

BCC204 - Teoria dos Grafos

Marco Antonio M. Carvalho

(baseado nas notas de aula do prof. Haroldo Gambini Santos)

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto



1 Introdução

2 Algoritmo de Dijkstra

Fonte

Este material é baseado no livro

- ▶ Goldbarg, M., & Goldbarg, E. (2012). *Grafos: conceitos, algoritmos e aplicações*. Elsevier.

Licença

Este material está licenciado sob a Creative Commons BY-NC-SA 4.0. Isto significa que o material pode ser compartilhado e adaptado, desde que seja atribuído o devido crédito, que o material não seja utilizado de forma comercial e que o material resultante seja distribuído de acordo com a mesma licença.

Caminho Mais Curto – Grafo Não Ponderado

O **caminho mais curto** entre os vértices v e w de um determinado grafo não ponderado é aquele que possui o menor número de arestas entre os referidos vértices.

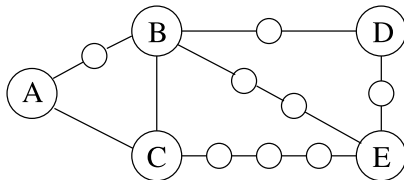
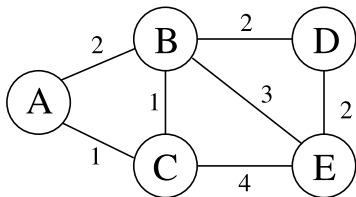
Pode ser obtido pela aplicação da BFS.

Caminho Mais Curto – Grafo Ponderado

O **caminho mais curto** entre os vértices v e w de um determinado grafo ponderado é aquele cuja soma dos pesos das arestas possui o menor valor possível dentre todos os caminhos existentes entre os referidos vértices.

Claramente, em grafos ponderados, o menor caminho pode não ser aquele com menor número de arestas.

Usando BFS em Grafos Ponderados



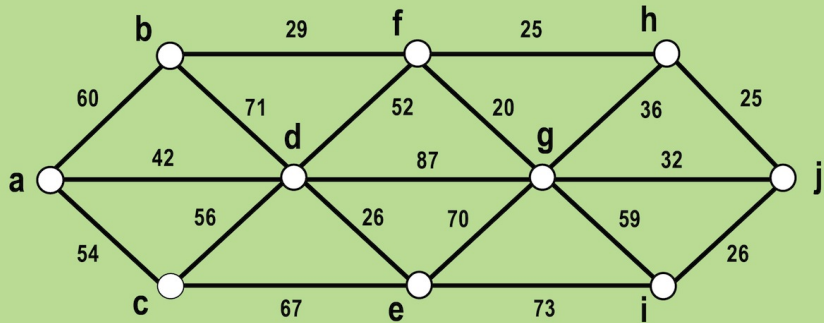
Complexidade

Em um grafo ponderado, se o peso das arestas for inteiro e limitado por um número k , então a complexidade é limitada por $O(km)$.

Algoritmo Guloso

Rápido e Rasteiro! – *Quick and Dirty!*

A cada instante, selecione a aresta de menor peso.



Princípio

O algoritmo, proposto em 1959, **rotula** os vértices durante a exploração de um grafo (orientado ou não), para encontrar o menor caminho entre um vértice de origem e todos os demais vértices

- ▶ Grafos ponderados somente com pesos positivos;
- ▶ Estruturalmente semelhante à BFS
 - ▶ Calcula a menor distância do vértice inicial aos seus vizinhos;
 - ▶ Calcula a menor distância dos vizinhos do vértice inicial aos seus próprios vizinhos;
 - ▶ E assim sucessivamente...
 - ▶ Noção de camadas;
 - ▶ Atualiza as distâncias sempre que descobre uma menor.
- ▶ Pode ser provado por indução!

