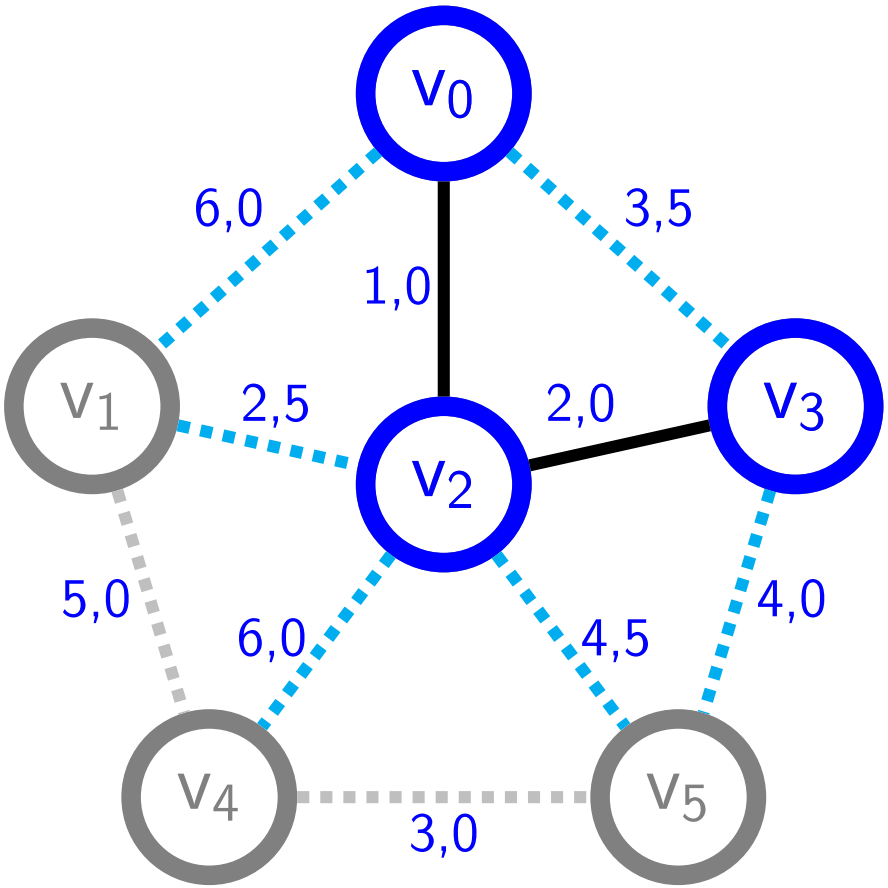
Fila de Prioridade:

1 (2,5)

5 (4,0)

4 (6,0)

