Caminhos Mínimos

Marcos Castro

Caminhos Mínimos

Um motorista deseja obter o caminho mínimo (mais curto) entre Campinas e Araçatuba, duas cidades de São Paulo. Dado um mapa do estado de São Paulo contendo as distâncias entre cada par de interseções adjacentes, como obter o caminho mínimo entre as duas cidades?

Nesse caso nós podemos modelar o mapa rodoviário como um grafo em que vértices representam as interseções, arestas representam segmentos de estrada entre interseções e o peso de cada aresta, a distância entre interseções.

Caminhos Mínimos

Nesta seção trataremos do problema de encontrar o caminho mínimo entre dois vértices de um grafo orientado valorado G = (V, E).

O problema do motorista descrito anteriormente é equivalente a obter os caminhos mínimos a partir de uma única origem.

Dado um grafo orientado valorado G = (V, E), o **peso** de um caminho $c = (v_0, v_1, v_2, ..., v_k)$ é a soma de todos os pesos das arestas do caminho:

$$p(c) = \sum_{i=1}^{k} = p(v_{i-1}, v_i)$$

Caminhos Mínimos

O caminho mínimo é definido por:

$$\sigma(u,v) = \begin{cases} \min\{p(c) : u \rightarrow v\}, \text{ se existir um caminho de u a v,} \\ \infty, \text{ caso contrário.} \end{cases}$$

Um **caminho mínimo** do vértice **u** ao vértice **v** é então definido como qualquer caminho c com peso $p(c) = \sigma(u, v)$.

Caminhos Mínimos

O peso das arestas pode ser interpretado como outras métricas diferentes de distâncias, tais como tempo, custo, penalidades, etc.

Um caminho mínimo em um grafo G = (V, E) não pode conter ciclo algum, uma vez que a remoção do ciclo do caminho produz um caminho com mesmos vértices origem e destino e um caminho de menor peso. Assim, podemos assumir que caminhos mínimos não possuem ciclos.

Caminhos Mínimos

Uma vez que qualquer caminho acíclico em G contém no máximo |V| vértices, então o caminho contém no máximo |V|-1 arestas.

A representação de caminhos mínimos em um grafo G=(V, E) pode ser realizada pela variável **Antecessor**. Para cada vértice v ∈ V o Antecessor[v] é outro vértice u ∈ V ou nil.

O algoritmo para obter caminhos mínimos atribui a **Antecessor** os rótulos de vértices de um caminho de antecessores com origem em um vértice v e que anda para trás ao longo de um caminho mínimo de um vértice origem s até v.

Caminhos Mínimos

Durante a execução do algoritmo para obter caminhos mínimos, os valores em Antecessor[v] não necessariamente indicam caminhos mais curtos. Entretanto, ao final do processamento Antecessor contém, de fato, uma árvore de caminhos mínimos.

Algoritmo de Dijkstra

O algoritmo de **Dijkstra** encontra o menor caminho entre quaisquer dois vértices do grafo, quando todos os arcos têm comprimento não-negativos.

O algoritmo de Dijkstra utiliza um procedimento iterativo, determinando, na iteração 1, o vértice mais próximo de 1, na segunda iteração, o segundo vértice mais próximo do vértice 1, e assim sucessivamente, até que em alguma iteração o vértice N seja atingido.

O algoritmo mantém um conjunto S de vértices cujos caminhos mínimos até um vértice origem já são conhecidos.

Algoritmo de Dijkstra

Este algoritmo utiliza a técnica de relaxamento. Para cada vértice $v \in V$ o atributo p[v] é um limite superior do peso de um caminho mínimo do vértice origem s até v.

O vetor p[v] contém uma estimativa de um caminho mais curto.

No passo da **inicialização** é executado:

Antecessor[v] = nil	$\forall v \in V$
p[v] = 0	para o vértice origem s
p[v] = ∞	para $v \in V - \{s\}$

Algoritmo de Dijkstra

O processo de **relaxamento** de uma aresta (u, v) consiste em verificar se é possível melhorar o melhor caminho obtido até o momento até v se passarmos por u. Se isto acontecer então p[v] e Antecessor[v] devem ser atualizados.

