PCC173/BCC463 - Otimização em Redes

Marco Antonio M. Carvalho

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

Conteúdo

Aviso

Fonte

Este material é baseado nos livros

- Goldbarg, M., & Goldbarg, E. (2012). *Grafos: conceitos, algoritmos e aplicações*. Elsevier.
- Goldbarg, M. C., & Luna, H. P. L. (2005). *Otimização combinatória e programação linear: modelos e algoritmos*. Elsevier.

Licença

Single-Source Shortest Path Problem

Definição

Dado um grafo direcionado $G = (V, A)$ e ponderado, em que $w(u, v)$ denota o peso do arco (u, v), consiste em determinar o caminho mais curto d_v entre um vértice de origem s e todos os demais vértices v do grafo.

Algoritmos: Dijkstra, Bellman-Ford, Busca Em Largura (GND), etc.

Single-Source Shortest Path Problem

Formulação PLI

As variáveis d_v indicam a menor distância entre o vértice de origem s e o vértice v.

Os valores c_{ij} indicam o custo do arco que liga os vértices i e j.

A função objetivo visa minimizar o custo dos caminhos.

As restrições são relacionadas ao comprimento dos caminhos: eles devem ter seu comprimento aumentado o mínimo possível.

O domínio das variáveis depende da aplicação (inteiras positivas, inteiras, contínuas positivas, contínuas, etc.).

Single-Source Shortest Path Problem

Formulação PLI

Uma maneira simples de interpretar o modelo a seguir é:

- Considere que o grafo é maleável e que as arestas possuem comprimento de acordo com o custo;
- Considere que o grafo está preso ao chão pelo vértice s;
- Comece a esticar o grafo;
- Em alguns pontos, será necessário parar e fixar alguns vértices no chão, porque senão algumas arestas irão arrebentar;
- Estes pontos são exatamente os menores caminhos.

Single-Source Shortest Path Problem

Formulação PLI

$$max \sum_{v \in V} d_v \tag{1}$$

$$sujeito\ a:$$

$$d_v - d_u \leq c(u,v), \forall (u,v) \in A \tag{2}$$

$$d_s = 0 \tag{3}$$

Formulação PLI

Na restrição (2), o maior valor de d_v que satisfaz a desigualdade é na verdade aquele que satisfaz $d_v = min_u(d_u + c_{uv})$, que é exatamente a definição de caminho mais curto em um grafo direcionado.

Este PL possui V variáveis e E restrições e pode ser resolvido em tempo $O(V^2E)$ em casos típicos e em tempo $O(V^8)$ usando o método da elipsóide no pior caso.

Ambos tempos de execução são piores do que os algoritmos especializados, como Dijkstra e Bellman-Ford.

O Algoritmo de Dijkstra

Dijkstra

Edsger W. Dijkstra $\star$ 1930 – $\dagger$ 2002

1959 - Algoritmo de Dijkstra para Caminhos Mínimos

Algoritmo de Dijkstra

Princípio

O algoritmo 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!

Algoritmo de Dijkstra

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 $G = (V, E)$, matriz de pesos D=$\{d_{ij}\}$ ($\{i, j\} \in A$)

```
1  dt[1] ← 0;
2  rot[1] ← 1;
3  para i ← 2 até n faça
4  |    dt[i] ← ∞;
5  |    rot[i] ← 0;
6  fim
7  A ← V;
8  F ← ∅;
9  enquanto F ≠ V faça
10 |    r ← j ∈ A, tal que dt[j] é o mínimo dentre os elementos de A;
11 |    F ← F ∪ {r};
12 |    A ← A \ {r};
13 |    N ← N \ F;
14 |    para i ∈ N faça
15 |    |    p ← min{dt[i], (dt[r]+d_ri)};
16 |    |    se p < dt[i] então
17 |    |    |    dt[i] ← p;
18 |    |    |    rot[i] ← r;
19 |    |    fim
20 |    fim
21 fim
```

Algoritmo de Dijkstra - Grafos Direcionados Arbitrários e Pesos Positivos

Complexidade 1

- O laço **enquanto** da linha 9 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 $O(n^2)$.

Complexidade 2 - Grafos Direcionados Arbitrários e Pesos Positivos

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

---

[a] Fredman, Michael Lawrence; Tarjan, Robert E. (1984). Fibonacci heaps and their uses in improved network optimization algorithms. 25th Annual Symposium on Foundations of Computer Science. IEEE. pp. 338–346.

Algoritmo de Dijkstra

Casos Especiais (1)

Grafos direcionados com pesos inteiros limitados por um parâmetro C: $O(mloglogn)$.

Thorup, Mikkel (2000). "On RAM priority Queues". SIAM Journal on Computing. 30 (1): 86–109c.

Casos Especiais (2)

Grafos direcionados acíclicos: $O(m+n)$.

