Algoritmos e Estruturas de Dados II
Aula 06 – Grafos: Busca em Profundidade

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

Grafos – Percursos

Como podemos navegar em um grafo?
Isto é, visitar seus vértices caminhando por meio das arestas, fazer buscas etc?

Grafos - Percursos

Há duas formas mais comuns de se fazer isso:

Grafos - Percursos

Há duas formas mais comuns de se fazer isso:

- Busca em Profundidade

Grafos - Percursos

Há duas formas mais comuns de se fazer isso:

- Busca em Profundidade
- Busca em Largura

Grafos - Percursos

Há duas formas mais comuns de se fazer isso:

- **Busca em Profundidade**
- Busca em Largura

Grafos – Busca em Profundidade

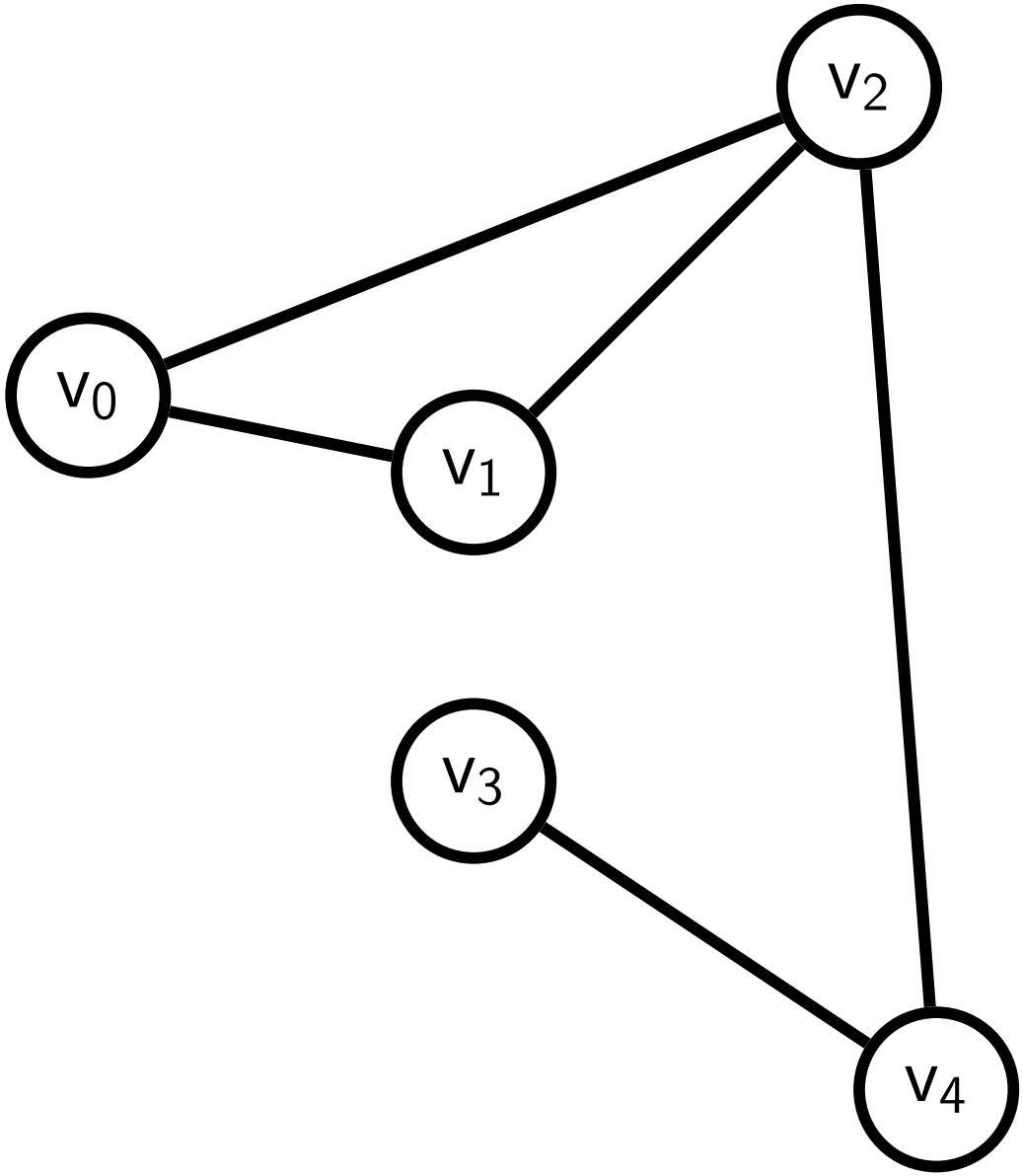
A Busca em Profundidade (ou do inglês *Depth-First Search - DFS*) é um algoritmo ou estratégia que pode ser utilizada para percorrer estruturas como árvores e grafos.

Grafos – Busca em Profundidade

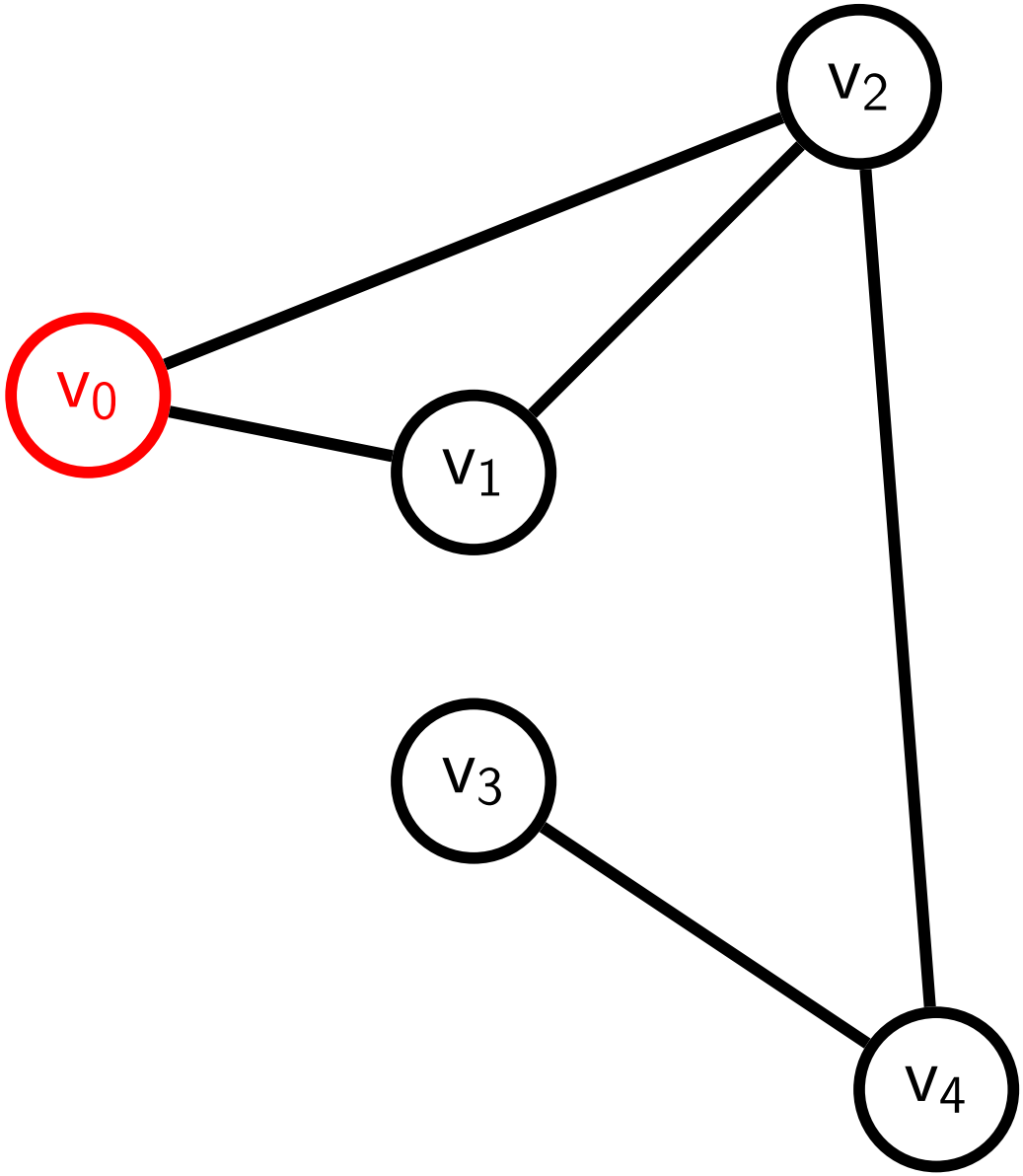
A Busca em Profundidade (ou do inglês *Depth-First Search - DFS*) é um algoritmo ou estratégia que pode ser utilizada para percorrer estruturas como árvores e grafos.

Em grafos, o algoritmo começa a partir de um vértice e explora a maior quantidade possível de vértices (navegando pelas arestas) usando a seguinte regra: a partir do primeiro vértice (potencialmente um vértice arbitrário) visita-se, recursivamente, seus vizinhos.

