Caminhos Mínimos a partir de um vértice
Definição do Problema

- Considere um grafo $G = (V, E)$ e uma função de custo das arestas $w : E \to \mathbb{R}$

Caminhos Mínimos a partir de um vértice
Definição do Problema

- Considere um grafo $G = (V, E)$ e uma função de custo das arestas $w : E \to \mathbb{R}$
- O **custo** de um caminho P em G é dado por

$$\sum_{e \in E(P)} w(e)$$

Caminhos Mínimos a partir de um vértice
Definição do Problema

- Considere um grafo $G = (V, E)$ e uma função de custo das arestas $w : E \to \mathbb{R}$
- O **custo** de um caminho P em G é dado por

$$\sum_{e \in E(P)} w(e)$$

- Um **st-caminho** é um caminho cujos extremos são os vértices s e t

Caminhos Mínimos a partir de um vértice
Definição do Problema

Caminho mínimo a partir de um vértice

Considere um grafo $G = (V, E)$ e uma função de custo das arestas $w\colon E \to \mathbb{R}$, e um vértice $s \in V$. Queremos encontrar, para todo vértice u de G, um su-caminho P de custo mínimo, ou seja, para qualquer su-caminho P' em G, temos que

$$w(P) \leq w(P').$$

Caminhos Mínimos a partir de um vértice
Definição do Problema

Caminho mínimo a partir de um vértice

Considere um grafo $G = (V, E)$ e uma função de custo das arestas $w\colon E \to \mathbb{R}$, e um vértice $s \in V$. Queremos encontrar, para todo vértice u de G, um su-caminho P de custo mínimo, ou seja, para qualquer su-caminho P' em G, temos que

$$w(P) \leq w(P').$$

- A **distância** de s a v é o custo de um sv-caminho minímo.

Caminhos Mínimos a partir de um vértice
Definição do Problema

- Encontrar o caminho mínimo entre **um par** de vértices, s a u não é mais fácil que encontrar os caminhos mínimos de s a todos os vértices do grafo.

Caminhos Mínimos a partir de um vértice
Definição do Problema

- Encontrar o caminho mínimo entre **um par** de vértices, s a u não é mais fácil que encontrar os caminhos mínimos de s a todos os vértices do grafo.
- Notem que o caminho mínimo de s a u pode passar por todos os outros vértices do grafo.

Caminhos Mínimos a partir de um vértice
Definição do Problema

- Encontrar o caminho mínimo entre **um par** de vértices, s a u não é mais fácil que encontrar os caminhos mínimos de s a todos os vértices do grafo.
- Notem que o caminho mínimo de s a u pode passar por todos os outros vértices do grafo.
- Além disso, se o caminho mínimo de s a u passa por v, então este é composto pelo caminho mínimo de s a v unido ao caminho mínimo de v a u.

Caminhos Mínimos a partir de um vértice
Definição do Problema

- Encontrar o caminho mínimo entre **um par** de vértices, *s* a *u* não é mais fácil que encontrar os caminhos mínimos de *s* a todos os vértices do grafo.
- Notem que o caminho mínimo de *s* a *u* pode passar por todos os outros vértices do grafo.
- Além disso, se o caminho mínimo de *s* a *u* passa por *v*, então este é composto pelo caminho mínimo de *s* a *v* unido ao caminho mínimo de *v* a *u*.
- O algoritmo de Dijkstra que veremos a seguir baseia-se no fato de que conseguimos determinar, na *k*-ésima iteração, os *k* vértices mais próximos (distância) de *r*.

Notações

Complemento de conjunto

Seja V um conjunto e $X \subseteq V$. Definimos $\overline{X} = V \setminus X$

Notações

Complemento de conjunto

Seja V um conjunto e $X \subseteq V$. Definimos $\overline{X} = V \setminus X$

Corte de arestas

Dado um grafo G e um conjunto de vértices $X \subseteq V(G)$, definimos

$$\partial_G(X) = \{uv \in E(G) : u \in X \text{ e } v \in V(G) \setminus X\}$$

Algoritmo de Dijkstra: projeto por indução

- **Hipótese:** Dado um grafo $G = (V, E)$, uma função de custos $w\colon E \to \mathbb{R}^+$, e um vértice s de G, sabemos encontrar os k vértices mais próximos de s, $1 \leq k \leq |V| - 1$, e as respectivas distâncias mínimas.

Algoritmo de Dijkstra: projeto por indução

- **Hipótese:** Dado um grafo $G = (V, E)$, uma função de custos $w\colon E \to \mathbb{R}^+$, e um vértice s de G, sabemos encontrar os k vértices mais próximos de s, $1 \leq k \leq |V| - 1$, e as respectivas distâncias mínimas.
- **Base:** Para $k = 1$, o próprio vértice origem s é o vértice mais próximo, com distância 0.

Algoritmo de Dijkstra: projeto por indução

- **Hipótese:** Dado um grafo $G = (V, E)$, uma função de custos $w\colon E \to \mathbb{R}^+$, e um vértice s de G, sabemos encontrar os k vértices mais próximos de s, $1 \leq k \leq |V| - 1$, e as respectivas distâncias mínimas.
- **Base:** Para $k = 1$, o próprio vértice origem s é o vértice mais próximo, com distância 0.
  Para $k = 2$, o segundo vértice mais próximo de s é um vértice u adjacente a s por uma aresta $e = su$ com $w(e)$ mínimo, que é o valor da distância mínima.