Terminologia

- ▶ Um vértice é dito **fechado** caso o caminho mínimo da origem até ele já tenha sido calculado;
- ▶ Caso contrário, o vértice é considerado **aberto**;
- ▶ F : Conjunto de vértices fechados;
- ▶ A : Conjunto de vértices abertos;
- ▶ N : Conjunto de vértices vizinhos ao vértice atual;
- ▶ $dt[i]$: Vetor que armazena a distância entre o vértice de origem e o vértice i ;
- ▶ $rot[i]$: Vetor que armazena o índice do vértice anterior ao vértice i , no caminho cuja distância está armazenada em $dt[i]$;
- ▶ \setminus : subtração em conjuntos.

Algoritmo de Dijkstra

Entrada: Grafo $G = (V, E)$ e matriz de pesos $D = \{d_{ij}\}$ para todas as arestas $\{i, j\}$

```
1   $dt[1] \leftarrow 0$ ; //considerando o vértice 1 como o inicial
2   $rot[1] \leftarrow 0$ ;
3  para  $i \leftarrow 2$  até  $n$  faça
4       $dt[i] \leftarrow \infty$ ;
5       $rot[i] \leftarrow 0$ ;
6   $A \leftarrow V$ ;
7   $F \leftarrow \emptyset$ ;
8  enquanto  $F \neq V$  faça
9       $v \leftarrow$  elemento de  $A$ , tal que  $dt[v]$  é o mínimo dentre os elementos de  $A$ ;
10      $F \leftarrow F \cup \{v\}$ ;
11      $A \leftarrow A \setminus \{v\}$ ;
12      $N \leftarrow N \setminus F$ ; //conjunto de vizinhos do vértice  $v$  menos os vértices já fechados
13     para  $u \in N$  faça
14         se  $dt[v] + d_{vu} < dt[u]$  então
15              $dt[u] \leftarrow dt[v] + d_{vu}$ ;
16              $rot[u] \leftarrow v$ ;
```

Algoritmo de Dijkstra

Complexidade

- ▶ O laço **enquanto** da linha 8 é repetido $O(n)$ vezes;
- ▶ Usando estruturas simples, examinar o conjunto A no pior caso pode exigir $O(n)$ comparações;
- ▶ Caso o conjunto N seja grande, pode ser necessário atualizar $O(n)$ vértices.
- ▶ Logo, em uma implementação simples, a complexidade é $O(n^2)$.

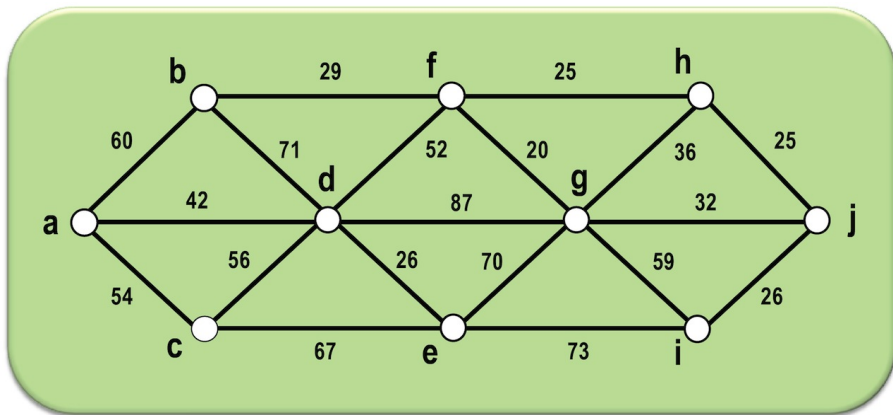
Alternativa

Se utilizarmos um *heap* e listas de adjacências para representar o grafo, a complexidade cai para $O((n + m)\log m)$, porque determinar o menor elemento e atualizar o *heap* pode ser feito em tempo logarítmico.