Grafos – Relembrando

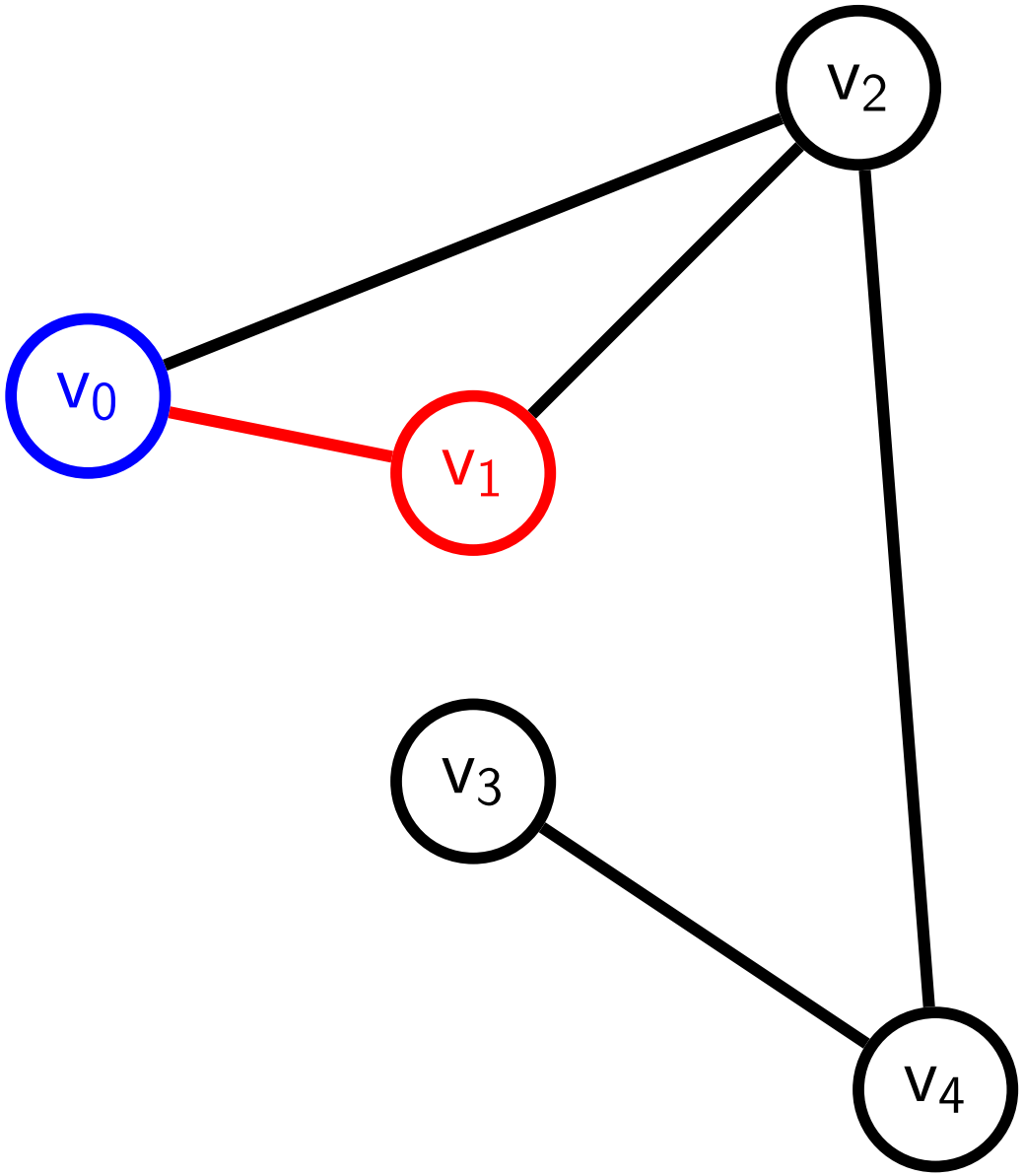


Grafos – Relembrando



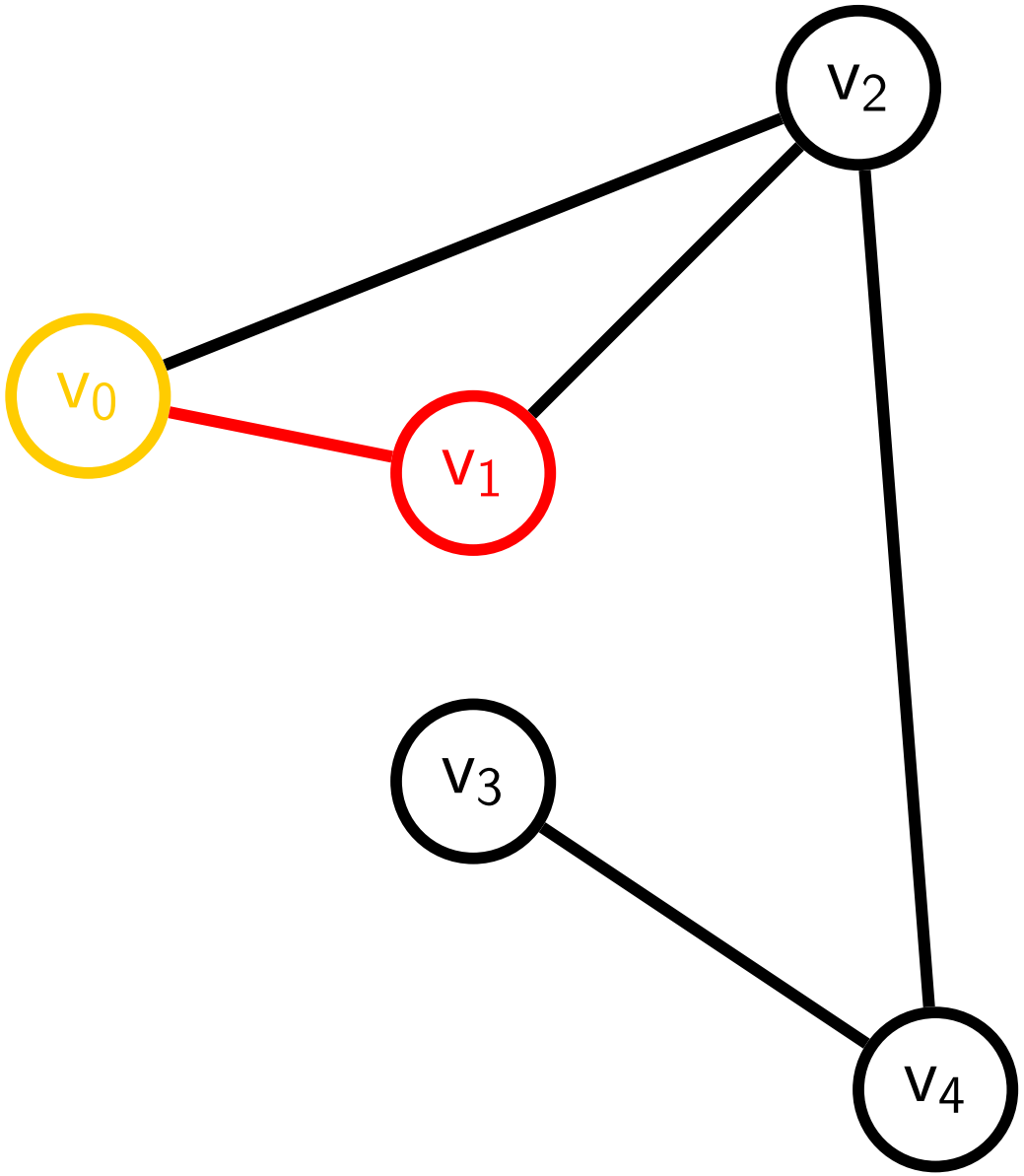
vértice	anterior
v0	-

Grafos – Relembrando



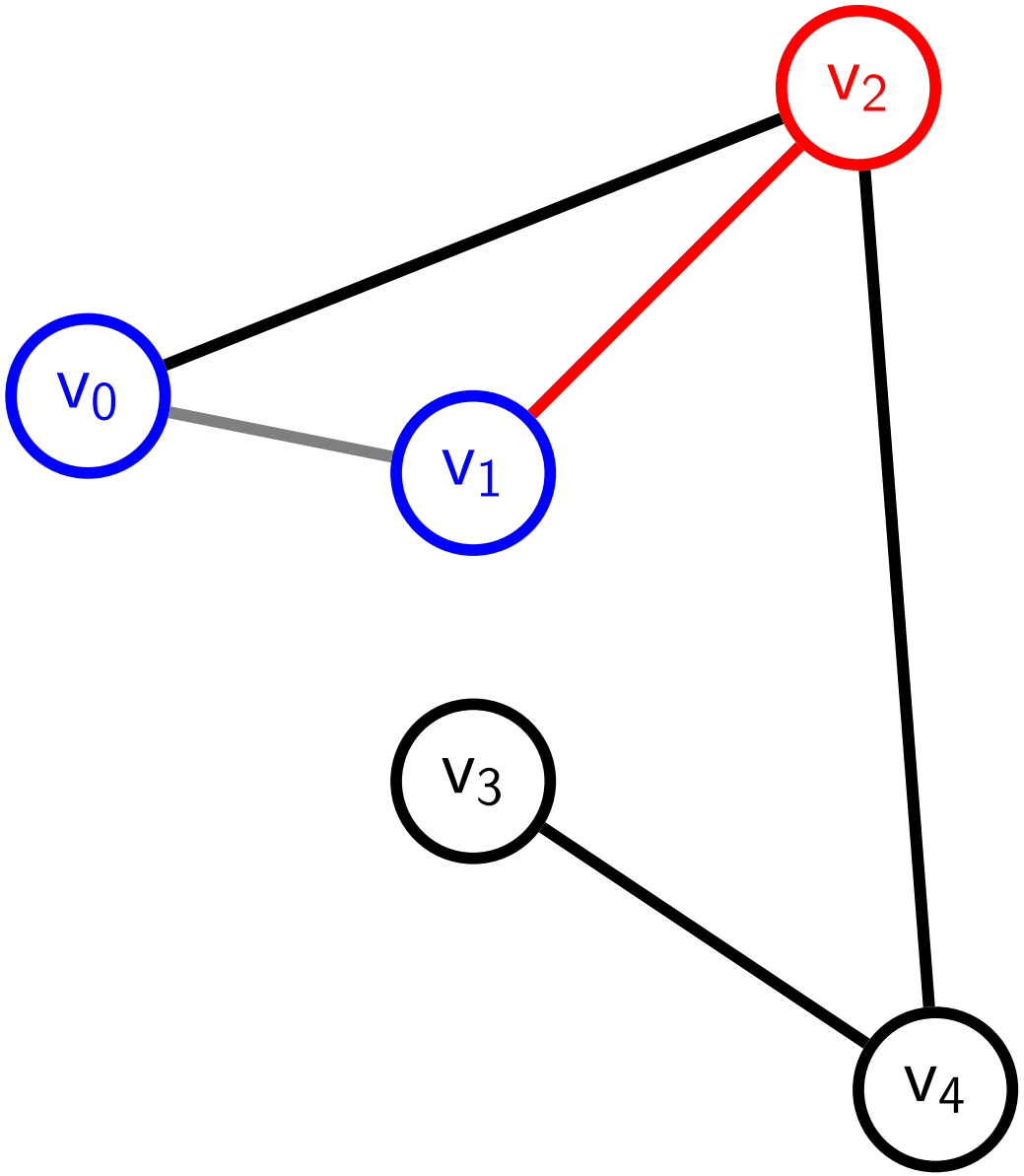
vértice	anterior
v0	-
v1	v0

Grafos – Relembrando



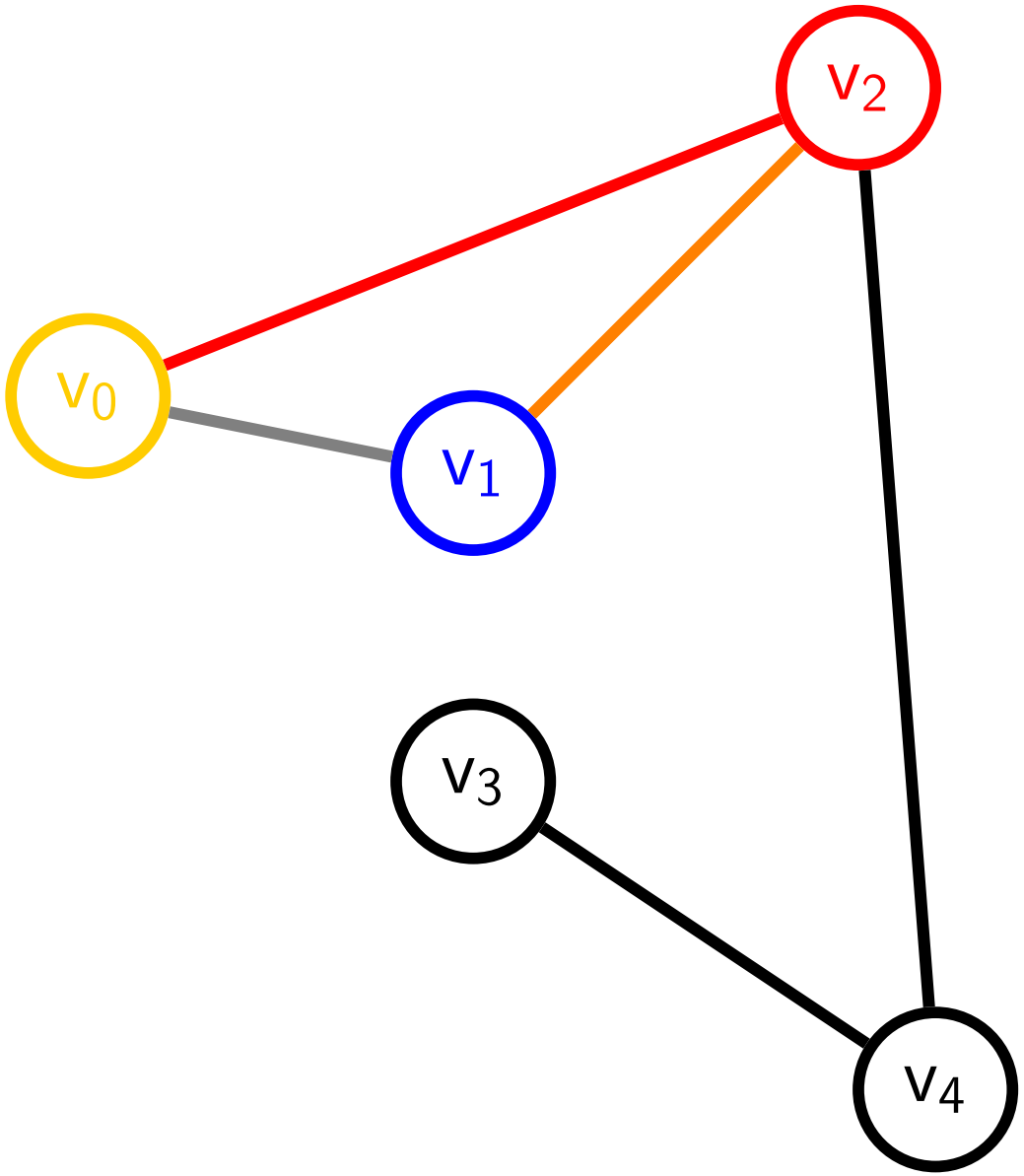
vértice	anterior
v0	-
v1	v0

Grafos – Relembrando



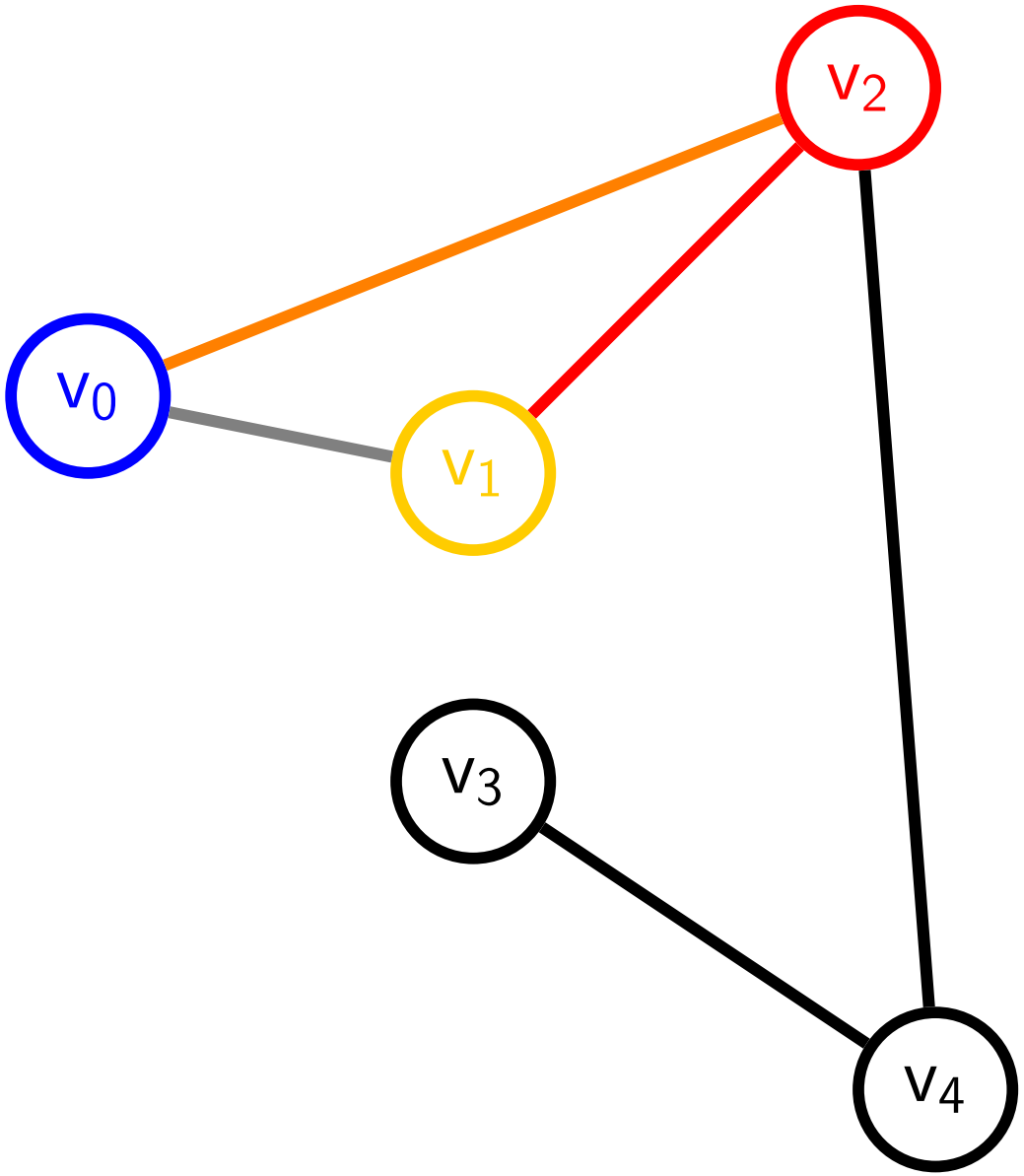
vértice	anterior
v0	-
v1	v0
v2	v1

Grafos – Relembrando



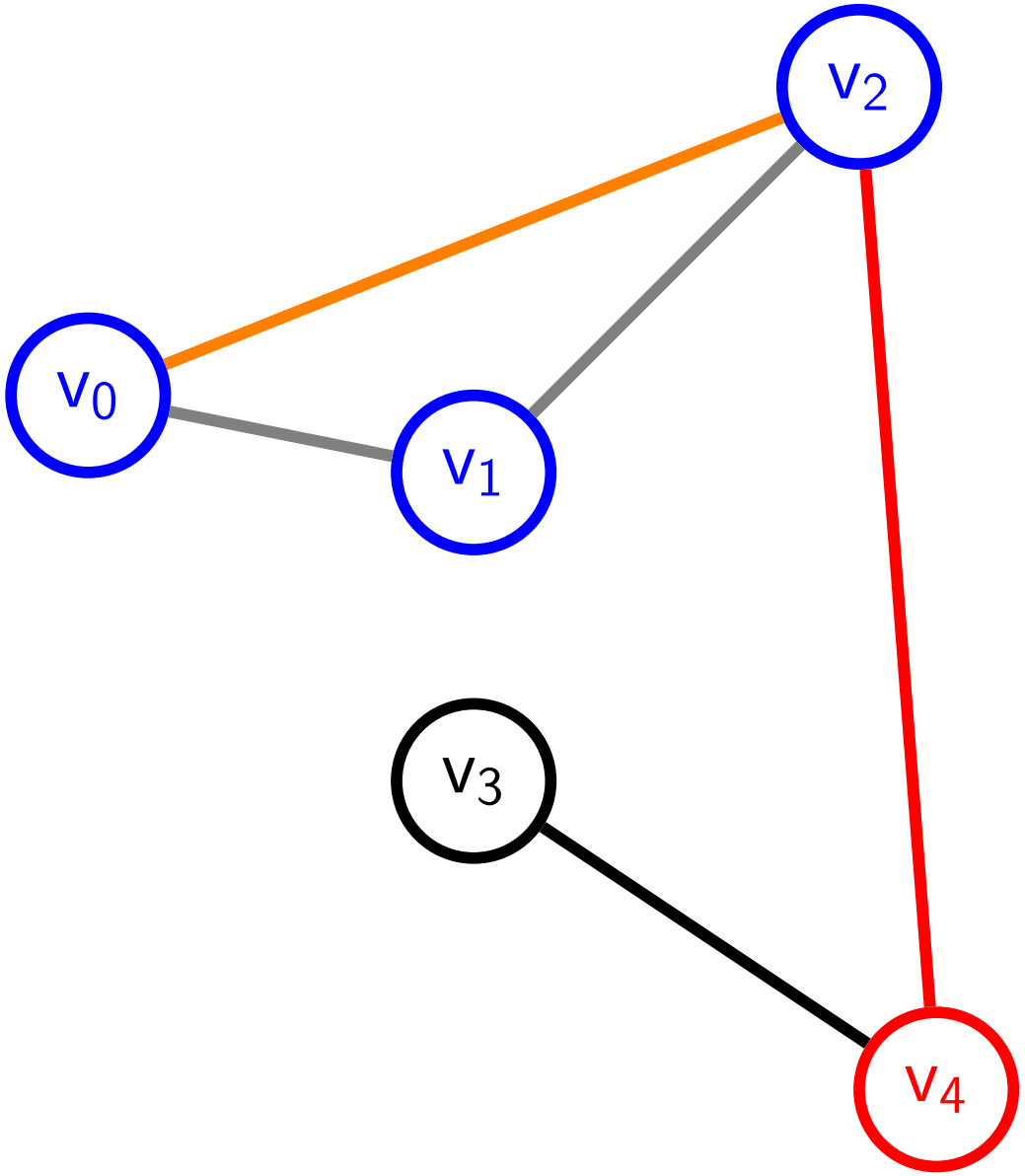
vértice	anterior
v0	-
v1	v0
v2	v1

Grafos – Relembrando



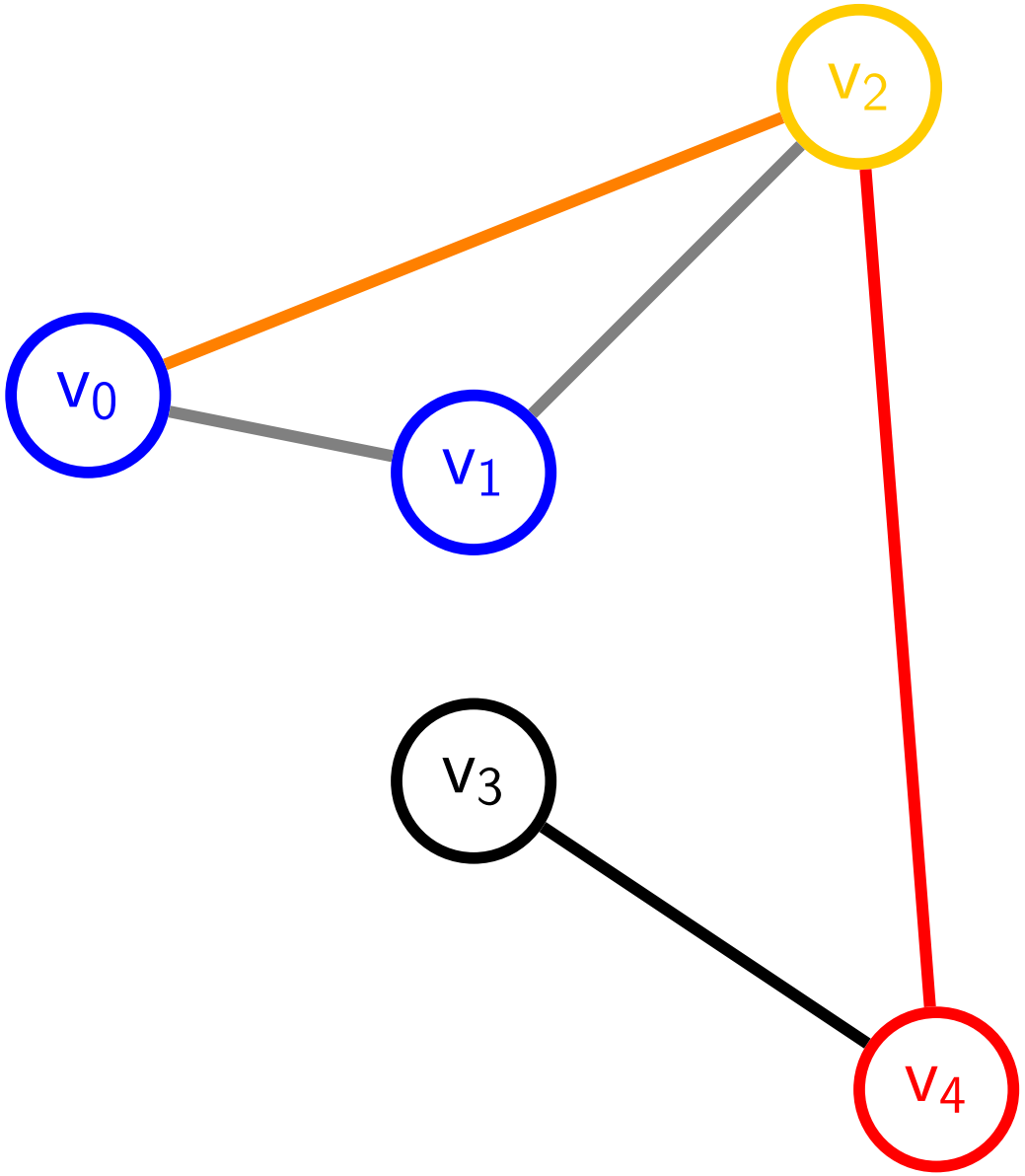
vértice	anterior
v0	-
v1	v0
v2	v1

Grafos – Relembrando



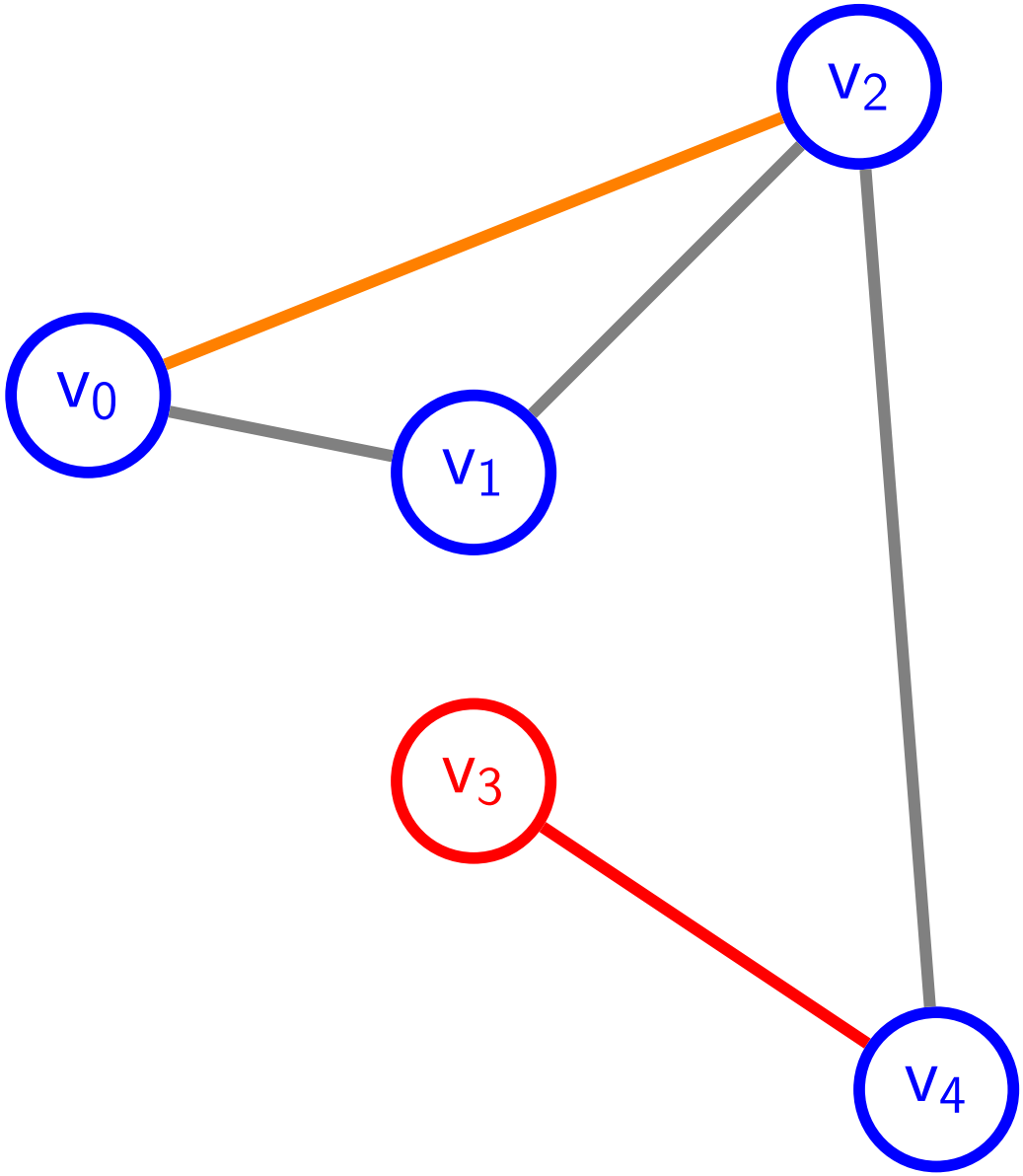
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2

Grafos – Relembrando



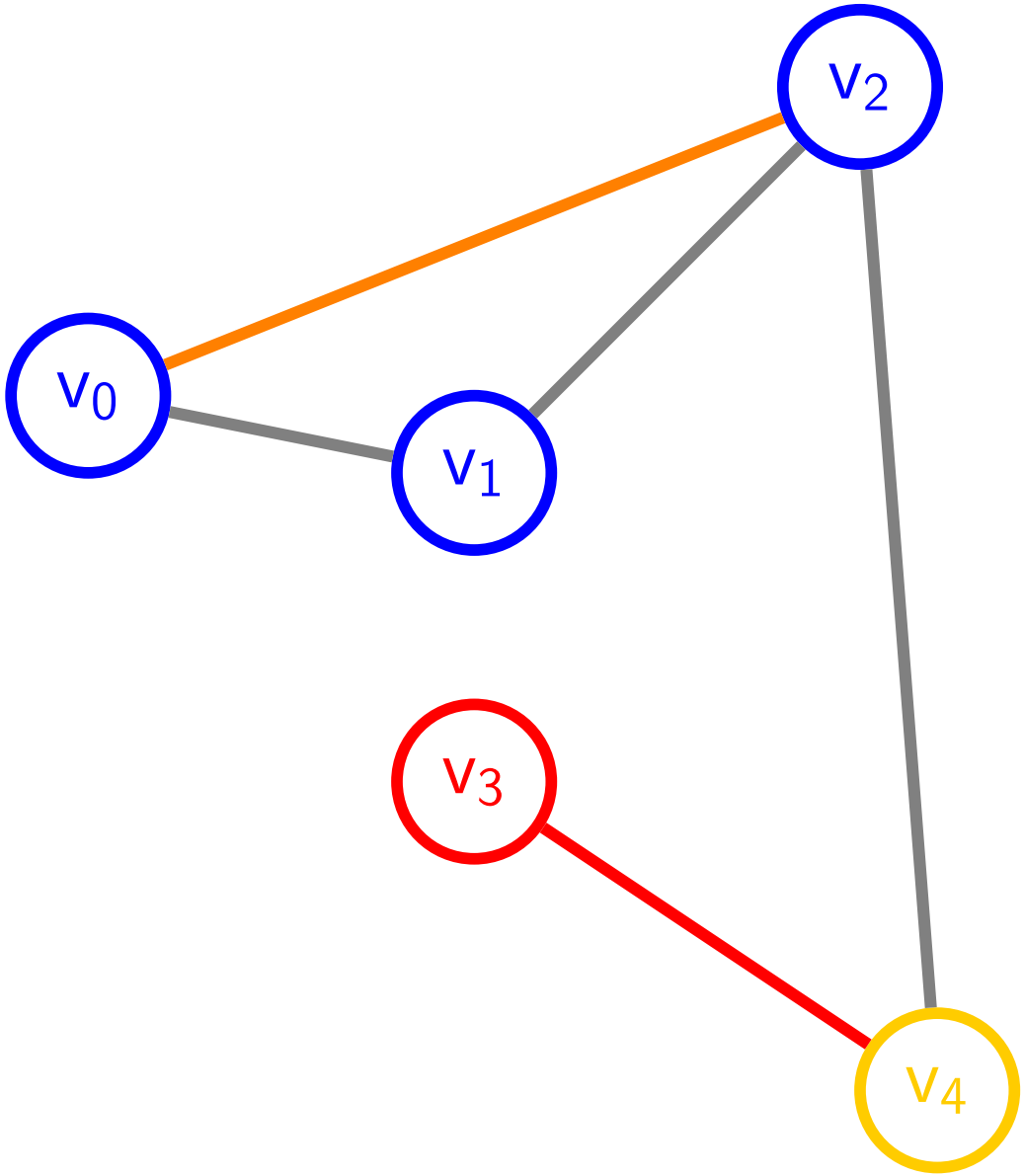
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2

Grafos – Relembrando



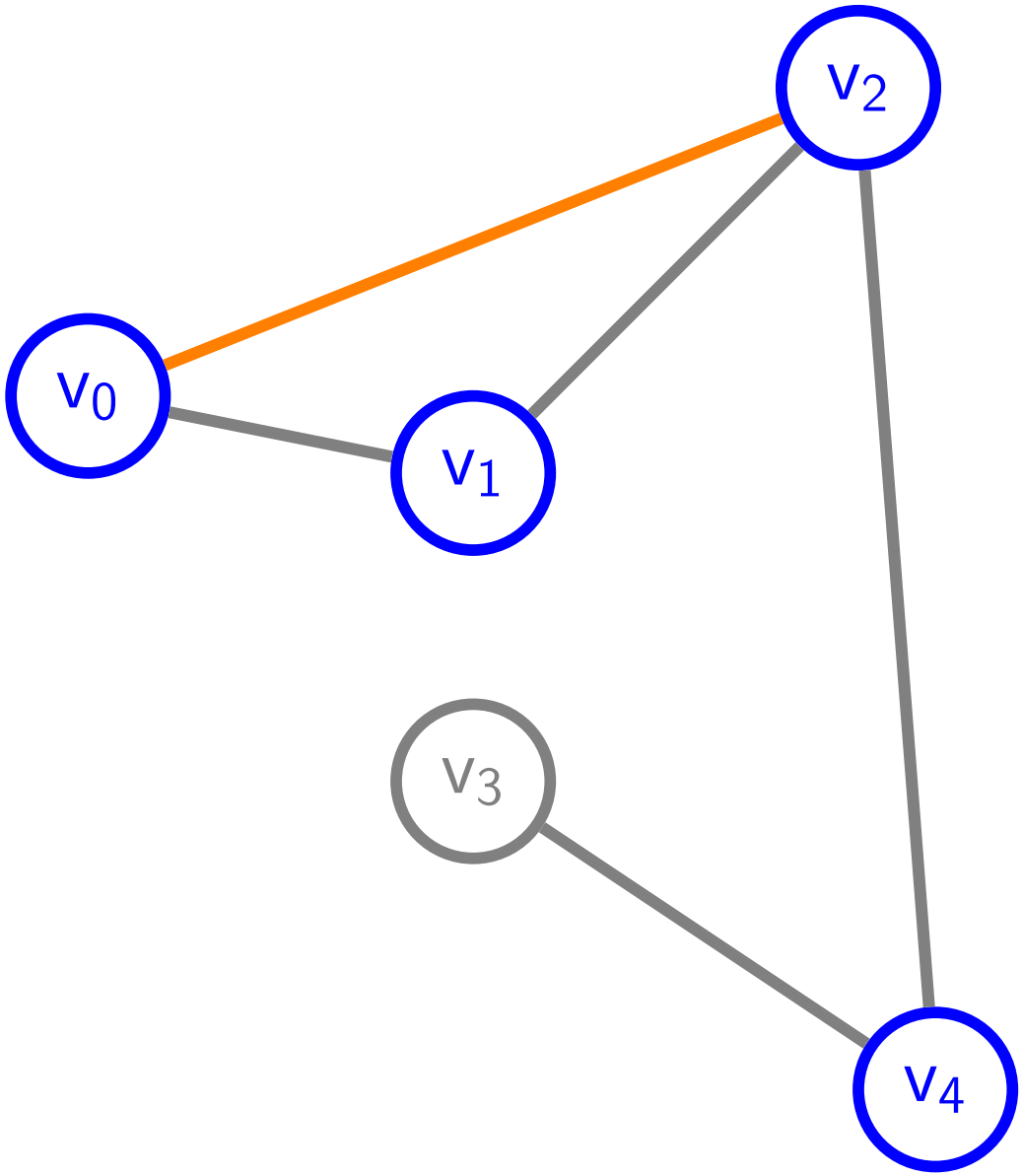
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



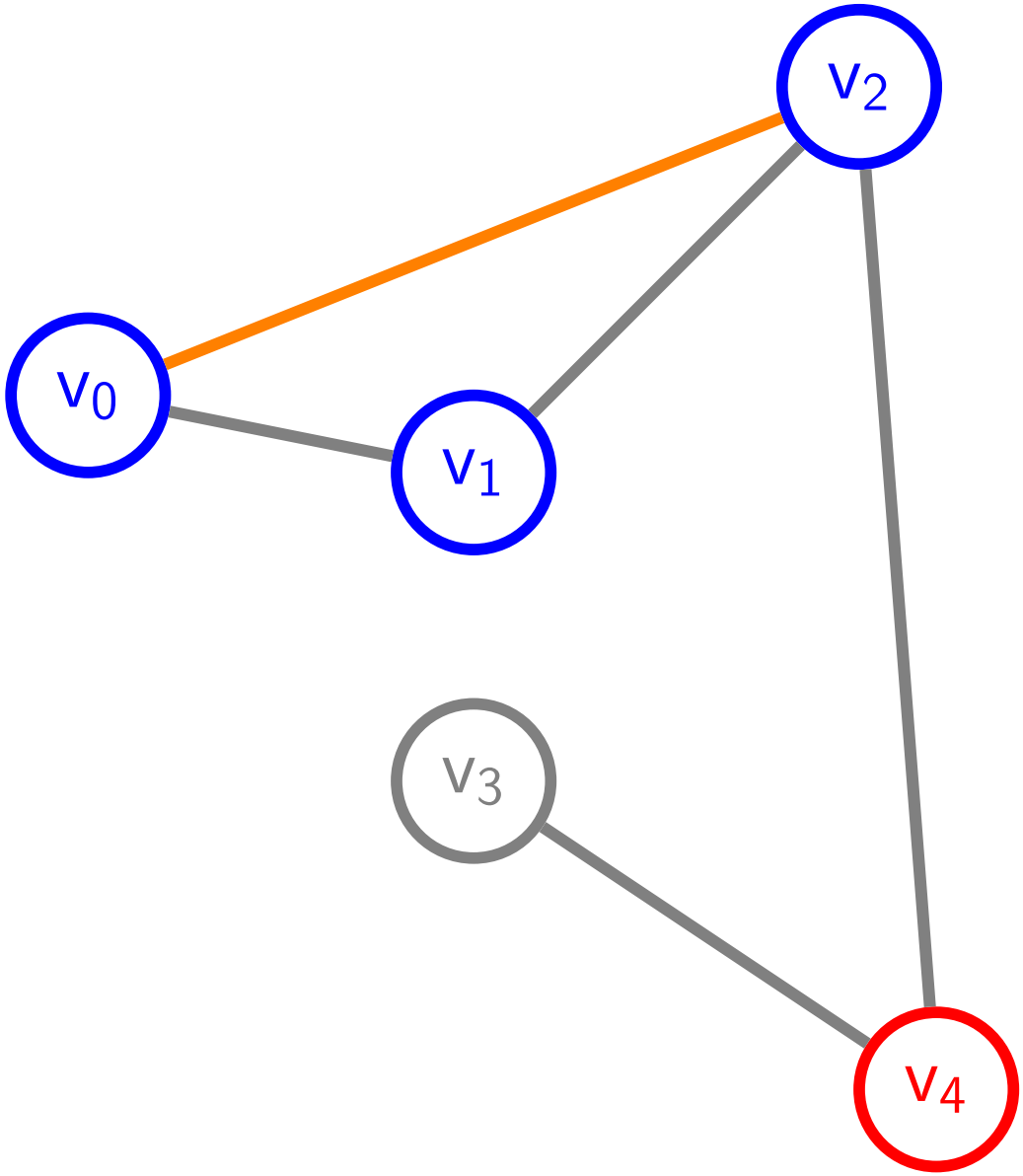
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



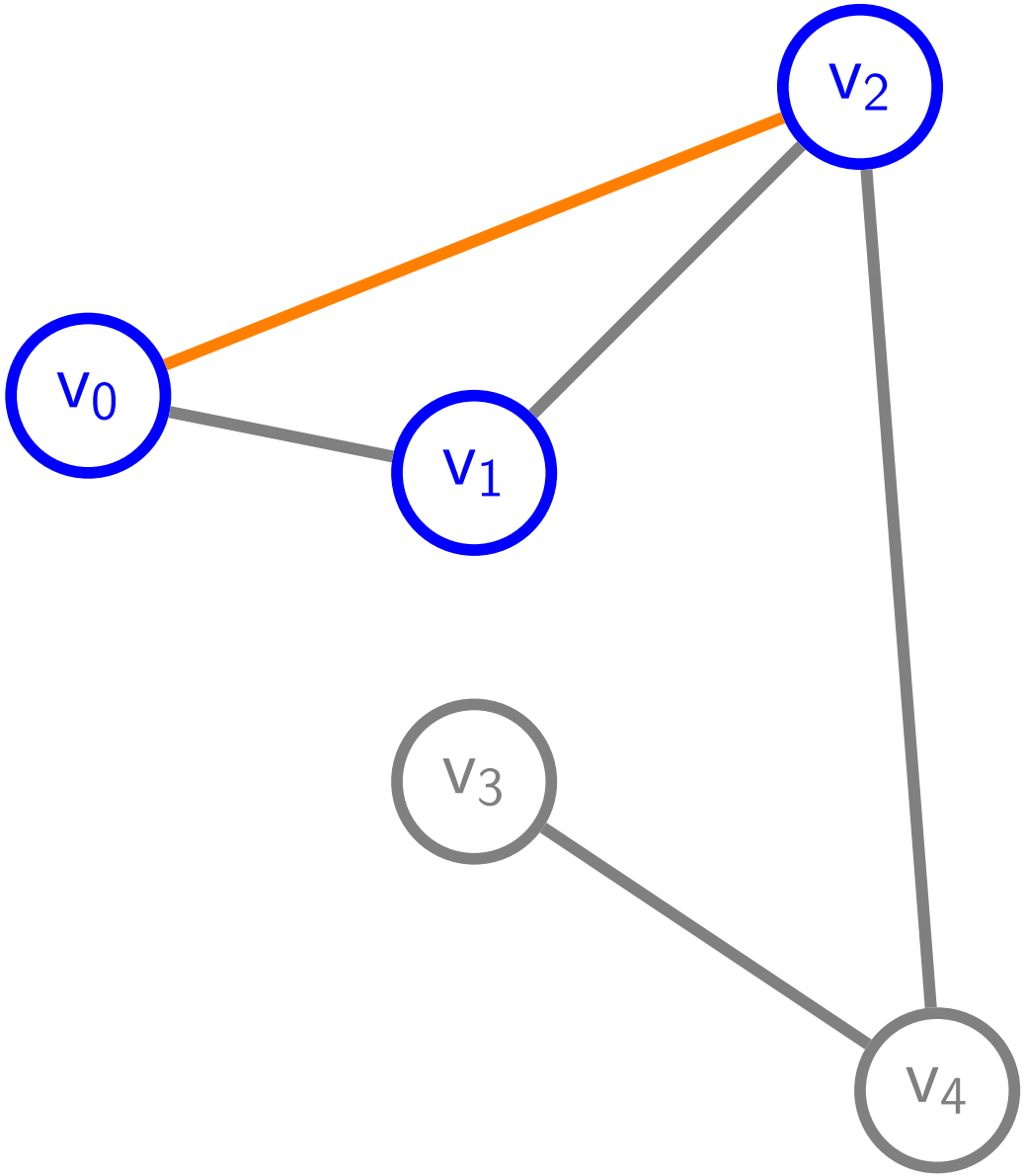
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



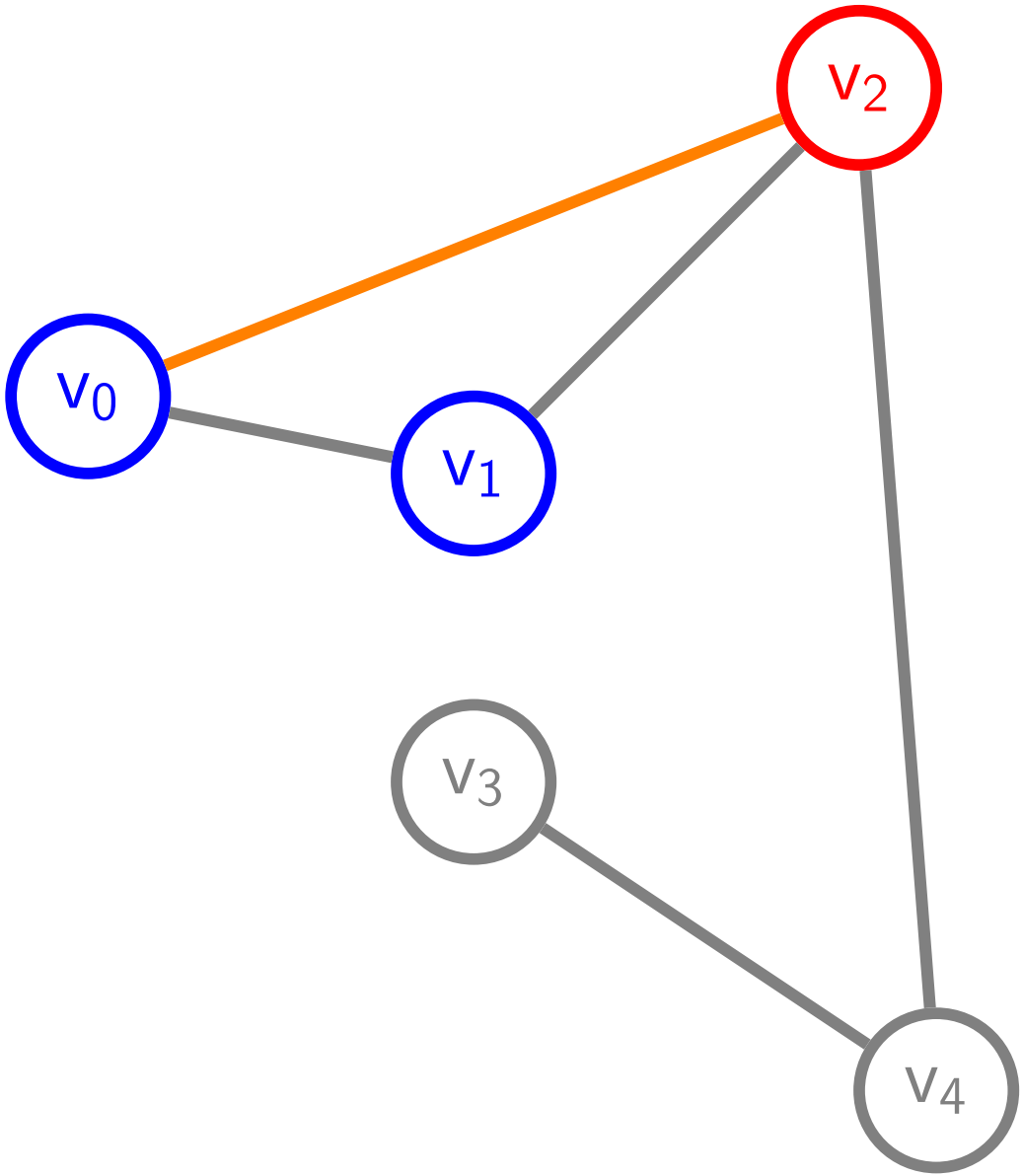
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