Algoritmo de Dijkstra: projeto por indução

- **Hipótese:** Dado um grafo $G = (V, E)$, uma função de custos $w\colon E \to \mathbb{R}^+$, e um vértice s de G, sabemos encontrar os k vértices mais próximos de s, $1 \leq k \leq |V| - 1$, e as respectivas distâncias mínimas.
- **Base:** Para $k = 1$, o próprio vértice origem s é o vértice mais próximo, com distância 0.
  Para $k = 2$, o segundo vértice mais próximo de s é um vértice u adjacente a s por uma aresta $e = su$ com $w(e)$ mínimo, que é o valor da distância mínima.
- **Passo:** Por hipótese de indução, sabemos encontrar os $k - 1$ vértices mais próximos do vértice origem s e suas distâncias. Seja S o conjunto dos $k - 1$ vértices mais próximos de s e $\mathrm{dist}[u]$ a distância mínima de s a u, para todo $u \in S$.

Algoritmo de Dijkstra: projeto por indução

Algoritmo de Dijkstra: projeto por indução

- **Passo (cont.):** O k-ésimo vértice mais próximo de s é um vértice de $\overline{S}$ e o caminho mínimo de s a ele necessariamente passa por uma aresta de $\partial(S)$.

Algoritmo de Dijkstra: projeto por indução

- **Passo (cont.):** O k-ésimo vértice mais próximo de s é um vértice de $\overline{S}$ e o caminho mínimo de s a ele necessariamente passa por uma aresta de $\partial(S)$.
  Defina, para todo vértice v de $\overline{S}$ que é extremo de alguma aresta uv de $\partial(S)$,

$$D(v) = \min_{uv \in \partial(S)} \{\mathrm{dist}[u] + w(uv)\}.$$

Algoritmo de Dijkstra: projeto por indução

- **Passo (cont.):** O k-ésimo vértice mais próximo de s é um vértice de $\overline{S}$ e o caminho mínimo de s a ele necessariamente passa por uma aresta de $\partial(S)$.
  Defina, para todo vértice v de $\overline{S}$ que é extremo de alguma aresta uv de $\partial(S)$,

$$D(v) = \min_{uv \in \partial(S)} \{\mathrm{dist}[u] + w(uv)\}.$$

  O vértice v^* com $D(v^*)$ mínimo será o k-ésimo vértice mais próximo de s, pois qualquer caminho de s a um vértice $y \in \overline{S}$, $y \neq v^*$ necessariamente passaria por um vértice x extremo de alguma aresta de $\partial(S)$.

Algoritmo de Dijkstra: projeto por indução

- **Passo (cont.):** O k-ésimo vértice mais próximo de s é um vértice de $\overline{S}$ e o caminho mínimo de s a ele necessariamente passa por uma aresta de $\partial(S)$.
  Defina, para todo vértice v de $\overline{S}$ que é extremo de alguma aresta uv de $\partial(S)$,

$$D(v) = \min_{uv \in \partial(S)} \{\mathrm{dist}[u] + w(uv)\}.$$

  O vértice v^* com $D(v^*)$ mínimo será o k-ésimo vértice mais próximo de s, pois qualquer caminho de s a um vértice $y \in \overline{S}$, $y \neq v^*$ necessariamente passaria por um vértice x extremo de alguma aresta de $\partial(S)$.
  Como $D(v^*) \leq D(y)$ e os pesos das arestas são não negativos tal caminho de s a y certamente não terá custo menor que $D(v)$. □

Algoritmo de Dijkstra

Da demonstração indutiva obtemos um algoritmo para determinar não apenas as distâncias mínimas de s aos demais vértices, mas também os caminhos mínimos:

Algoritmo de Dijkstra

Da demonstração indutiva obtemos um algoritmo para determinar não apenas as distâncias mínimas de s aos demais vértices, mas também os caminhos mínimos:

Principais passos do algoritmo

1. Inicialmente, tome $S = \{s\}$.
2. Enquanto $\overline{S} \neq \emptyset$ repita
   - 2.1. Calcule $D(v)$ para todos os vértices $v \in \overline{S}$ que são extremos de arestas de $\partial(S)$.
   - 2.2. Tome $v^* \in (\overline{S})$ com $D(v^*)$ mínimo
   - 2.3. Acrescente v^* a S, tomando como caminho mínimo de s a v^* a concatenação do caminho mínimo de s a u e da aresta uv^*.

Árvore de Caminhos Mínimos

- É interessante notar que a união dos caminhos mínimos de s aos demais vértices é uma árvore geradora com raiz em s.

Árvore de Caminhos Mínimos

- É interessante notar que a união dos caminhos mínimos de s aos demais vértices é uma árvore geradora com raiz em s.
- Esta árvore é formada exatamente pelas arestas que conectam o vértice a ser inserido em S no passo 2.3 a um vértice de S.

Deixando o algoritmo mais esperto

- Quando adicionamos o vértice v^* ao conjunto S, podemos reaproveitar alguns valores computados para $D[u]$, onde $u \in \overline{S} \setminus \{v^*\}$. Note que apenas um vértice u que é adjacente ao vértice v^* pode ter o valor de $D[u]$ modificado, e isso apenas ocorre quando $dist[v^*] + w(v^*u) < D[u]$.

Deixando o algoritmo mais esperto

- Quando adicionamos o vértice v^* ao conjunto S, podemos reaproveitar alguns valores computados para $D[u]$, onde $u \in \overline{S} \setminus \{v^*\}$. Note que apenas um vértice u que é adjacente ao vértice v^* pode ter o valor de $D[u]$ modificado, e isso apenas ocorre quando $dist[v^*] + w(v^*u) < D[u]$.
  - Isso significa que todo vértice precisa ter um valor de $D[u]$ atribuído ao longo de toda execução: ∞

Deixando o algoritmo mais esperto

- Quando adicionamos o vértice v^* ao conjunto S, podemos reaproveitar alguns valores computados para $D[u]$, onde $u \in \overline{S} \setminus \{v^*\}$. Note que apenas um vértice u que é adjacente ao vértice v^* pode ter o valor de $D[u]$ modificado, e isso apenas ocorre quando $dist[v^*] + w(v^*u) < D[u]$.
  - Isso significa que todo vértice precisa ter um valor de $D[u]$ atribuído ao longo de toda execução: ∞
- Por causa do ∞, fica mais fácil encontrar o vértice v^*: podemos simplesmente procurar pelo vértice v em $\overline{S}$ com o menor valor de $D[v]$.