Relaxamento de uma aresta:

Se p[v] > p[u] + peso da aresta (u, v)
então p[v] = p[u] + peso da aresta (u, v)
Antecessor[v] := u

Algoritmo de Dijkstra

```
procedure Dijkstra (Grafo, Raiz);
        for v := 0 to Grafo.NumVertices-1 do
                p[v] := Infinito;
                Antecessor[v] := -1;
        p[Raiz] := 0;
        Constroi heap no vetor A;
        S := Ø;
        While heap > 1 do
                u := RetiraMin(A);
                S := S + u;
                for v ∈ ListaAdjacentes[u] do
                        if p[v] > p[u] + peso da aresta (u,v)
                        then p[v] = p[u] + peso da aresta (u,v)
                                Antecessor[v] := u
```

Algoritmo de Dijkstra

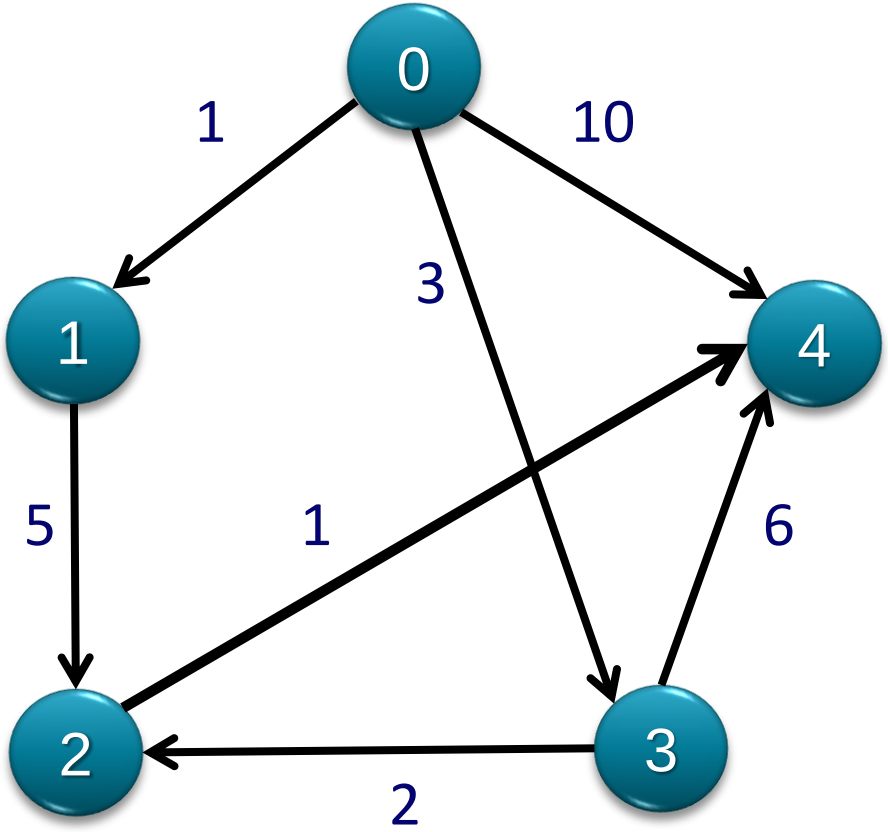
S – conjunto solução

heap A – é uma fila de prioridades.

RetiraMin(A) – retirar o vértice **u** que tem a menor estimativa de caminhos mínimos em comparação com qualquer vértice em V - S.