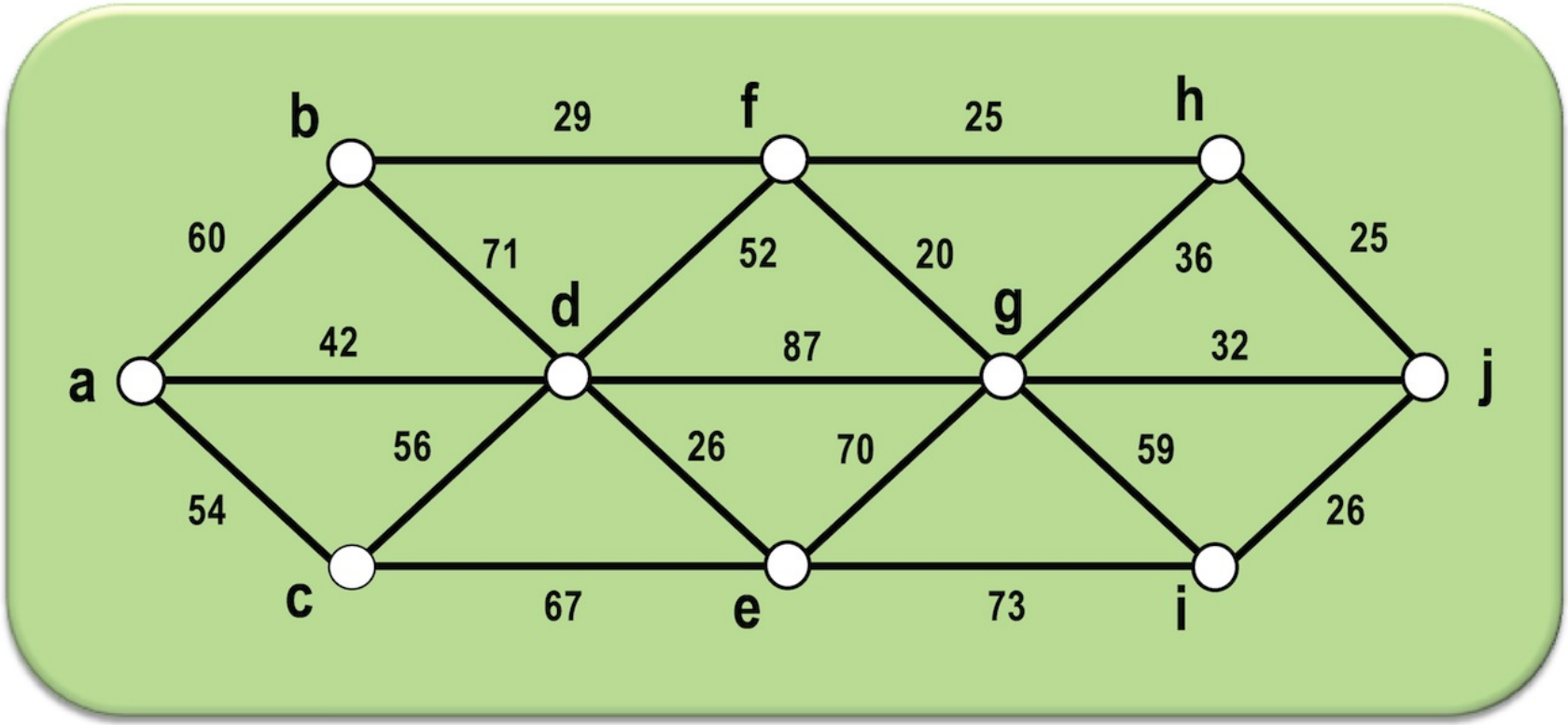
Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGraw-Hill. pp. 595–601.

Casos Especiais (3)

Grafos não direcionados com pesos inteiros: $O(m+n)$.

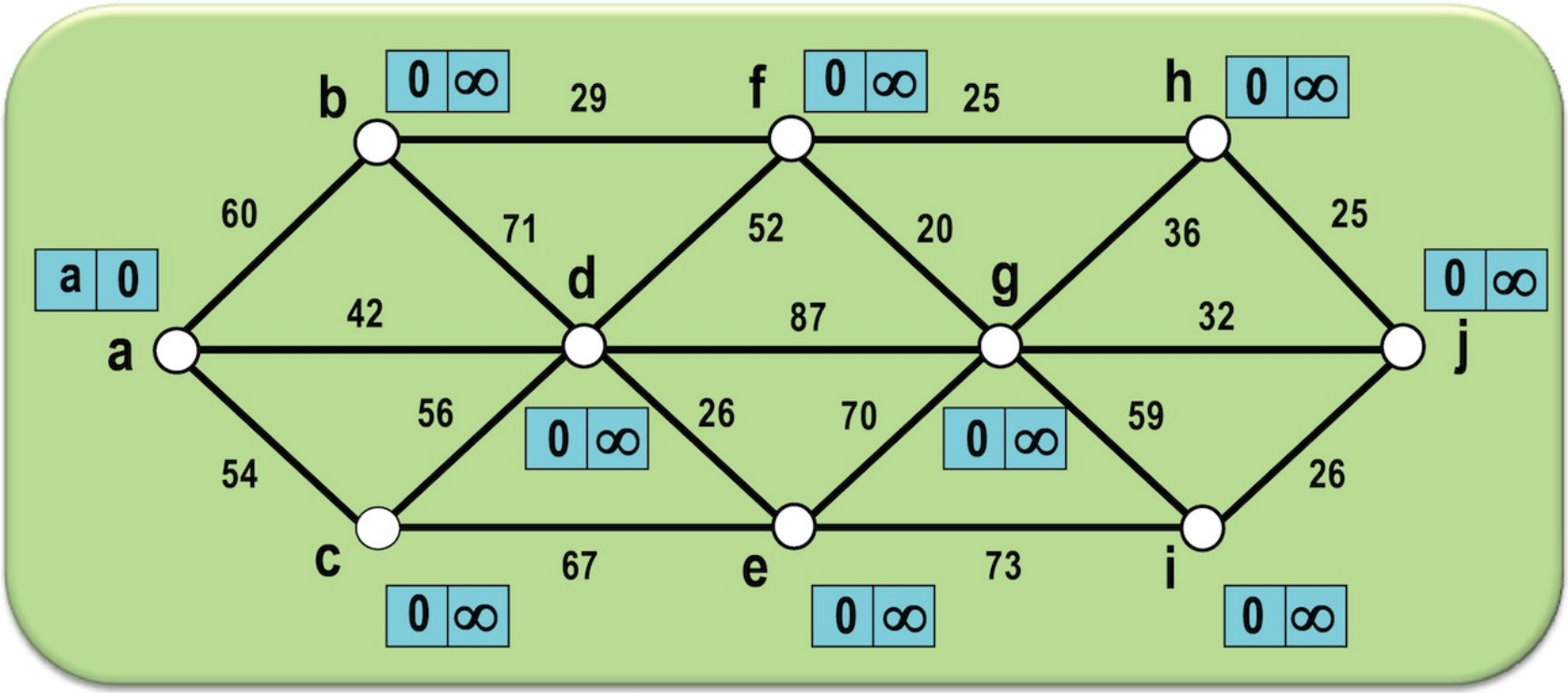
Thorup, Michael (1999). "Undirected single-source shortest paths with positive integer weights in linear time". Journal of the ACM. 46 (3): 362–394.