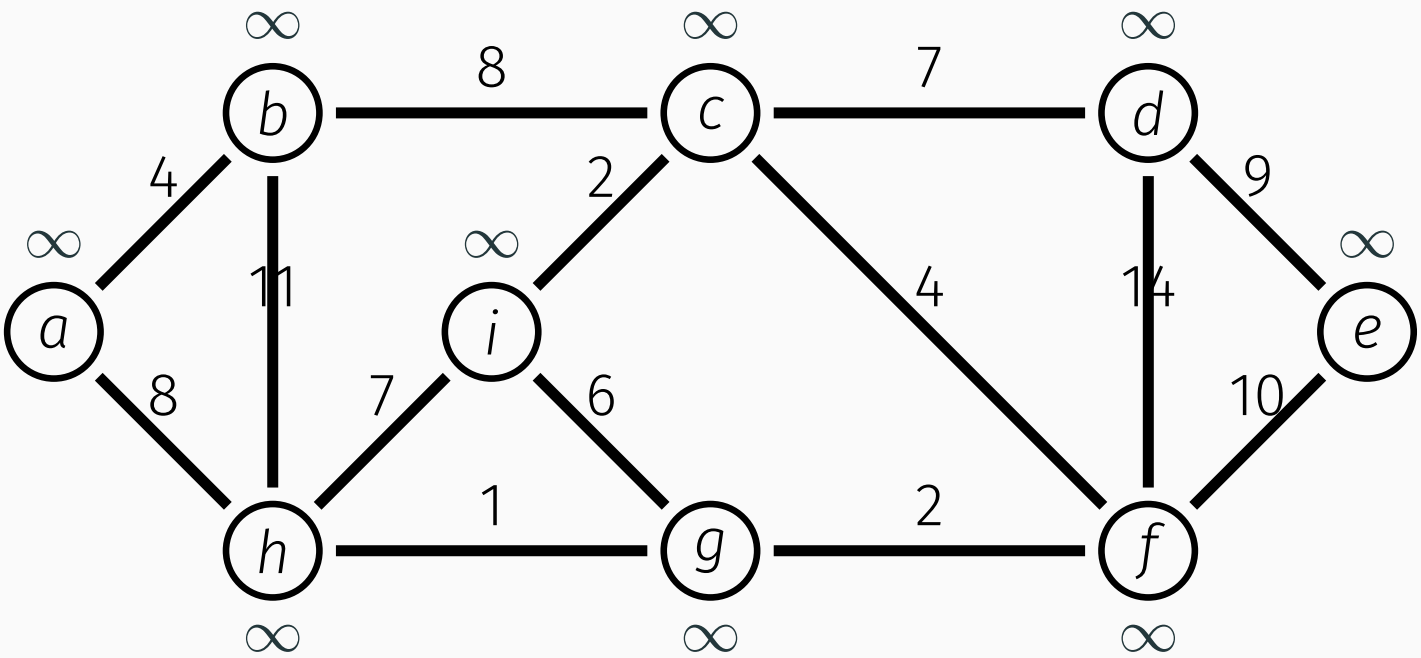
Deixando o algoritmo mais esperto

- Quando adicionamos o vértice v^* ao conjunto S, podemos reaproveitar alguns valores computados para $D[u]$, onde $u \in \overline{S} \setminus \{v^*\}$. Note que apenas um vértice u que é adjacente ao vértice v^* pode ter o valor de $D[u]$ modificado, e isso apenas ocorre quando $dist[v^*] + w(v^*u) < D[u]$.
  - Isso significa que todo vértice precisa ter um valor de $D[u]$ atribuído ao longo de toda execução: ∞
- Por causa do ∞, fica mais fácil encontrar o vértice v^*: podemos simplesmente procurar pelo vértice v em $\overline{S}$ com o menor valor de $D[v]$.
- Após adicionar v^* em S, temos que $\mathrm{dist}[v^*] = D[v^*]$, então não precisamos de uma estrutura adicional para armazenar $\mathrm{dist}[v^*]$.