Recorde

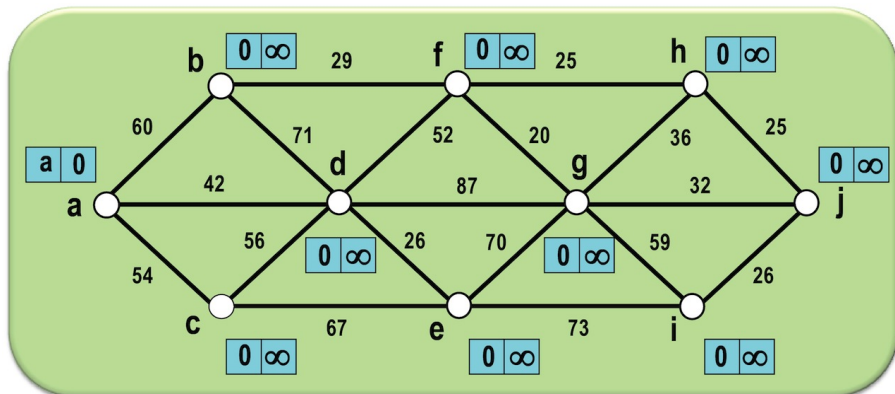
A melhor implementação do algoritmo, do ano 2000, possui complexidade $O(n\log\log m)$.

Dijkstra – Exemplo



Grafo G. O vértice inicial será 'a'.

Dijkstra – Exemplo

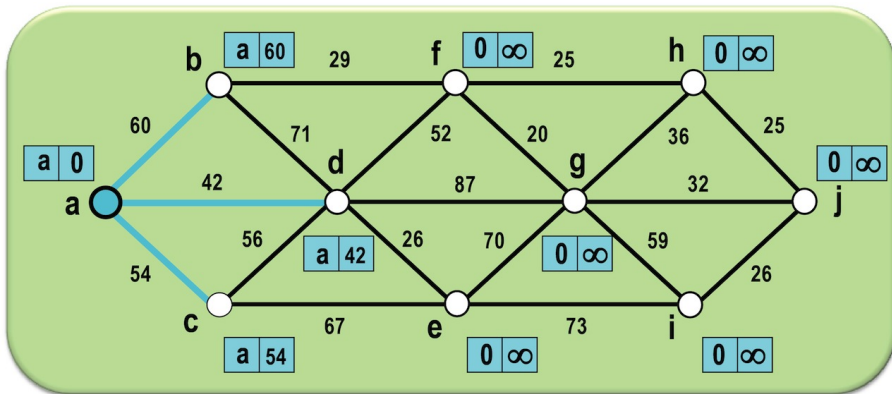


Rotulação após a primeira iteração do algoritmo.

O primeiro número é $rot[i]$ e o segundo, $dt[i]$.

Os vértices de F serão marcados em azul.

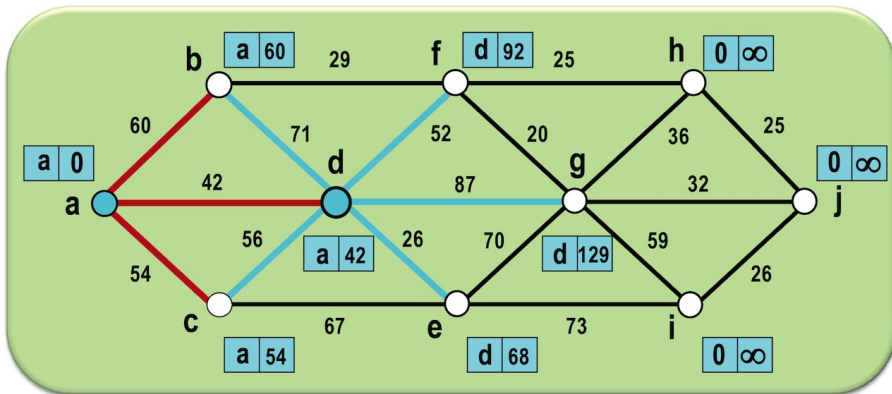
Dijkstra – Exemplo



Exame do vértice a .

$rot[b]$, $dt[b]$, $rot[c]$, $dt[c]$, $rot[d]$ e $dt[d]$ atualizados.