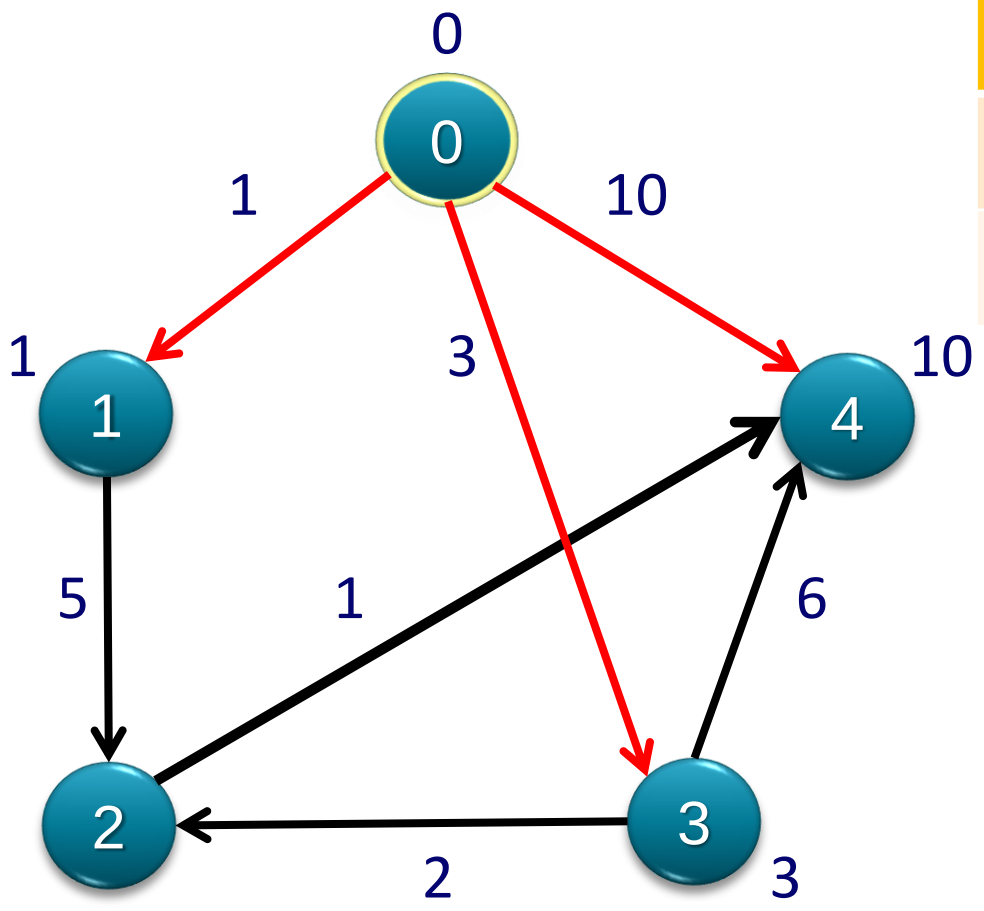
Exemplo 1

Caminho mínimo do vértice 0 ao vértice 4.



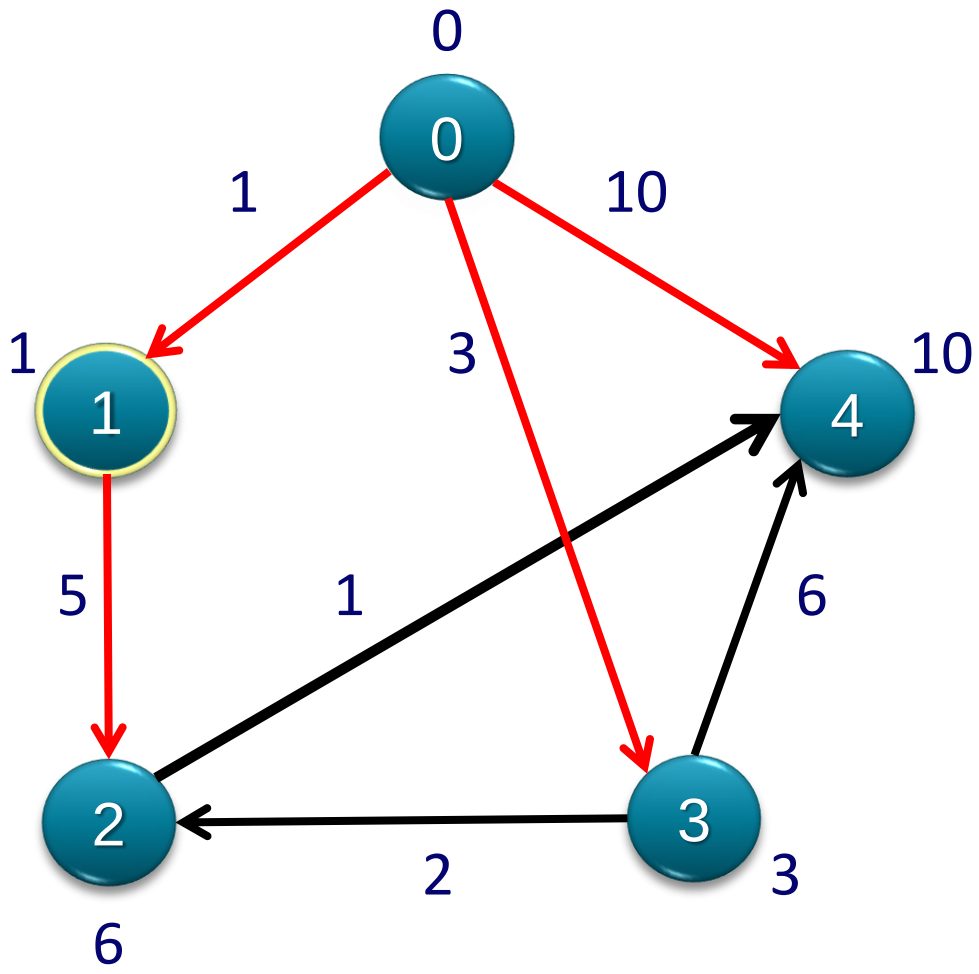
Iteração	S	d[0]	d[1]	d[2]	d[3]	d[4]
1	∅	∞	∞	∞	∞	∞