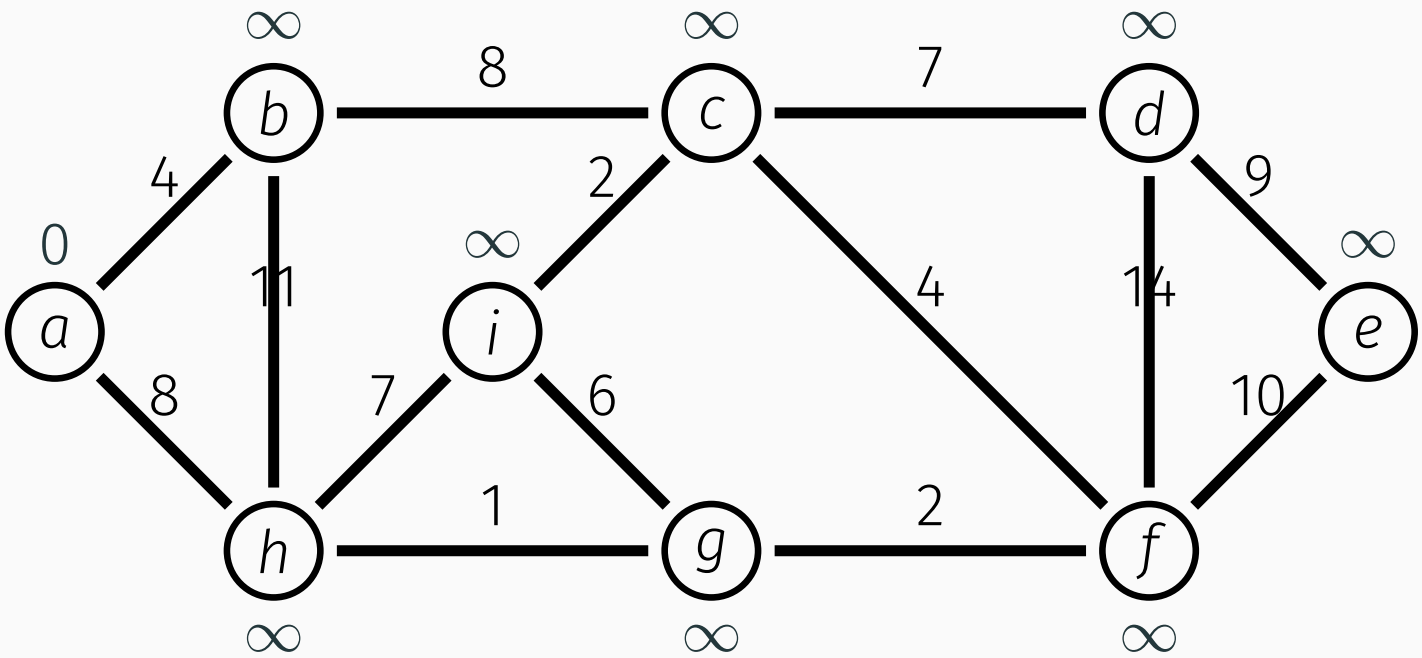
Algoritmo de Dijkstra

1: **Função** DIJKSTRA(G, w, s)
$\triangleright$ G é um grafo, $w\colon E(G) \to \mathbb{R}^+$, e $s \in V(G)$
2: **Para cada** $v \in V(G)$ **faça**
3: $D[v] \leftarrow \infty$
4: $pred[v] \leftarrow null$
5: $D[s] \leftarrow 0$
6: $S \leftarrow \emptyset$
7: **Enquanto** $\overline{S} \neq \emptyset$ **faça**
8: Seja $v^* \in \overline{S}$ tal $D[v^*]$ seja mínimo
9: $S \leftarrow S \cup \{v^*\}$
10: **Para cada** $u \in N(v^*)$ **faça**
11: **Se** $D[v^*] + w(v^*u) < D[u]$ **então**
12: $D[u] = D[v^*] + w(v^*u)$
13: $pred[u] = v^*$