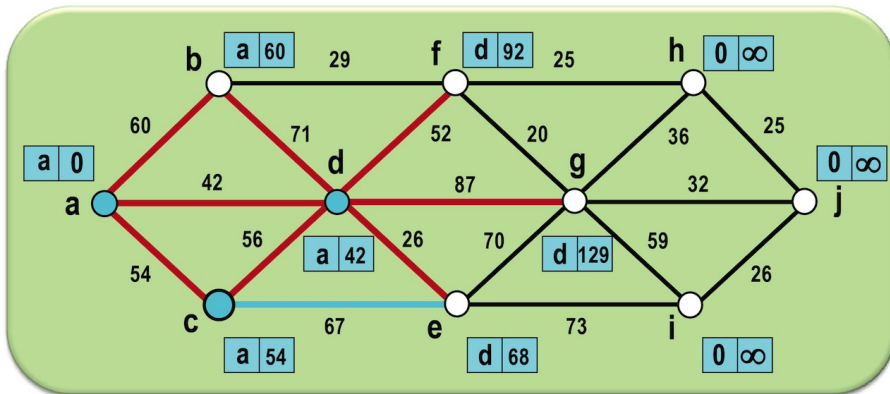
Dijkstra – Exemplo



Exame do vértice d .

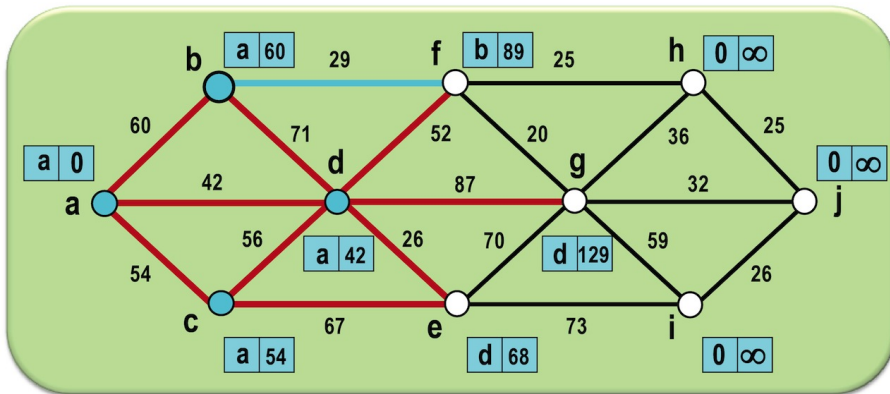
$rot[e]$, $dt[e]$, $rot[f]$, $dt[f]$, $rot[g]$ e $dt[g]$ atualizados.

Dijkstra – Exemplo



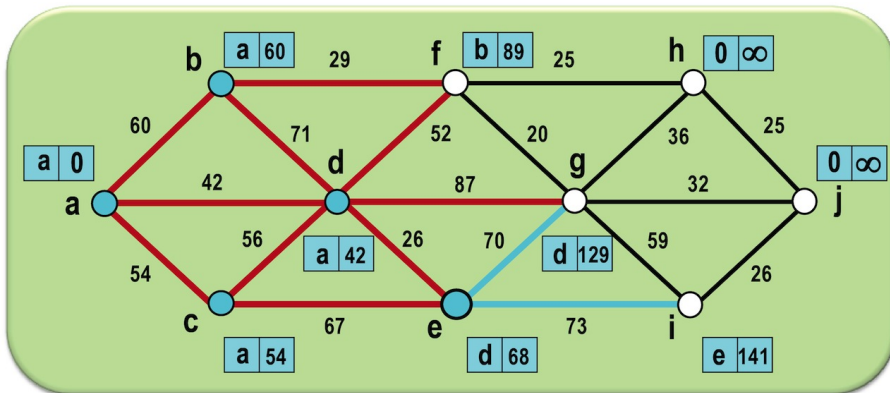
Exame do vértice c.
 $rot[e]$ e $dt[e]$ não são atualizados.