Exemplo 1



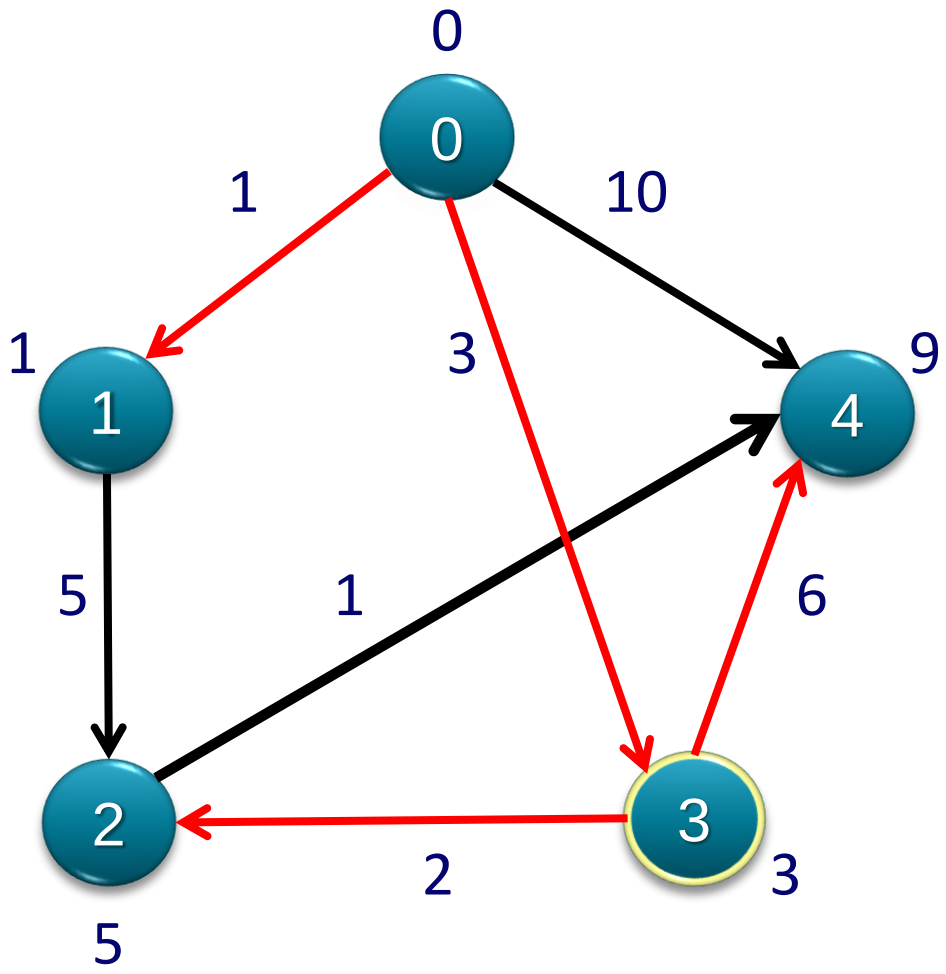
Iteração	S	d[0]	d[1]	d[2]	d[3]	d[4]
1	Ø	∞	∞	∞	∞	∞
2	{0}	0	1	∞	3	10

Exemplo 1



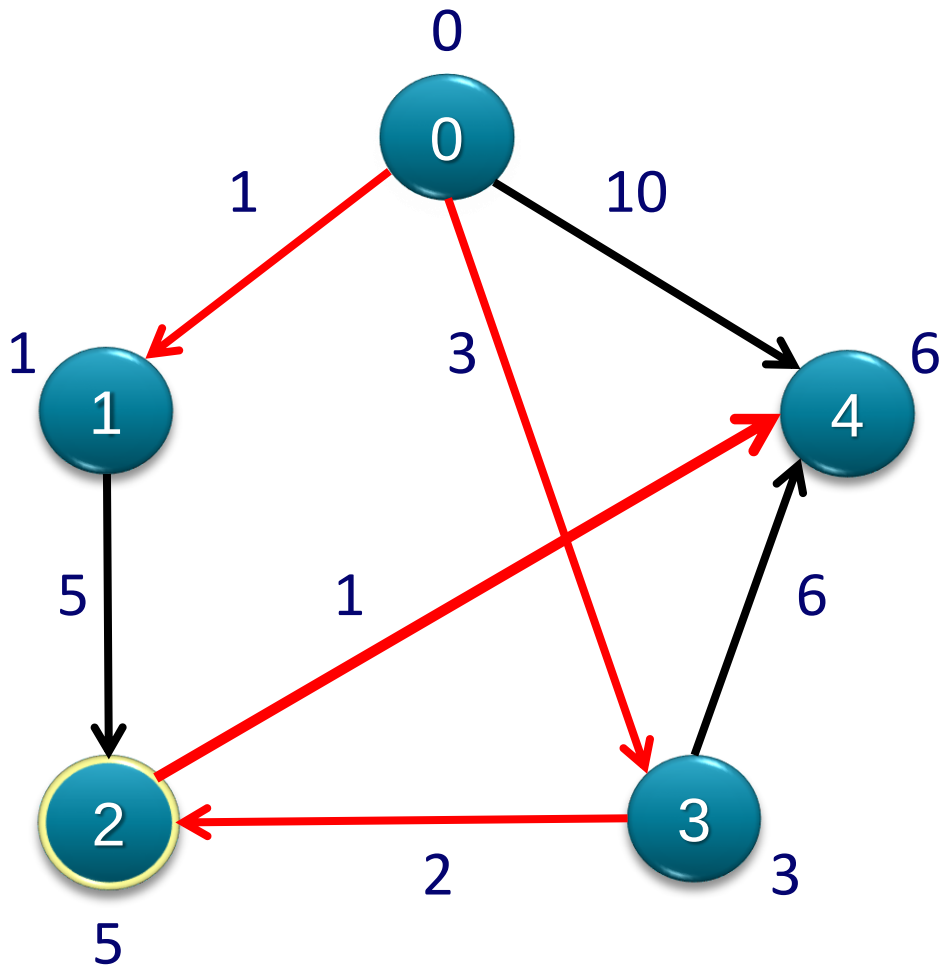
Iteração	S	d[0]	d[1]	d[2]	d[3]	d[4]
1	Ø	∞	∞	∞	∞	∞
2	{0}	0	1	∞	3	10
3	{0,1}	0	1	6	3	10