Árvore Geradora de Custo Mínimo - Prim



Custo: 3,0

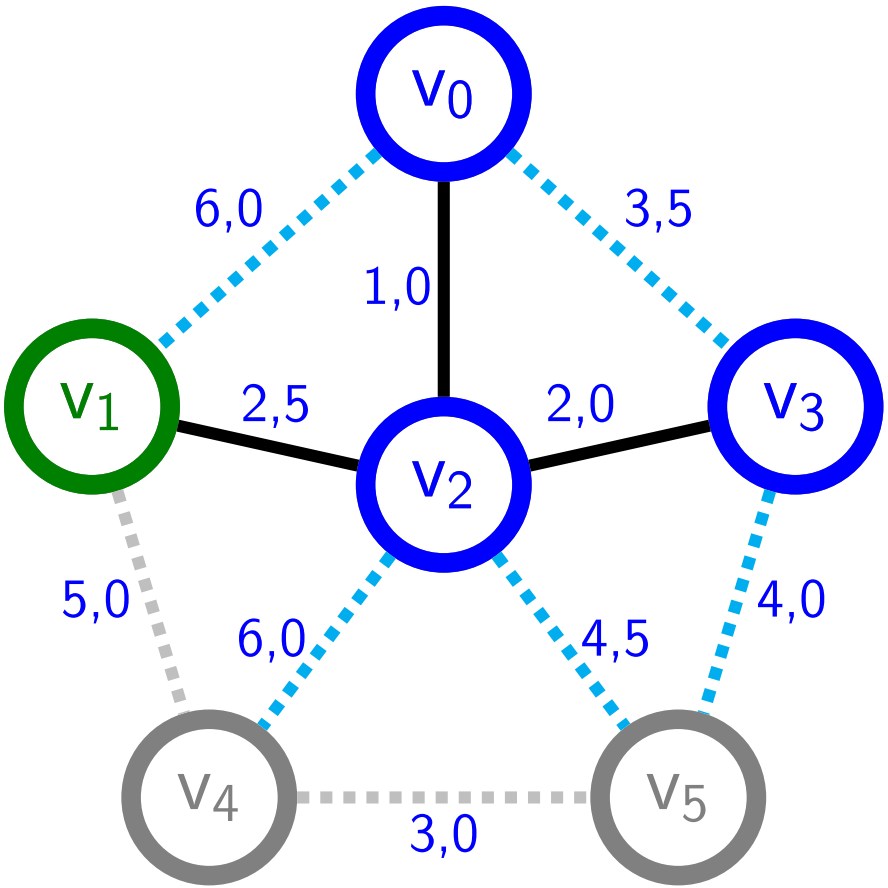
Fila de Prioridade:

1 (2,5)

5 (4,0)

4 (6,0)

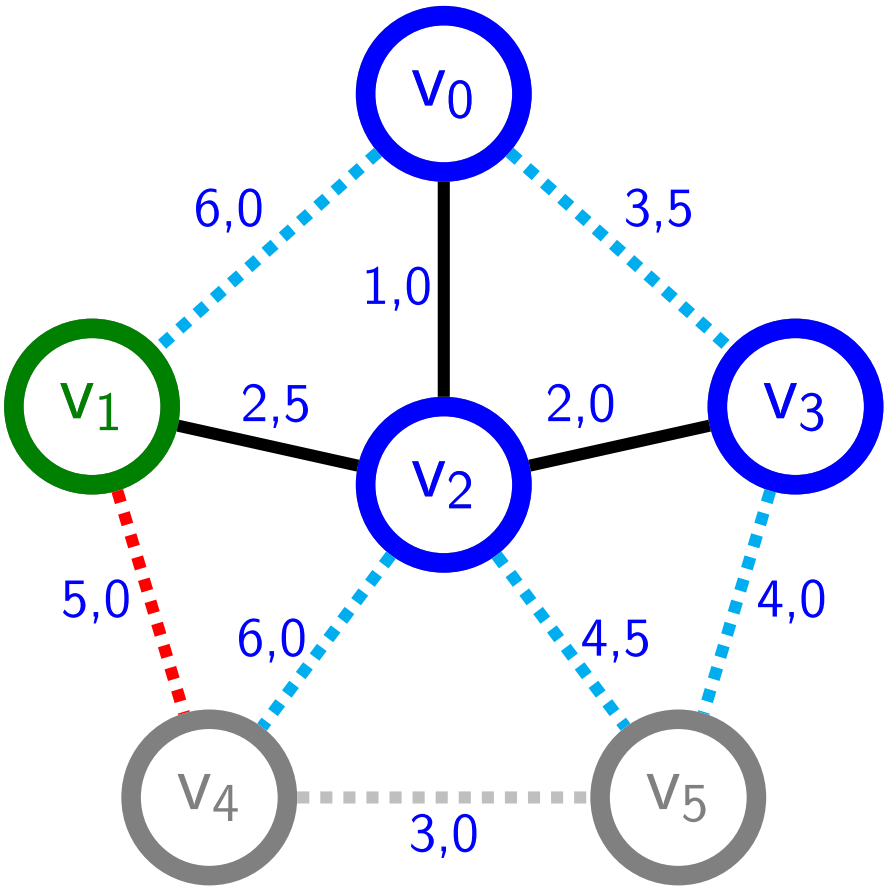
Árvore Geradora de Custo Mínimo - Prim



Custo: 5,5

Fila de Prioridade:
5 (4,0)
4 (6,0)