Dijkstra – Exemplo



Exame do vértice b .
 $rot[f]$ e $dt[f]$ são atualizados.

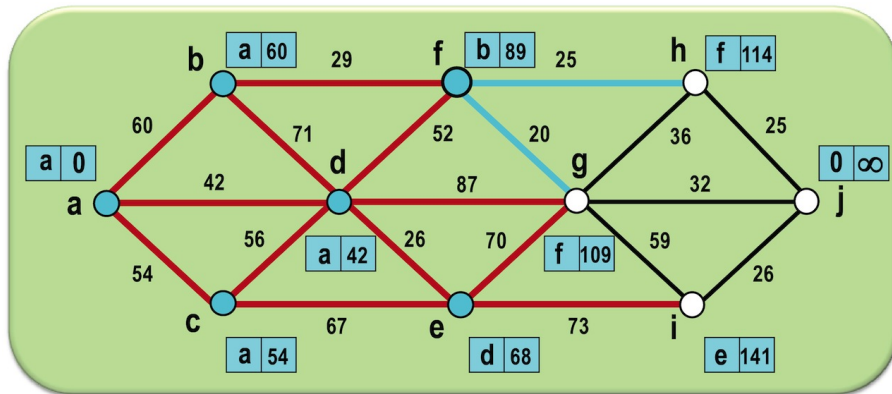
Dijkstra – Exemplo



Exame do vértice e.

Apenas $rot[i]$ e $dt[i]$ são atualizados.

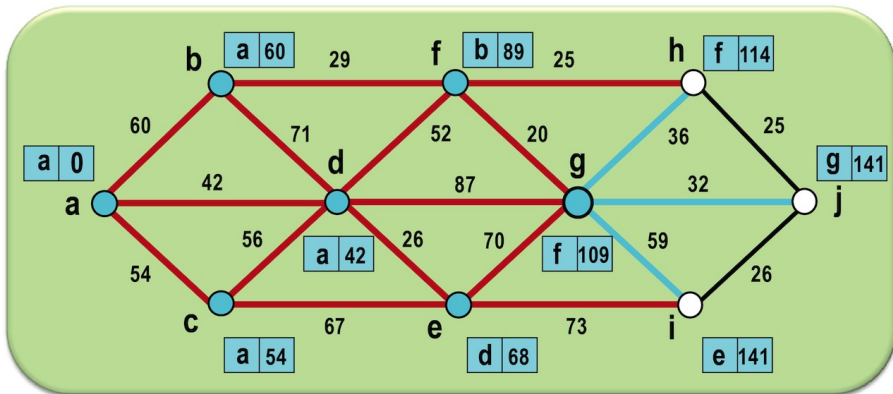
Dijkstra – Exemplo



Exame do vértice f .

$rot[g]$, $dt[g]$, $rot[h]$ e $dt[h]$ atualizados.

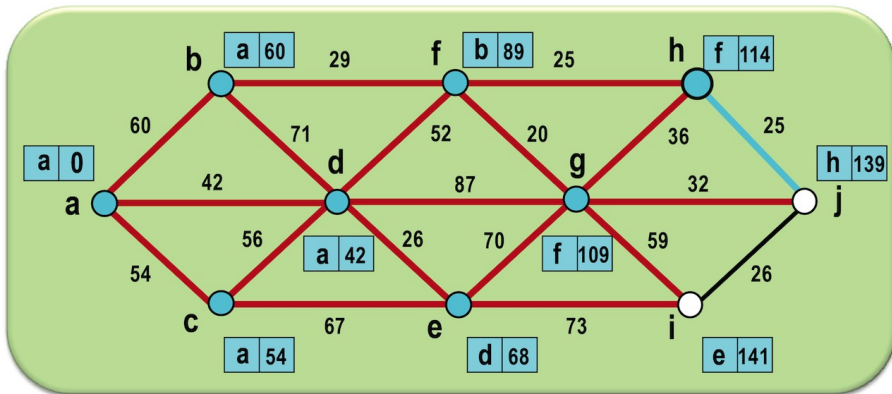
Dijkstra – Exemplo



Exame do vértice g .