Exemplo 1



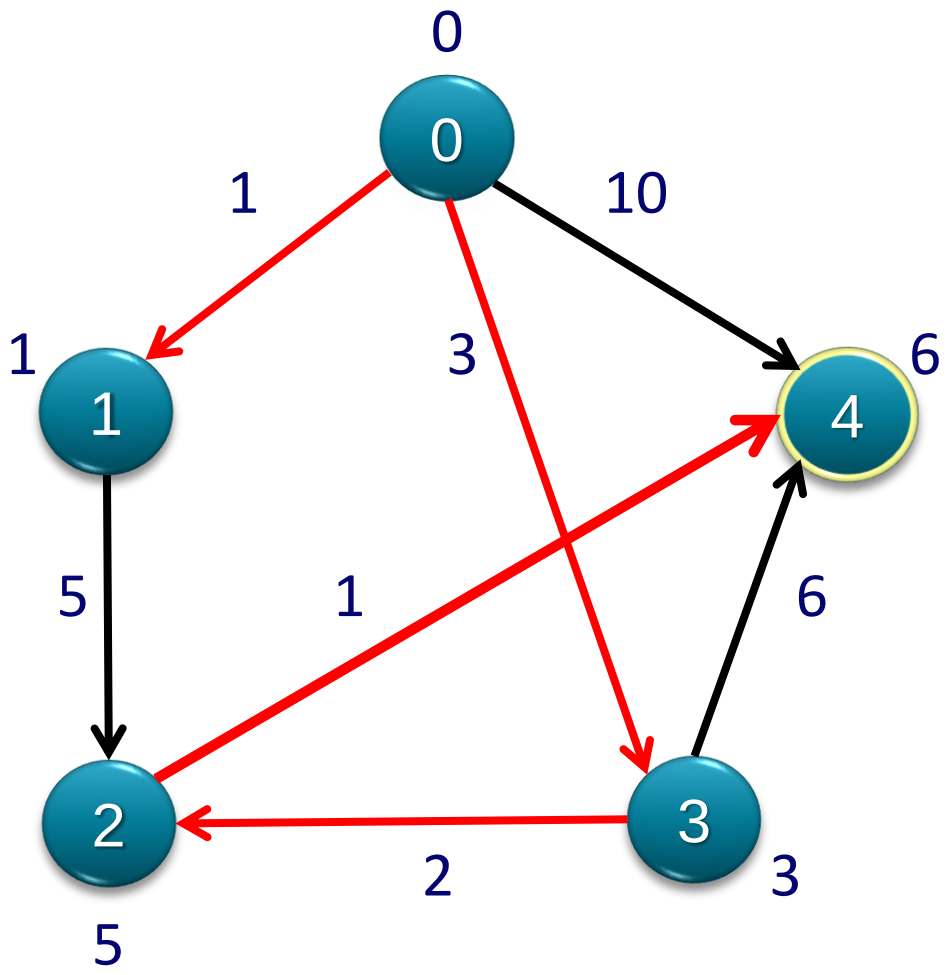
Iteração	S	d[0]	d[1]	d[2]	d[3]	d[4]
1	Ø	∞	∞	∞	∞	∞
2	{0}	0	1	∞	3	10
3	{0,1}	0	1	6	3	10
4	{0,1,3}	0	1	5	3	9