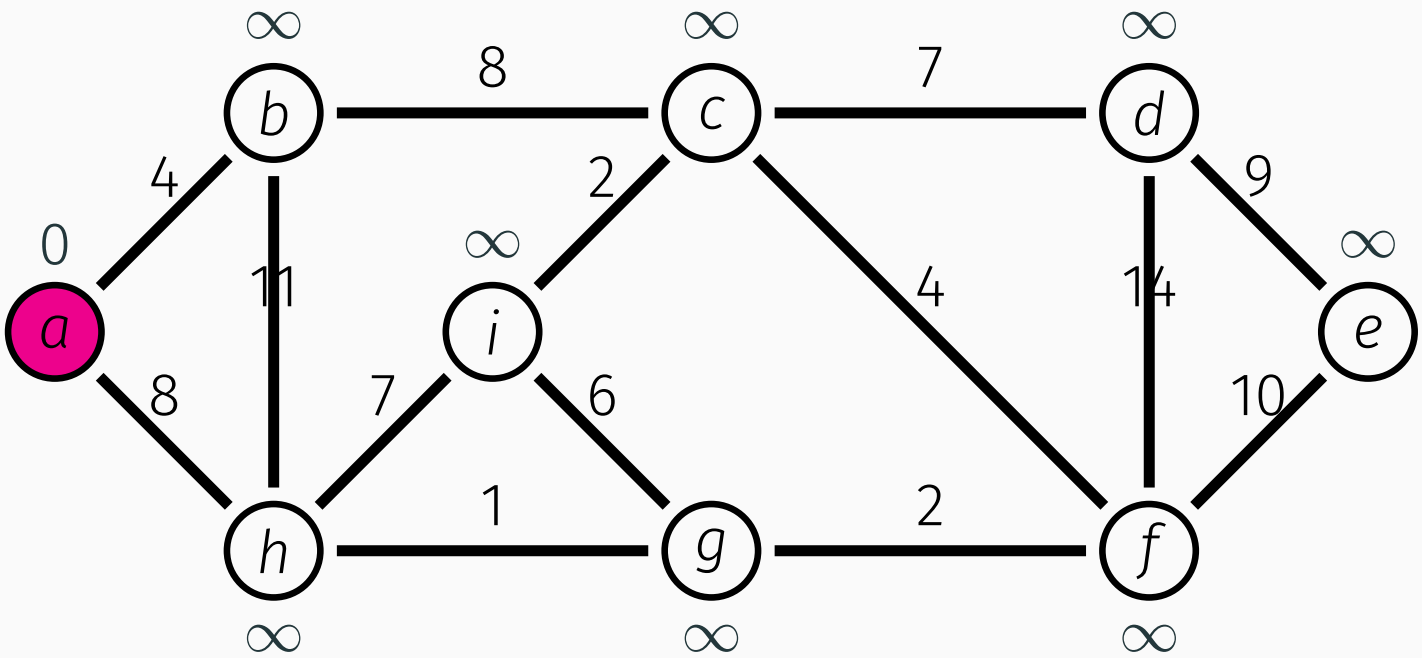
O algoritmo de Dijkstra: exemplo



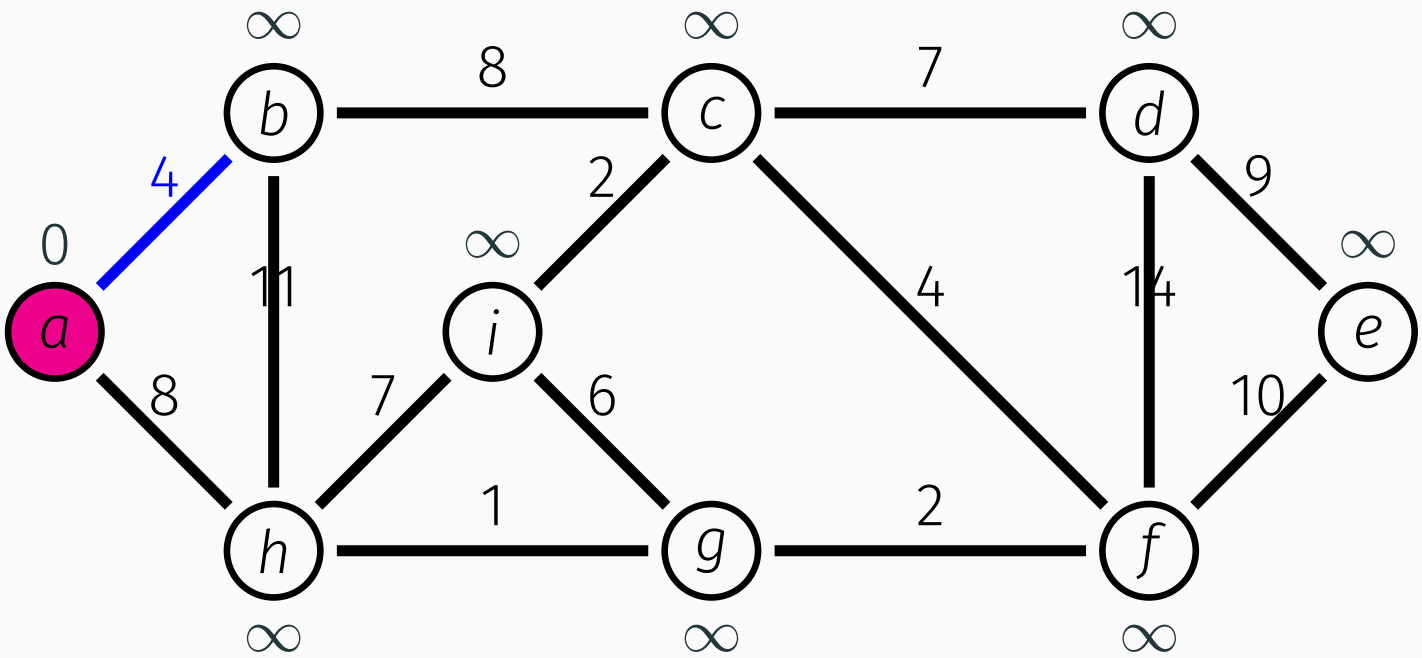
O algoritmo de Dijkstra: exemplo



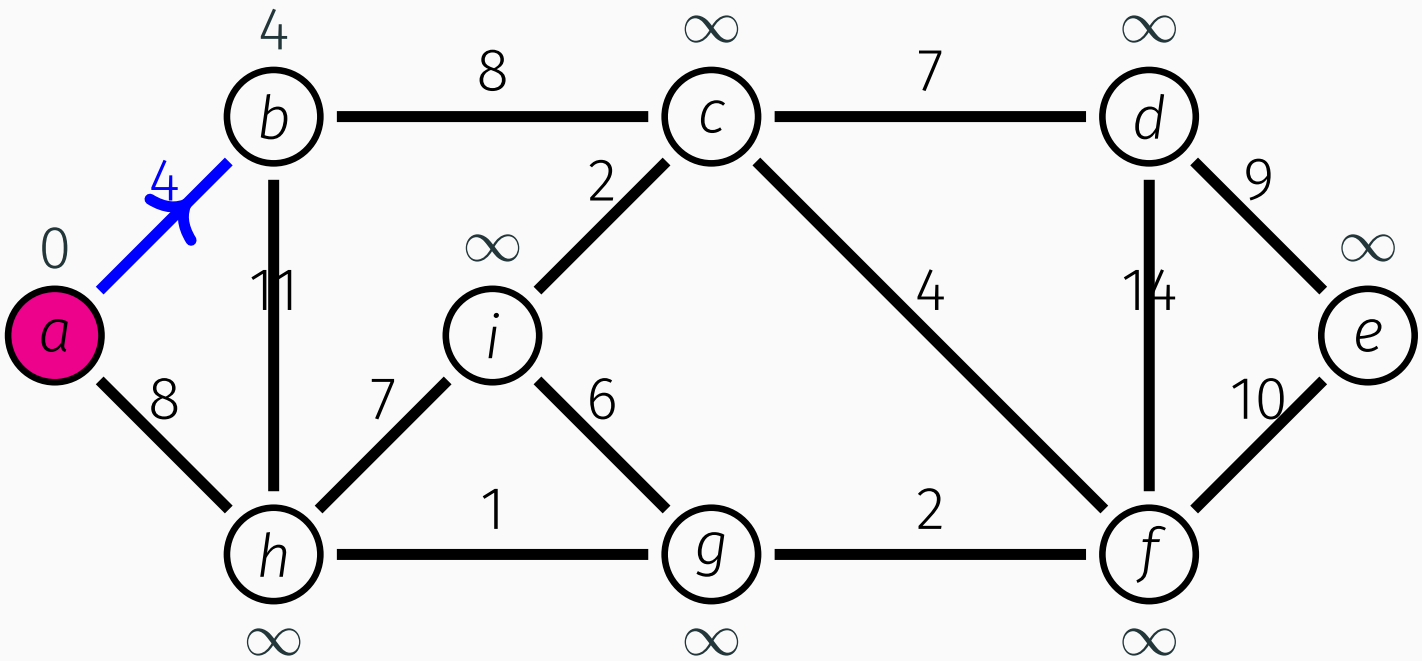
O algoritmo de Dijkstra: exemplo



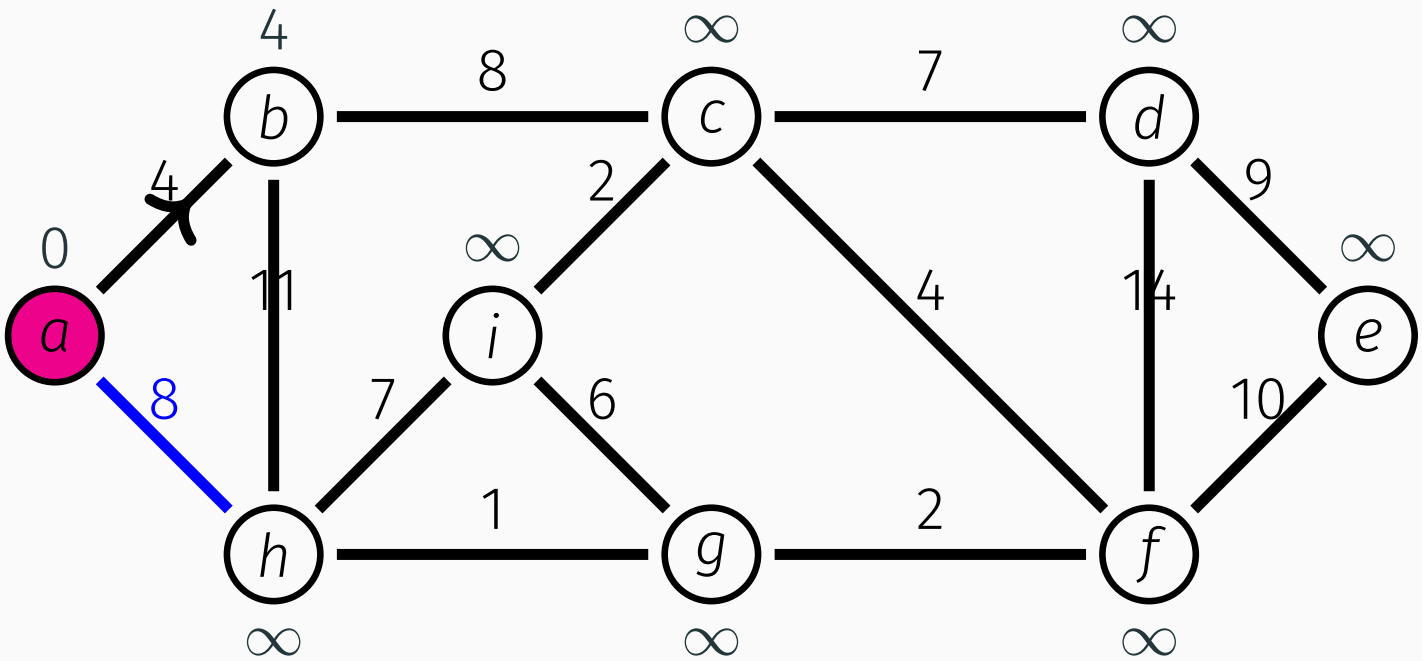
O algoritmo de Dijkstra: exemplo



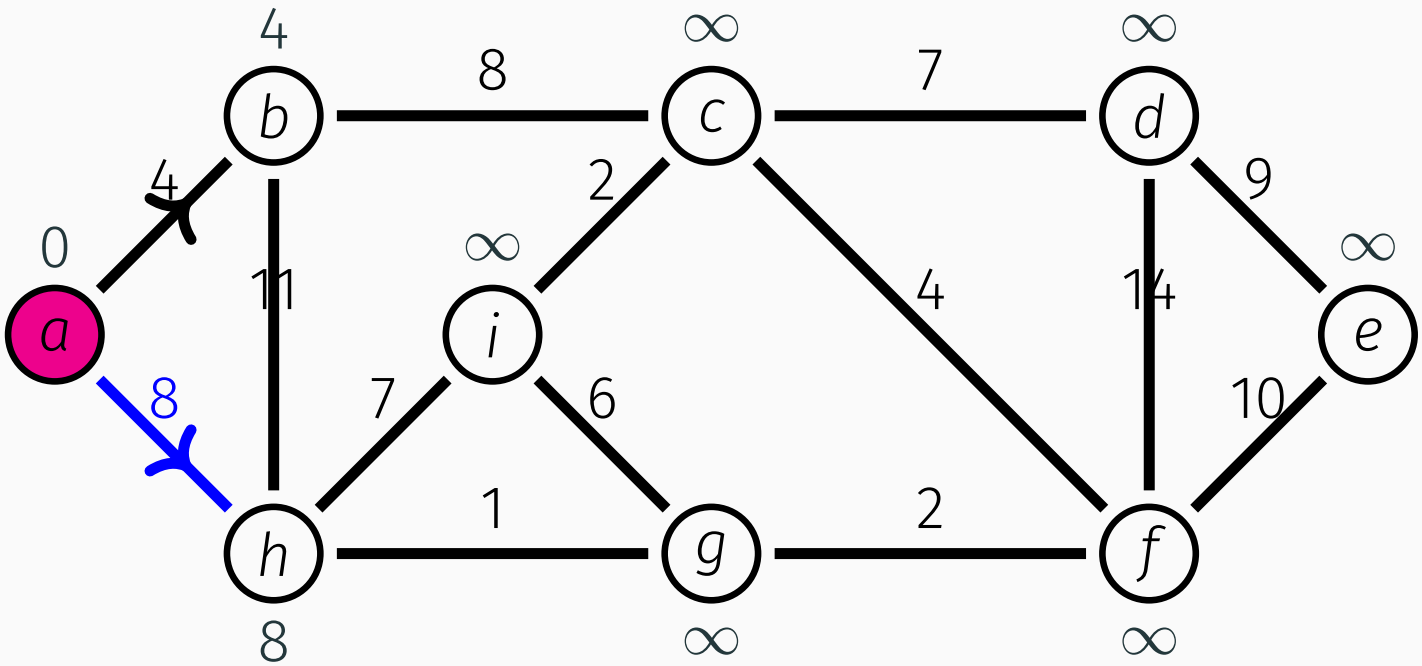
O algoritmo de Dijkstra: exemplo



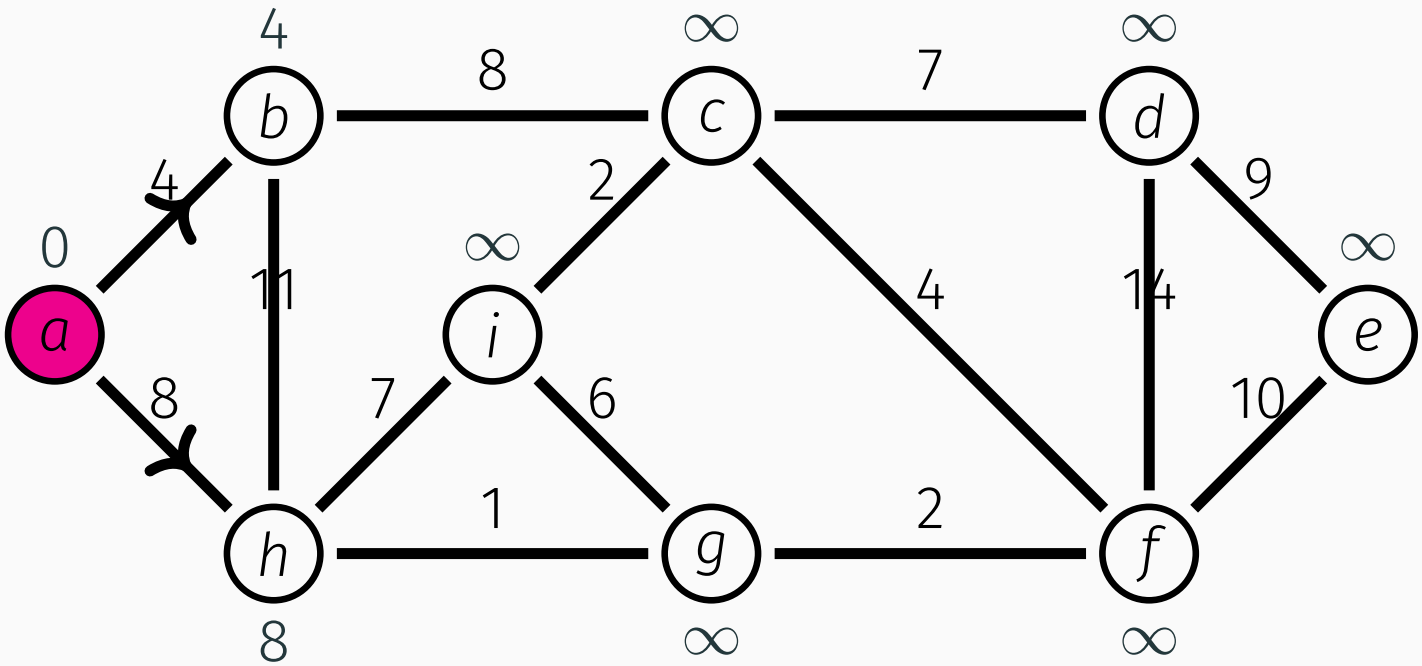
O algoritmo de Dijkstra: exemplo



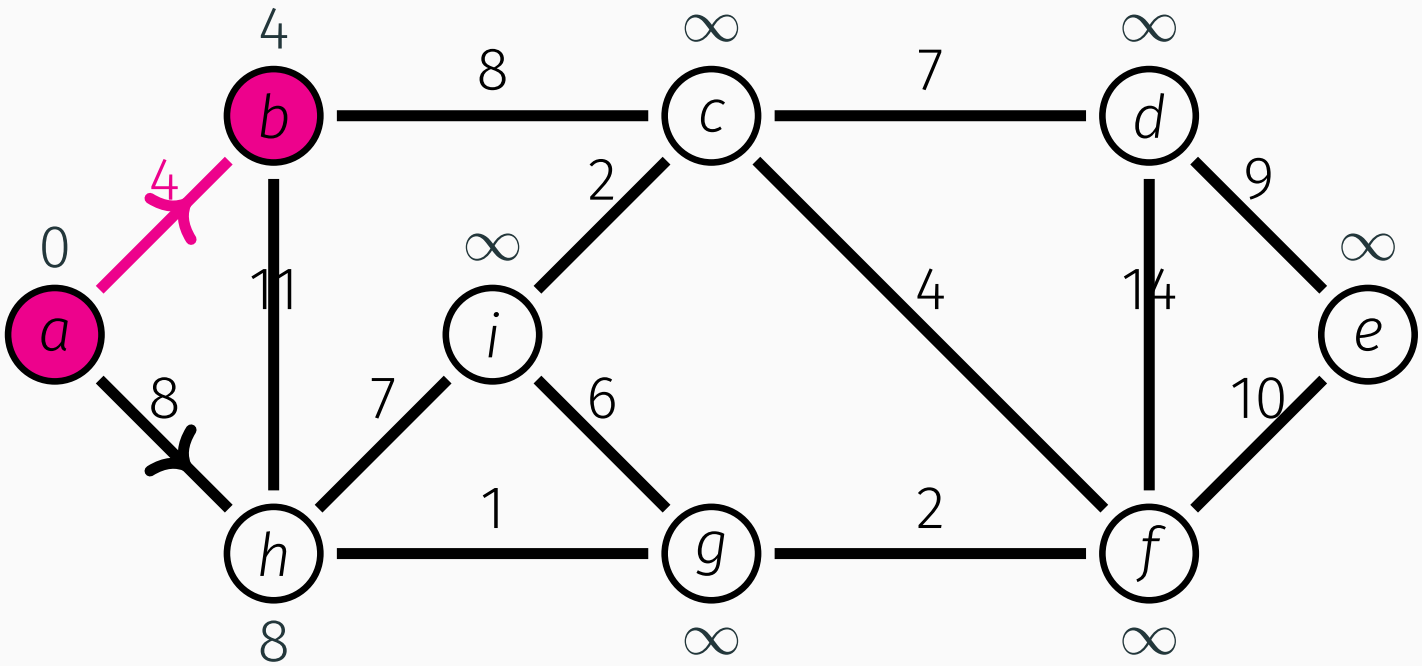