Apenas $rot[j]$ e $dt[j]$ são atualizados, pois os outros caminhos são mais curtos.

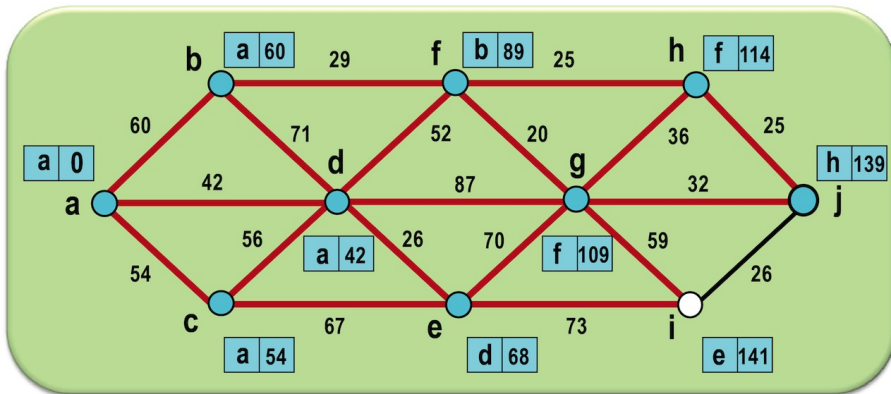
Dijkstra – Exemplo



Exame do vértice h .

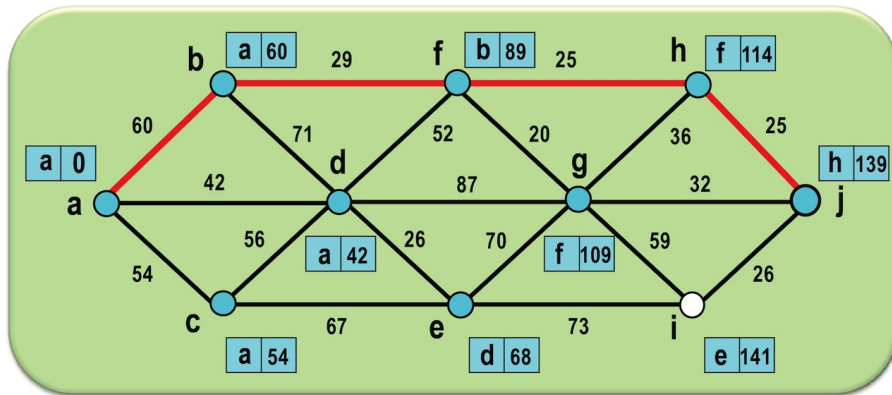
$rot[j]$ e $dt[j]$ são atualizados, pois o novo caminho é mais curto.

Dijkstra – Exemplo



Exame do vértice j .
Nenhuma atualização.

Dijkstra – Exemplo



Caminho mais curto entre *a* e *j*.

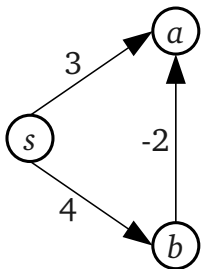
Crítica

O algoritmo é incapaz de calcular os caminhos mínimos caso existam arestas com custo negativo.

O algoritmo só calcula os caminhos mínimos a partir de uma única origem.

Para calcular os caminhos mínimos de todos os vértices para todos os vértices, o algoritmo deve ser executado uma vez para cada vértice do grafo, com complexidade total $O(n^3)$ na implementação simples.

Dijkstra - Limitação



- ▶ As instâncias devem obedecer à *desigualdade triangular*.
- ▶ No algoritmo, qualquer caminho de s para outro vértice v deve passar apenas por **vértices mais próximos** de s .
- ▶ No exemplo, o caminho mais curto entre s e a passa por b , que é mais distante do que a !

Dúvidas?