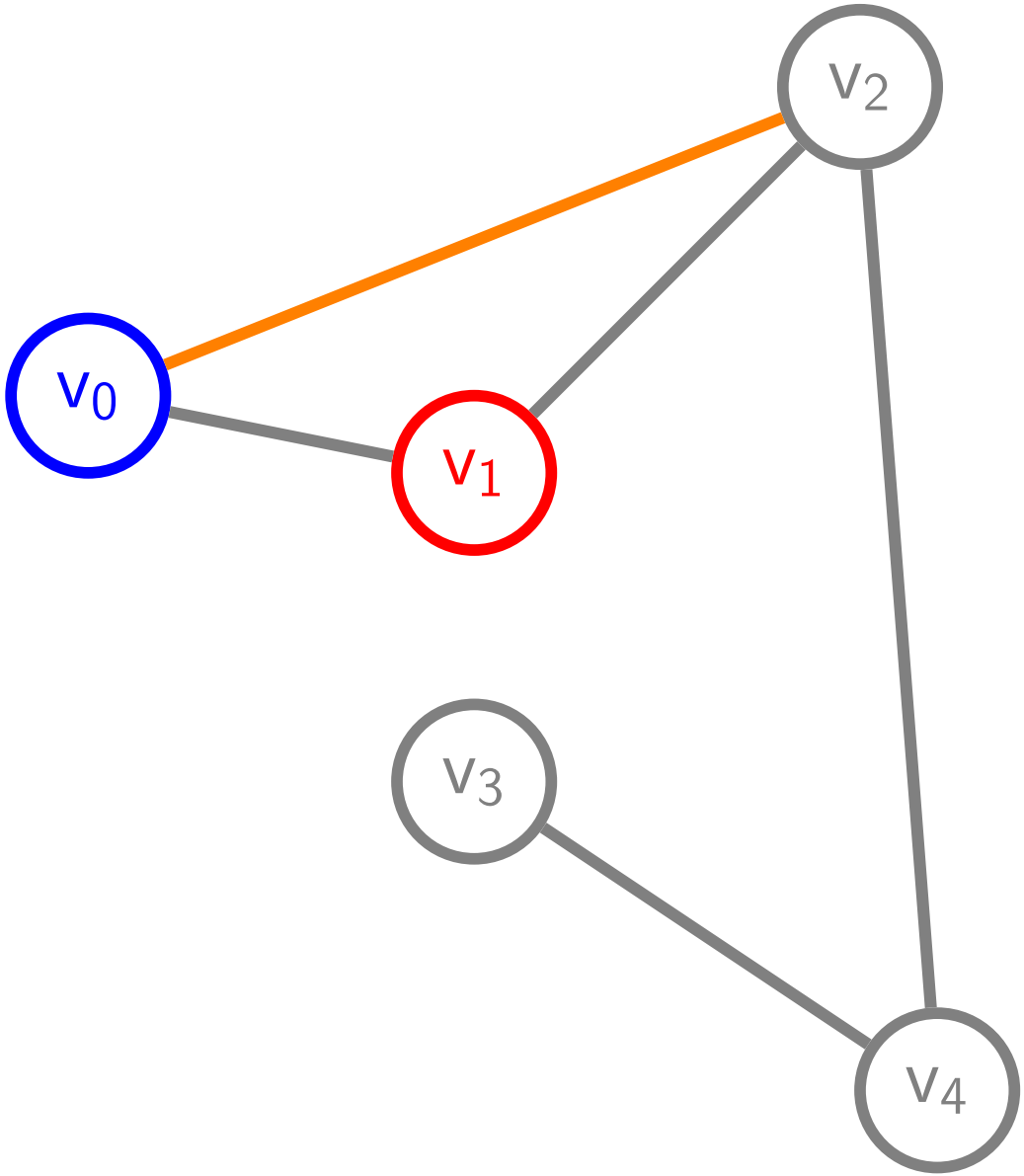
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



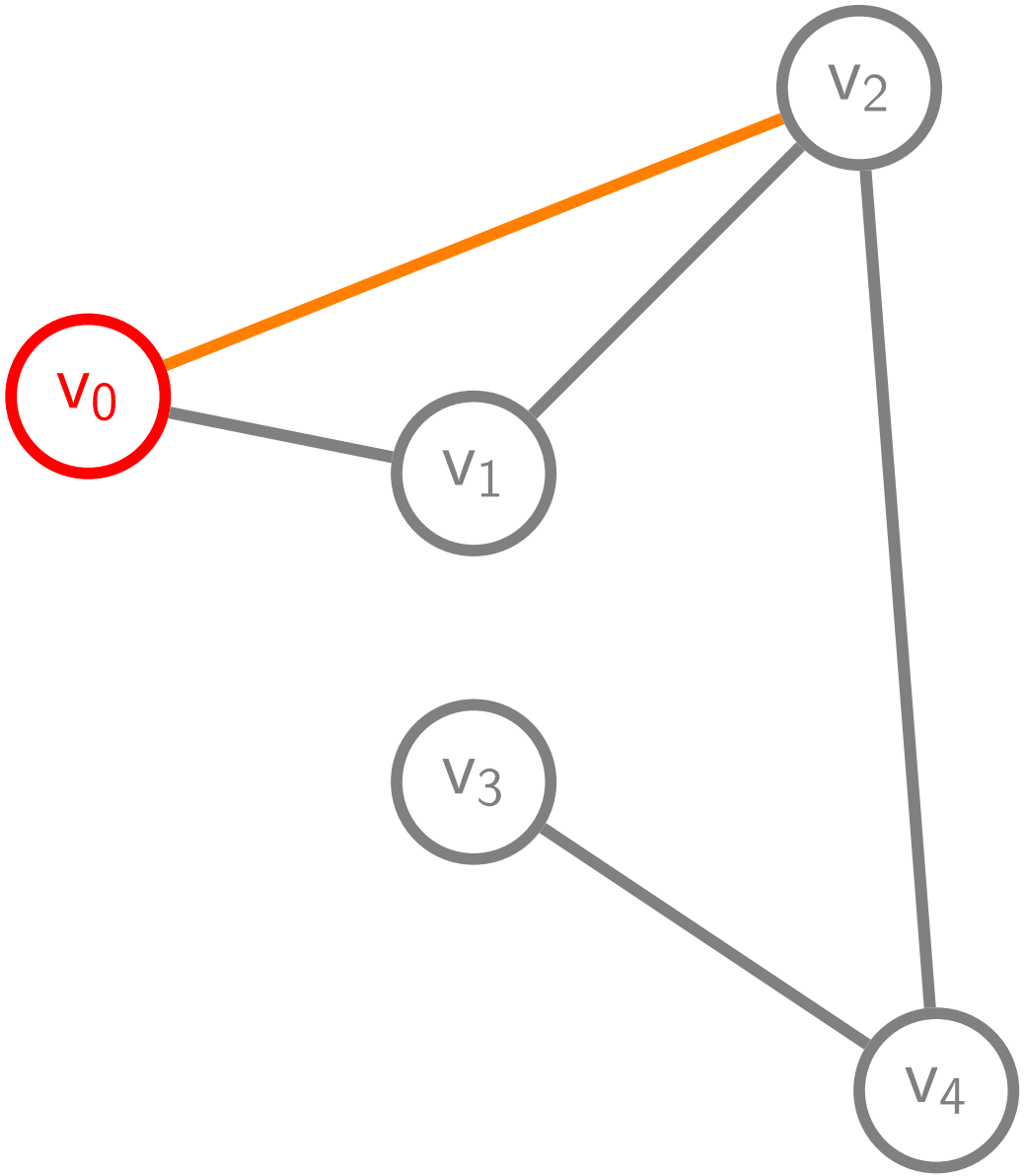
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



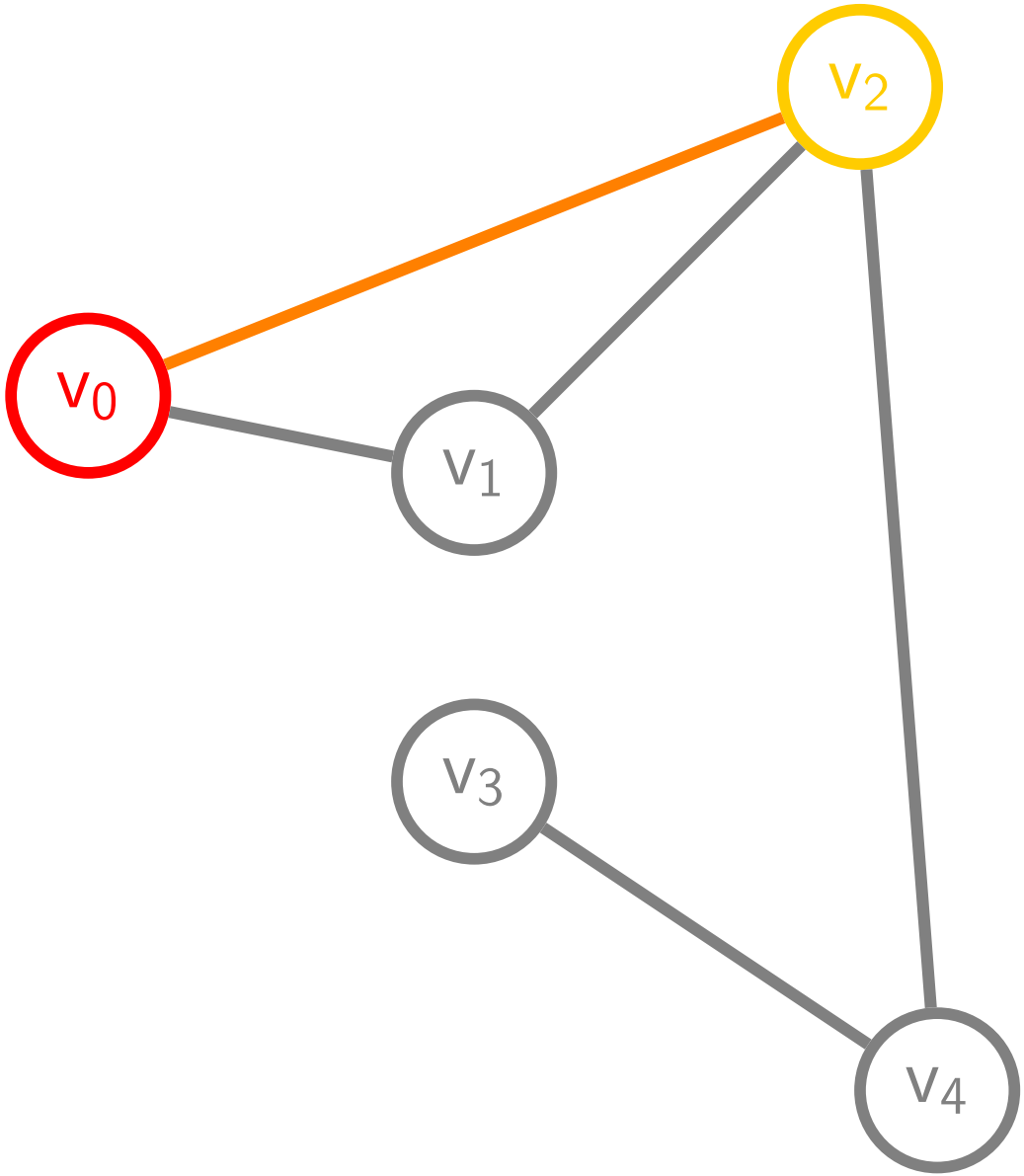
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



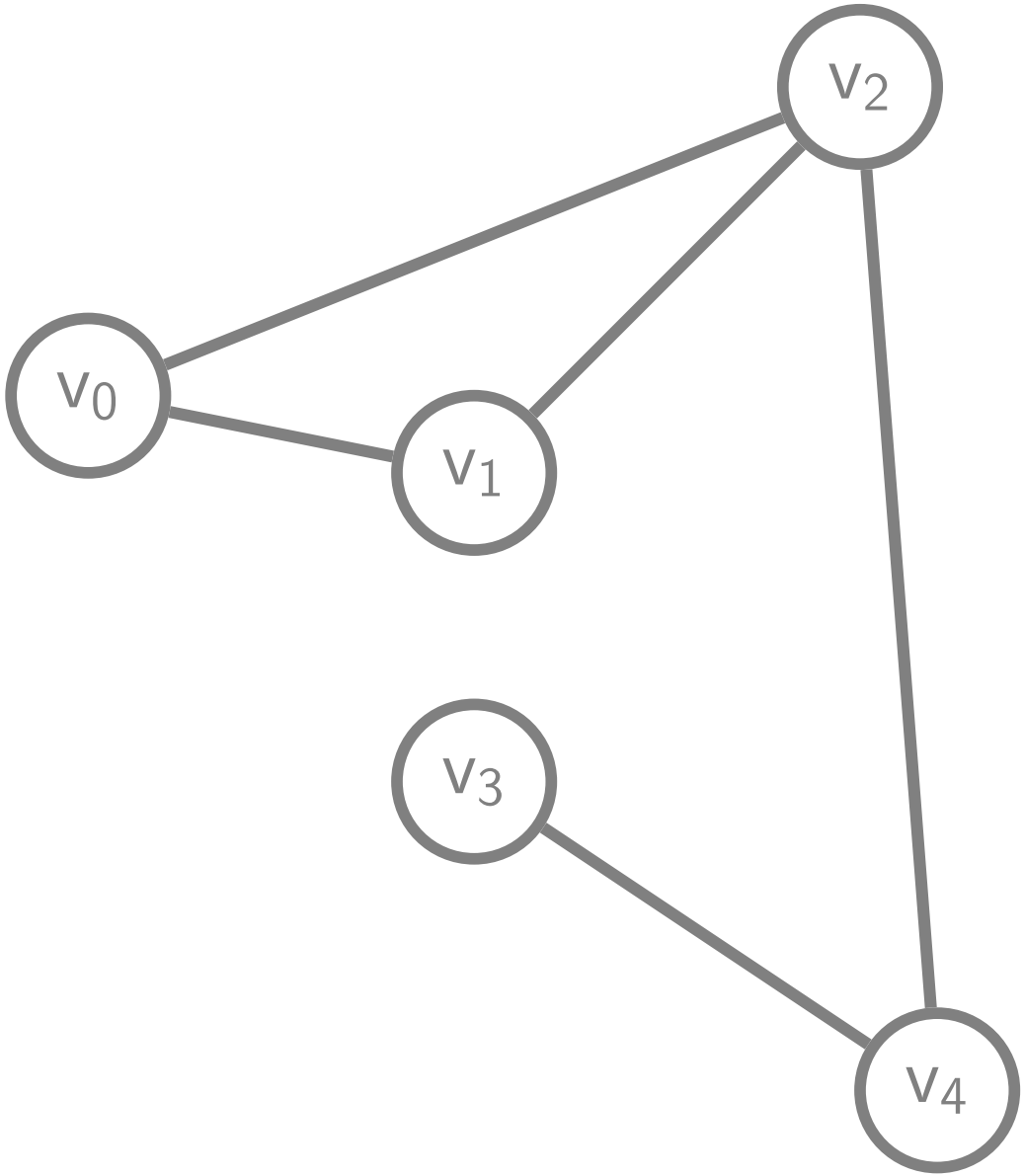
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



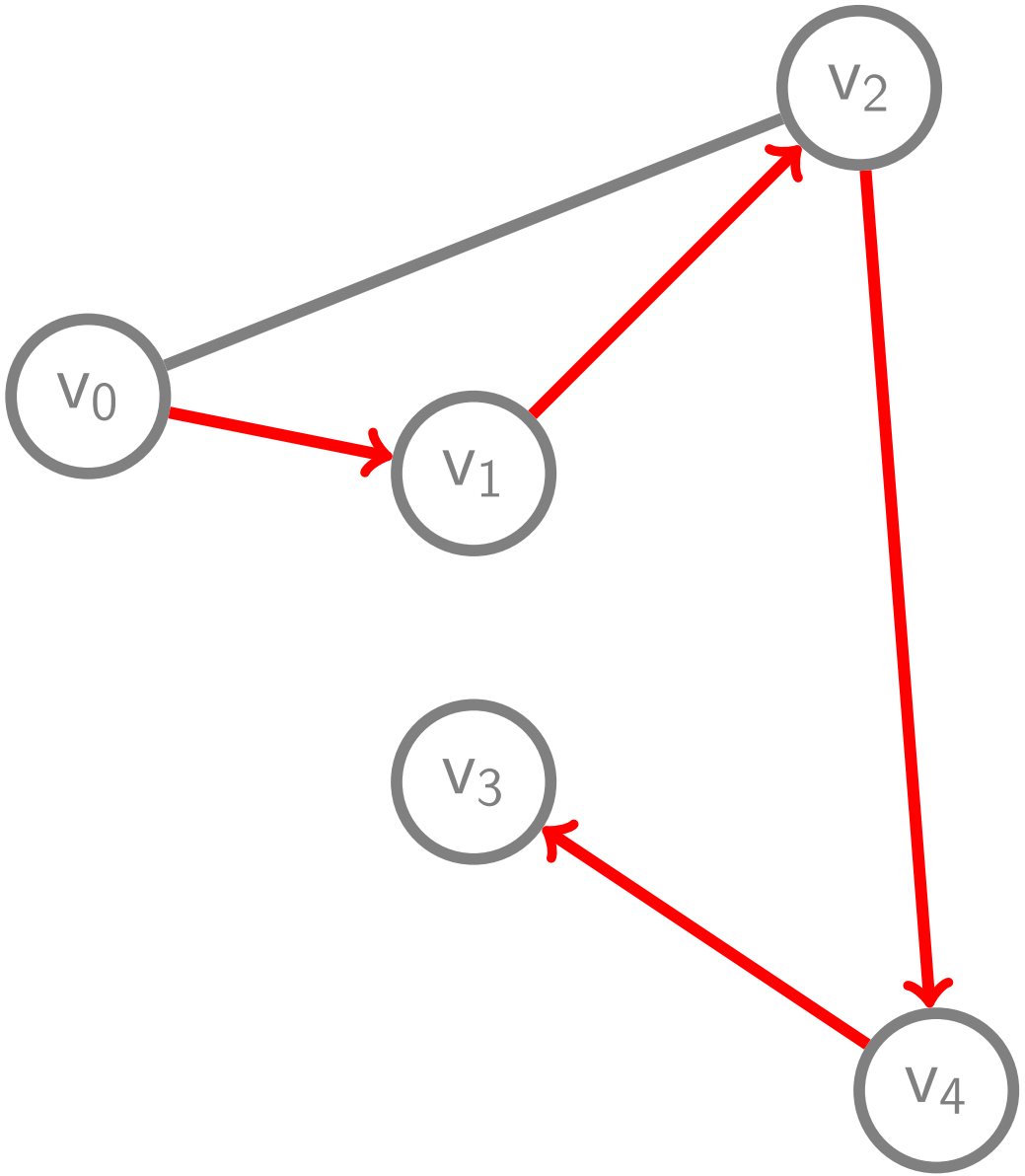
vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

Grafos – Relembrando



vértice	anterior
v0	-
v1	v0
v2	v1
v4	v2
v3	v4

O caminho percorrido durante a busca ao visitar os vértices forma árvores conhecidas como árvores de busca em profundidade.

Grafos – Busca em Profundidade

Observações:

Grafos – Busca em Profundidade

Observações:

Nem sempre é possível, a partir de um único vértice, percorrer o grafo todo usando Busca em Profundidade (não será possível, por exemplo, em um grafo não direcionado que não seja conexo).

Grafos – Busca em Profundidade

Observações:

Nem sempre é possível, a partir de um único vértice, percorrer o grafo todo usando Busca em Profundidade (não será possível, por exemplo, em um grafo não direcionado que não seja conexo).

Pela abordagem usada na Busca em Profundidade, é comum implementá-la usando recursão.

Grafos - Busca em Profundidade

Pseudocódigo:

```
Função DFS(G, v)
  marque v como visitado
  PARA TODAS as arestas de v para w
    SE w ainda não foi visitado
      DFS(G, w)
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos **Não Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){




}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências: Grafos ou Digrafos **Não Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);




}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos **Não Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;




}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências: Grafos ou Digrafos **Não Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  int x;
  for (x=0; x<g->numVertices; x++)


}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências: Grafos ou Digrafos **Não Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  int x;
  for (x=0; x<g->numVertices; x++)
    if (g->matriz[atual][x] && !visitado[x])

}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências: Grafos ou Digrafos **Não Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  int x;
  for (x=0; x<g->numVertices; x++)
    if (g->matriz[atual][x] && !visitado[x])
      visitaProfundidade(g, x, visitado, atual);
}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências: Grafos ou Digrafos **Não Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  int x;
  for (x=0; x<g->numVertices; x++)
    if (g->matriz[atual][x] && !visitado[x])
      visitaProfundidade(g, x, visitado, atual);
}
```

$$O(V^2)$$

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências: Grafos ou Digrafos **Ponderados**

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  int x;
  for (x=0; x<g->numVertices; x++)
    if (g->matriz[atual][x]!=ARESTA_INVALIDA &&
                                    !visitado[x])
      visitaProfundidade(g, x, visitado, atual);
}
```

$O(V^2)$

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){



}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos Não Ponderados ou Ponderados

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









}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;




}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;
  for (x=0;x<g->numVertices;x++)



}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;
  for (x=0;x<g->numVertices;x++)
    if (!visitado[x])



}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;
  for (x=0;x<g->numVertices;x++)
    if (!visitado[x])
      visitaProfundidade(g, x, visitado, -1);

}
```

Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;
  for (x=0;x<g->numVertices;x++)
    if (!visitado[x])
      visitaProfundidade(g, x, visitado, -1);
  free(visitado);
}
```

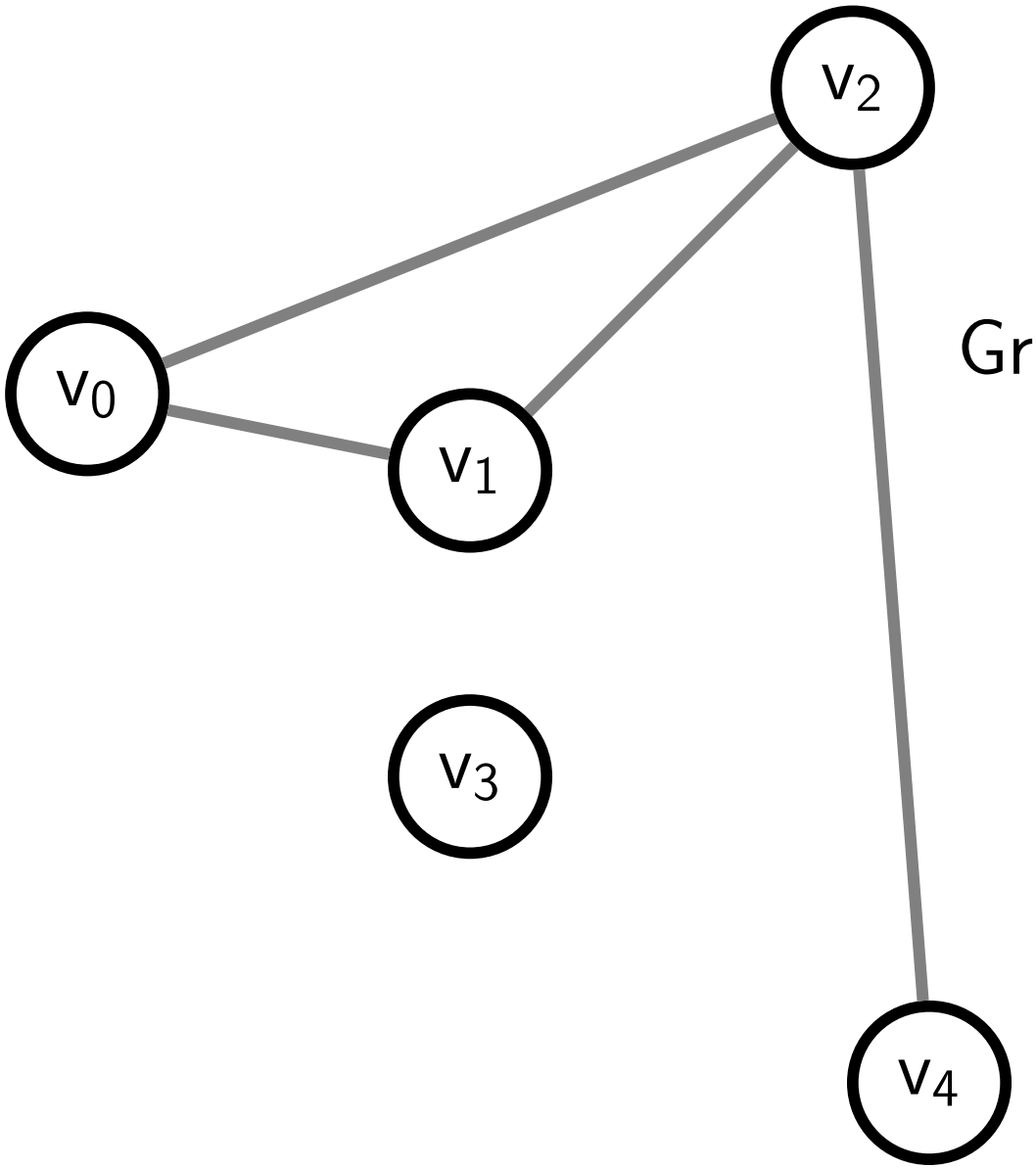
Grafos - Busca em Profundidade

Representação por Matrizes de Adjacências: Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;
  for (x=0;x<g->numVertices;x++)
    if (!visitado[x])
      visitaProfundidade(g, x, visitado, -1);
  free(visitado);
}
```