Árvore Geradora de Custo Mínimo - Prim



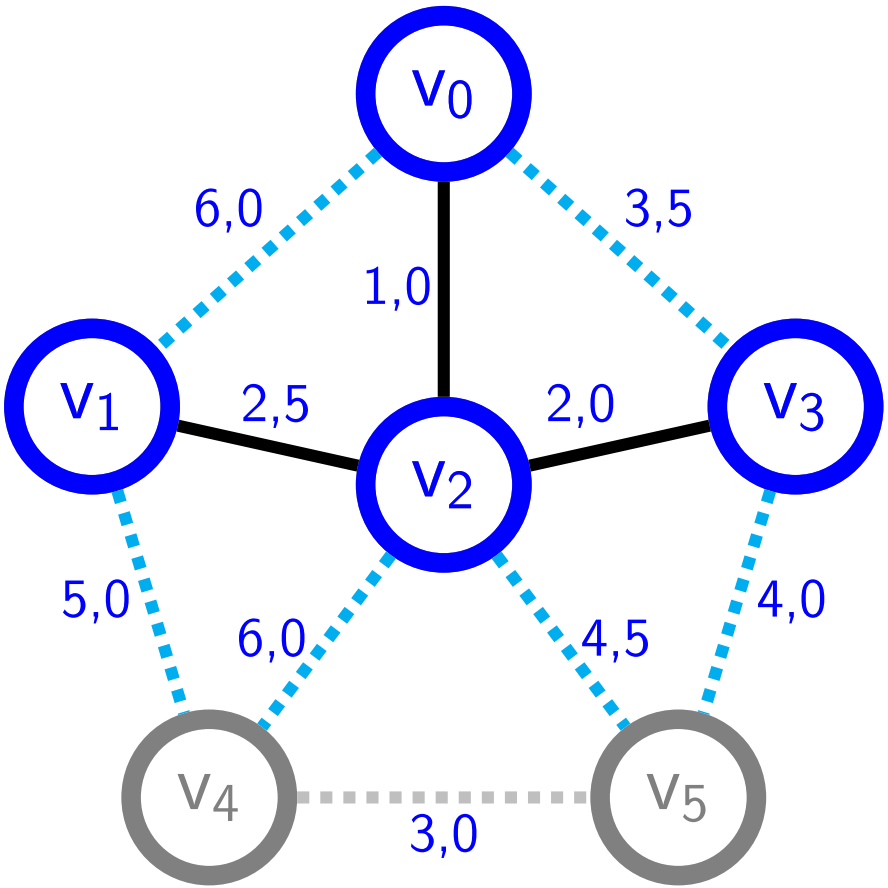
Custo: 5,5

Fila de Prioridade:

5 (4,0)

4 (5,0)