Exemplo 1



Iteração	S	d[0]	d[1]	d[2]	d[3]	d[4]
1	Ø	∞	∞	∞	∞	∞
2	{0}	0	1	∞	3	10
3	{0,1}	0	1	6	3	10
4	{0,1,3}	0	1	5	3	9
5	{0,1,3,2}	0	1	5	3	6

Exemplo 1

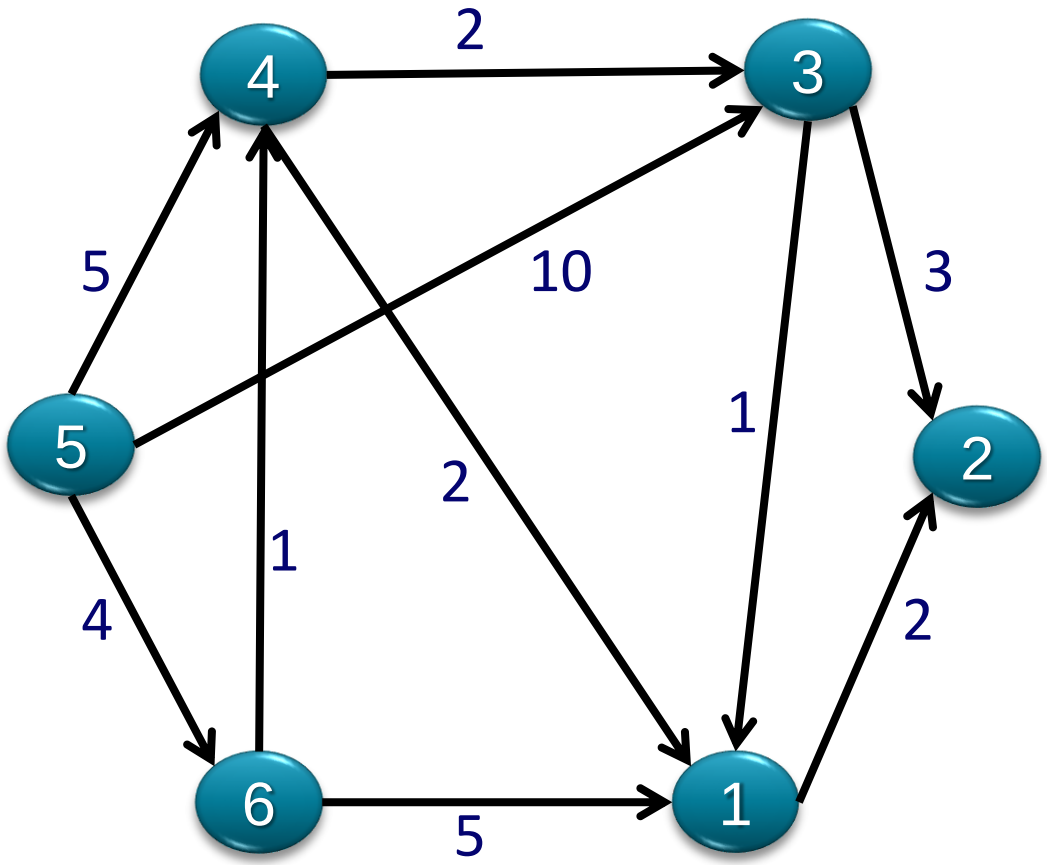


Iteração	S	d[0]	d[1]	d[2]	d[3]	d[4]
1	Ø	∞	∞	∞	∞	∞
2	{0}	0	1	∞	3	10
3	{0,1}	0	1	6	3	10
4	{0,1,3}	0	1	5	3	9
5	{0,1,3,2}	0	1	5	3	6
6	{0,1,3,2,4}	0	1	5	3	6