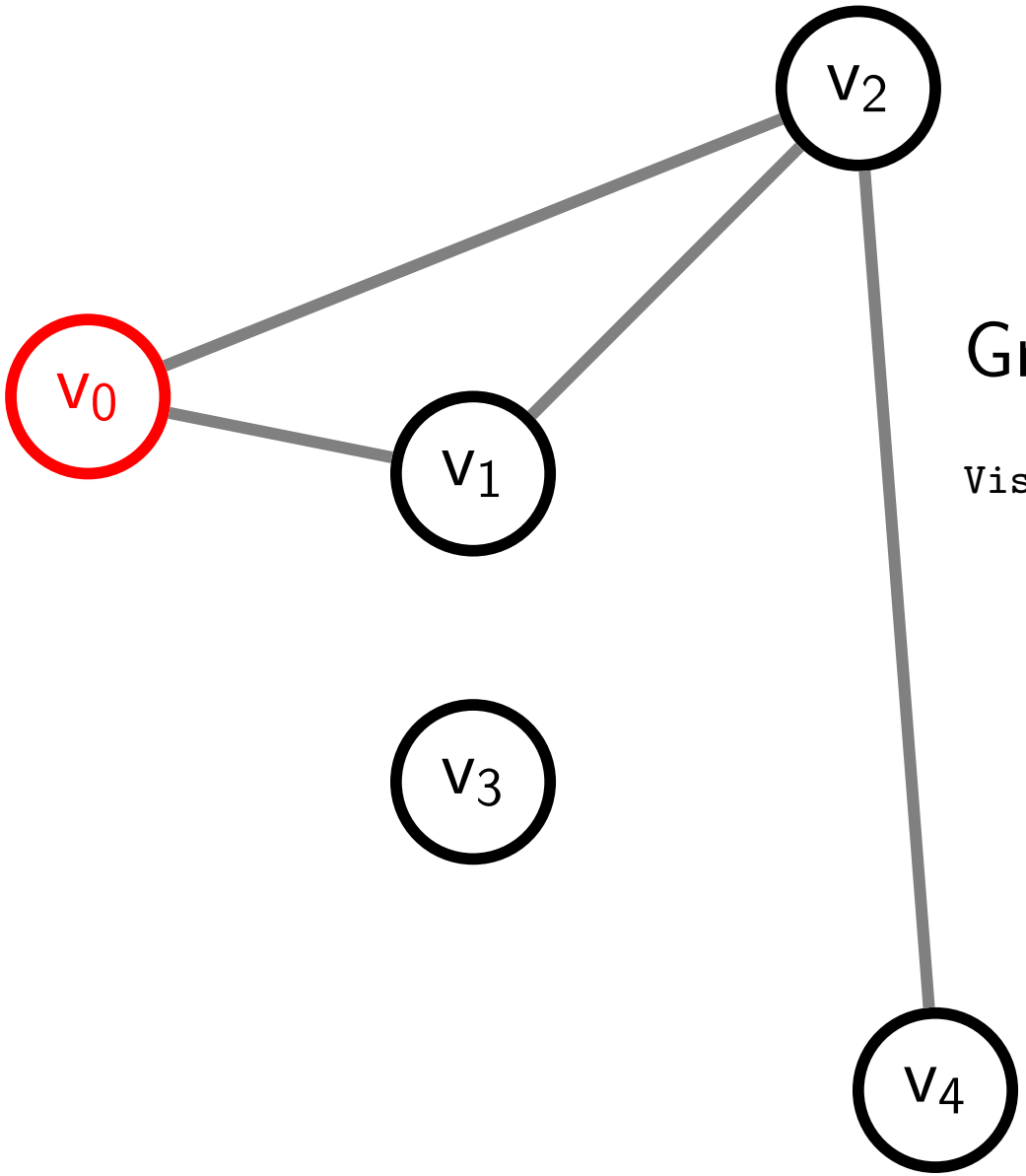
$O(V^2)$

Grafos – Busca em Profundidade



Grafo não direcionado

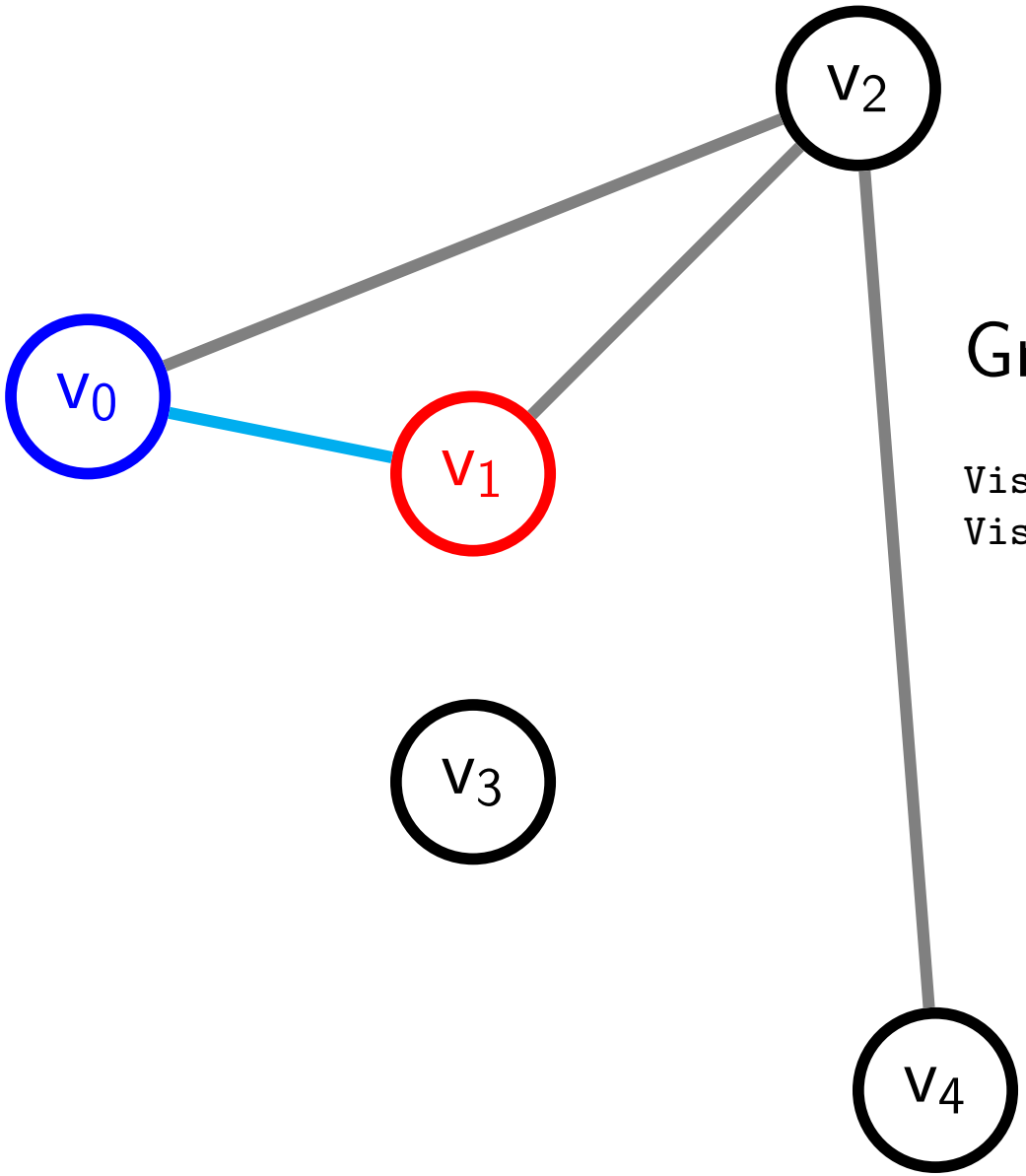
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
```

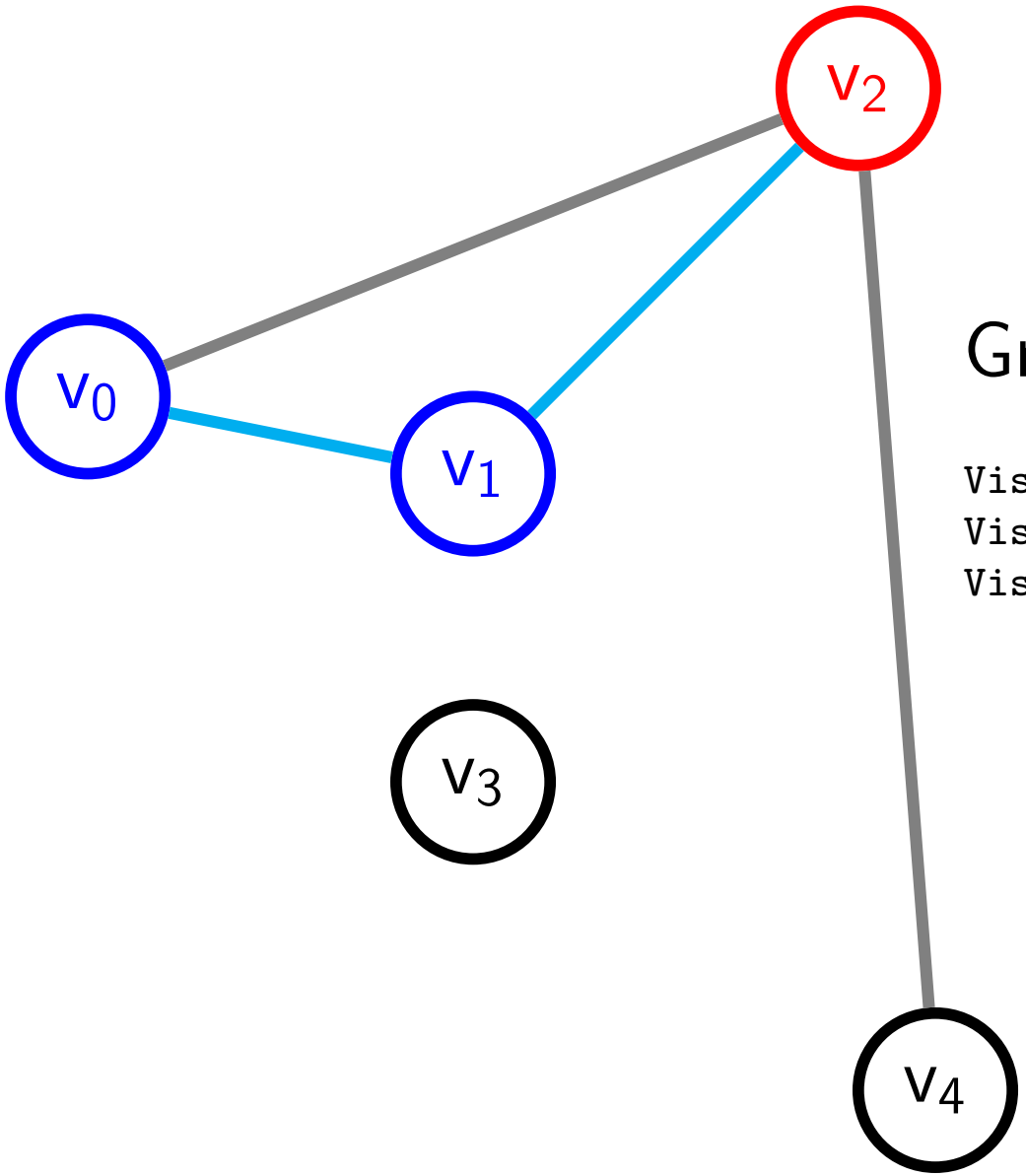
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
```

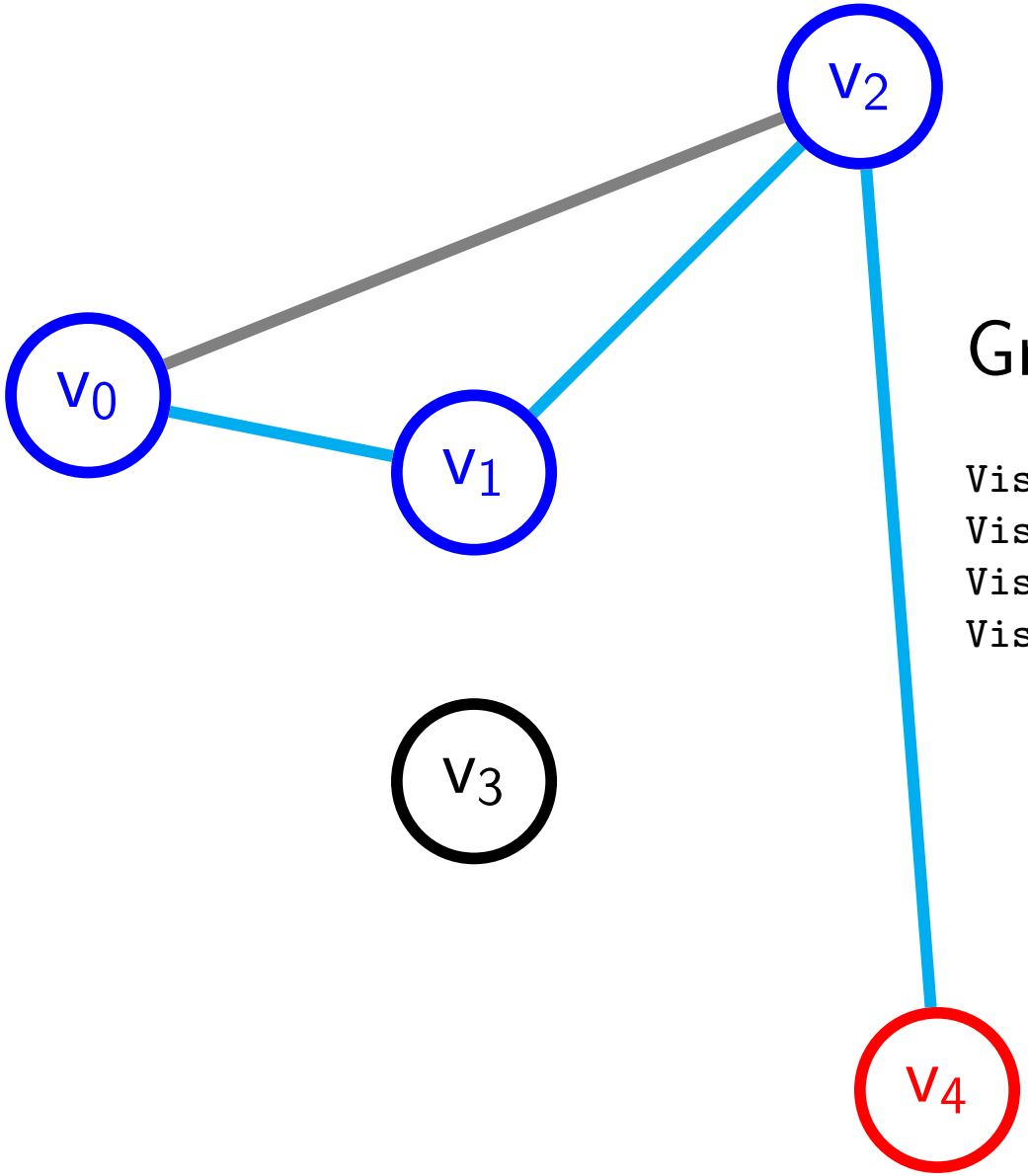
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
```

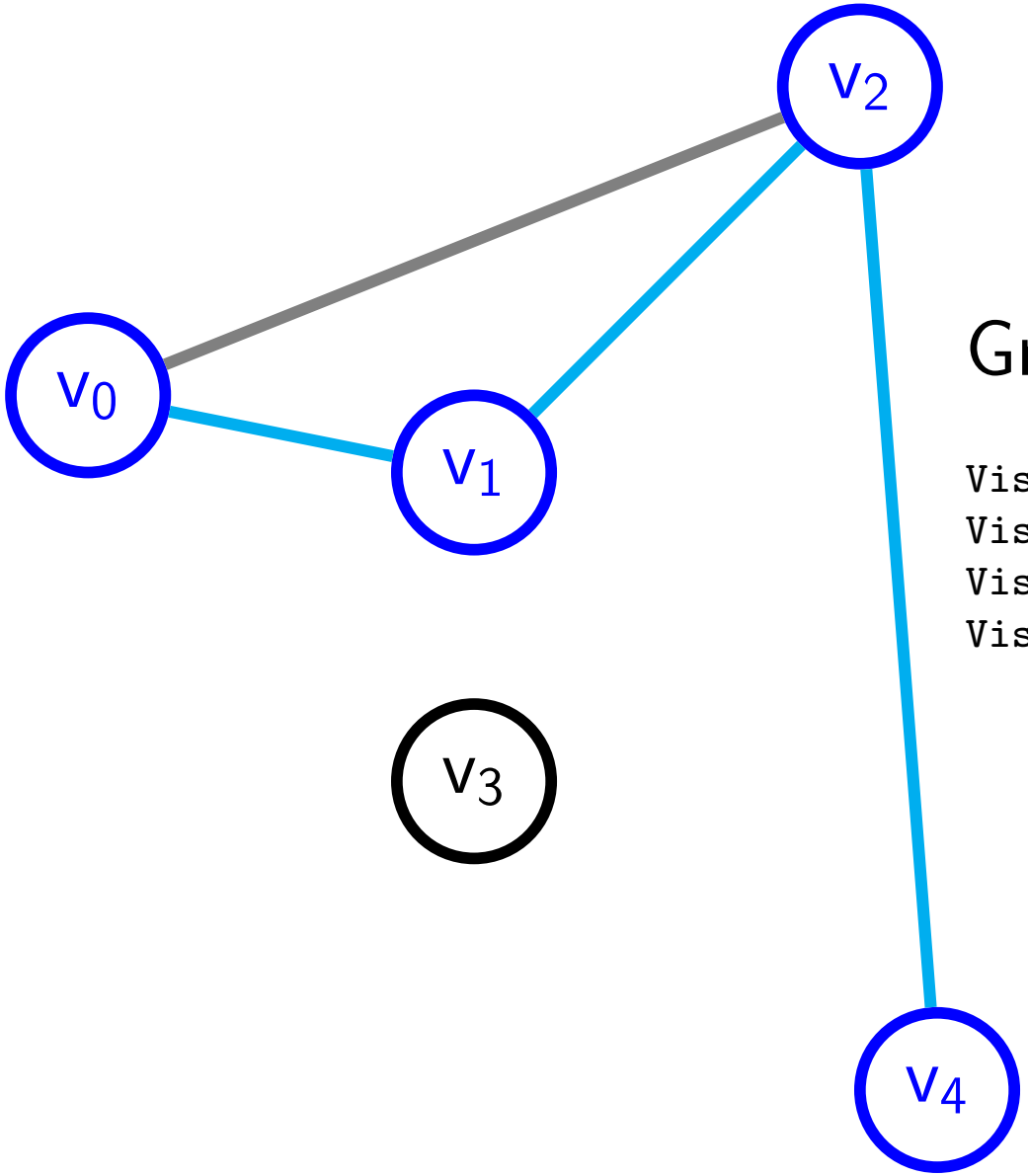
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

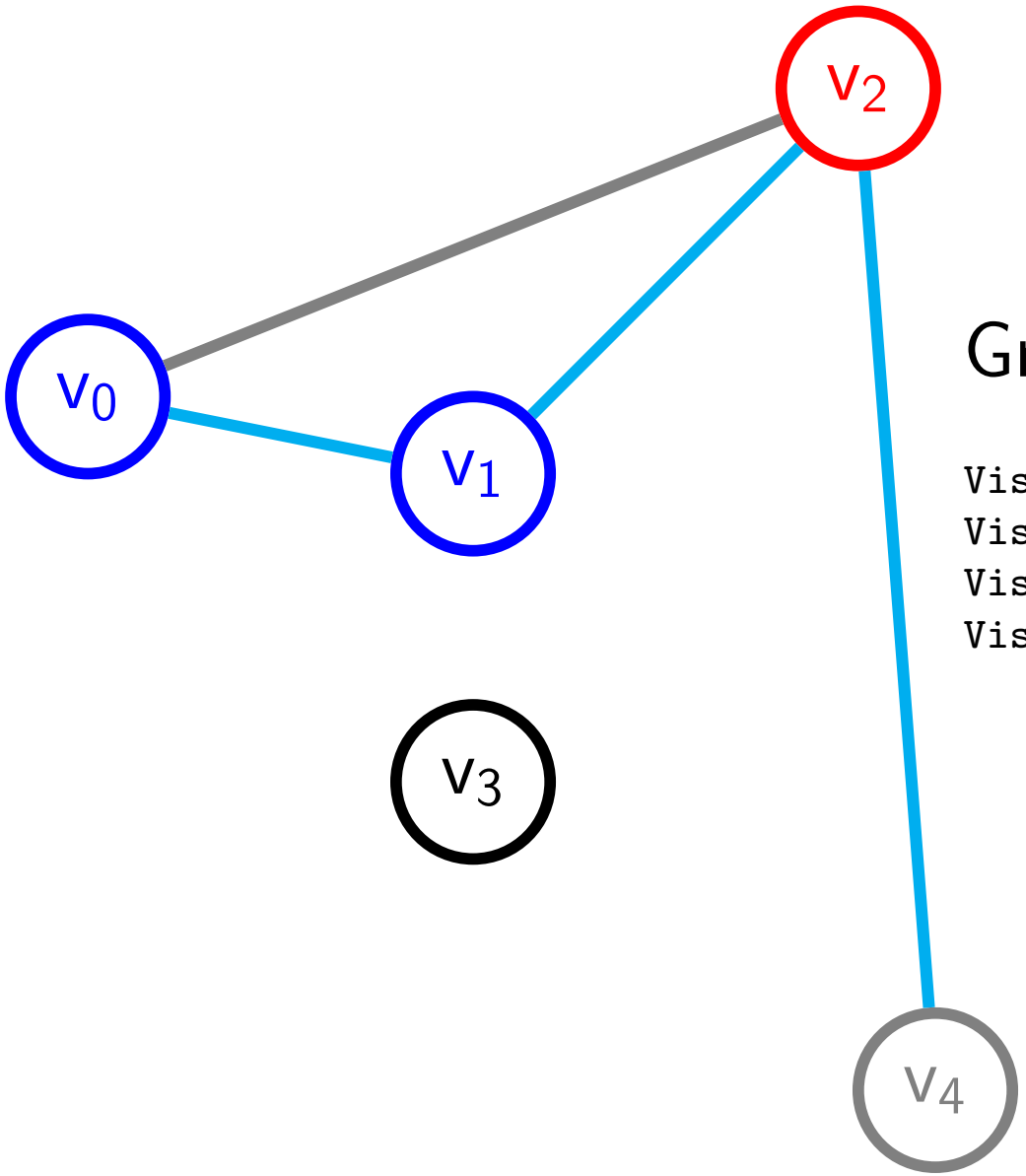
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

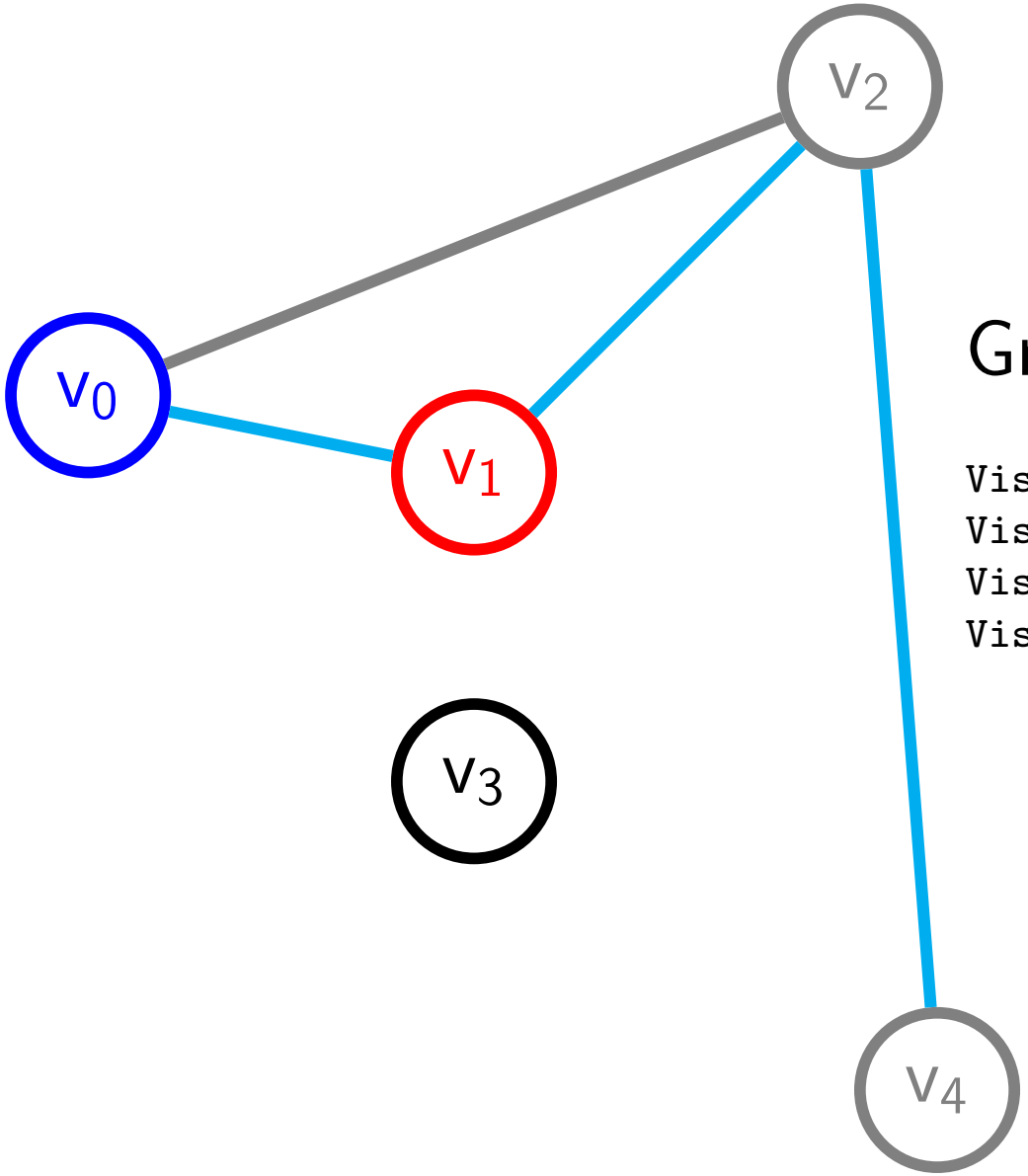
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

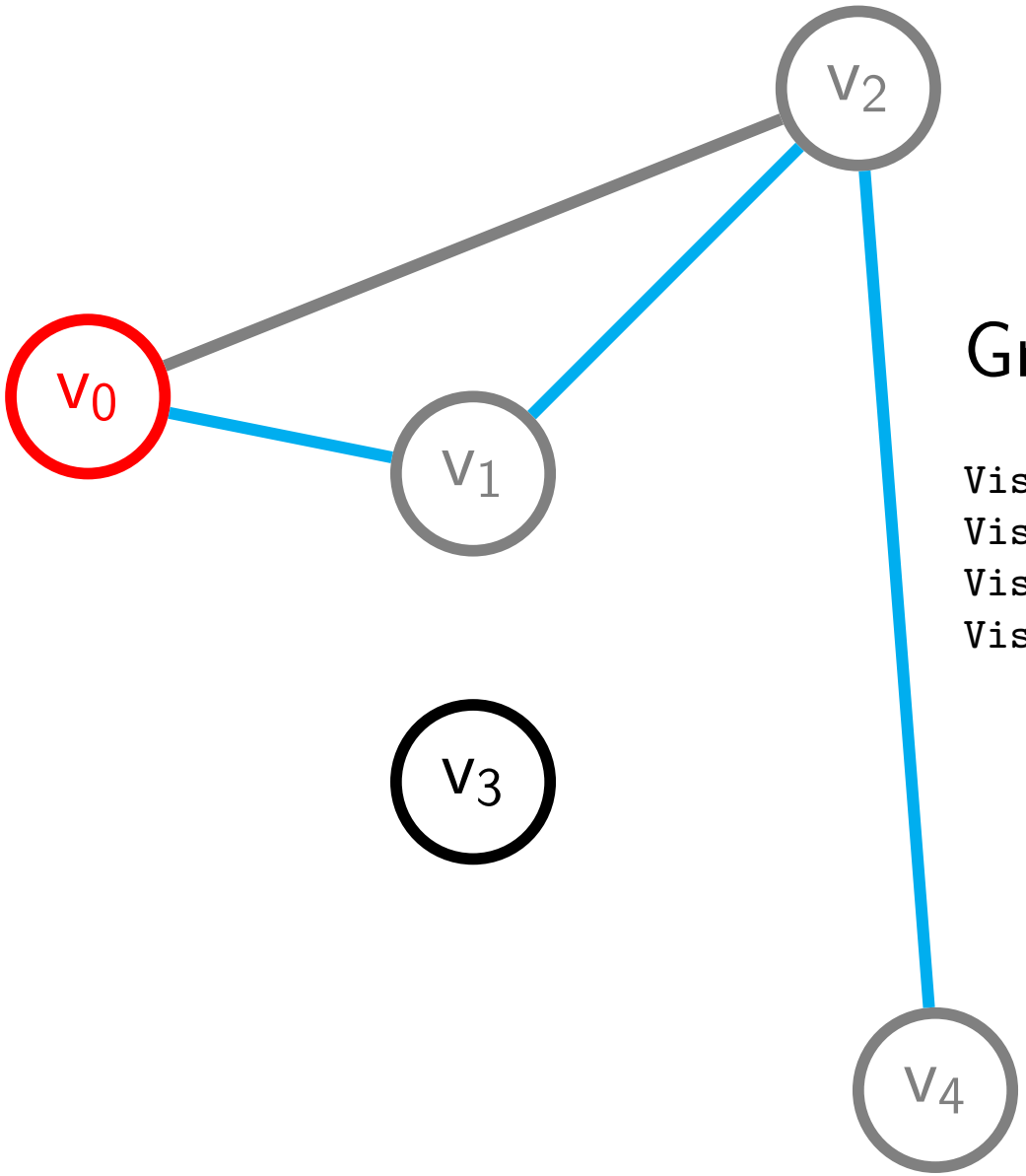
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

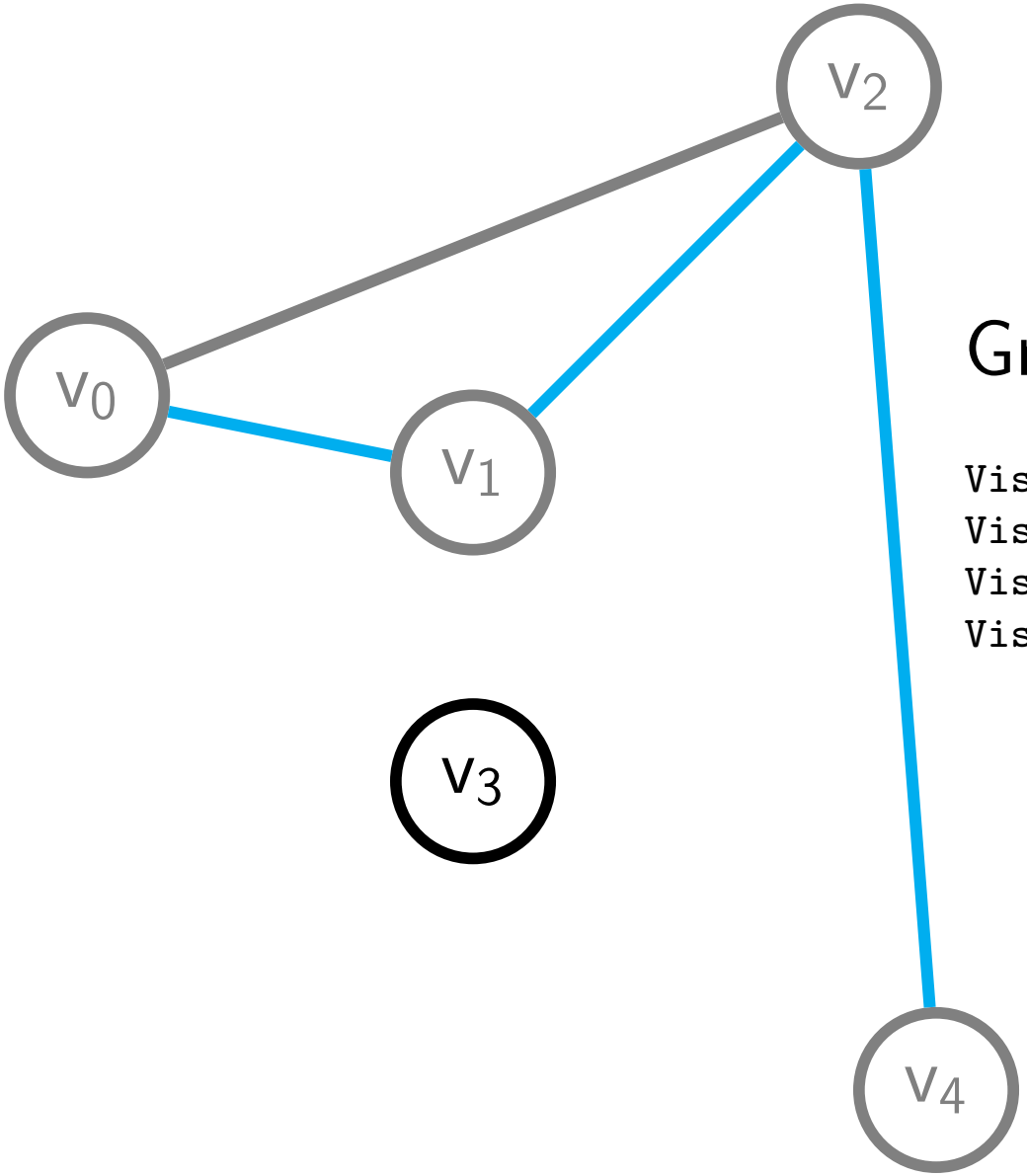
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

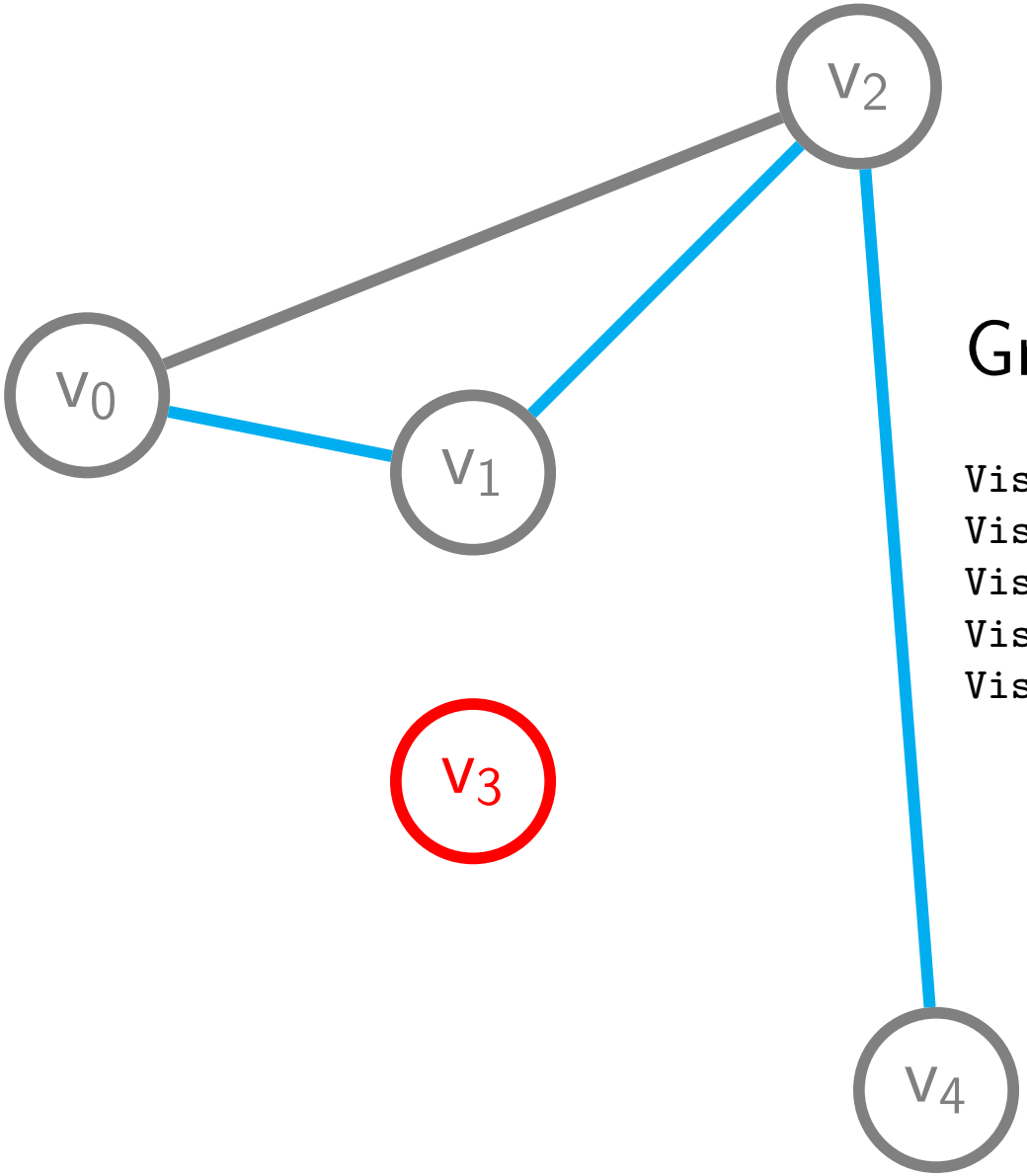
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

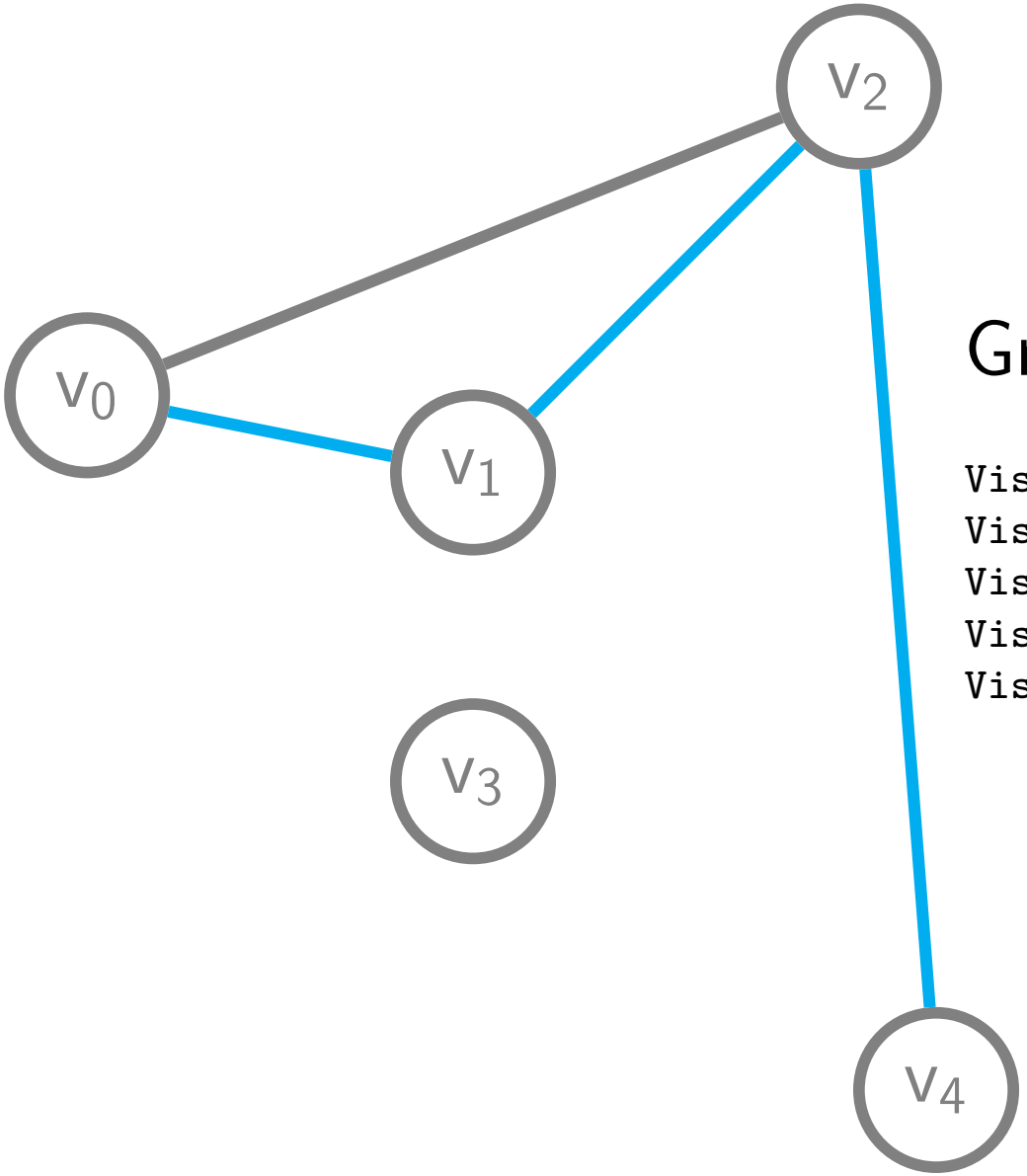
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
Visitando vertice:   3 (anterior:  -1)
```