Árvore Geradora de Custo Mínimo - Prim



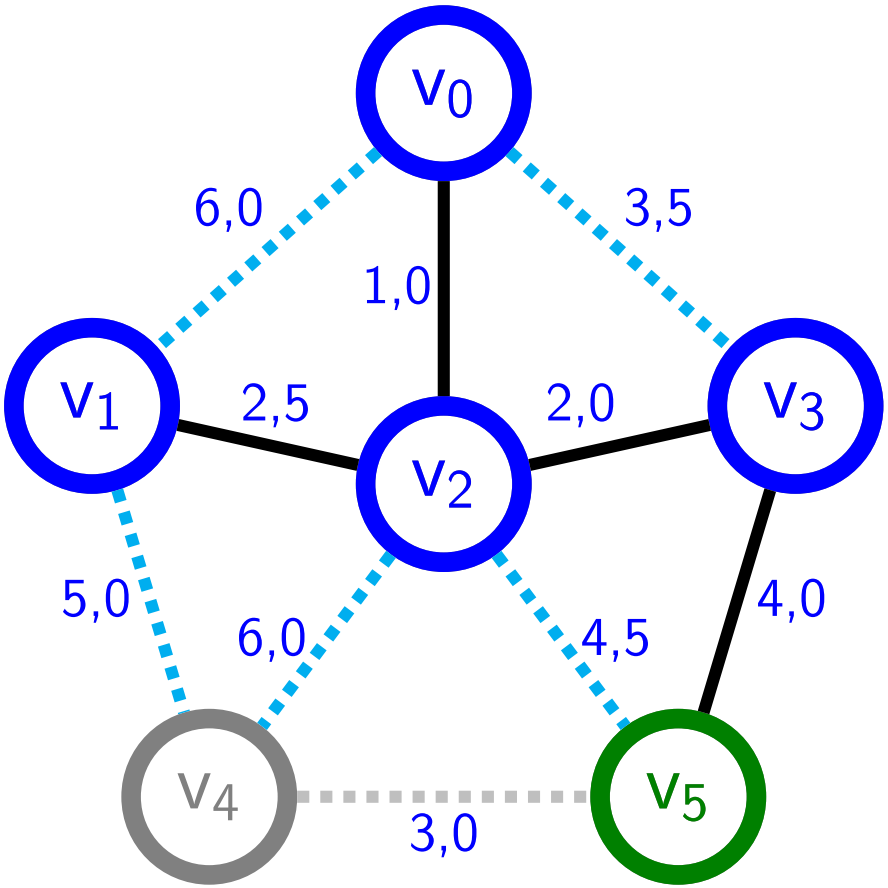
Custo: 5,5

Fila de Prioridade:

5 (4,0)

4 (5,0)