Dijkstra – Exemplo



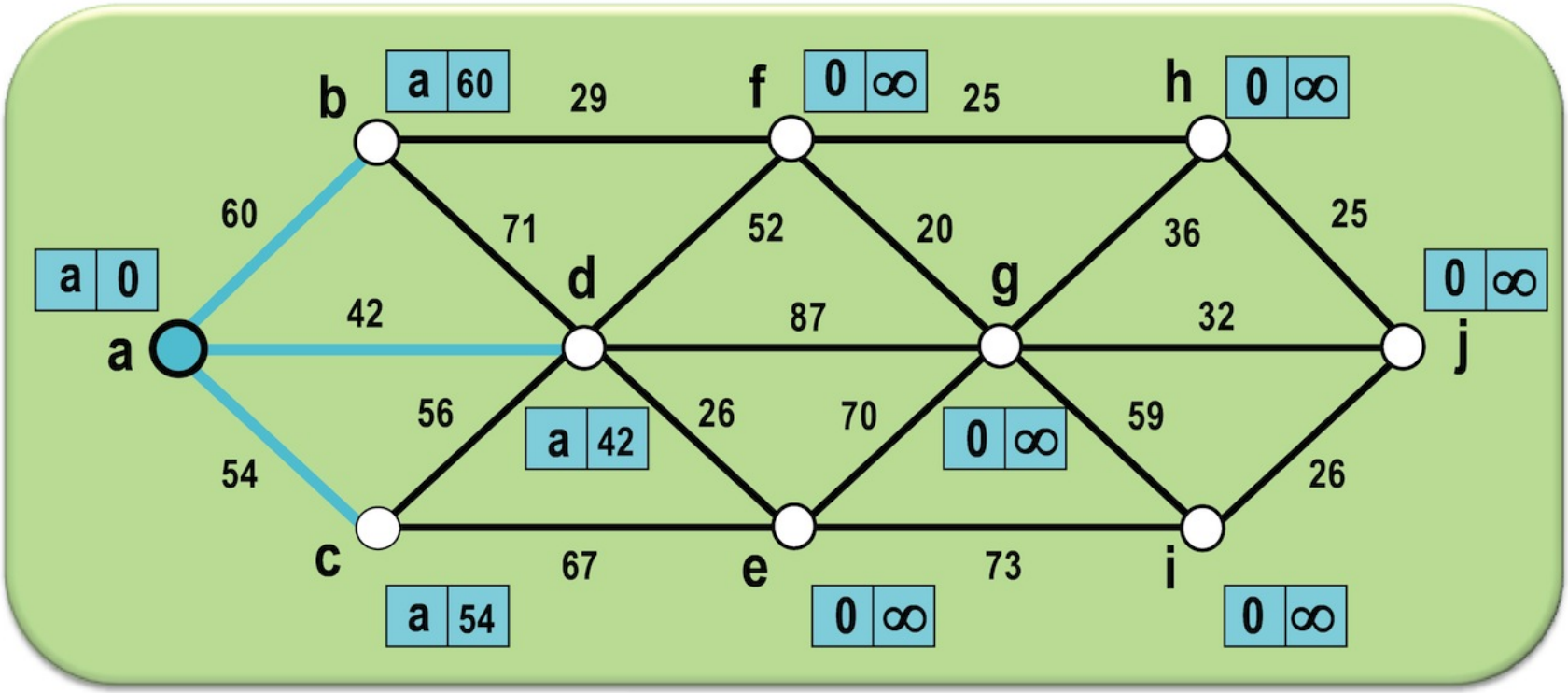
Grafo G. O vértice inicial será 'a'.

Dijkstra – Exemplo



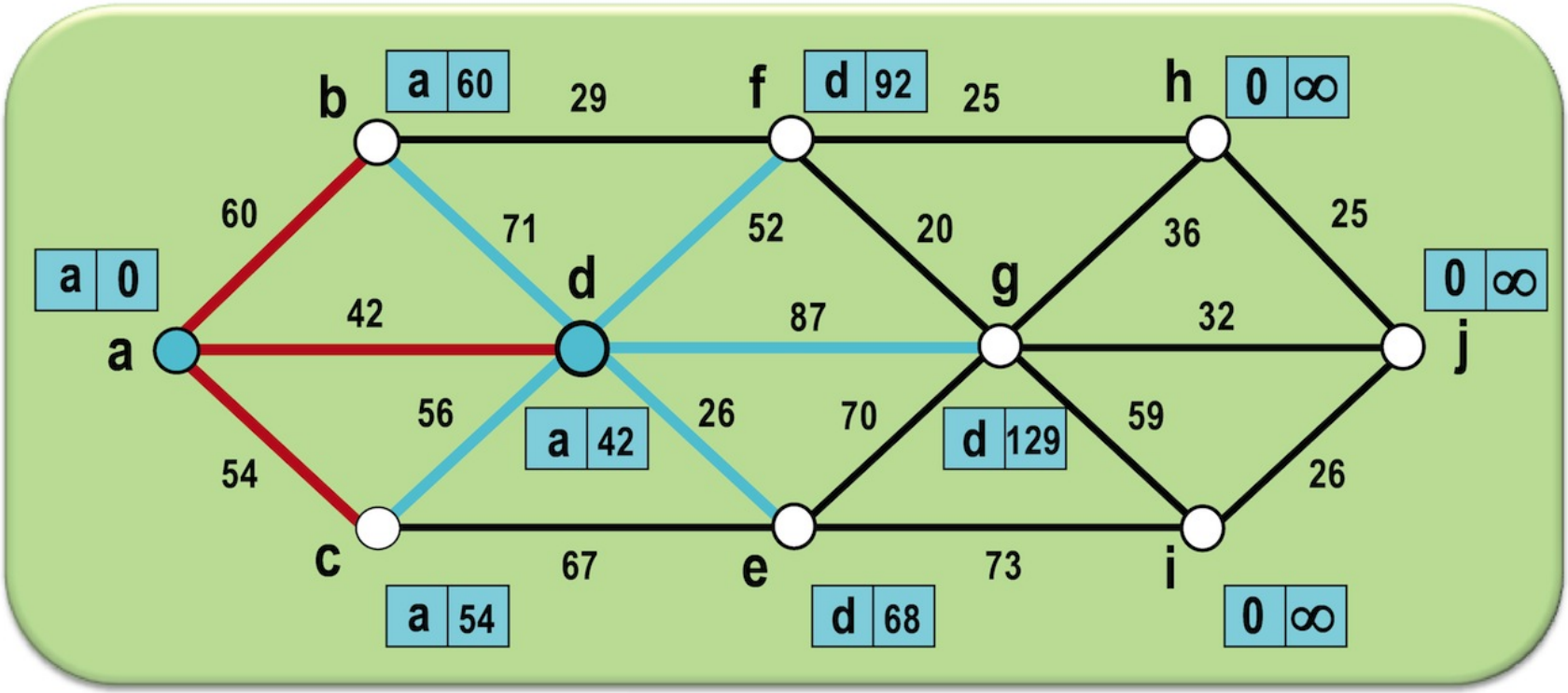
Rotulação após a primeira iteração do algoritmo.
O primeiro número é $rot[i]$ e o segundo, $dt[i]$.
Os vértices de F serão marcados em azul.