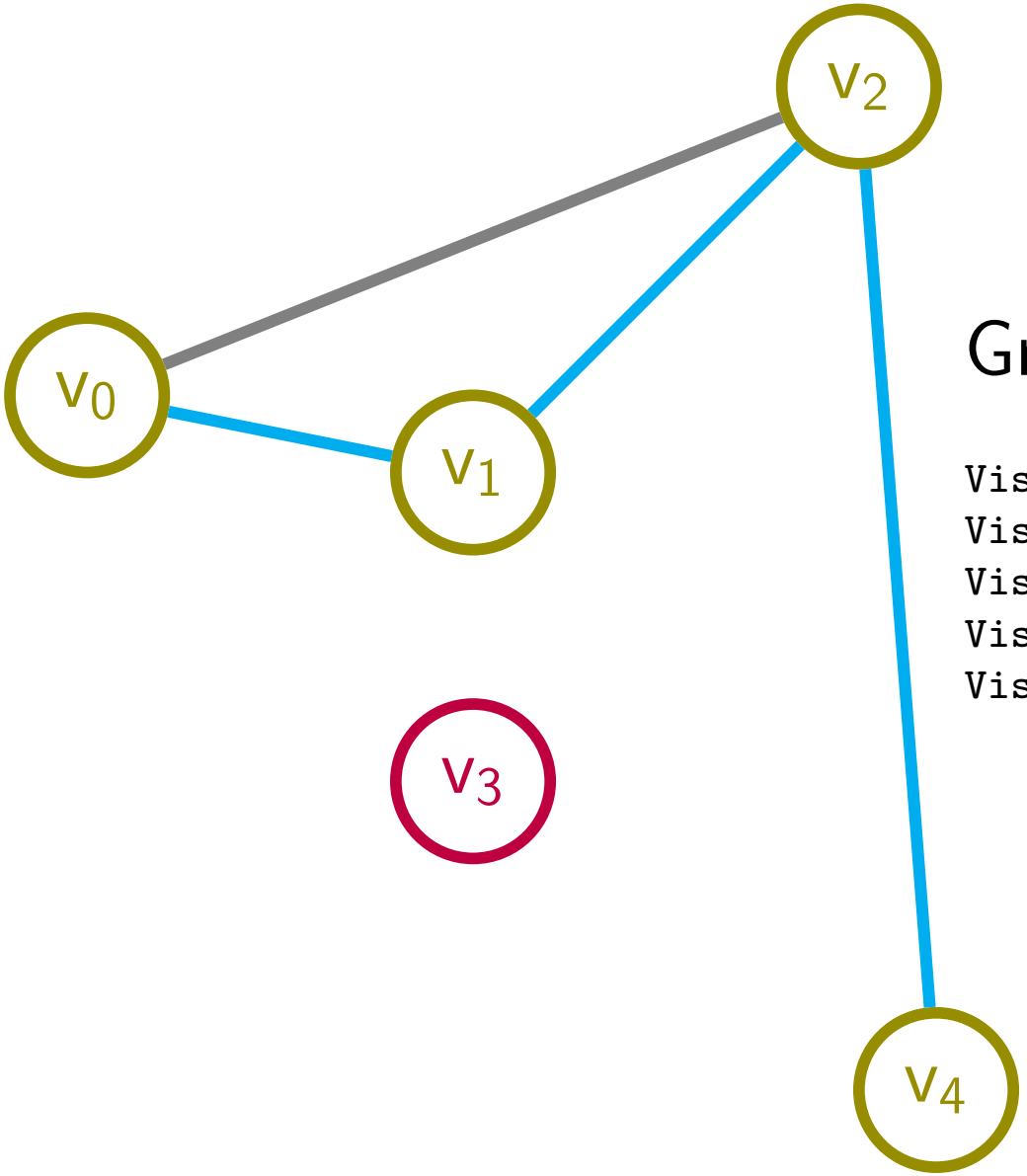
Grafos – Busca em Profundidade



Grafo não direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
Visitando vertice:   3 (anterior:  -1)
```

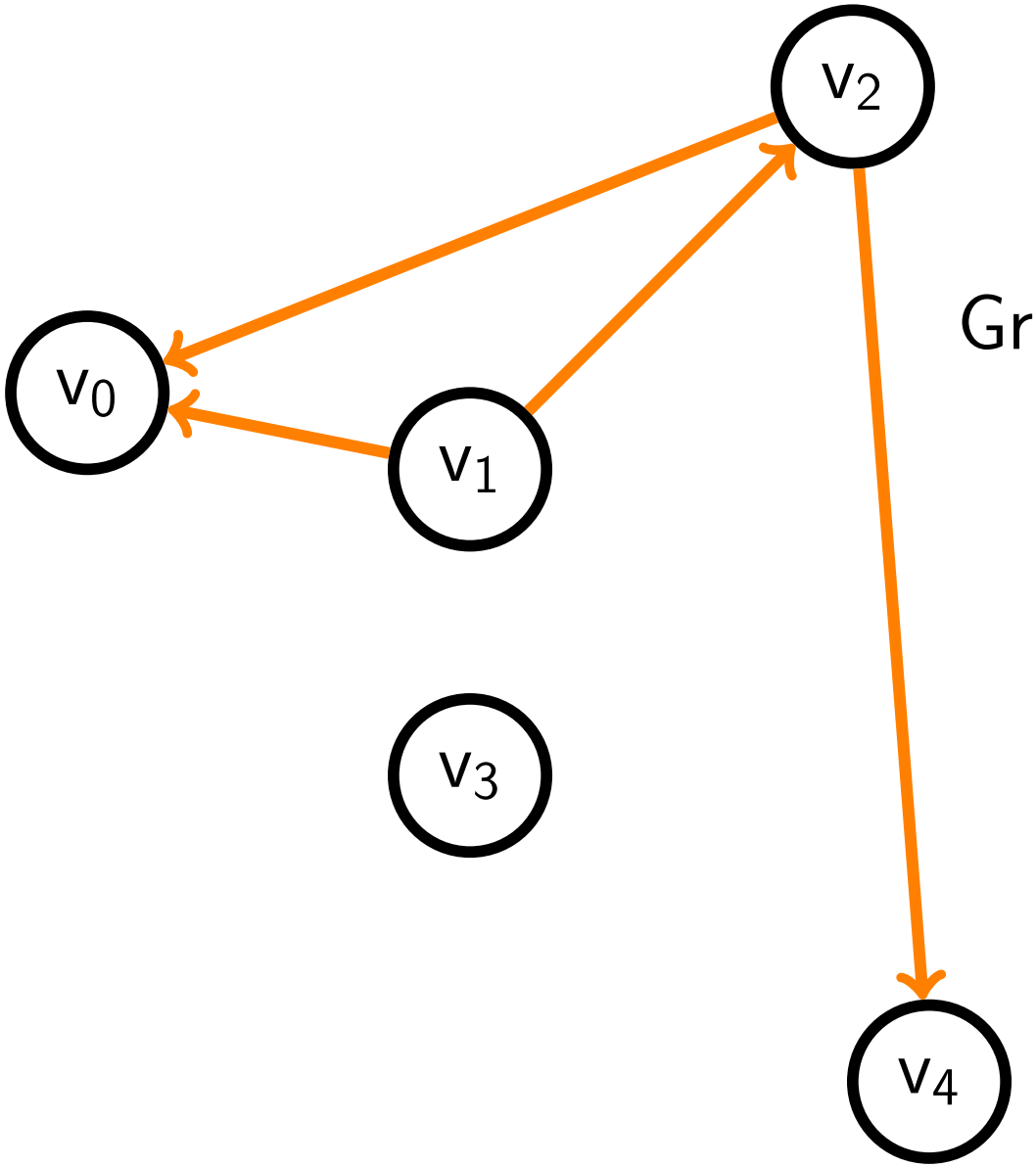
Grafos – Busca em Profundidade



Grafo não direcionado

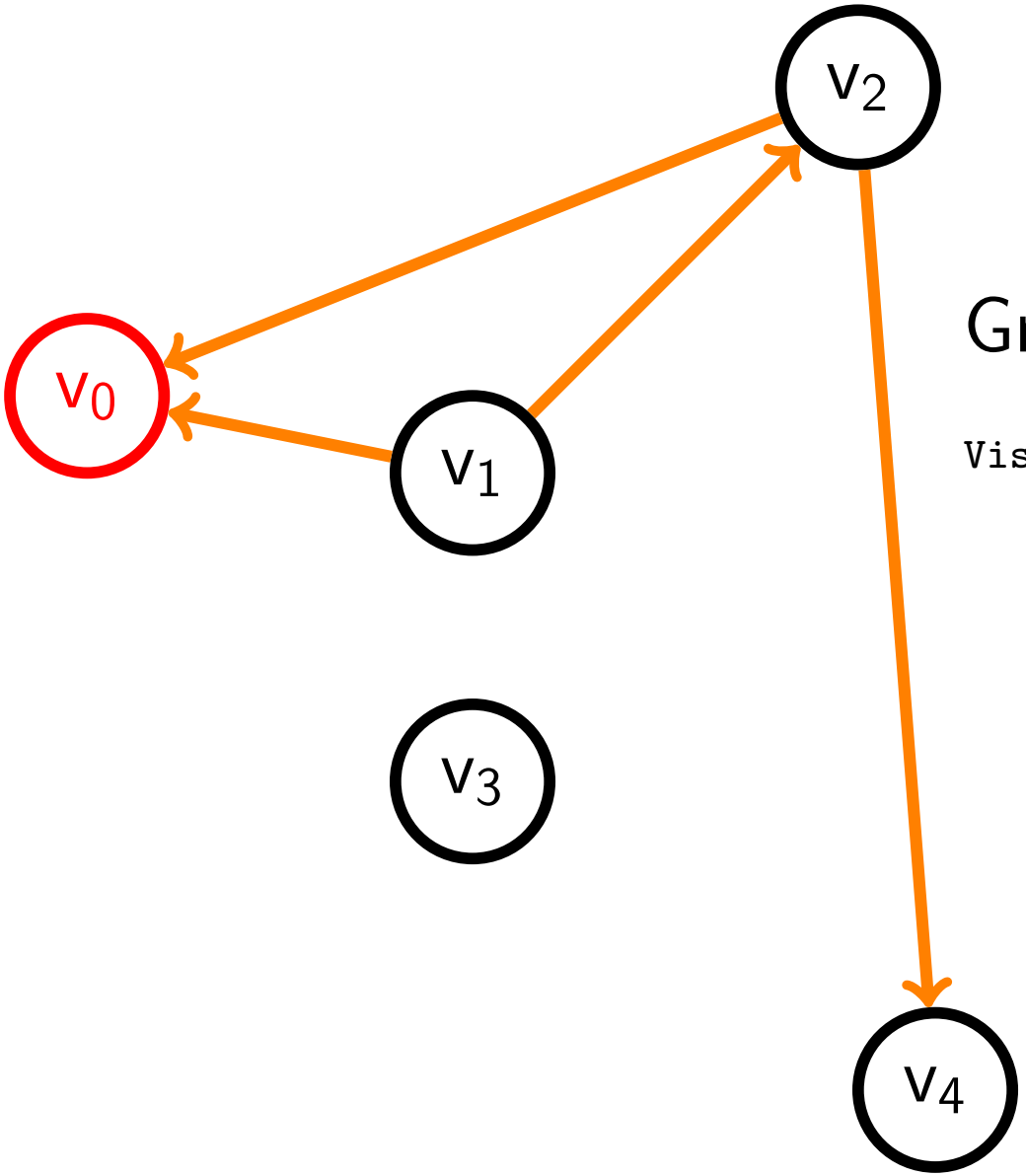
```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:   0)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
Visitando vertice:   3 (anterior:  -1)
```

Digrafos – Busca em Profundidade



Grafo direcionado

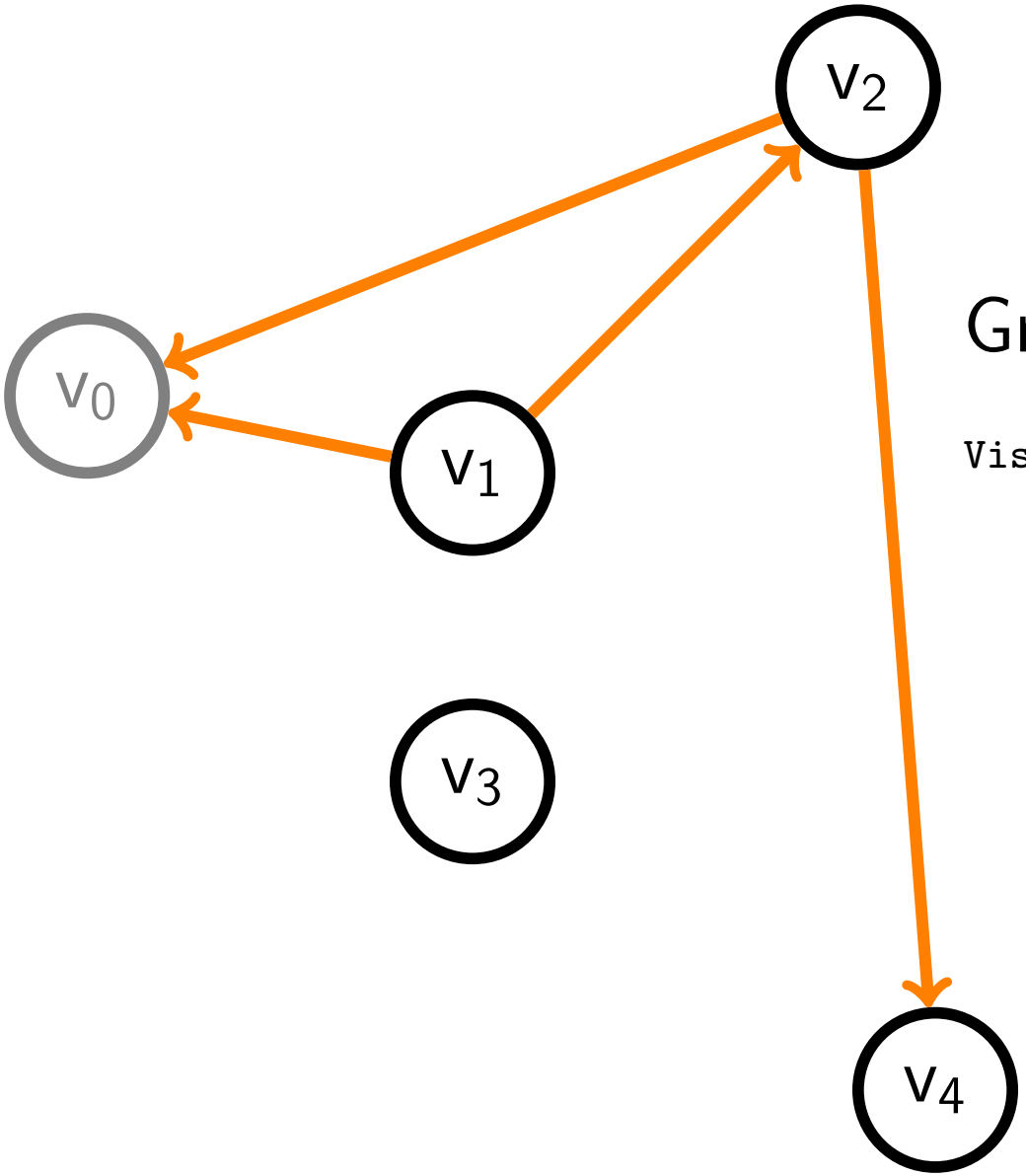
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
```

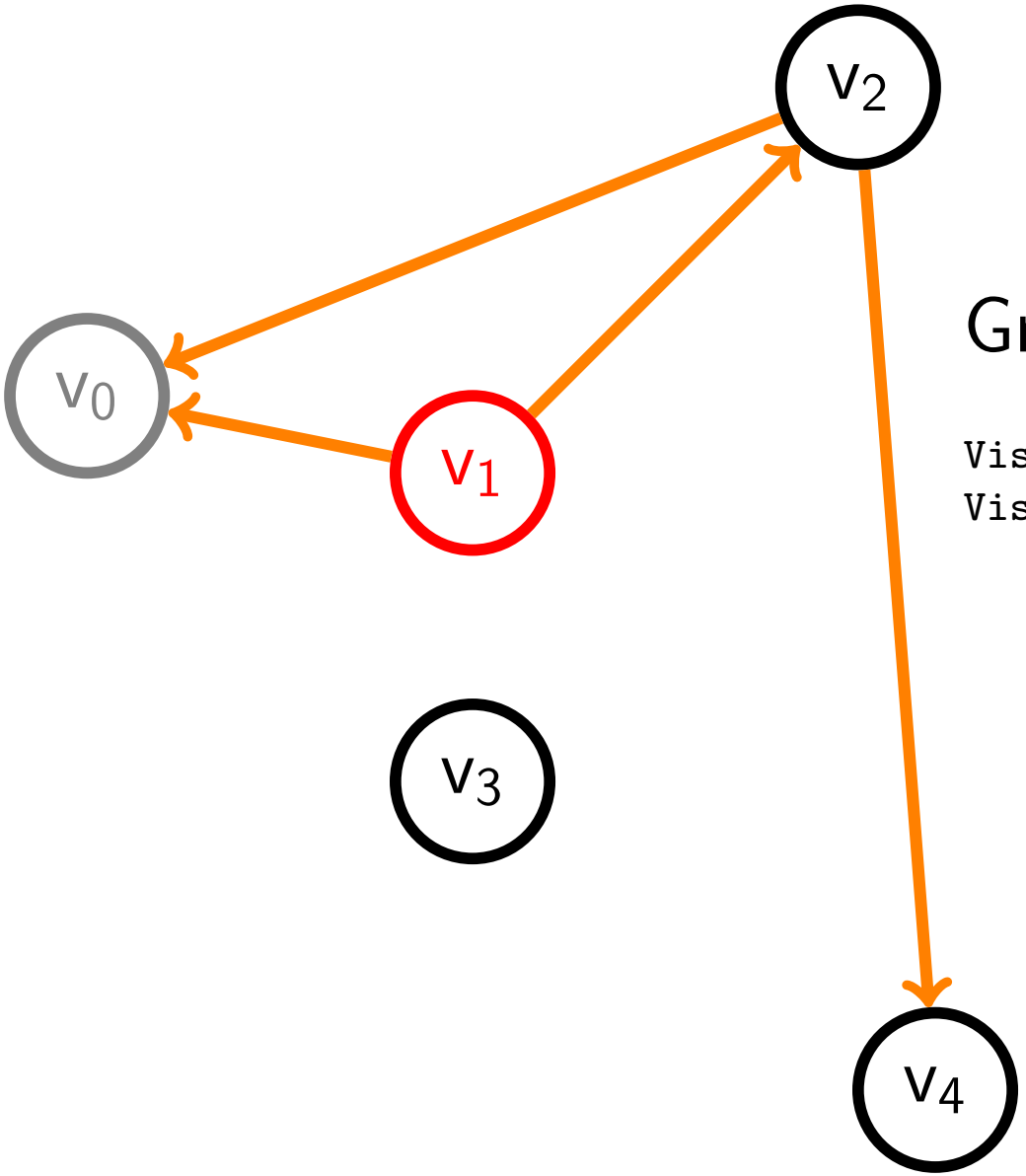
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
```

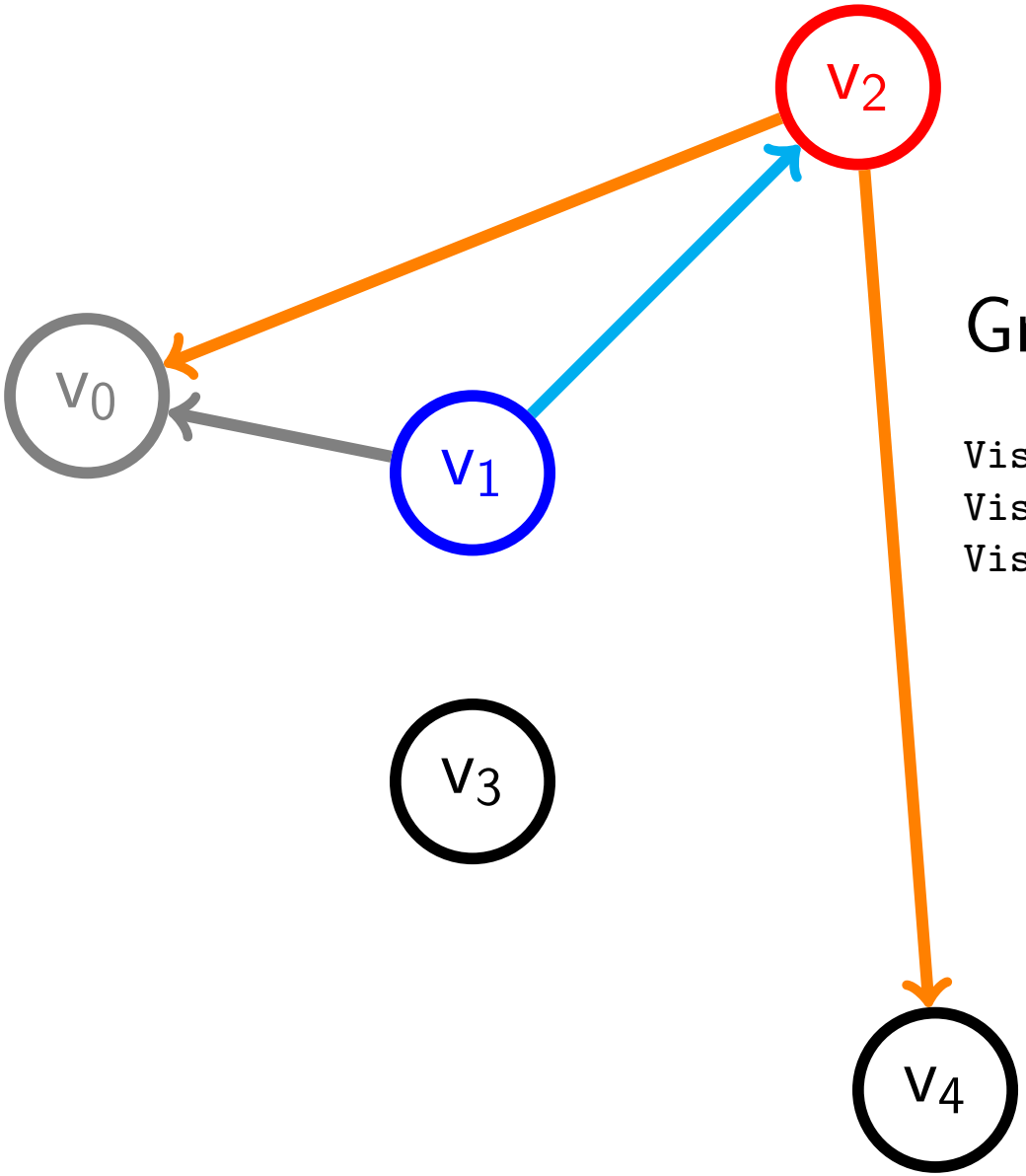
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:    0 (anterior:   -1)
Visitando vertice:    1 (anterior:   -1)
```