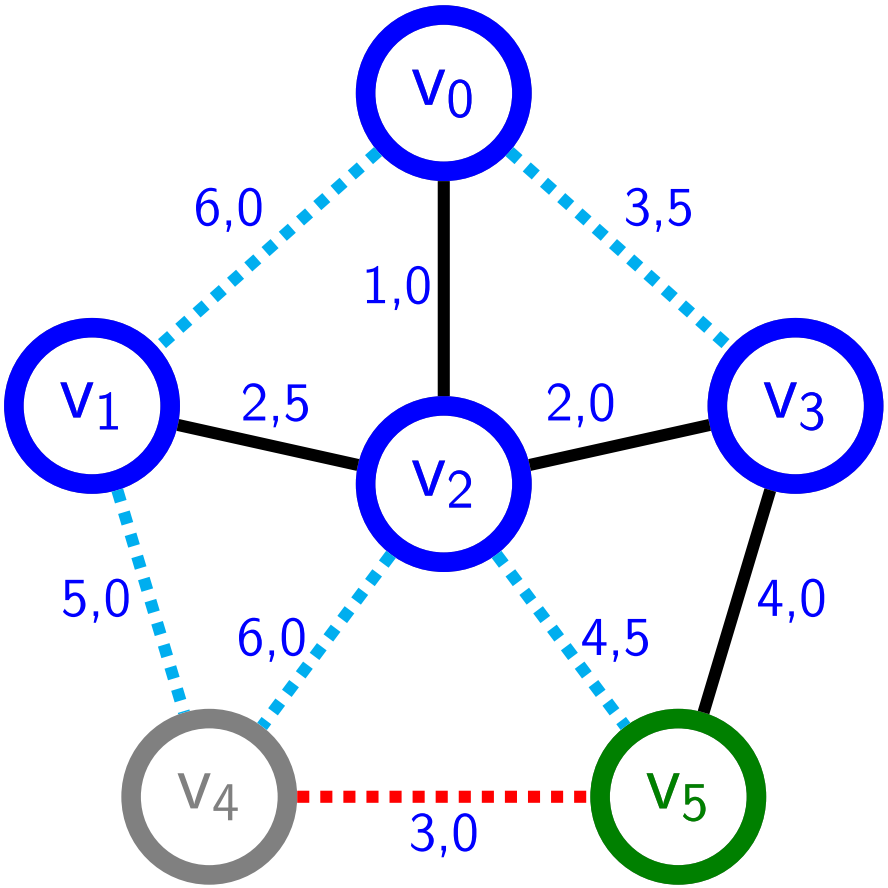
Árvore Geradora de Custo Mínimo - Prim



Custo: 9,5

Fila de Prioridade:
4 (5,0)

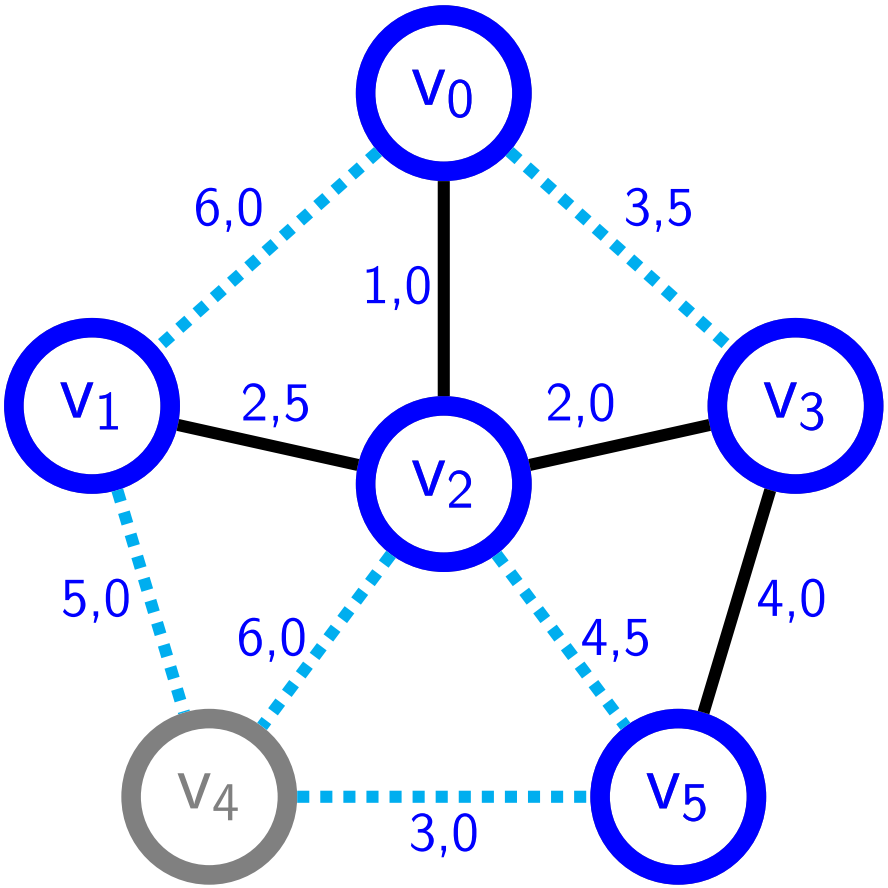
Árvore Geradora de Custo Mínimo - Prim



Custo: 9,5

Fila de Prioridade:
4 (3,0)