Dijkstra – Exemplo



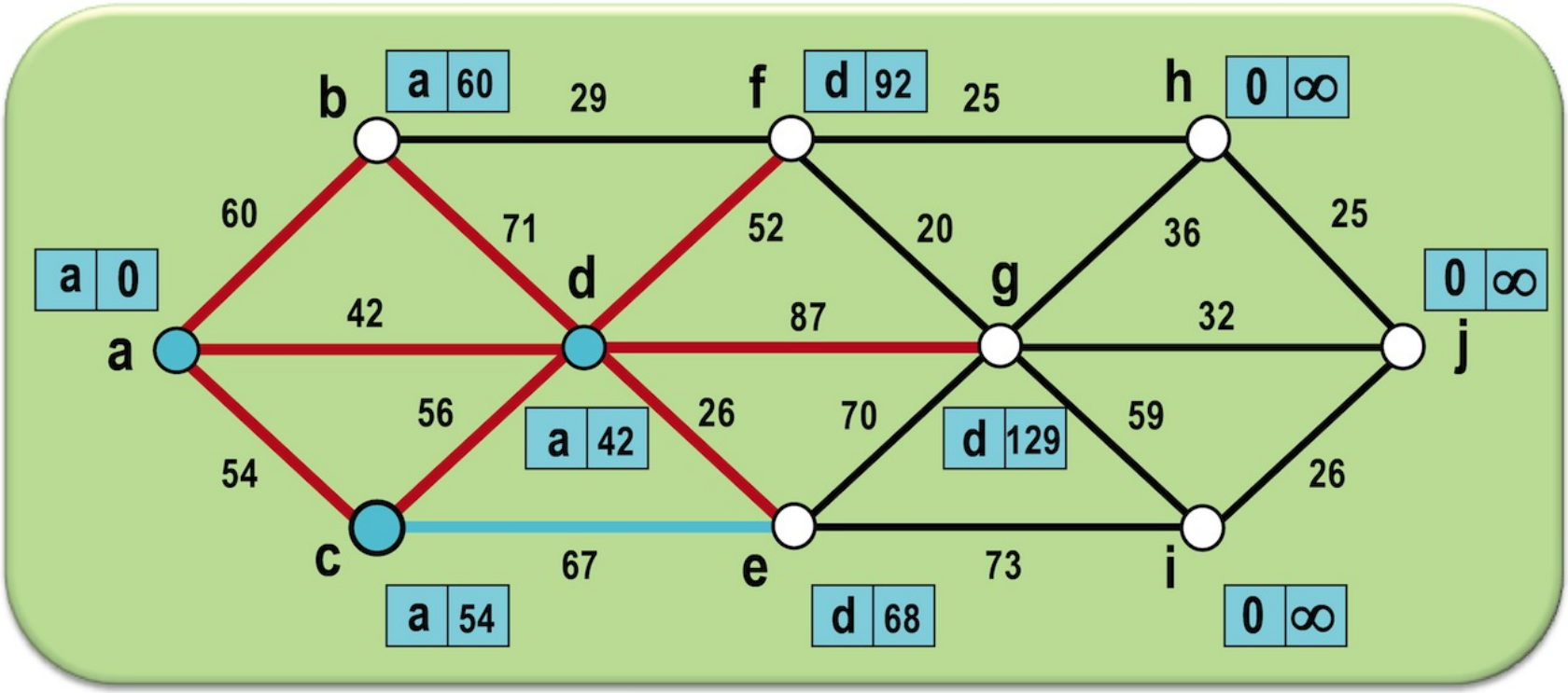
Exame do vértice a.
$rot[b]$, $dt[b]$, $rot[c]$, $dt[c]$, $rot[d]$ e $dt[d]$ atualizados.

Dijkstra – Exemplo



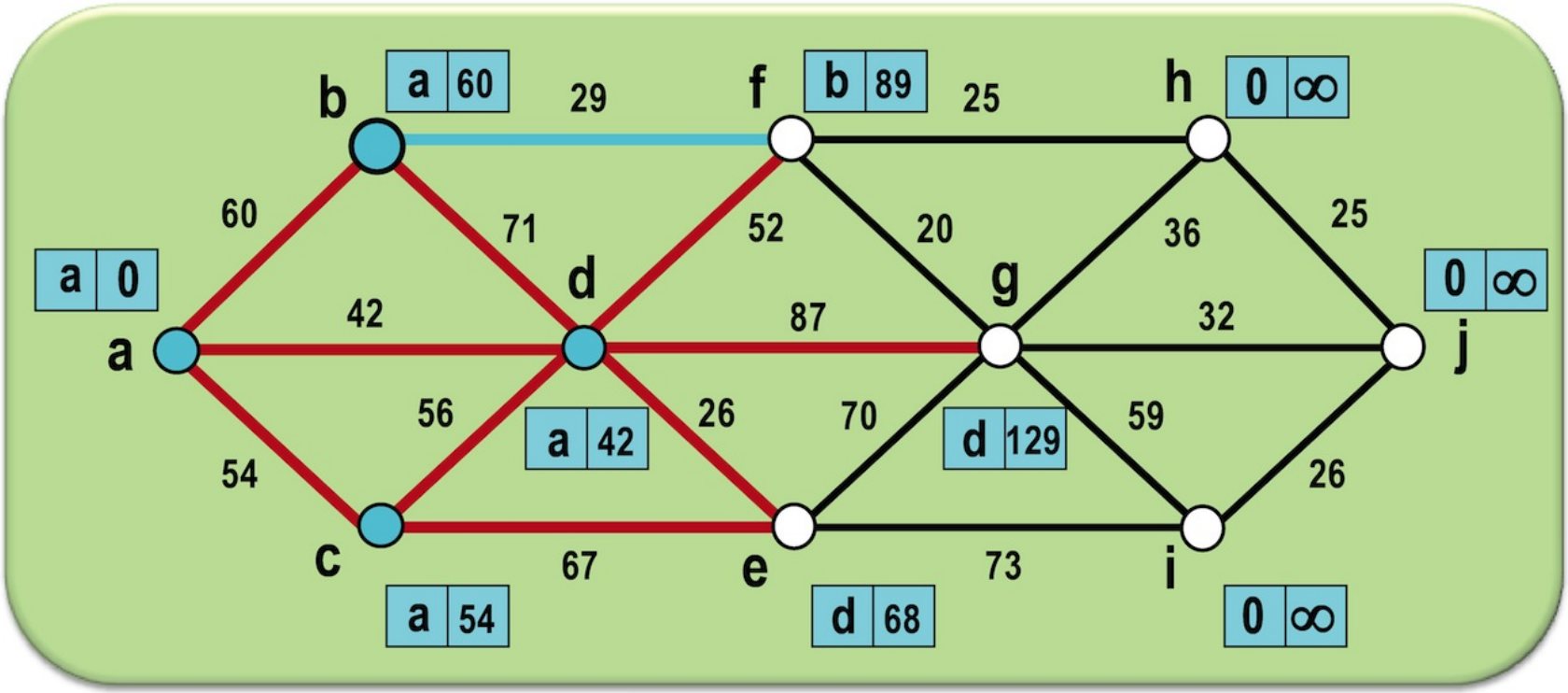
Exame do vértice d.
$rot[e]$, $dt[e]$, $rot[f]$, $dt[f]$, $rot[g]$ e $dt[g]$ atualizados.