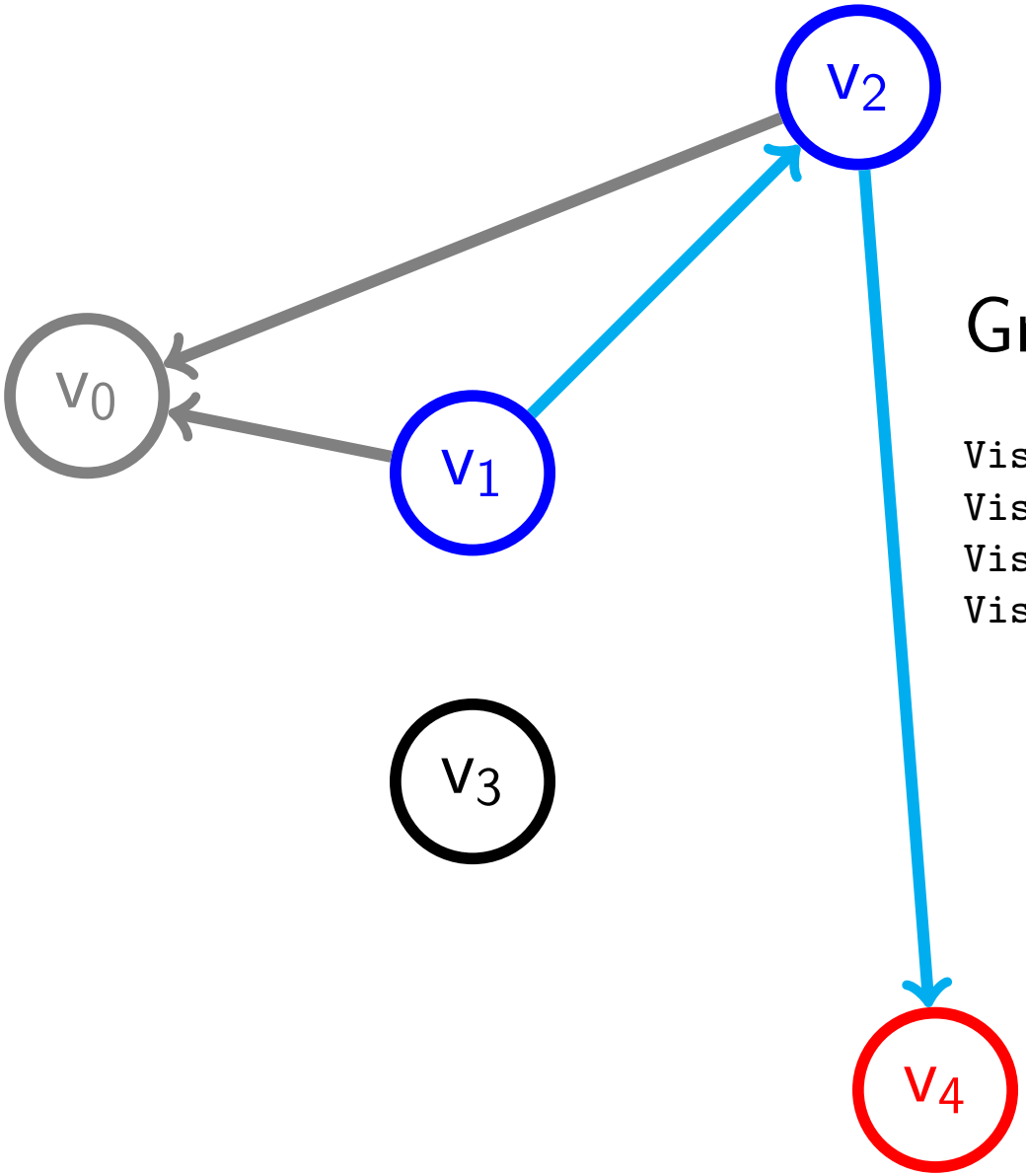
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
```

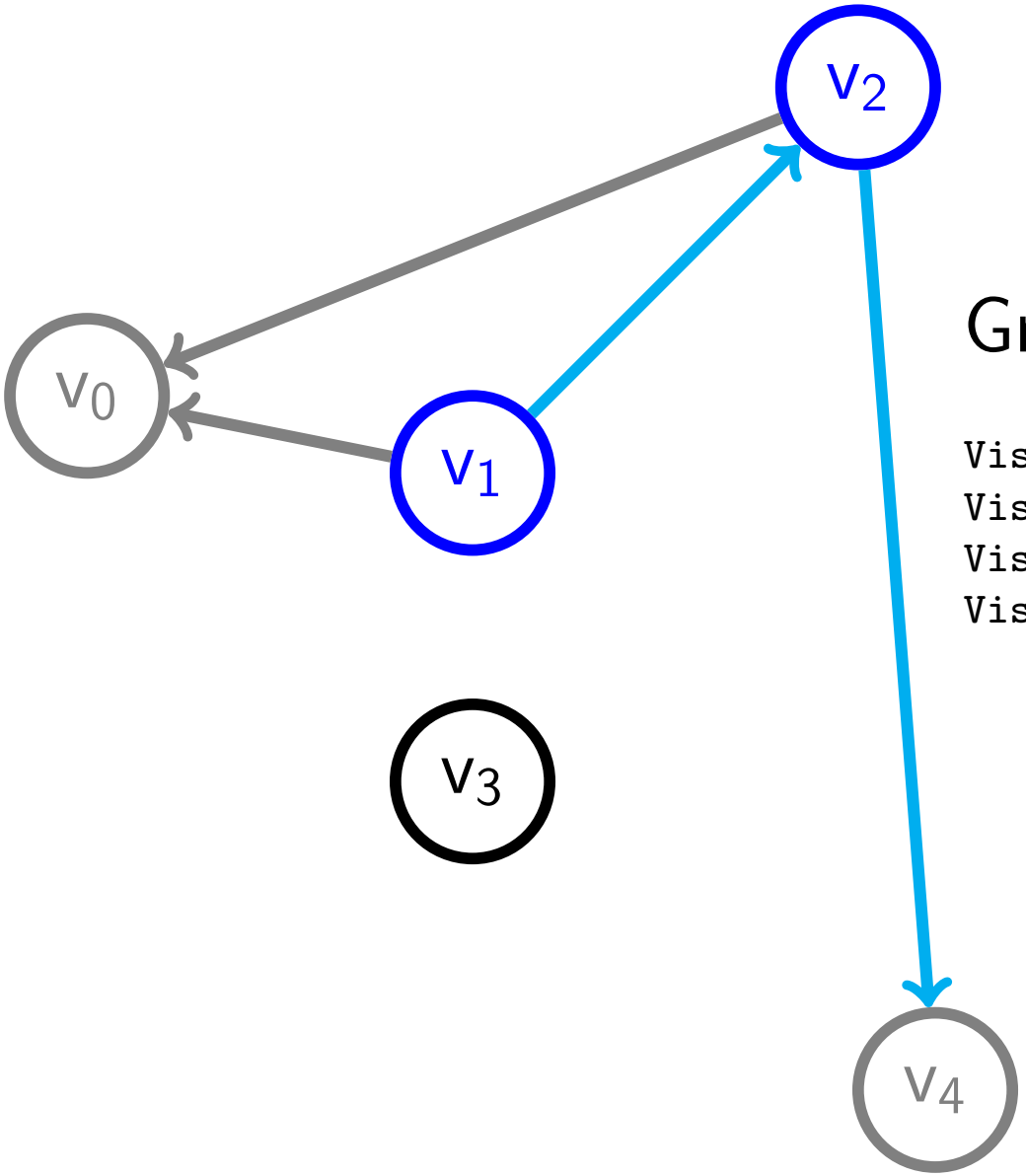
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

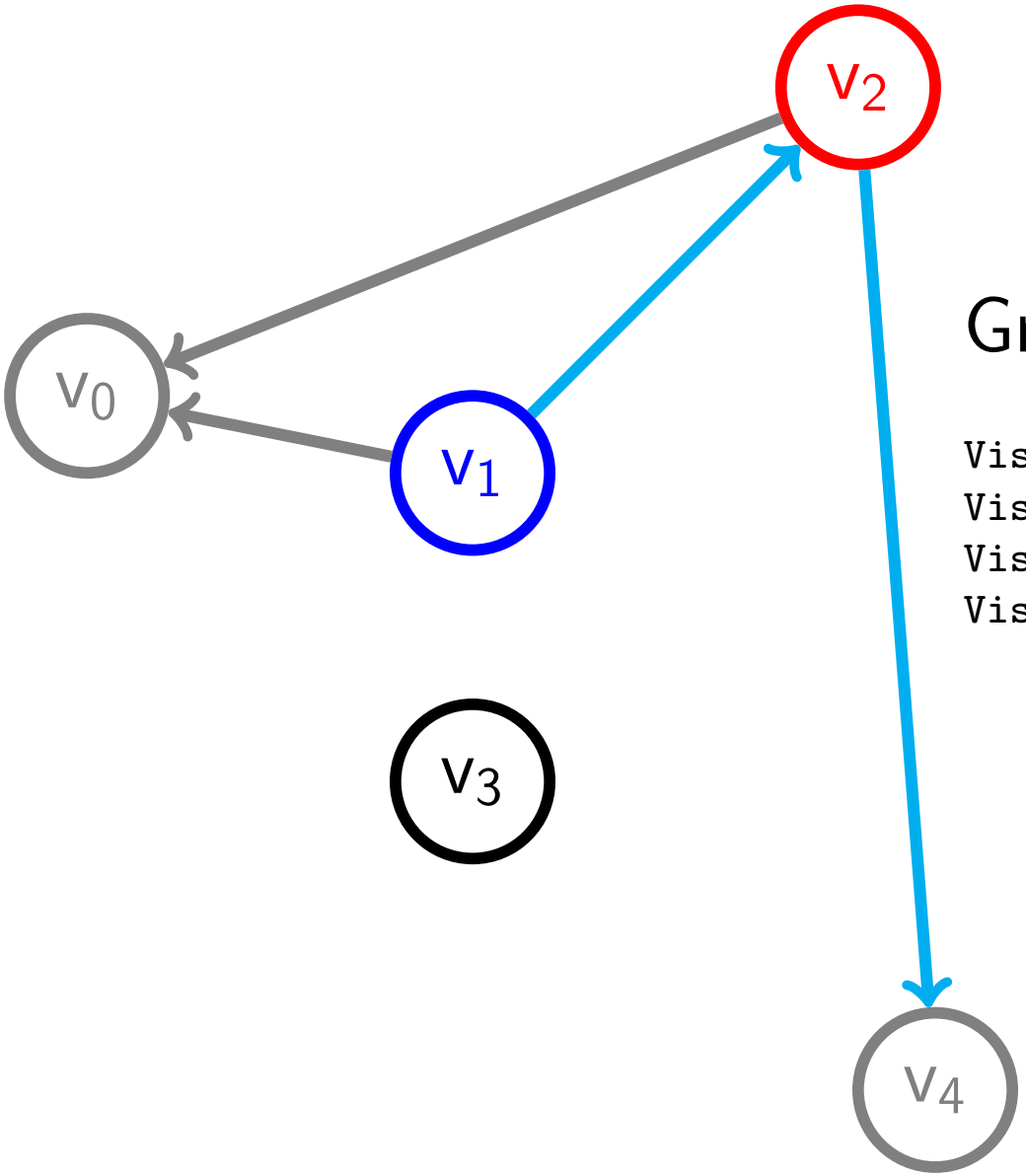
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

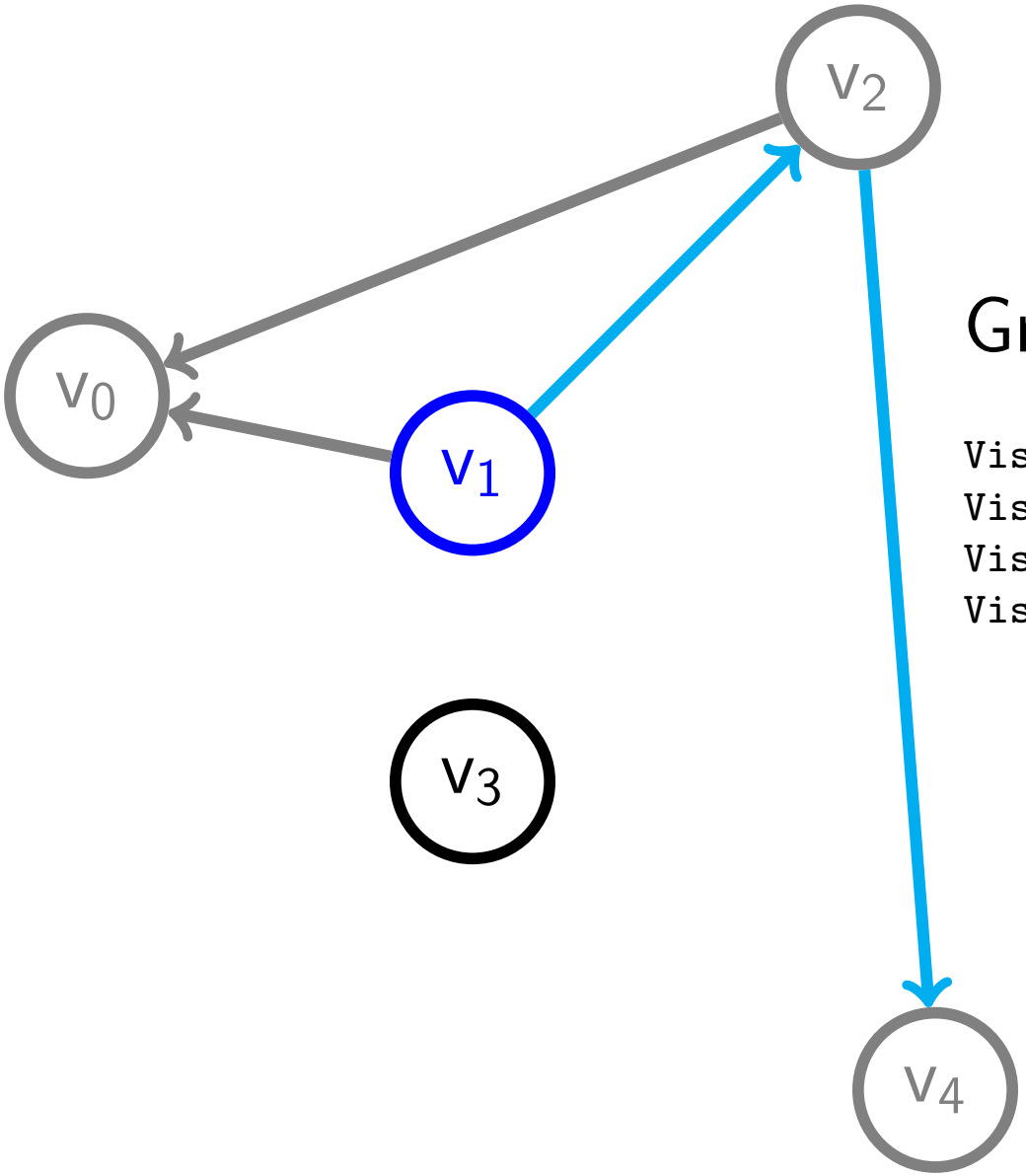
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

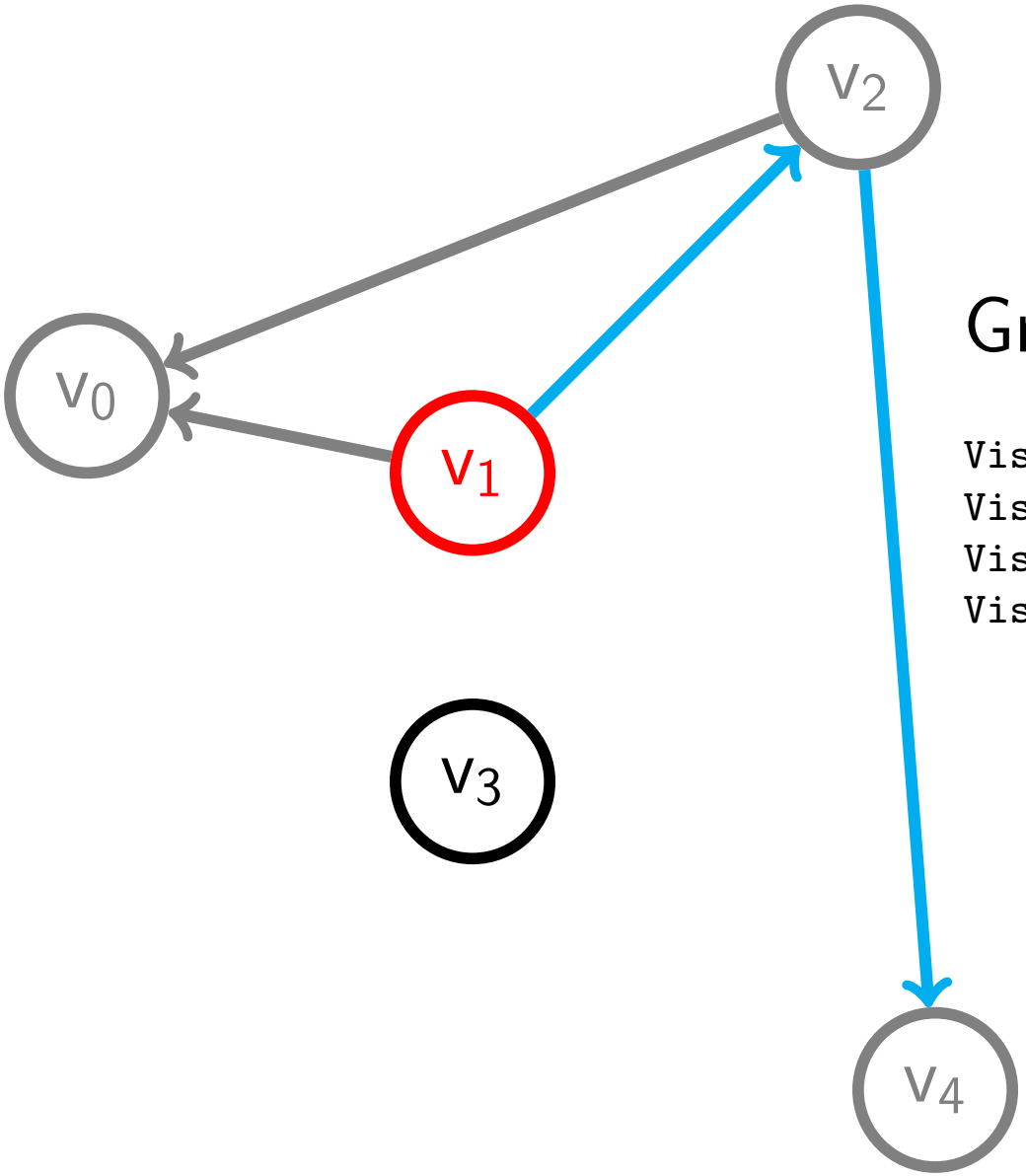
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

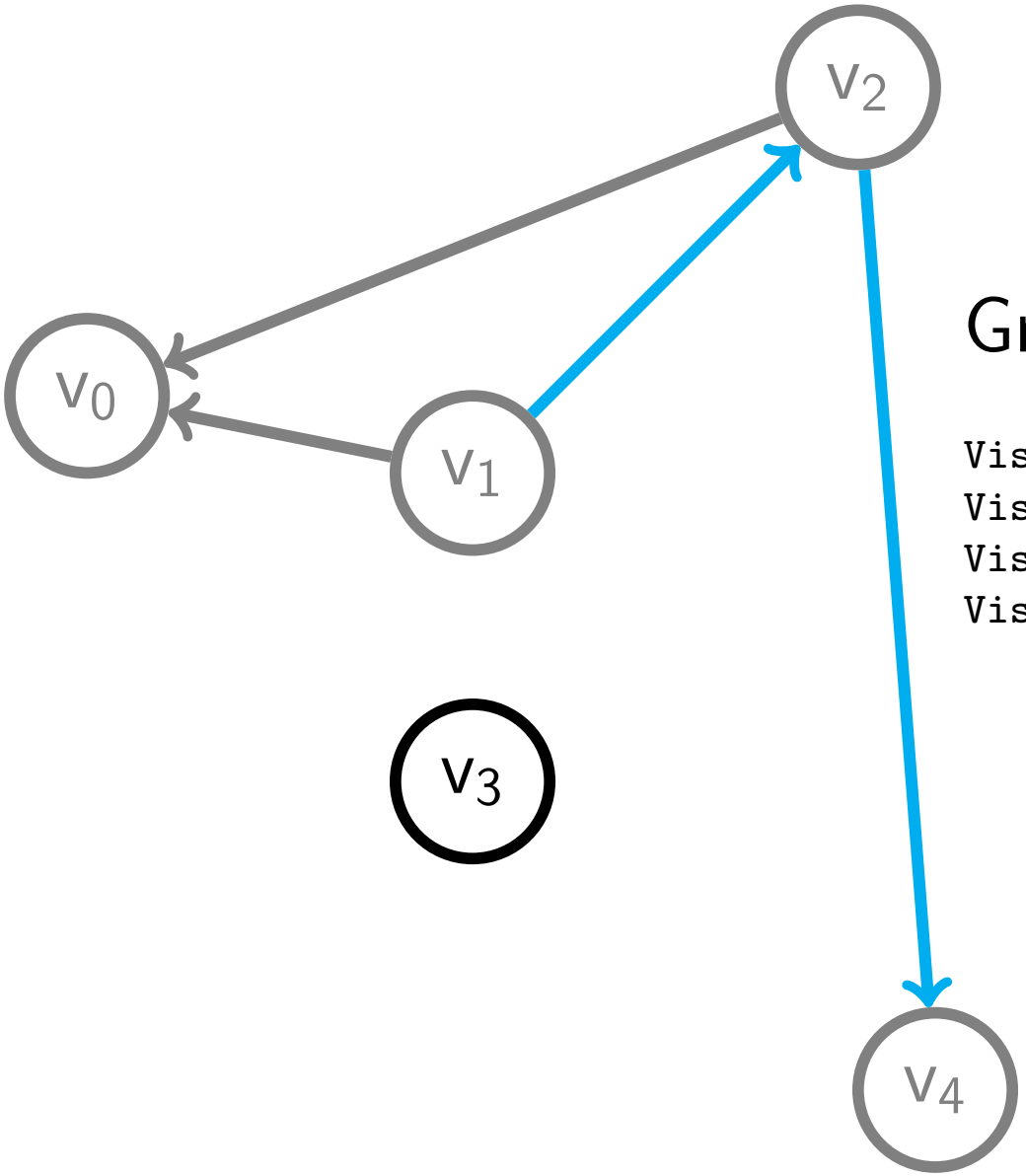
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

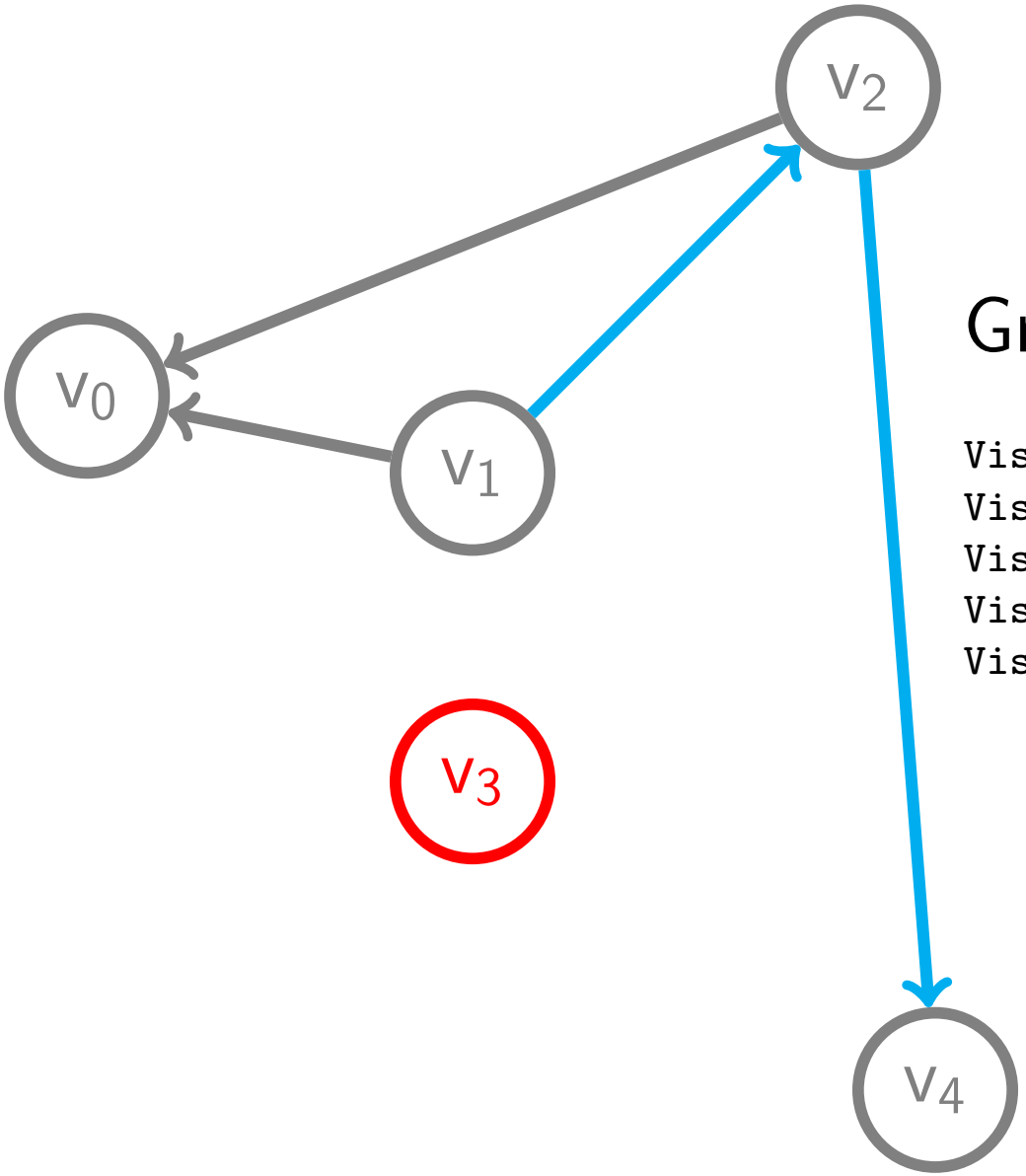
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
```