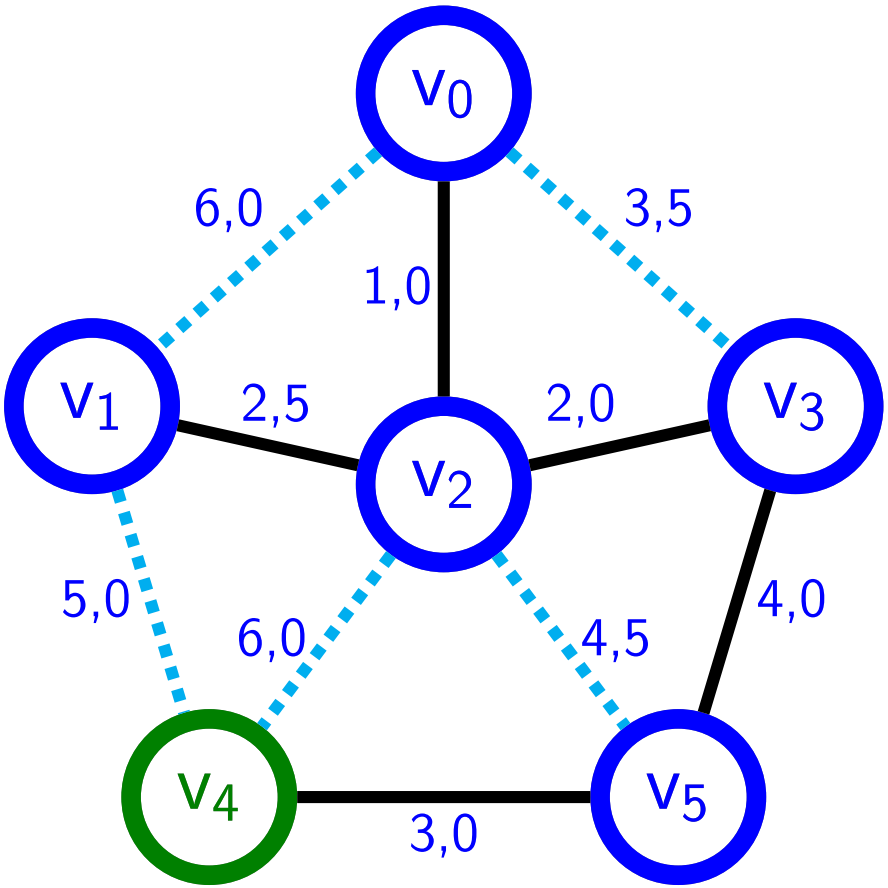
Árvore Geradora de Custo Mínimo - Prim



Custo: 9,5

Fila de Prioridade:
4 (3,0)