O algoritmo de Dijkstra: exemplo



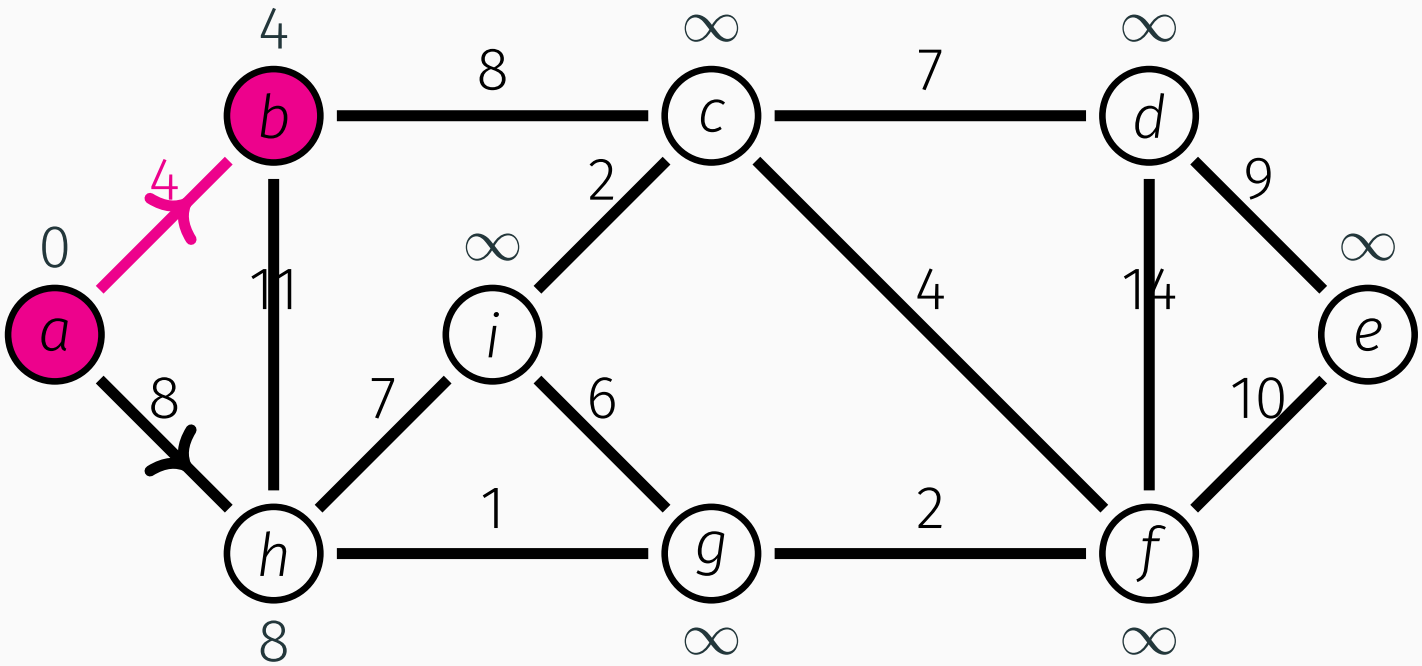
O algoritmo de Dijkstra: exemplo



O algoritmo de Dijkstra: exemplo



O algoritmo de Dijkstra: exemplo



Algoritmo de Dijkstra - Complexidade

- Se usarmos vetores simples para registrar os elementos de $\overline{S}$ e suas distâncias mínimas conhecidas até o momento, a complexidade do algoritmo será $O(|V|^2)$.

Algoritmo de Dijkstra - Complexidade

- Se usarmos vetores simples para registrar os elementos de $\overline{S}$ e suas distâncias mínimas conhecidas até o momento, a complexidade do algoritmo será $O(|V|^2)$.
- Se usarmos uma fila de prioridades (heap) para registrar os elementos de $\overline{S}$ e suas distâncias mínimas conhecidas até o momento, a complexidade do algoritmo será $O(|E| \log |V|)$.

Algoritmo de Dijkstra - Complexidade

- Se usarmos vetores simples para registrar os elementos de $\overline{S}$ e suas distâncias mínimas conhecidas até o momento, a complexidade do algoritmo será $O(|V|^2)$.
- Se usarmos uma fila de prioridades (heap) para registrar os elementos de $\overline{S}$ e suas distâncias mínimas conhecidas até o momento, a complexidade do algoritmo será $O(|E| \log |V|)$.
- Portanto, para grafos *densos* ($|E| \in O(|V|^2)$), a melhor alternativa é usar um vetor simples, enquanto que para grafos *esparsos* ($|E| \in O(|V|)$), deve-se optar pela implementação com uma fila de prioridades.

- O algoritmo de Dijkstra funciona desde que não haja arestas de peso negativo. (Por quê ?)