Dijkstra – Exemplo



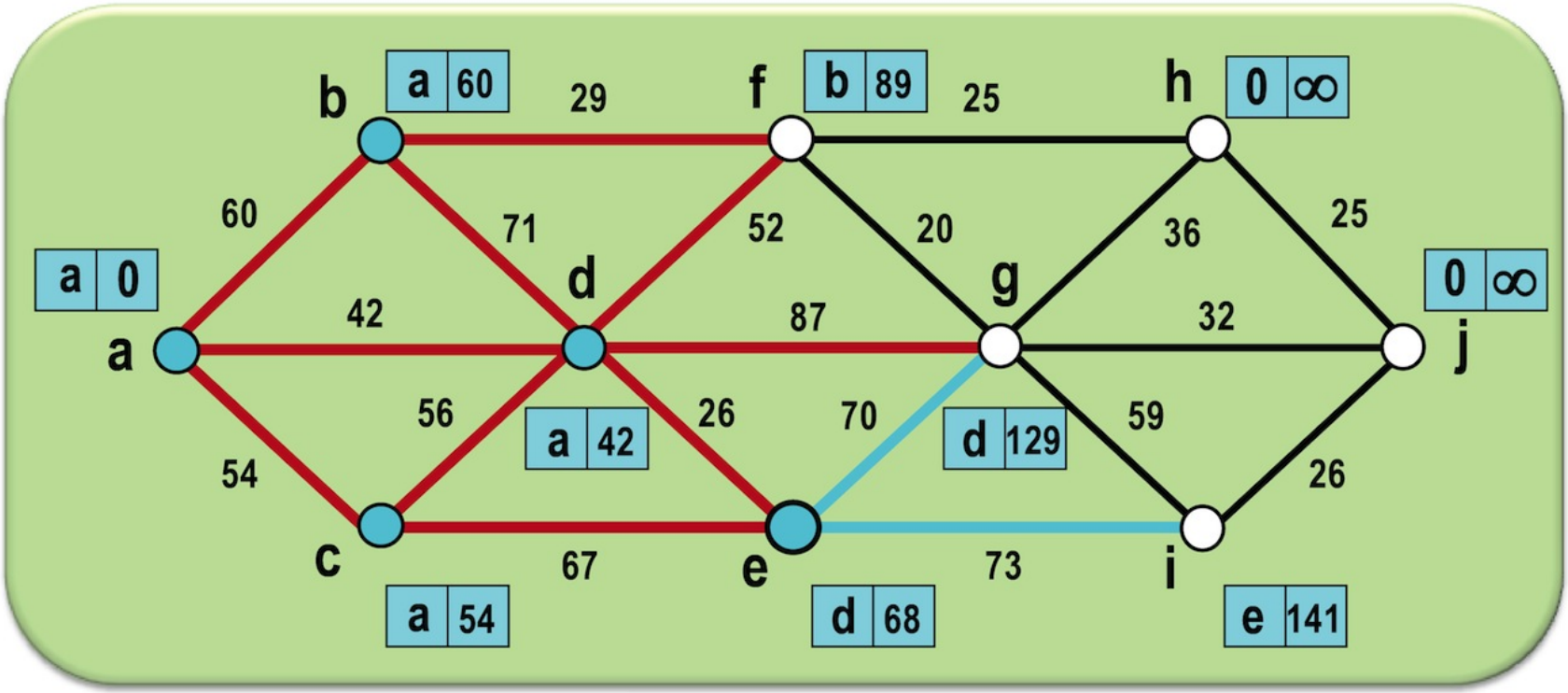
Exame do vértice c.
$rot[e]$ e $dt[e]$ não são atualizados.

