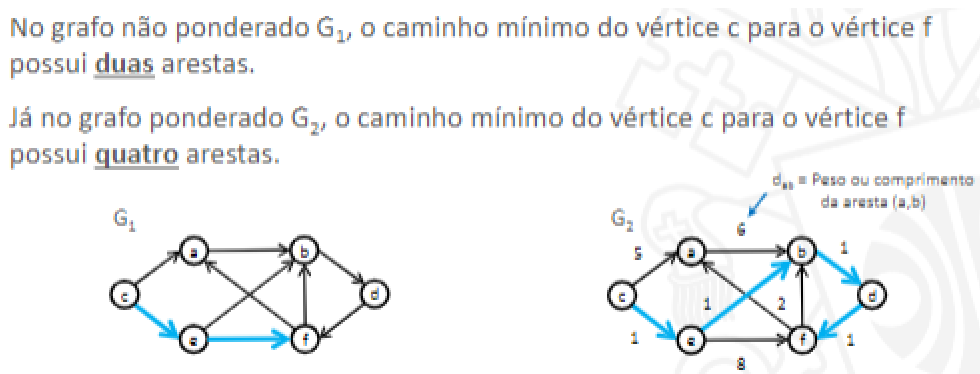
13 - Caminho mínimo - Dijkstra

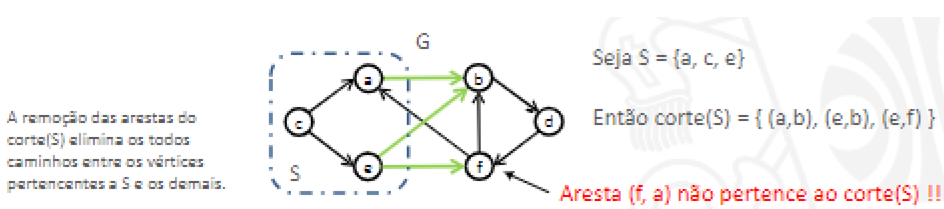
O caminho mais curto entre os vértices de um grafo não ponderado é aquele que possui o menor número de arestas e pode ser obtido por mio de uma busca em largura.

Já o caminho mais curto entre os vértices de um grafo ponderado é aquele cuja soma dos pesos das arestas possui o menor valor possível dentre todos os caminhos existentes. Por isso o menor caminho pode não ser aquele com o menor número de arestas.

Exemplo:



Dado um grafo direcionado e um subconjunto de vértice, um corte é um subconjunto das arestas com um extremo em S e o outro não, isto é, o conjunto de vértices S não pode ser vazio nem conter todos os vértices.



Método de Dijkstra:

Busca o menor caminho entre um vértice de origem e os demais. Inicialmente um conjunto contendo somente a raiz é usado para se definir um corte. A cada iteração, calcula-se uma estimativa da distância entre os vértices já explorados e seus vizinhos não explorados (analisar apenas as arestas do corte). A aresta fornecer a menor estimativa é selecionada e seu extremo não selecionado passa a ser explorado e sua distância é atualizada.

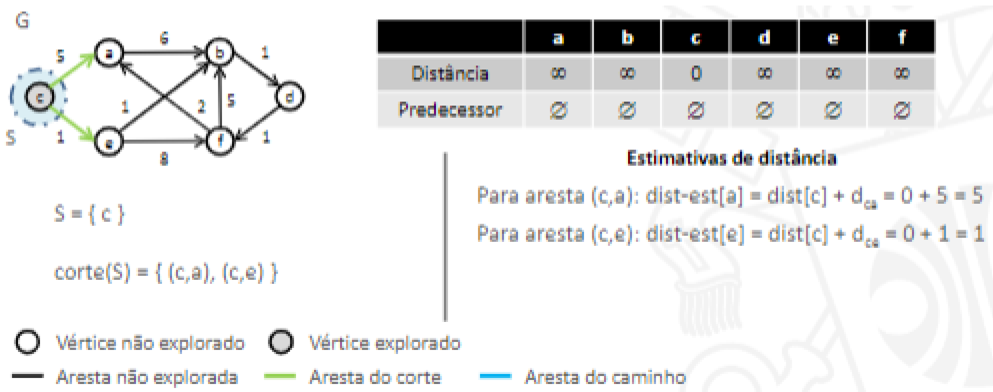
Durante o cálculo dos caminhos, alguns atributos são definidos para os vértices:

- Inicialmente a raiz está presente no conjunto S de vértices selecionados e que define um corte.
- Cada um dos cortes restantes é inicialmente não explorado (ou branco) e se torna explorado (ou cinza) quando for selecionado e inserido no conjunto S.
- Cada vértice tem dois valores associados a ele, a distância (comprimento da raiz até ele) e predecessor (vértice pelo qual passa até chegar a ele).
- Inicialmente, todos os predecessores são definidos como vazio, enquanto os valores de distância são infinitos (exceto para raiz que é zero).
- A cada iteração pode usar o peso da aresta pertencente ao corte para estimar a distância da raiz até w $dist[v] + d$, onde d é o peso.
- Assim, seleciona-se a aresta cujo valor seja mínimo e se atualiza $dist[w]$.
- Uma vez selecionada uma aresta do corte, o caminho do vértice W deve passar pelo vértice V, então o predecessor do vértice W será V, ou ainda, $pred[w] = v$.