Custo: 6

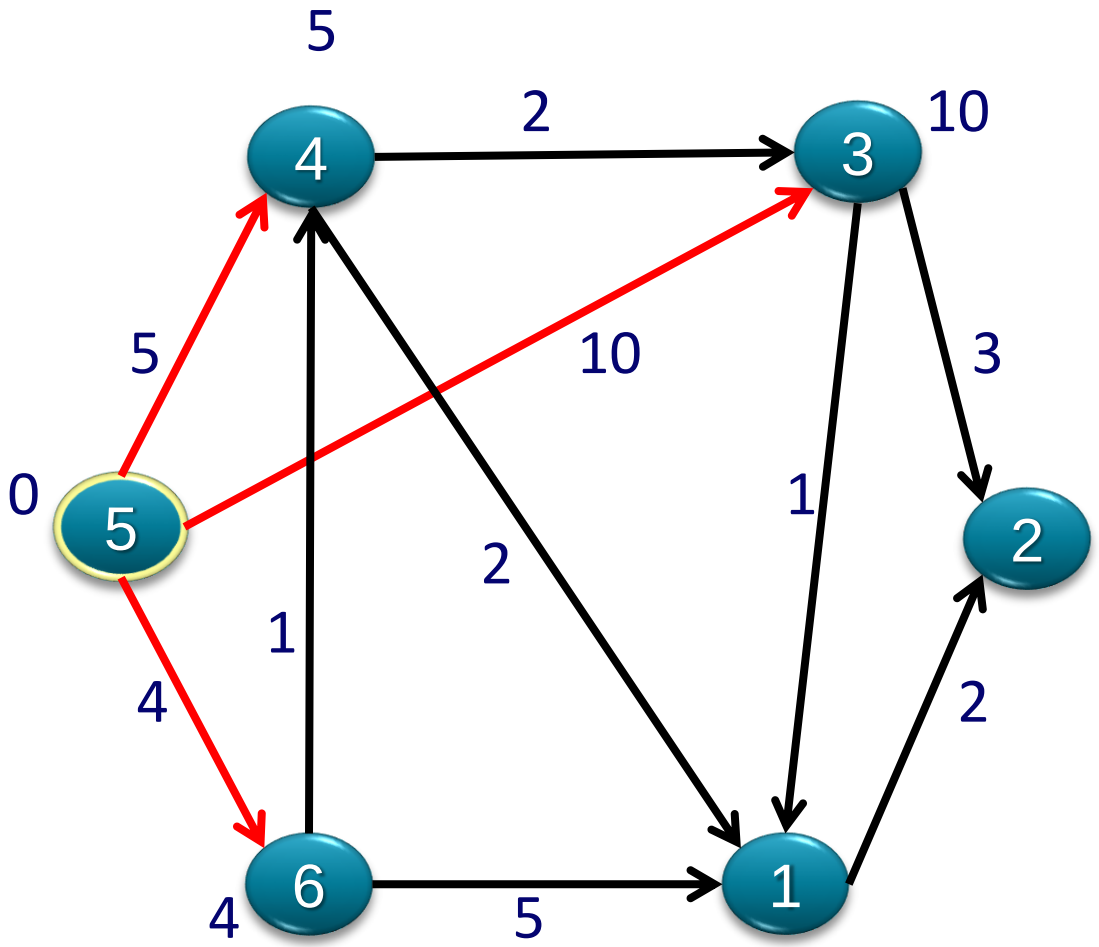
Exemplo 2

Caminho mínimo do vértice 5 ao vértice 2.



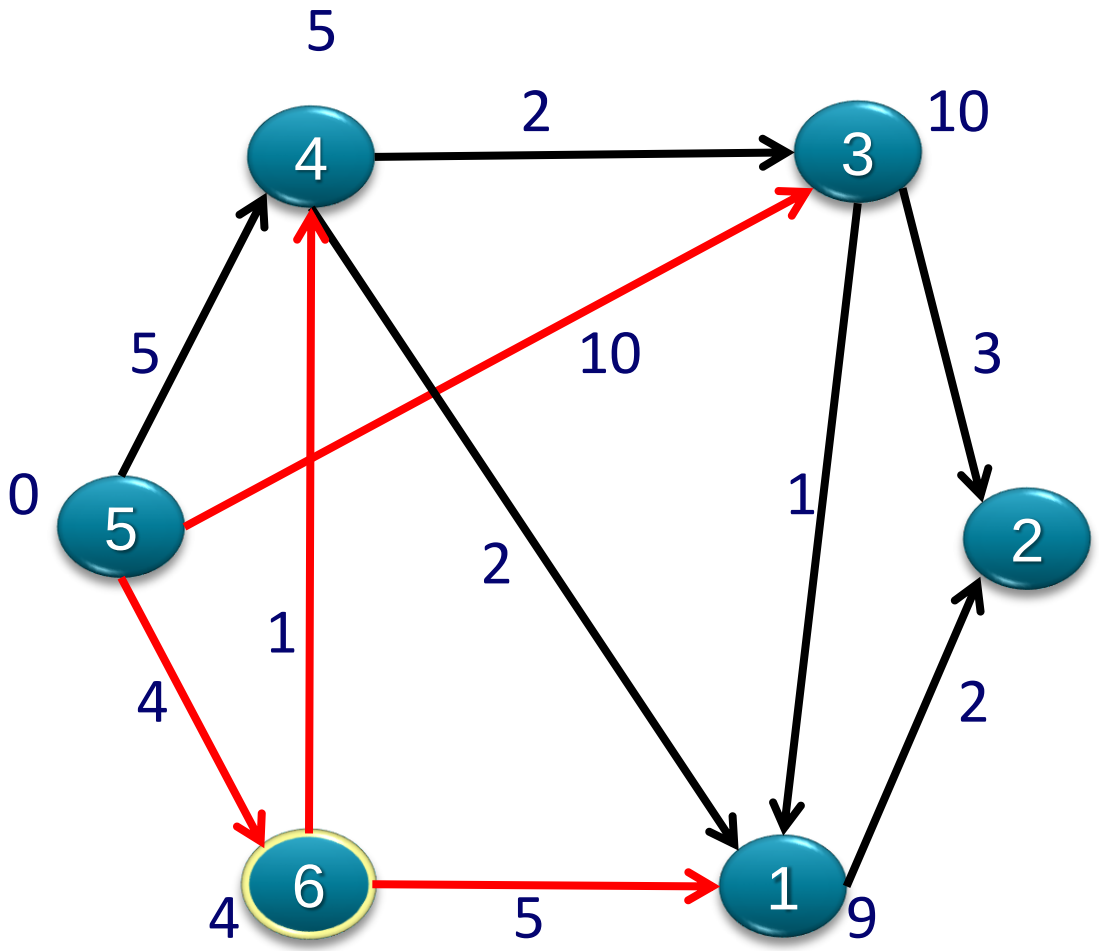
Iter	S	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
1	∅	∞	∞	∞	∞	∞	∞