Dijkstra – Exemplo



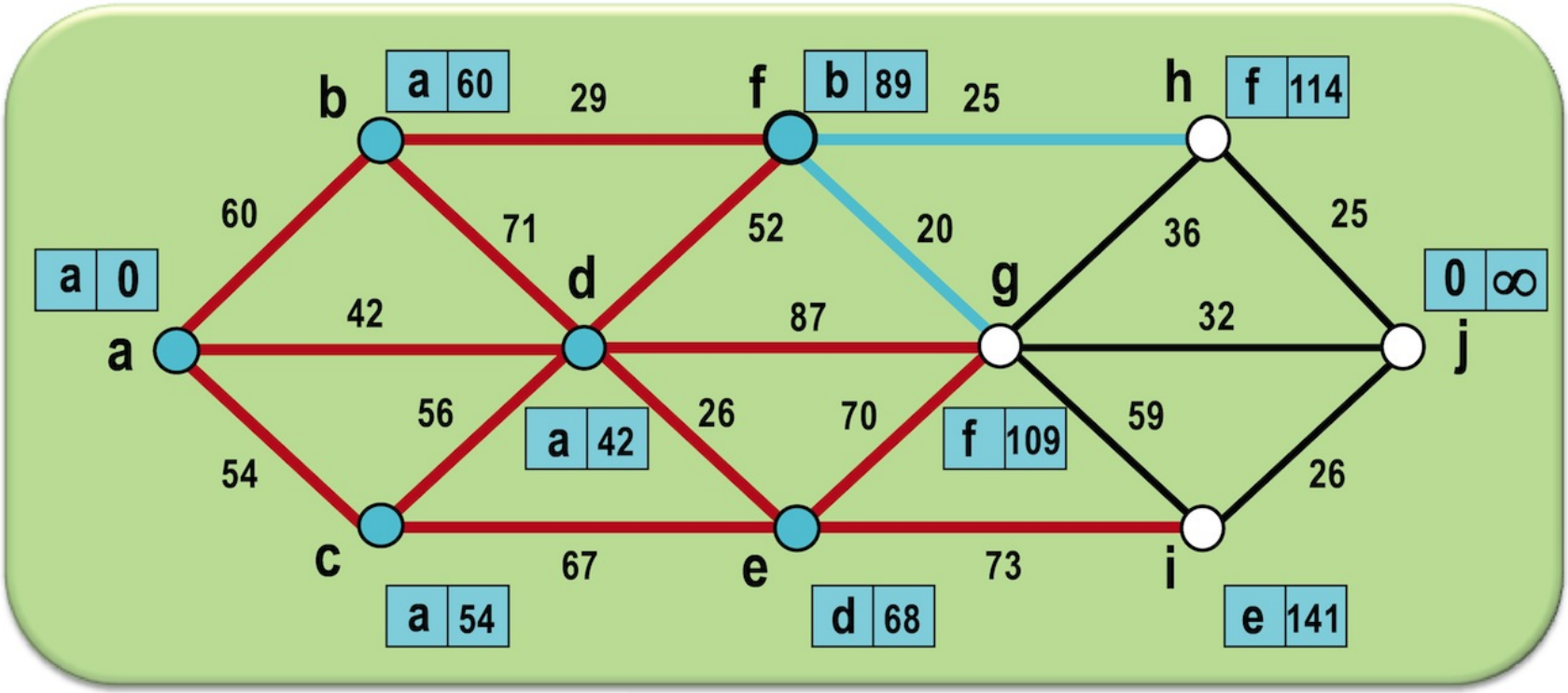
Exame do vértice b.
$rot[f]$ e $dt[f]$ são atualizados.

Dijkstra – Exemplo



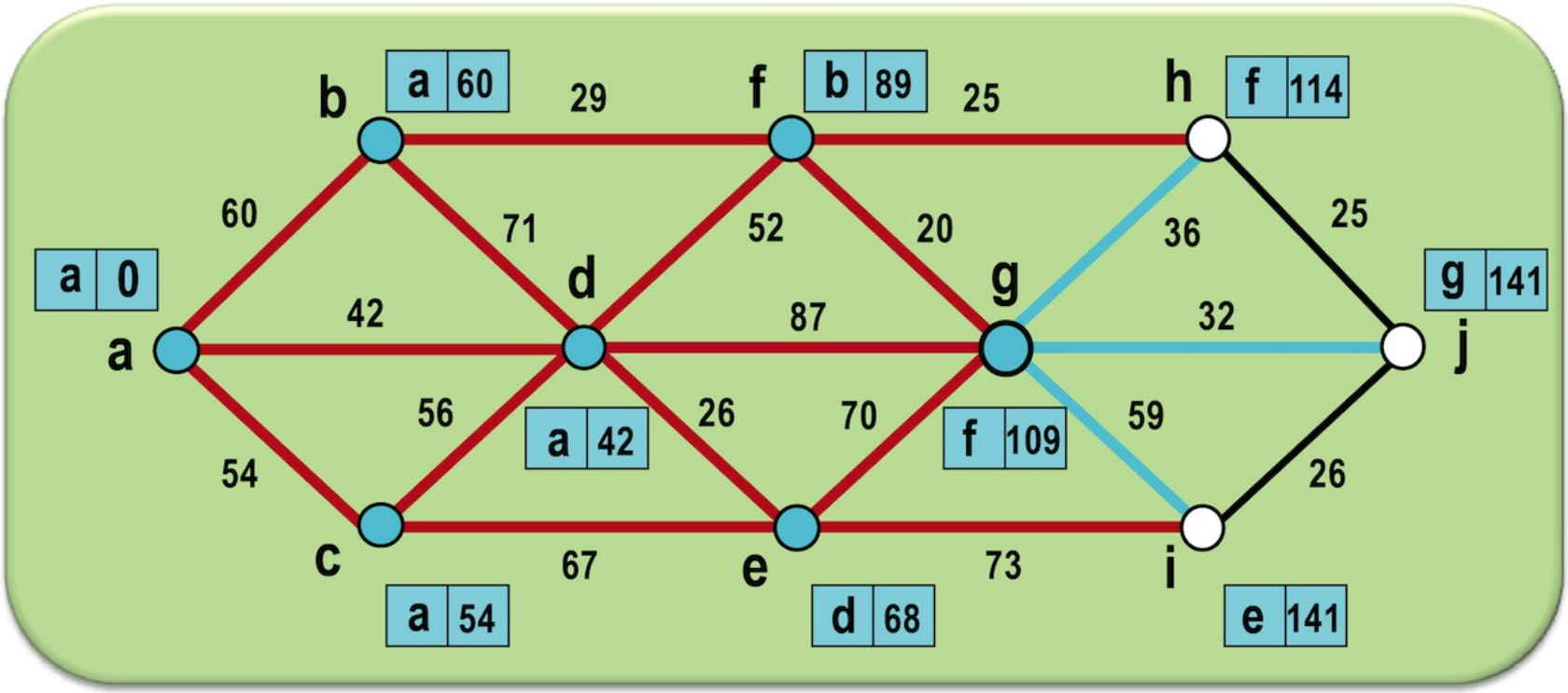
Exame do vértice e.
Apenas $rot[i]$ e $dt[i]$ são atualizados.

Dijkstra – Exemplo



Exame do vértice f.
$rot[g]$, $dt[g]$, $rot[h]$ e $dt[h]$ atualizados.