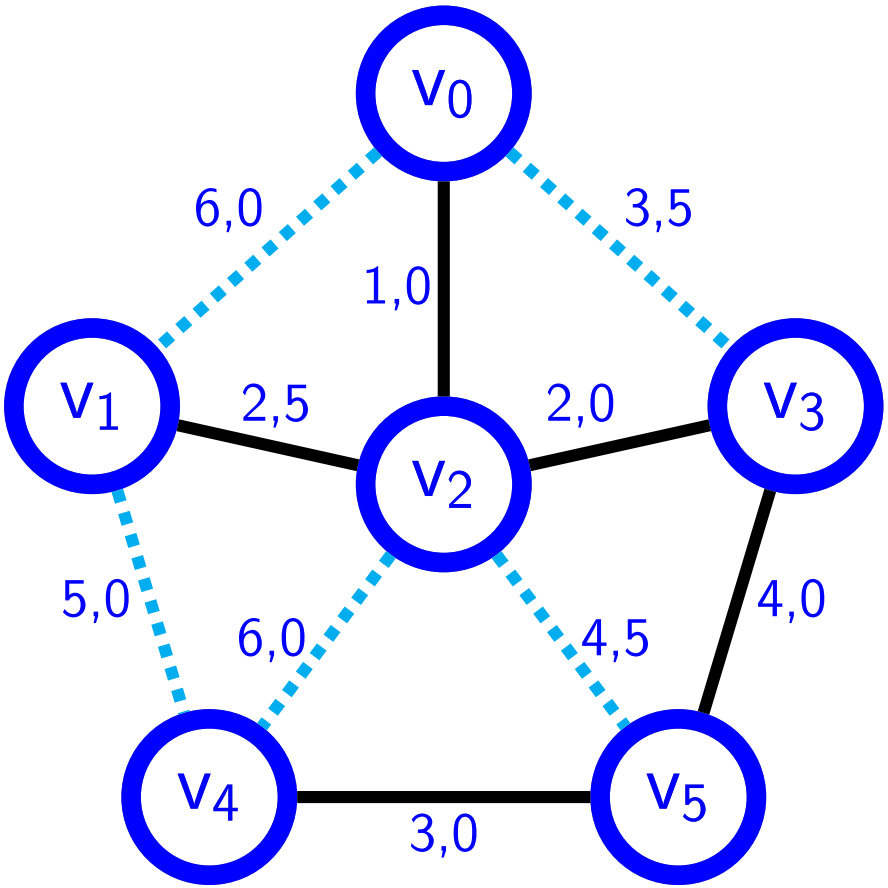
Árvore Geradora de Custo Mínimo - Prim



Custo: 12,5

Fila de Prioridade:

Árvore Geradora de Custo Mínimo - Prim



Custo: 12,5

Fila de Prioridade:

Árvore Geradora Mínima - Prim

```
Grafo* MSTPrim(Grafo* g){

...
```

Árvore Geradora Mínima - Prim

```
Grafo* MSTPrim(Grafo* g){
  if (!g || g->numVertices < 1) return NULL;

  ...
```

Árvore Geradora Mínima - Prim

```
Grafo* MSTPrim(Grafo* g){
  if (!g || g->numVertices < 1) return NULL;
  int n, e, x;
  n = g->numVertices;
  e = g->numArestas;
  Grafo* mst = (Grafo*)malloc(sizeof(Grafo));
  inicializaGrafo(mst, n);
```

...

Árvore Geradora Mínima - Prim

```
Grafo* MSTPrim(Grafo* g){
  if (!g || g->numVertices < 1) return NULL;
  int n, e, x;
  n = g->numVertices;
  e = g->numArestas;
  Grafo* mst = (Grafo*)malloc(sizeof(Grafo));
  inicializaGrafo(mst, n);
  Peso custo[n];
  int anterior[n];
  bool visitados[n];
  for (x=0;x<n;x++){
    visitados[x] = false;
    anterior[x] = -1;
    custo[x] = INFINITO;
  }

  ...
```

Árvore Geradora Mínima - Prim

. . .

. . .

Árvore Geradora Mínima - Prim

```
...

FilaDePrioridade fp;
inicializaFilaDePrioridade(&fp, n);

...
```

Árvore Geradora Mínima - Prim

```
...

FilaDePrioridade fp;
inicializaFilaDePrioridade(&fp, n);
int atual = 0;
PONT vizinhos = listaDeVizinhos(g, atual);
visitados[atual] = true;

...
```

Árvore Geradora Mínima - Prim

```
...

FilaDePrioridade fp;
inicializaFilaDePrioridade(&fp, n);
int atual = 0;
PONT vizinhos = listaDeVizinhos(g, atual);
visitados[atual] = true;
while (vizinhos){



  vizinhos = vizinhos->prox;
}

...
```

Árvore Geradora Mínima - Prim

```
...

FilaDePrioridade fp;
inicializaFilaDePrioridade(&fp, n);
int atual = 0;
PONT vizinhos = listaDeVizinhos(g, atual);
visitados[atual] = true;
while (vizinhos){
   adicionaElemento(&fp,vizinhos->vertice, vizinhos->peso);
   anterior[vizinhos->vertice] = atual;
   custo[vizinhos->vertice] = vizinhos->peso;
   vizinhos = vizinhos->prox;
}

...
```