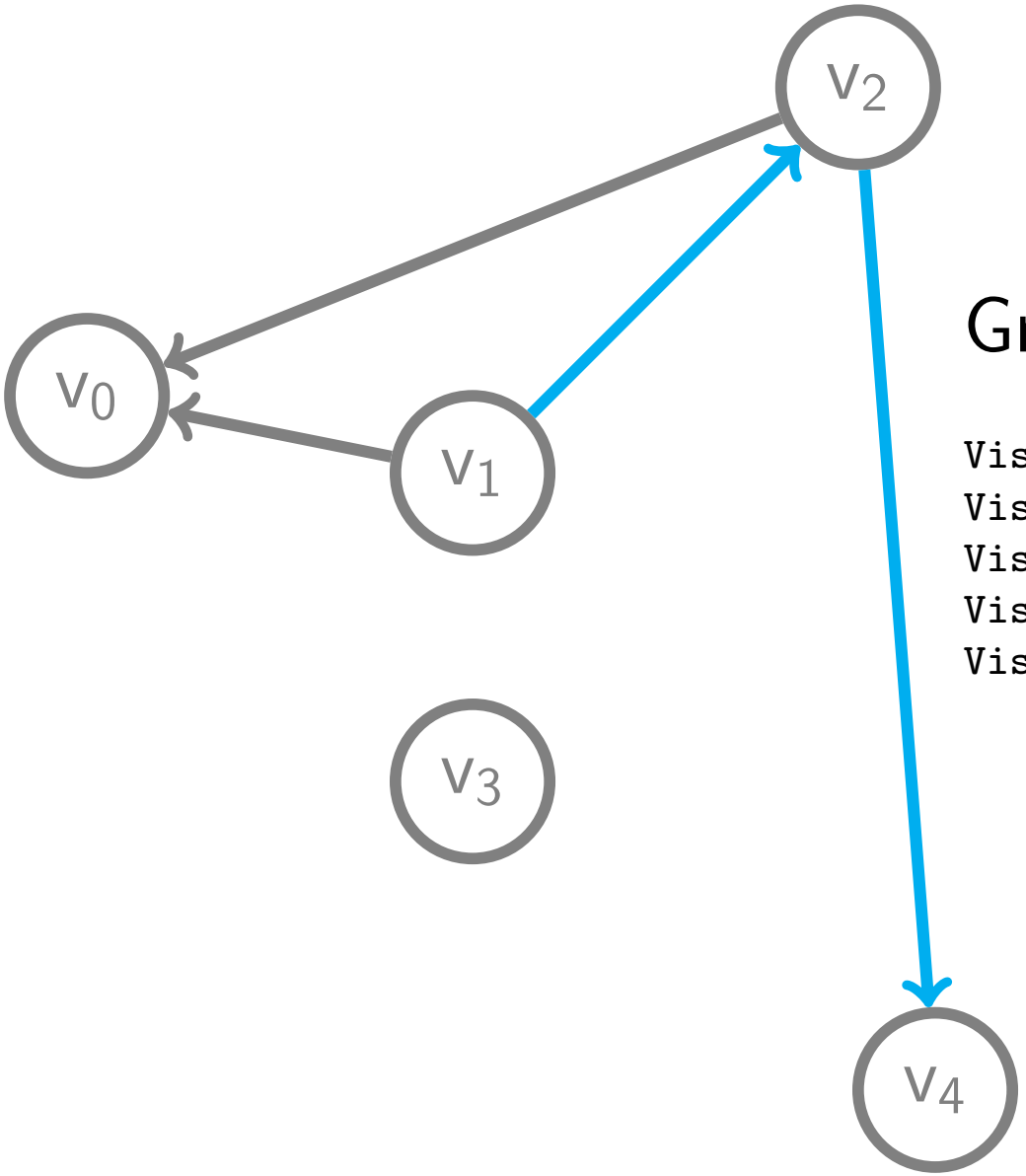
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
Visitando vertice:   3 (anterior:  -1)
```

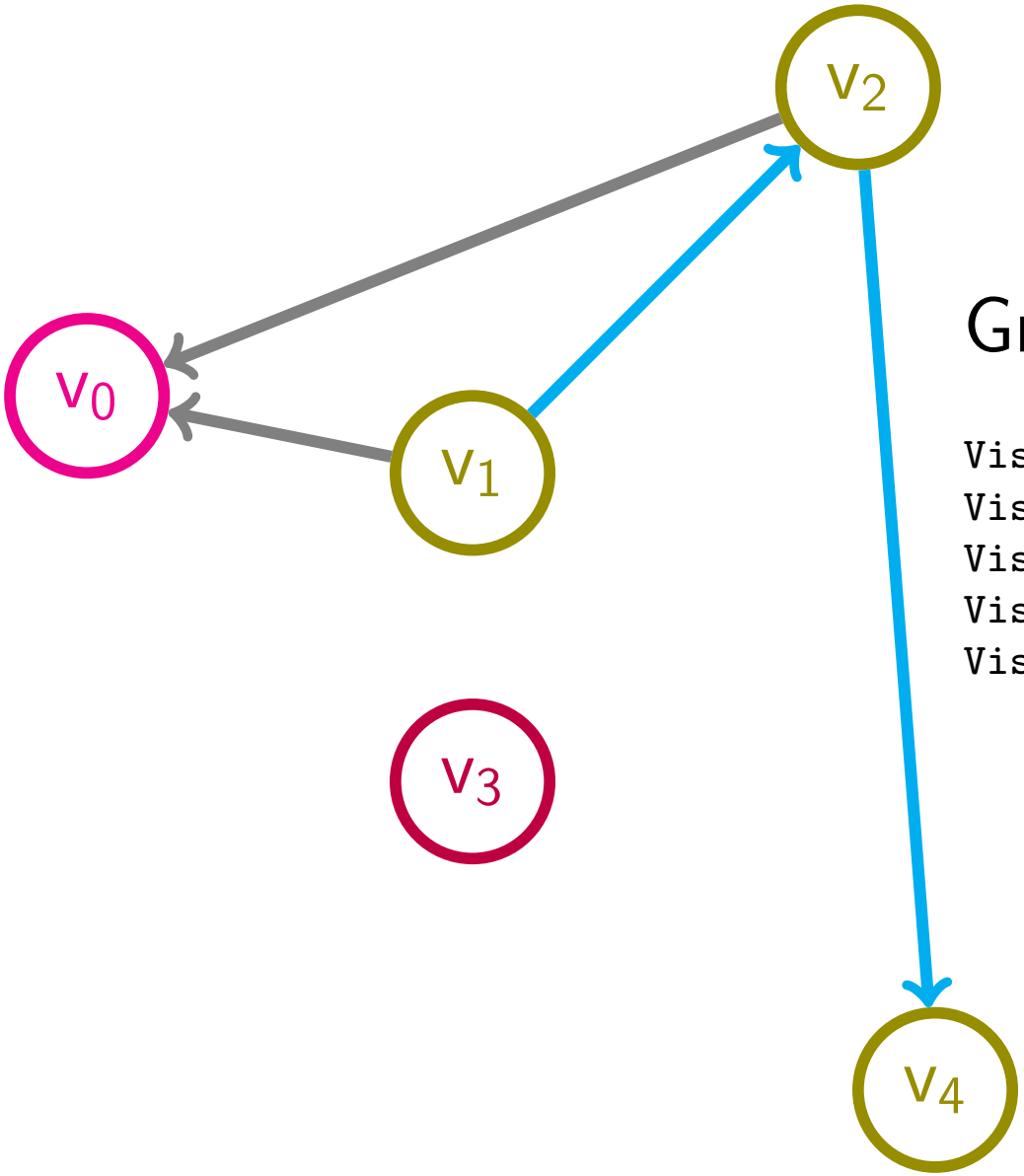
Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
Visitando vertice:   3 (anterior:  -1)
```

Digrafos – Busca em Profundidade



Grafo direcionado

```
Visitando vertice:   0 (anterior:  -1)
Visitando vertice:   1 (anterior:  -1)
Visitando vertice:   2 (anterior:   1)
Visitando vertice:   4 (anterior:   2)
Visitando vertice:   3 (anterior:  -1)
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){



}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);




}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;




}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  ElemLista* end = g->A[atual];




}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  ElemLista* end = g->A[atual];
  while (end) {



  }
}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  ElemLista* end = g->A[atual];
  while (end) {
    if (!visitado[end->vertice])
      visitaProfundidade(g,end->vertice,visitado,atual);

  }
}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  ElemLista* end = g->A[atual];
  while (end) {
    if (!visitado[end->vertice])
      visitaProfundidade(g,end->vertice,visitado,atual);
    end = end->prox;
  }
}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void visitaProfundidade(Grafo* g, int atual,
                        bool* visitado, int anterior){
  printf("Visitando vertice: %3i (anterior: %3i)\n",
                                    atual, anterior);
  visitado[atual] = true;
  ElemLista* end = g->A[atual];
  while (end) {
    if (!visitado[end->vertice])
      visitaProfundidade(g,end->vertice,visitado,atual);
    end = end->prox;
  }
}
```

$$O(V + E)$$

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){




}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;
  for (x=0;x<g->numVertices;x++)
    if (!visitado[x])
      visitaProfundidade(g, x, visitado, -1);
  free(visitado);
}
```

Grafos - Busca em Profundidade

Representação por **Listas de Adjacências**:
Grafos ou Digrafos Não Ponderados ou Ponderados

```
void buscaEmProfundidade(Grafo* g){
  if (!g || g->numVertices<1) return;
  int x;
  bool* visitado =
        (bool*) malloc(sizeof(bool)*g->numVertices);
  for (x=0;x<g->numVertices;x++) visitado[x] = false;
  for (x=0;x<g->numVertices;x++)
    if (!visitado[x])
      visitaProfundidade(g, x, visitado, -1);
  free(visitado);
}
```

$O(V + E)$

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

A Busca em Profundidade por ser utilizada (ou adaptada) para ajudar a resolver uma série de problemas:

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

A Busca em Profundidade por ser utilizada (ou adaptada) para ajudar a resolver uma série de problemas:

- Encontrar componentes conexos/conectados

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

A Busca em Profundidade por ser utilizada (ou adaptada) para ajudar a resolver uma série de problemas:

- Encontrar componentes conexos/conectados
- Encontrar componentes fortemente conexos/conectados

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

A Busca em Profundidade por ser utilizada (ou adaptada) para ajudar a resolver uma série de problemas:

- Encontrar componentes conexos/conectados
- Encontrar componentes fortemente conexos/conectados
- Verificar se um grafo é acíclico

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

A Busca em Profundidade por ser utilizada (ou adaptada) para ajudar a resolver uma série de problemas:

- Encontrar componentes conexos/conectados
- Encontrar componentes fortemente conexos/conectados
- Verificar se um grafo é acíclico
- Realizar a ordenação topológica

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

A Busca em Profundidade por ser utilizada (ou adaptada) para ajudar a resolver uma série de problemas:

- Encontrar componentes conexos/conectados
- Encontrar componentes fortemente conexos/conectados
- Verificar se um grafo é acíclico
- Realizar a ordenação topológica
- Encontrar soluções para labirintos

Grafos – Busca em Profundidade

Para que serve a Busca em Profundidade?

A Busca em Profundidade por ser utilizada (ou adaptada) para ajudar a resolver uma série de problemas:

- Encontrar componentes conexos/conectados
- Encontrar componentes fortemente conexos/conectados
- Verificar se um grafo é acíclico
- Realizar a ordenação topológica
- Encontrar soluções para labirintos
- ...

Busca em Profundidade - Aplicações

Faremos uma nova implementação da Busca em Profundidade que servirá de base para diversas aplicações.

Busca em Profundidade - Aplicações

Faremos uma nova implementação da Busca em Profundidade que servirá de base para diversas aplicações.

- Consideraremos três estados para os vértices:

Busca em Profundidade - Aplicações

Faremos uma nova implementação da Busca em Profundidade que servirá de base para diversas aplicações.

- Consideraremos três estados para os vértices:
  - BRANCO: vértice ainda não visitado

Busca em Profundidade - Aplicações

Faremos uma nova implementação da Busca em Profundidade que servirá de base para diversas aplicações.

- Consideraremos três estados para os vértices:
  - BRANCO: vértice ainda não visitado
  - CINZA: vértice visitado sendo processado (ainda estamos explorando sua lista de adjacências)

Busca em Profundidade - Aplicações

Faremos uma nova implementação da Busca em Profundidade que servirá de base para diversas aplicações.

- Consideraremos três estados para os vértices:
  - BRANCO: vértice ainda não visitado
  - CINZA: vértice visitado sendo processado (ainda estamos explorando sua lista de adjacências)
  - PRETO: vértice já processado (processamento concluído, chamada recursiva do vértice finalizada)

Busca em Profundidade - Aplicações

- O algoritmo controlará quatro arranjos (um valor para cada vértice):

Busca em Profundidade - Aplicações

- O algoritmo controlará quatro arranjos (um valor para cada vértice):
  - **cor**: cor de cada vértice (0 - BRANCO, 1 - CINZA ou 2 - PRETO);

Busca em Profundidade - Aplicações

- O algoritmo controlará quatro arranjos (um valor para cada vértice):
  - **cor**: cor de cada vértice (0 - BRANCO, 1 - CINZA ou 2 - PRETO);
  - **tDescoberta**: tempo em que o vértice foi descoberto (visitado pela primeira vez);

Busca em Profundidade - Aplicações

- O algoritmo controlará quatro arranjos (um valor para cada vértice):
  - **cor**: cor de cada vértice (0 - BRANCO, 1 - CINZA ou 2 - PRETO);
  - **tDescoberta**: tempo em que o vértice foi descoberto (visitado pela primeira vez);
  - **tTermino**: tempo em que o processamento do vértice terminou (momento em que se tornou PRETO);

Busca em Profundidade - Aplicações

- O algoritmo controlará quatro arranjos (um valor para cada vértice):
  - **cor**: cor de cada vértice (0 - BRANCO, 1 - CINZA ou 2 - PRETO);
  - **tDescoberta**: tempo em que o vértice foi descoberto (visitado pela primeira vez);
  - **tTermino**: tempo em que o processamento do vértice terminou (momento em que se tornou PRETO);
  - **anterior**: indicação do vértice anterior ao vértice atual (a partir do qual chegamos no vértice atual).

Busca em Profundidade - Outra Implementação

```
void DFSCores(Grafo* g){
  if (!g || g->numVertices<1) return;
  int* cor = (int*) malloc(sizeof(int)*g->numVertices);
  int* tDescoberta = (int*) malloc(sizeof(int)*g->numVertices);
  int* tTermino = (int*) malloc(sizeof(int)*g->numVertices);
  int* anterior = (int*) malloc(sizeof(int)*g->numVertices);
  int tempo = 0;
  int x;
  for (x=0;x<g->numVertices;x++){
    cor[x] = 0;  // BRANCO
    tDescoberta[x] = -1;
    tTermino[x] = -1;
    anterior[x] = -1;
  }

  ...
```

Busca em Profundidade - Outra Implementação

```
void DFSCores(Grafo* g){
  ...
  for (x=0;x<g->numVertices;x++)
    if (cor[x]==0)
      visitaDFSCores(g,x,&tempo,cor,tDescoberta,tTermino,anterior);

  printf("Resumo da Busca em Profundidade:\n");
  printf("No\tanterior\tDescoberta\tTermino\tCor:\n");
  for (x=0;x<g->numVertices;x++)
    printf("%2i\t%8i\t%10i\t%7i\t%3i\n",x,anterior[x],
                              tDescoberta[x],tTermino[x],cor[x]);
  printf("\n");
  free(cor);
  free(tDescoberta);
  free(tTermino);
  free(anterior);
}
```

Busca em Profundidade - Outra Implementação

```
void DFSCores(Grafo* g){
  ...
  for (x=0;x<g->numVertices;x++)
    if (cor[x]==0)
      visitaDFSCores(g,x,&tempo,cor,tDescoberta,tTermino,anterior);

  printf("Resumo da Busca em Profundidade:\n");
  printf("No\tanterior\tDescoberta\tTermino\tCor:\n");
  for (x=0;x<g->numVertices;x++)
    printf("%2i\t%8i\t%10i\t%7i\t%3i\n",x,anterior[x],
                              tDescoberta[x],tTermino[x],cor[x]);
  printf("\n");
  free(cor);
  free(tDescoberta);
  free(tTermino);
  free(anterior);
}
```

$O(V) + O(E)$

Busca em Profundidade - Outra Implementação

```
void visitaDFSCores(Grafo* g, int atual, int* tempo, int* cor,
                    int* tDescoberta, int* tTermino, int* anterior){
  (*tempo)++;
  cor[atual] = 1; // CINZA
  tDescoberta[atual] = *tempo;
  int w;
  ElemLista* end = g->A[atual];
  while (end){
    w = end->vertice;
    if(cor[w]==0){ // BRANCO
      anterior[w] = atual;
      visitaDFSCores(g,w,tempo,cor,tDescoberta,tTermino,anterior);
    }
    end = end->prox;
  }
  cor[atual] = 2; // PRETO
  (*tempo)++;
  tTermino[atual] = *tempo;
}
```

Busca em Profundidade - Outra Implementação

```
void visitaDFSCores(Grafo* g, int atual, int* tempo, int* cor,
                    int* tDescoberta, int* tTermino, int* anterior){
  (*tempo)++;
  cor[atual] = 1; // CINZA
  tDescoberta[atual] = *tempo;
  int w;
  ElemLista* end = g->A[atual];
  while (end){
    w = end->vertice;
    if(cor[w]==0){ // BRANCO
      anterior[w] = atual;
      visitaDFSCores(g,w,tempo,cor,tDescoberta,tTermino,anterior);
    }
    end = end->prox;
  }
  cor[atual] = 2; // PRETO
  (*tempo)++;
  tTermino[atual] = *tempo;
}
```

$$O(V) + O(E)$$

Grafos – Busca em Profundidade

- Consideraremos quatro tipos de arestas:

Grafos – Busca em Profundidade

- Consideraremos quatro tipos de arestas:
  - **Arestas de árvore:** arestas de uma árvore de busca em profundidade (arestas utilizadas para acessar vértices pela primeira vez);

Grafos – Busca em Profundidade

- Consideraremos quatro tipos de arestas:
  - **Arestas de árvore:** arestas de uma árvore de busca em profundidade (arestas utilizadas para acessar vértices pela primeira vez);
  - **Arestas de retorno:** conectam um vértice $v2$ a um antecessor $v1$ em uma árvore de busca em profundidade;

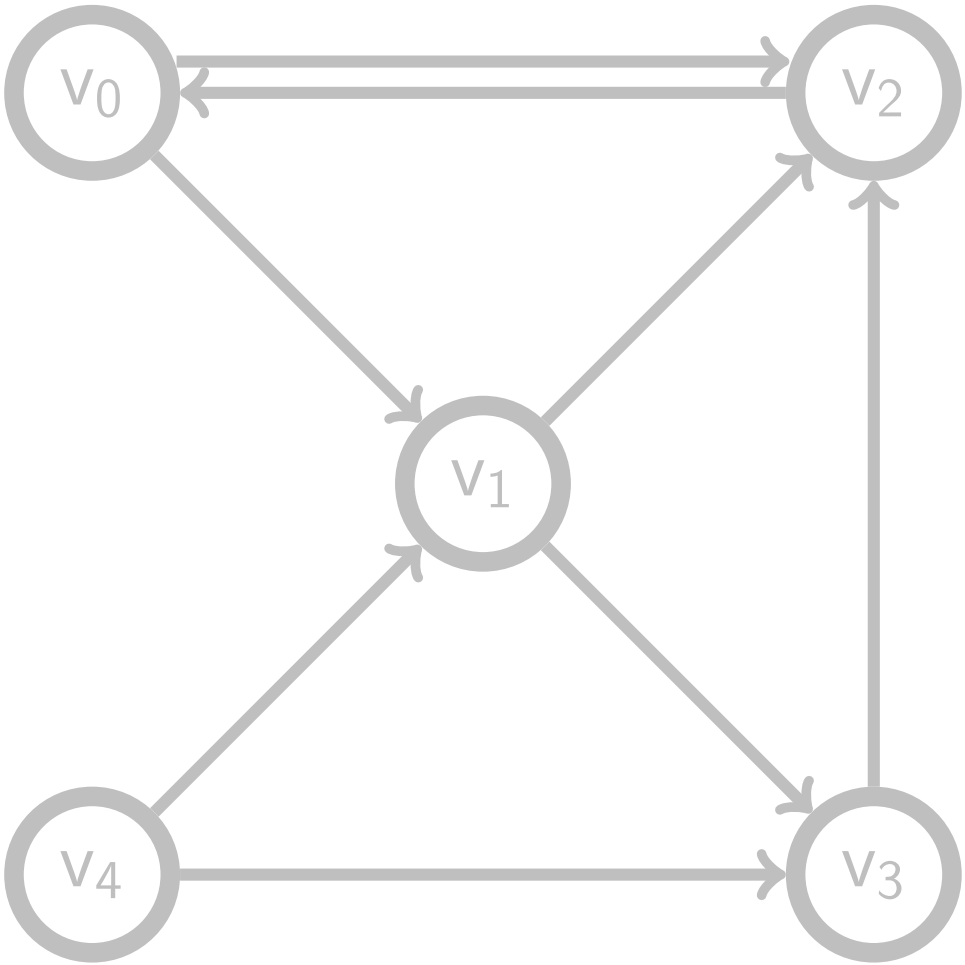
Grafos – Busca em Profundidade

- Consideraremos quatro tipos de arestas:
  - **Arestas de árvore:** arestas de uma árvore de busca em profundidade (arestas utilizadas para acessar vértices pela primeira vez);
  - **Arestas de retorno:** conectam um vértice $v2$ a um antecessor $v1$ em uma árvore de busca em profundidade;
  - **Arestas de avanço:** não pertencem à árvore de busca em profundidade, mas conectam um vértice $v1$ a um sucessor $v2$ na árvore;

Grafos – Busca em Profundidade

- Consideraremos quatro tipos de arestas:
  - **Arestas de árvore:** arestas de uma árvore de busca em profundidade (arestas utilizadas para acessar vértices pela primeira vez);
  - **Arestas de retorno:** conectam um vértice $v2$ a um antecessor $v1$ em uma árvore de busca em profundidade;
  - **Arestas de avanço:** não pertencem à árvore de busca em profundidade, mas conectam um vértice $v1$ a um sucessor $v2$ na árvore;
  - **Arestas de cruzamento:** conectam vértices de árvores diferentes de busca ou da mesma árvore, mas que não sejam nem antecessores nem sucessores (por exemplo, irmãos).

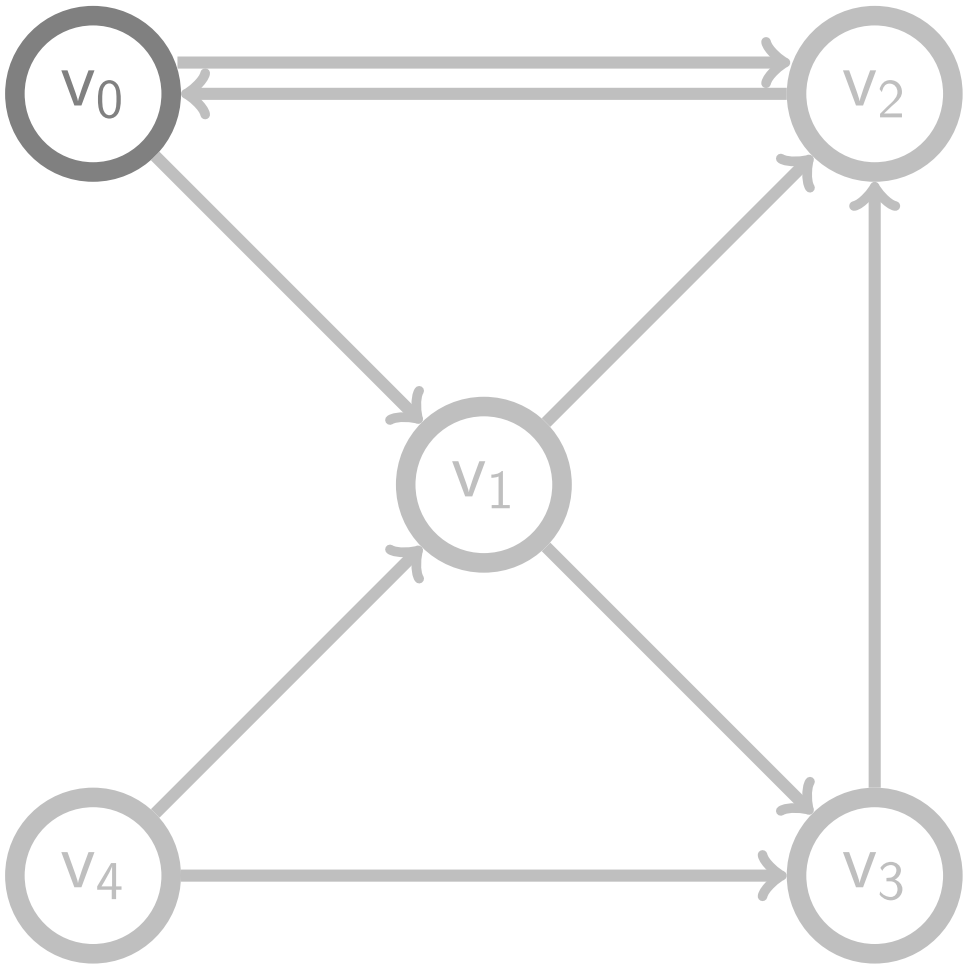
Digrafos – Busca em Profundidade




```
Resumo da Busca em Profundidade:
Nó Anterior Descoberta Término Cor:
 0       -1          1       8   2
 1        0          2       7   2
 2        1          3       4   2
 3        1          5       6   2
 4       -1          9      10   2
```

Tempo: 0

Digrafos – Busca em Profundidade

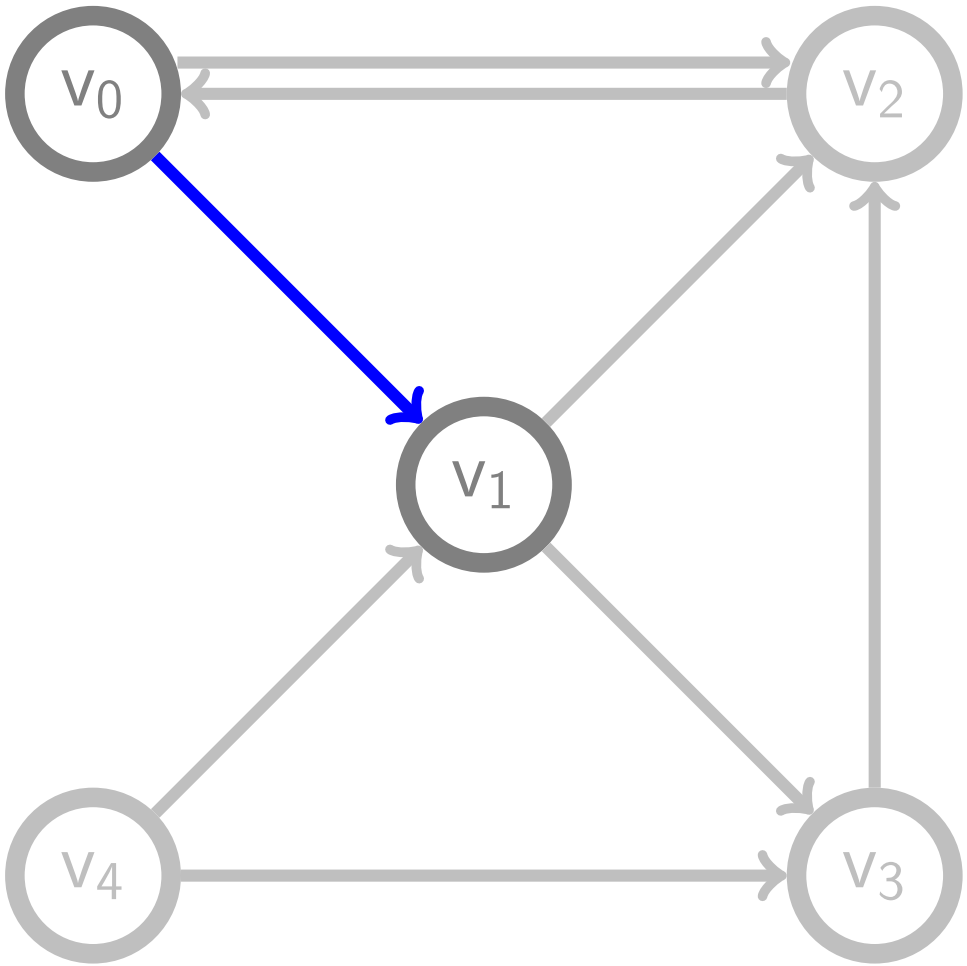


Resumo da Busca em Profundidade:

Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

Tempo: 1

Digrafos – Busca em Profundidade

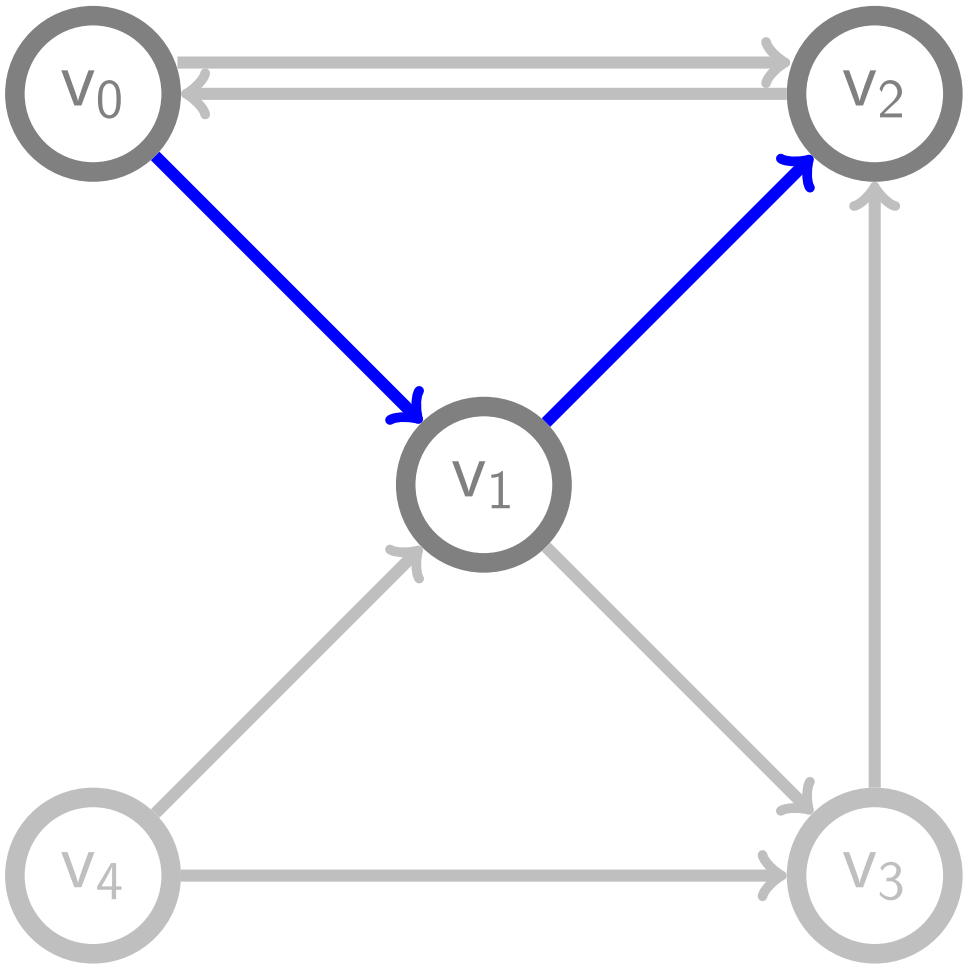