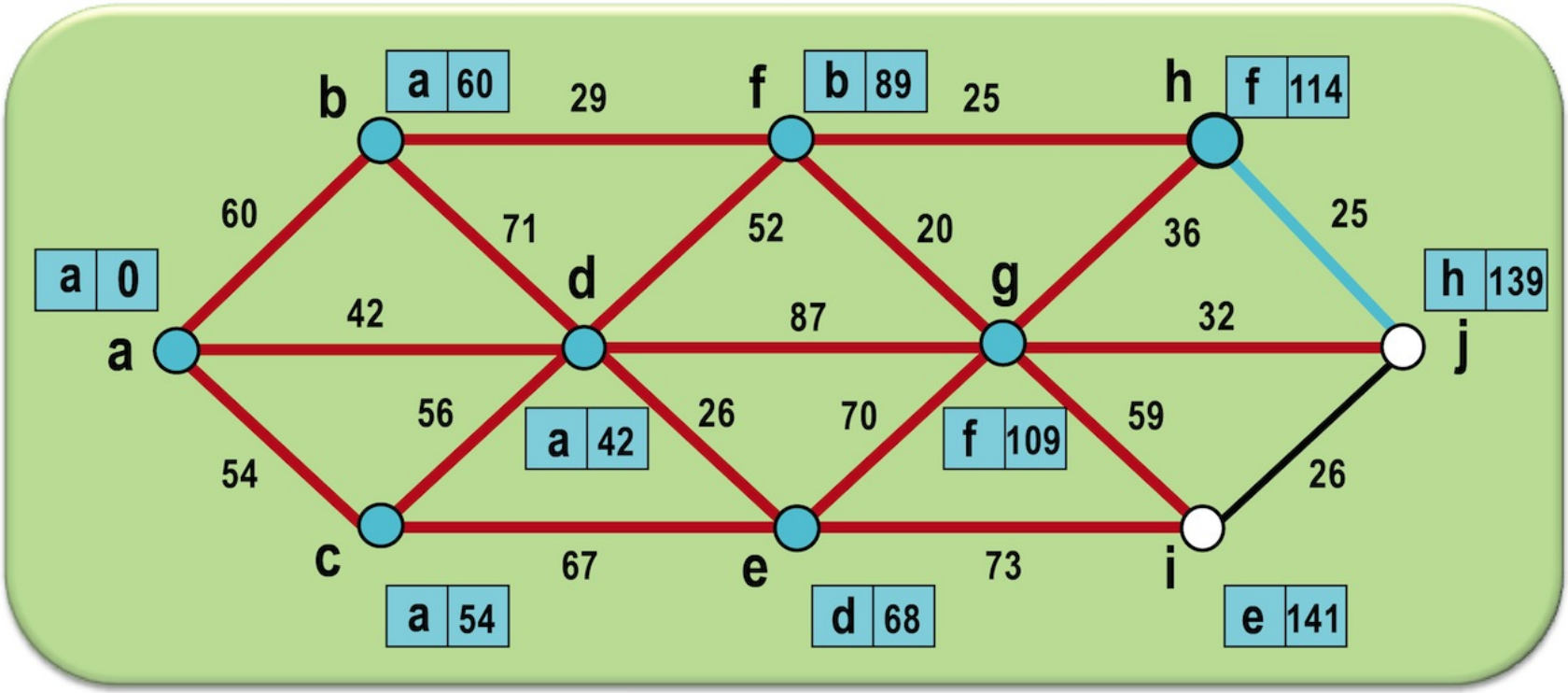
Dijkstra – Exemplo



Exame do vértice g.

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

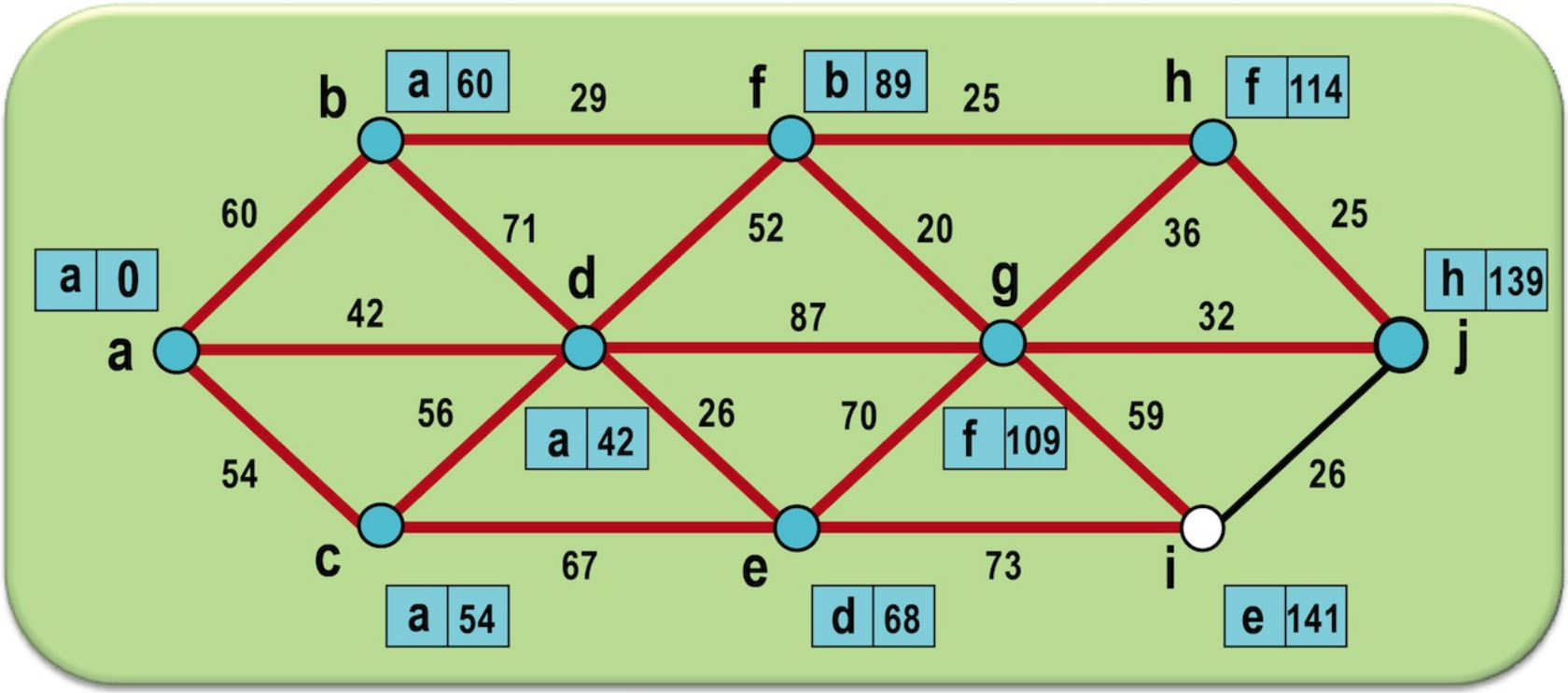
Dijkstra – Exemplo



Exame do vértice h.