Árvore Geradora Mínima - Prim

```
}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){









  }

}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;

  }

}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;
    vizinhos = listaDeVizinhos(g, atual) ;
    while (vizinhos){









      vizinhos = vizinhos->prox;
    }
  }


}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;
    vizinhos = listaDeVizinhos(g, atual) ;
    while (vizinhos){
      if (!visitados[vizinhos->vertice] &&
                        custo[vizinhos->vertice] > vizinhos->peso){









      }
      vizinhos = vizinhos->prox;
    }
  }


}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;
    vizinhos = listaDeVizinhos(g, atual) ;
    while (vizinhos){
      if (!visitados[vizinhos->vertice] &&
                        custo[vizinhos->vertice] > vizinhos->peso){
        if (anterior[vizinhos->vertice] == -1)
          adicionaElemento(&fp,vizinhos->vertice, vizinhos->peso) ;
      }
      vizinhos = vizinhos->prox;
    }
  }

}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;
    vizinhos = listaDeVizinhos(g, atual) ;
    while (vizinhos){
      if (!visitados[vizinhos->vertice] &&
                        custo[vizinhos->vertice] > vizinhos->peso){
        if (anterior[vizinhos->vertice] == -1)
          adicionaElemento(&fp,vizinhos->vertice, vizinhos->peso) ;
        else diminuiPrioridade(&fp,vizinhos->vertice, vizinhos->peso) ;


      }
      vizinhos = vizinhos->prox;
    }
  }


}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;
    vizinhos = listaDeVizinhos(g, atual) ;
    while (vizinhos){
      if (!visitados[vizinhos->vertice] &&
                        custo[vizinhos->vertice] > vizinhos->peso){
        if (anterior[vizinhos->vertice] == -1)
          adicionaElemento(&fp,vizinhos->vertice, vizinhos->peso) ;
        else diminuiPrioridade(&fp,vizinhos->vertice, vizinhos->peso) ;
        anterior[vizinhos->vertice] = atual;
        custo[vizinhos->vertice] = vizinhos->peso;
      }
      vizinhos = vizinhos->prox;
    }
  }


}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;
    vizinhos = listaDeVizinhos(g, atual) ;
    while (vizinhos){
      if (!visitados[vizinhos->vertice] &&
                        custo[vizinhos->vertice] > vizinhos->peso){
        if (anterior[vizinhos->vertice] == -1)
          adicionaElemento(&fp,vizinhos->vertice, vizinhos->peso) ;
        else diminuiPrioridade(&fp,vizinhos->vertice, vizinhos->peso) ;
        anterior[vizinhos->vertice] = atual;
        custo[vizinhos->vertice] = vizinhos->peso;
      }
      vizinhos = vizinhos->prox;
    }
  }
  liberaFilaDePrioridade(&fp) ;
  return mst;
}
```

Árvore Geradora Mínima - Prim

```
  while(! filaDePrioridadeEstaVazia(&fp) ){
    atual = excluiElemento(&fp) ;
    visitados[atual] = true;
    insereAresta(mst, anterior[atual], atual, custo[atual]) ;
    vizinhos = listaDeVizinhos(g, atual) ;
    while (vizinhos){
      if (!visitados[vizinhos->vertice] &&
                        custo[vizinhos->vertice] > vizinhos->peso){
        if (anterior[vizinhos->vertice] == -1)
            adicionaElemento(&fp,vizinhos->vertice, vizinhos->peso) ;
        else diminuiPrioridade(&fp,vizinhos->vertice, vizinhos->peso) ;
        anterior[vizinhos->vertice] = atual;
        custo[vizinhos->vertice] = vizinhos->peso;
      }
      vizinhos = vizinhos->prox;
    }
  }
  liberaFilaDePrioridade(&fp) ;
  return mst;
}
```

$O(Elog(V))$

Algoritmos e Estruturas de Dados II
Aula 10 – Árvores Geradoras de Custo Mínimo - Algoritmo de Prim

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