Algoritmo:

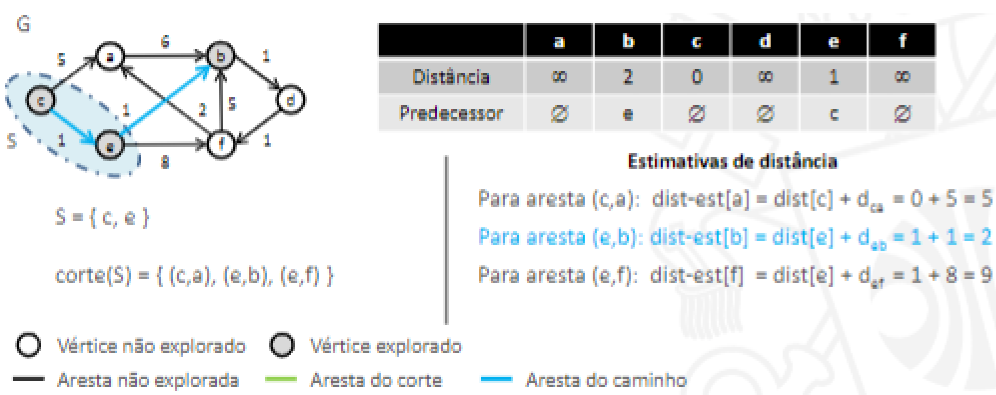
```
1. para todo vértice v ∈ V(G) faça
   a. dist[v] ← ∞;   pred[v] ← ∅;            // Inicializar distâncias e predecessores
2. S ← { s };                                 // Inserir raiz no conj. de explorados
3. dist[s] ← 0;                               // Fazer distância da raiz igual a zero
4. para  i = 1, . . ., |V(G)| − 1  faça
   a. Encontrar a aresta (v, w) ∈ corte(S) tal que dist[v] + d_vw seja mínimo
         // Isto é, aquela que representa a menor distância para um vértice não explorado
   b. dist[w] ← dist[v] + d_vw; pred[w] ← v;  // Atualizar a distância/predecessor de w
   c. S ← S ∪ { w };                          // Adicionar vértice w ao conj. explorados
```

Inicialização:

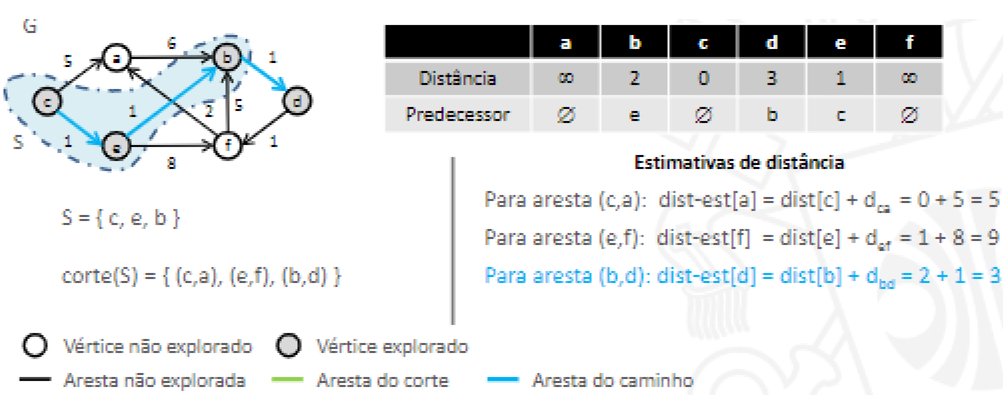


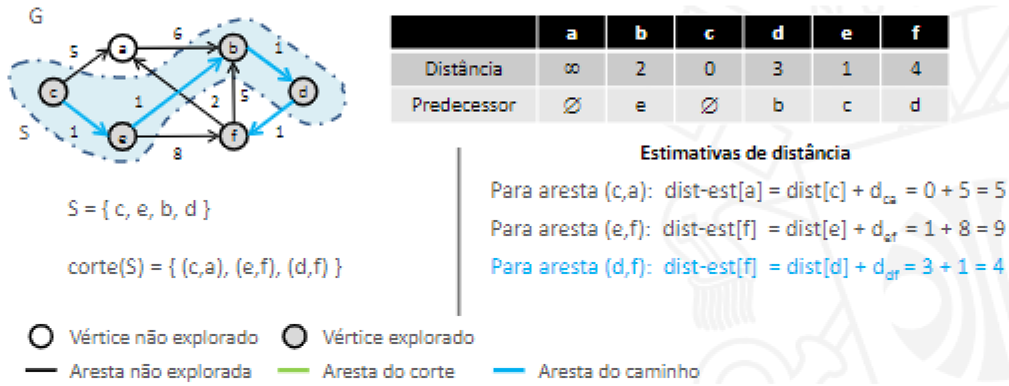
Inicializa o vértice raiz e olha todas as arestas incidentes a ele, assim analisando as distâncias escolhendo a de menor distância.