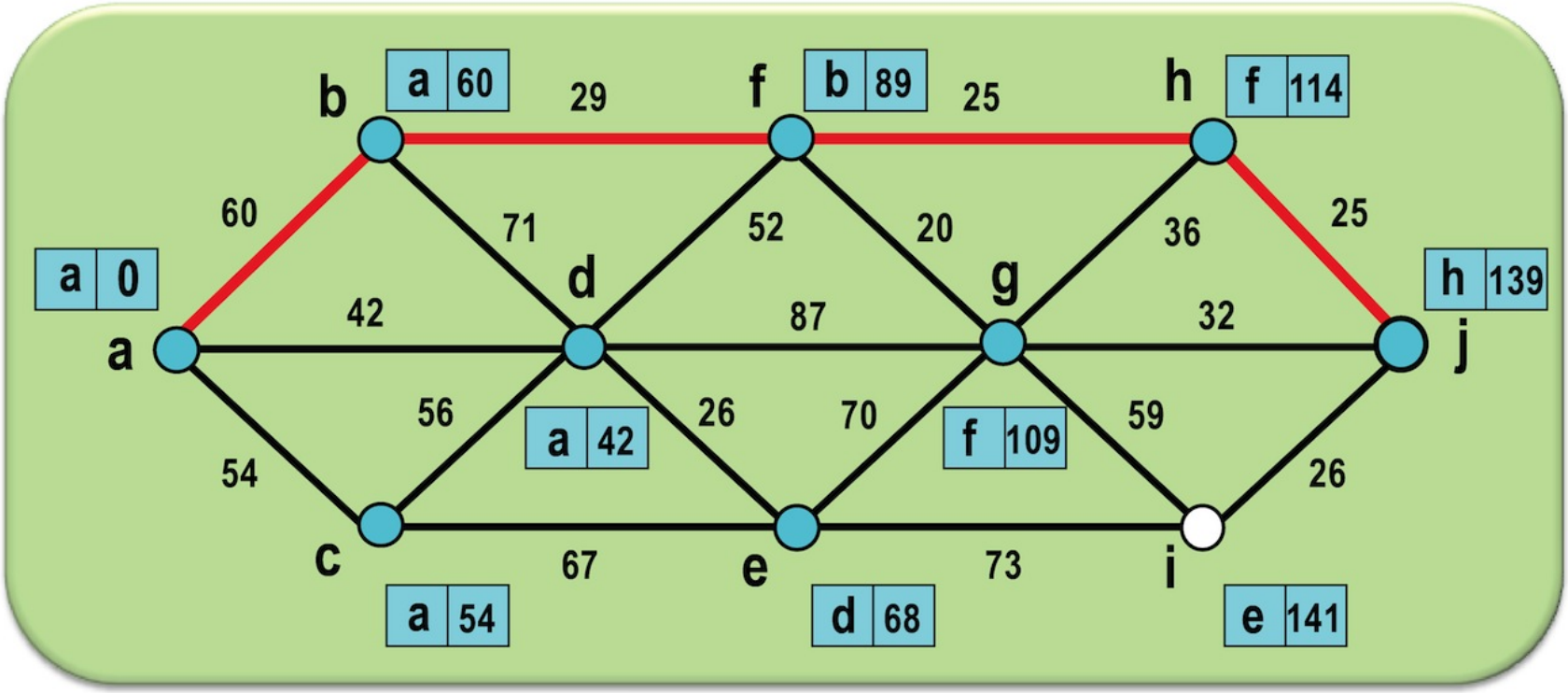
$rot[j]$ e $dt[j]$ são atualizados, pois o novo caminho é mais curto.

Dijkstra – Exemplo



Exame do vértice j.
Nenhuma atualização.

Dijkstra – Exemplo



Caminho mais curto entre a e j.

Algoritmo de Dijkstra

Prova de Corretude

A demonstração de que o algoritmo de Dijkstra está correto será feita por indução em F, considerando que, se o vértice i está em F, então dt[i] é o caminho mais curto da origem até o vértice i.

Base da Indução

No início, F contém o vértice s (a origem) e o teorema vale trivialmente.

Hipótese da Indução

A hipótese vale para todos os vértices de F até imediatamente antes da inserção de um vértice i.

Passo da Indução

Se o vértice i foi escolhido pelo algoritmo, então dt[i] é o menor dentre todos os vértices em A.

Deve-se mostrar que dt[i] é o comprimento do caminho mais curto entre a origem e i.

Supomos o contrário, ou seja, que existe pelo menos um vértice x no menor caminho entre a origem e i, que não pertence ao caminho atual, de comprimento dt[i], tal que dt[x] <dt[i] e $x \in A$.

Neste caso, o algoritmo deveria ter escolhido x ao invés de i. Mas escolheu i, significando que este nó x não existe.

Portanto, quando i é adicionado a F, o caminho mais curto entre a origem e i foi encontrado.

Algoritmo de Dijkstra

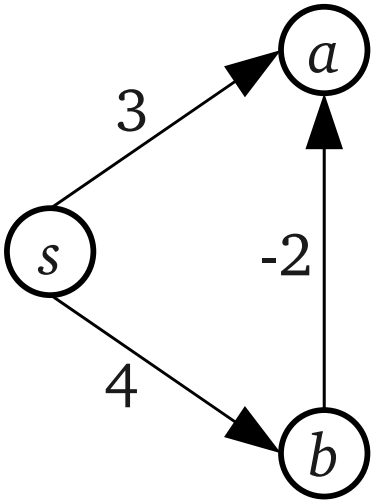
Crítica

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



- 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?