```
Resumo da Busca em Profundidade:
Nó Anterior Descoberta Término Cor:
 0       -1          1       8   2
 1        0          2       7   2
 2        1          3       4   2
 3        1          5       6   2
 4       -1          9      10   2
```

Arestas de Árvore

Tempo: 2

Digrafos – Busca em Profundidade



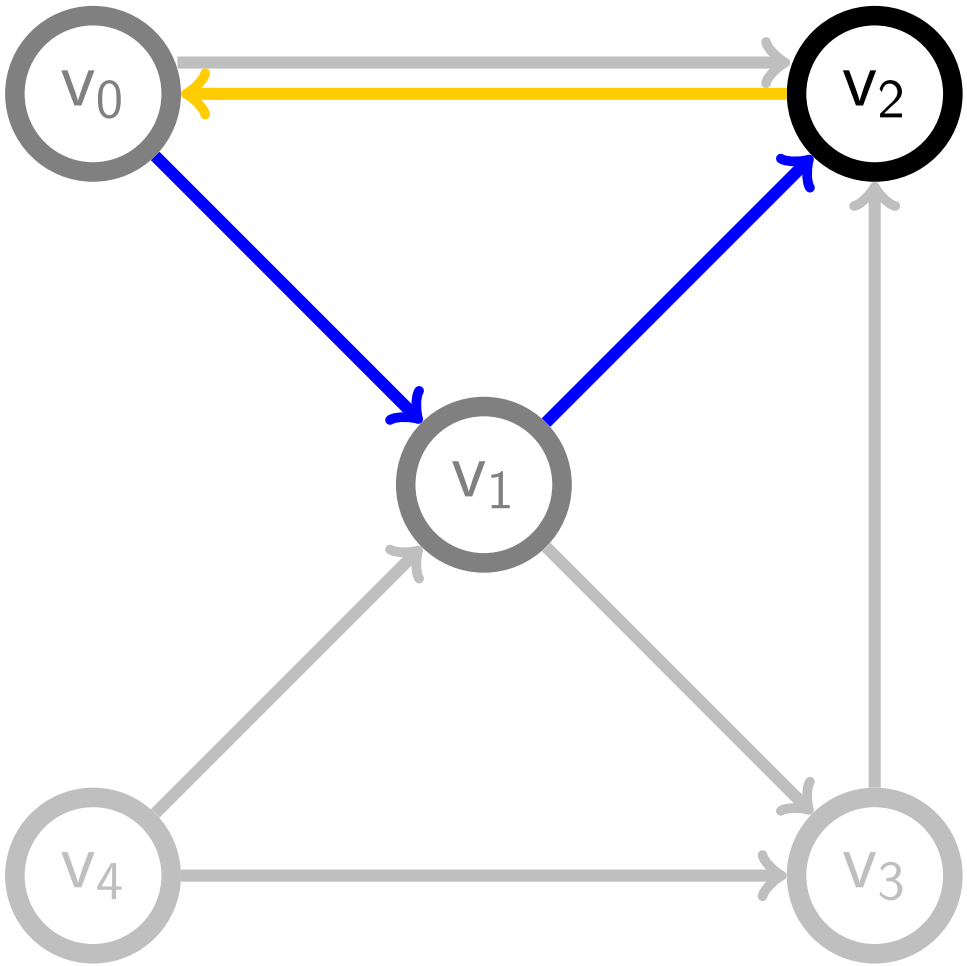
Resumo da Busca em Profundidade:

Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

Arestas de Árvore

Tempo: 3

Digrafos – Busca em Profundidade



Resumo da Busca em Profundidade:

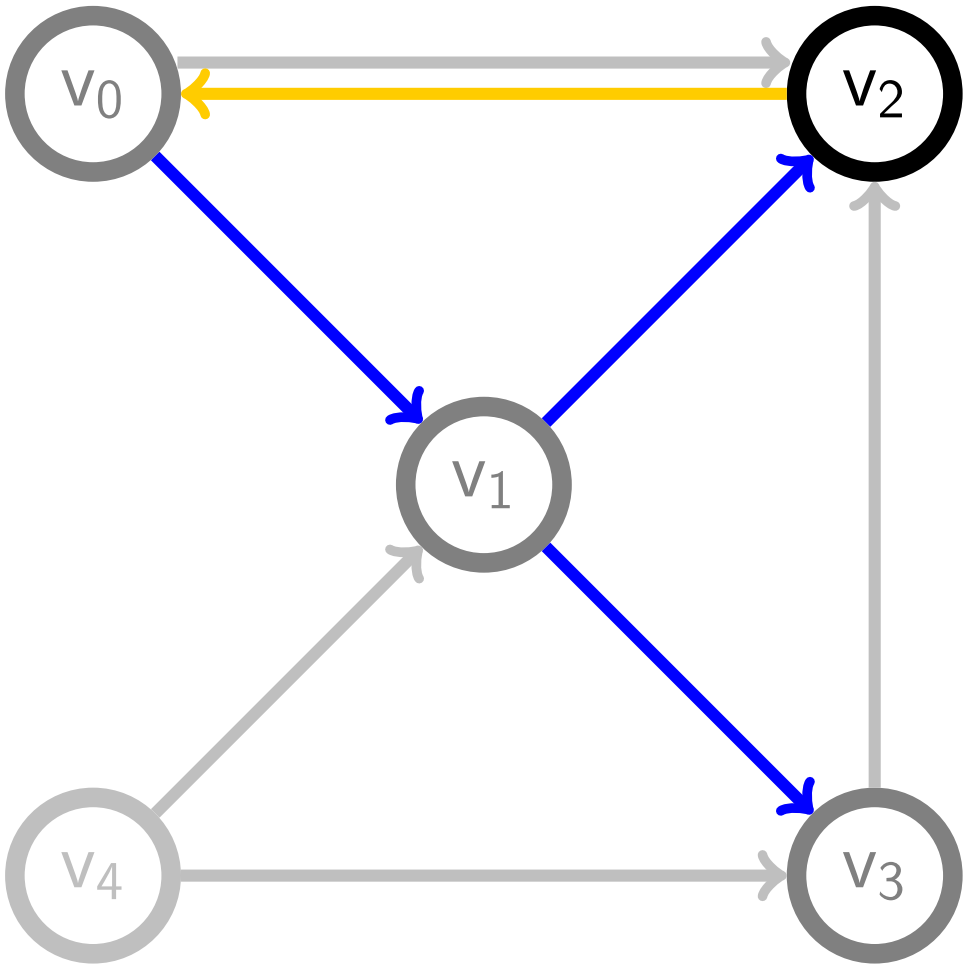
Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

Arestas de Árvore

Arestas de Retorno

Tempo: 4

Digrafos – Busca em Profundidade



Resumo da Busca em Profundidade:

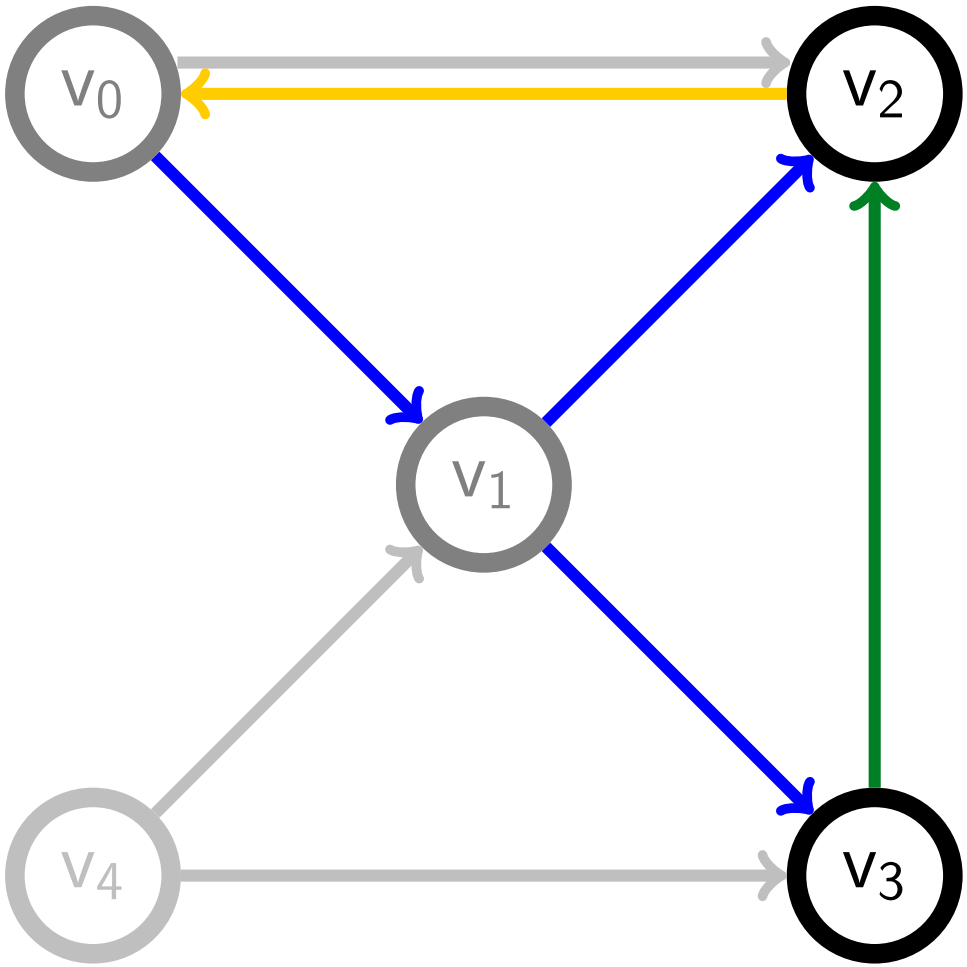
Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

Arestas de Árvore

Arestas de Retorno

Tempo: 5

Digrafos – Busca em Profundidade



Resumo da Busca em Profundidade:

Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

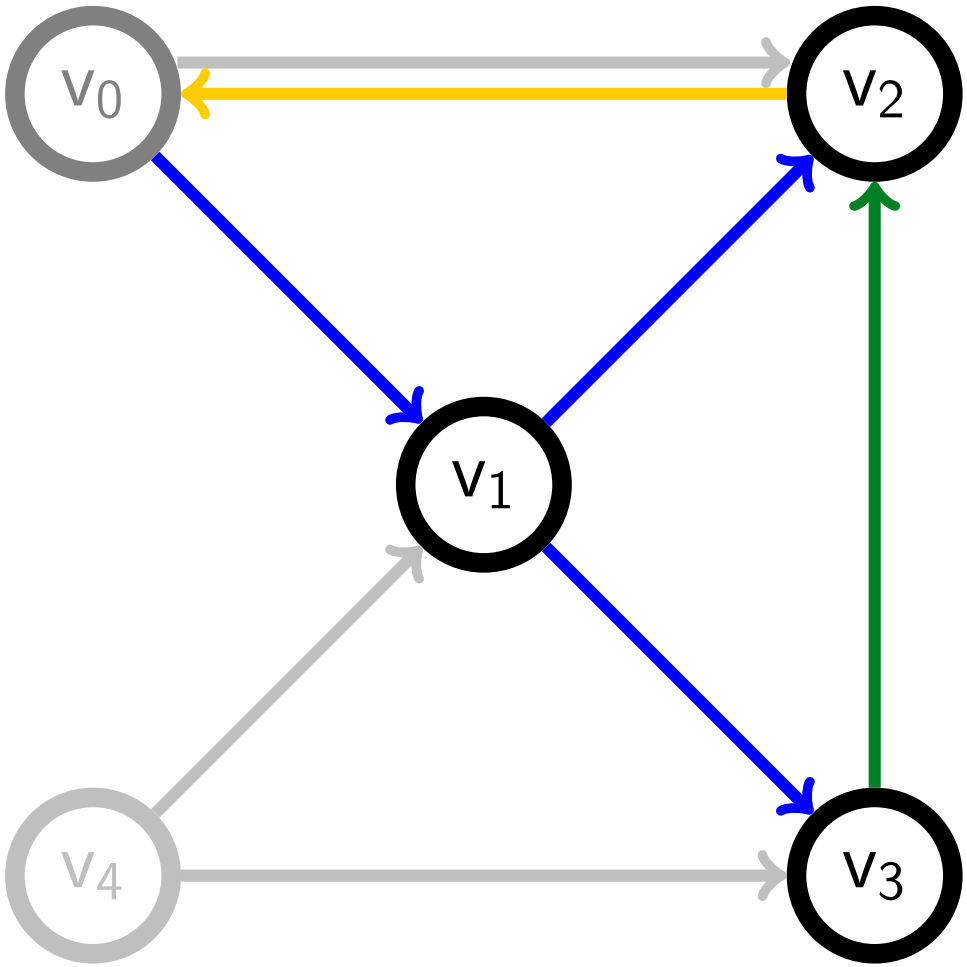
Arestas de Árvore

Arestas de Retorno

Arestas de Cruzamento

Tempo: 6

Digrafos – Busca em Profundidade



Resumo da Busca em Profundidade:

Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

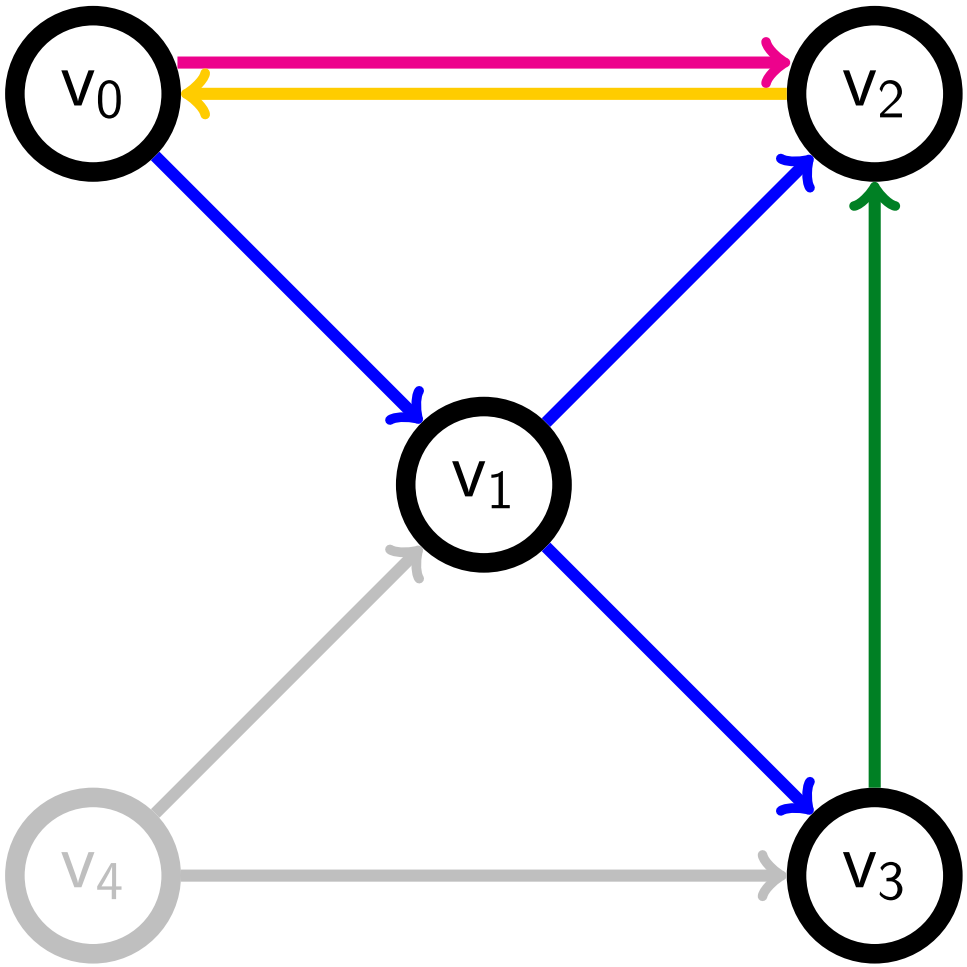
Arestas de Árvore

Arestas de Retorno

Arestas de Cruzamento

Tempo: 7

Digrafos – Busca em Profundidade



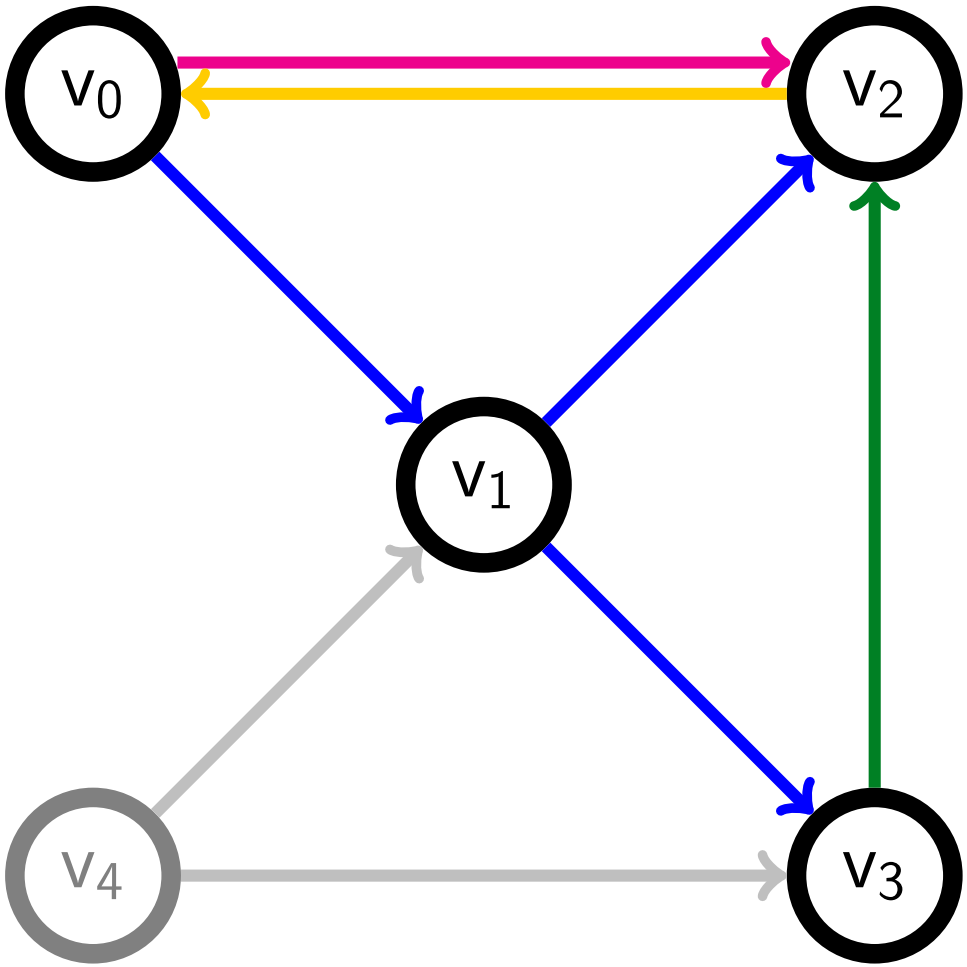
Resumo da Busca em Profundidade:

Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

Arestas de Árvore
Arestas de Retorno
Arestas de Cruzamento
Arestas de Avanço

Tempo: 8

Digrafos – Busca em Profundidade



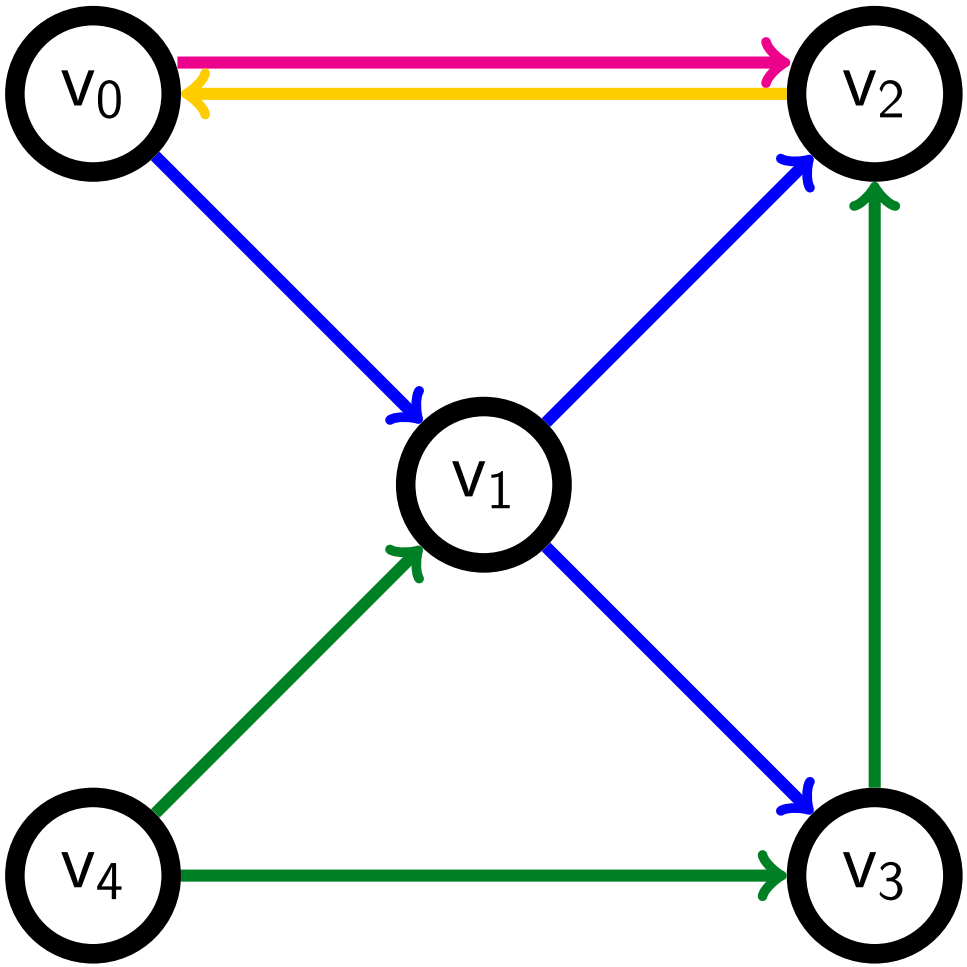
Resumo da Busca em Profundidade:

Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

Arestas de Árvore
Arestas de Retorno
Arestas de Cruzamento
Arestas de Avanço

Tempo: 9

Digrafos – Busca em Profundidade



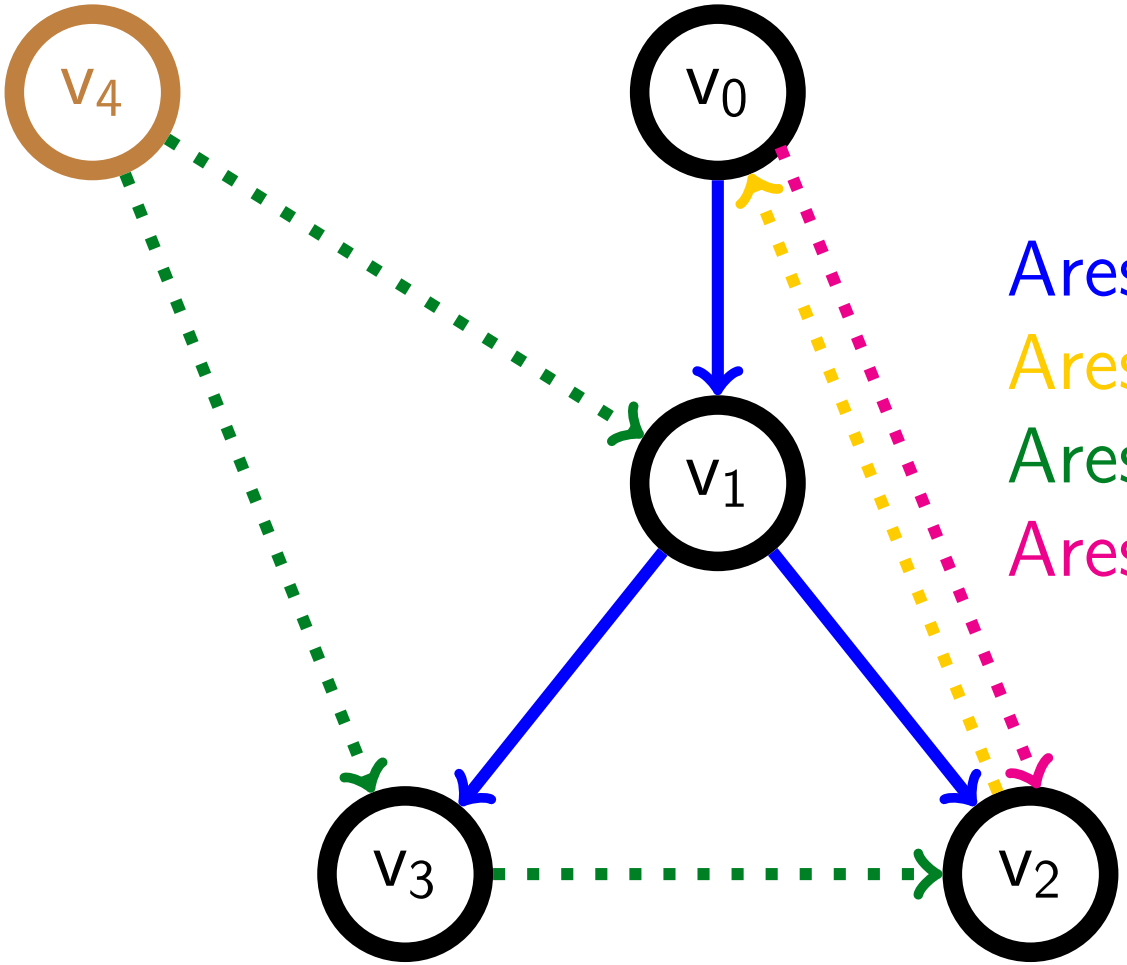
Resumo da Busca em Profundidade:

Nó	Anterior	Descoberta	Término	Cor:
0	-1	1	8	2
1	0	2	7	2
2	1	3	4	2
3	1	5	6	2
4	-1	9	10	2

Arestas de Árvore
Arestas de Retorno
Arestas de Cruzamento
Arestas de Avanço

Tempo: 10

Árvores de Busca em Profundidade



Arestas de Árvore
Arestas de Retorno
Arestas de Cruzamento
Arestas de Avanço

Algoritmos e Estruturas de Dados II
Aula 06 – Grafos: Busca em Profundidade

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