Após a inicialização:



Após escolher a aresta de menor distância, adiciona-se assim o vértice descoberto, tendo um conjunto e um aumento no corte de vértices, fazendo assim calcular novamente o menor caminho e escolher a aresta.

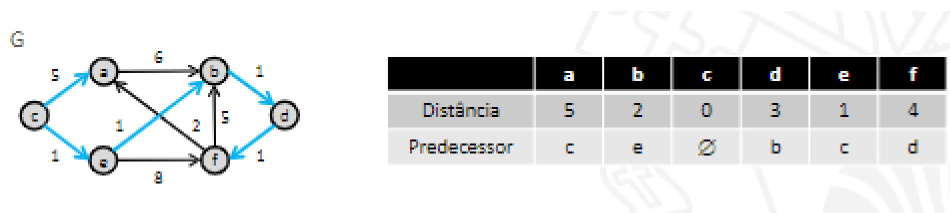




	a	b	c	d	e	f
Distância	∞	2	0	3	1	4
Predecessor	∅	e	∅	b	c	d

	a	b	c	d	e	f
Distância	5	2	0	3	1	4
Predecessor	c	e	∅	b	c	d

<u>Não tendo mais arestas de corte:</u>



	a	b	c	d	e	f
Distância	5	2	0	3	1	4
Predecessor	c	e	∅	b	c	d

Resultado do menor caminho para todos os vértices.

Potencial:

Um potencial para um grafo é uma numeração dos vértice, ou seja, um vetor que associa um número a cada vértice.

Em relação a um potencial, dizemos que:

- Arco está tenso → $h[w] - h[v] > c$
- Arco está relaxado → $h[w] - h[v] <= c$

- Arco está justo → $h[w] - h[v] = c$ (c é o custo do arco)

Em outras palavras, o arco (v, w) está:

- Tenso → $h[v] + c < h[w]$
- Relaxado → $h[v] + c >= h[w]$
- Justo → $h[v] + c = h[w]$

Potencial x Condições de minimalidade:

Um potencial é relaxado se todos os arcos de G estão relaxados em relação a ele. Qualquer potencial relaxado dá uma cota inferior para as distâncias entre vértices: se P é um caminho de um vértice X a um vértice Y então: $custo(P) >= h[y] - h[x]$, sendo $custo(P)$ o custo do caminho P.

Condição suficiente de minimalidade: para qualquer caminho de um vértice X a um vértice Y, se existe um potencial relaxado tal que $h[y] - h[x] = custo(P)$ então P é mínimo e portanto a diferença $h[y] - h[x]$ é a distância entre X e Y.

Suponha que P seja um caminho 0-1-2-3 então custo(P) = $d_{23} + d_{12} + d_{01}$. Para um potencial relaxado h[ ],

$$custo(P) = d_{23} + d_{12} + d_{01} \geq (h[3] - h[2]) + (h[2] - h[1]) + (h[1] - h[0])$$

$$custo(P) \geq h[3] - h[0]$$

Em particular, se P for um ciclo então custo(P) ≥ 0.

A recíproca da condição anterior é verdadeira se for restrita aos caminhos mínimos que têm uma origem comum (desde que todos os vértices sejam alcançáveis a partir da raiz s), isto é: Para qualquer vértice s, o vetor das distâncias a partir de s é um potencial relaxado.

Potencial x Método de Dijkstra:

Um vértice de um grafo é considerado maduro, se seu conjunto de sucessores já foi examinado, caso contrário, ele e considerado imaturo. Todo vértice maduro pertence a árvore de caminhos mínimos.

No inicio do processo, todos os vértices são imaturos, $dist[s] = 0$ para a raiz s, $dist[v] = infinito$ para todo V diferente de S e os predecessores est~~ao indefinidos.

O processo iterativo consiste no seguinte:

enquanto existir vértice imaturo faça

1. Selecionar o vértice imaturo *v* que minimiza dist[ ]
2. para cada arco (*v*, *w*) que está tenso faça
   - i. dist[*w*] ← dist[*v*] + d_{vw} // Relaxação da aresta
   - ii. pred[*w*] ← *v*
3. Declarar *v* como maduro