Exemplo 2



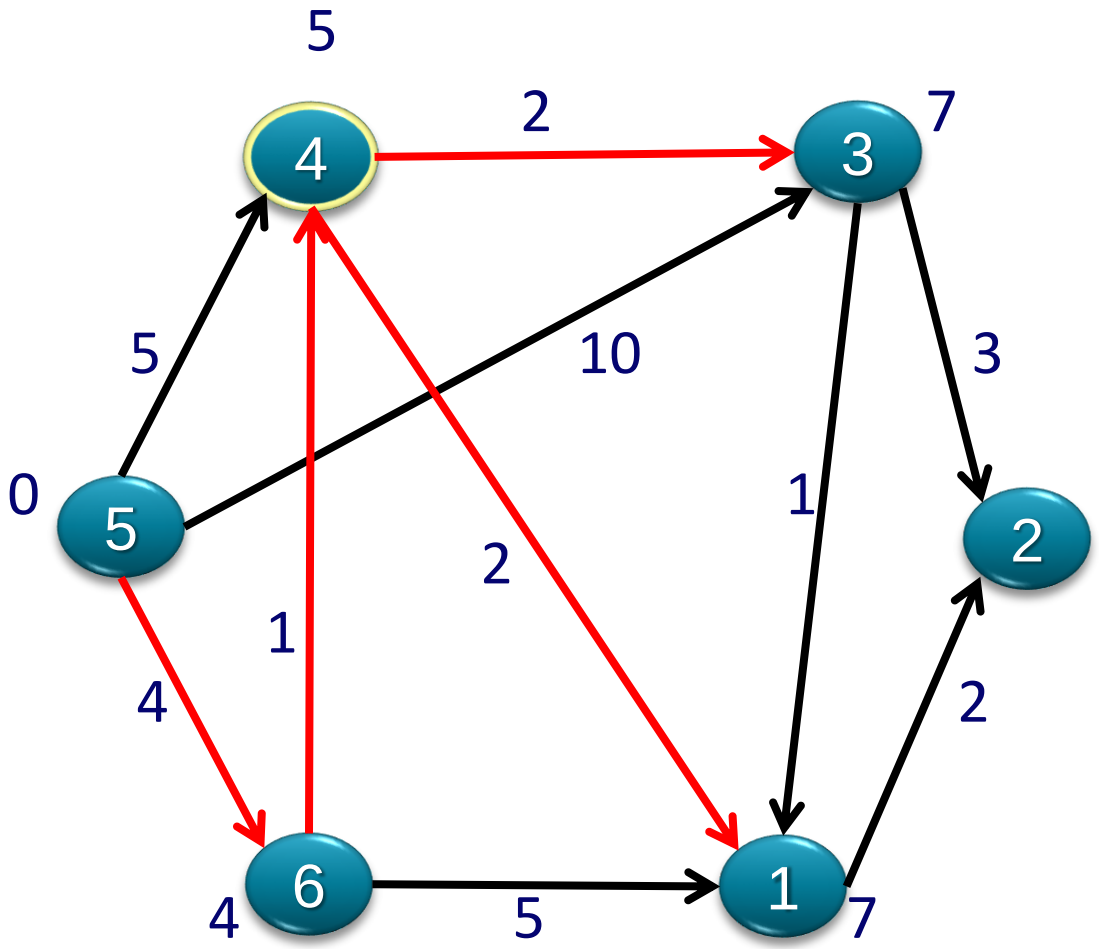
Iter	S	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
1	Ø	∞	∞	∞	∞	∞	∞
2	{5}	∞	∞	10	5	0	4

Exemplo 2



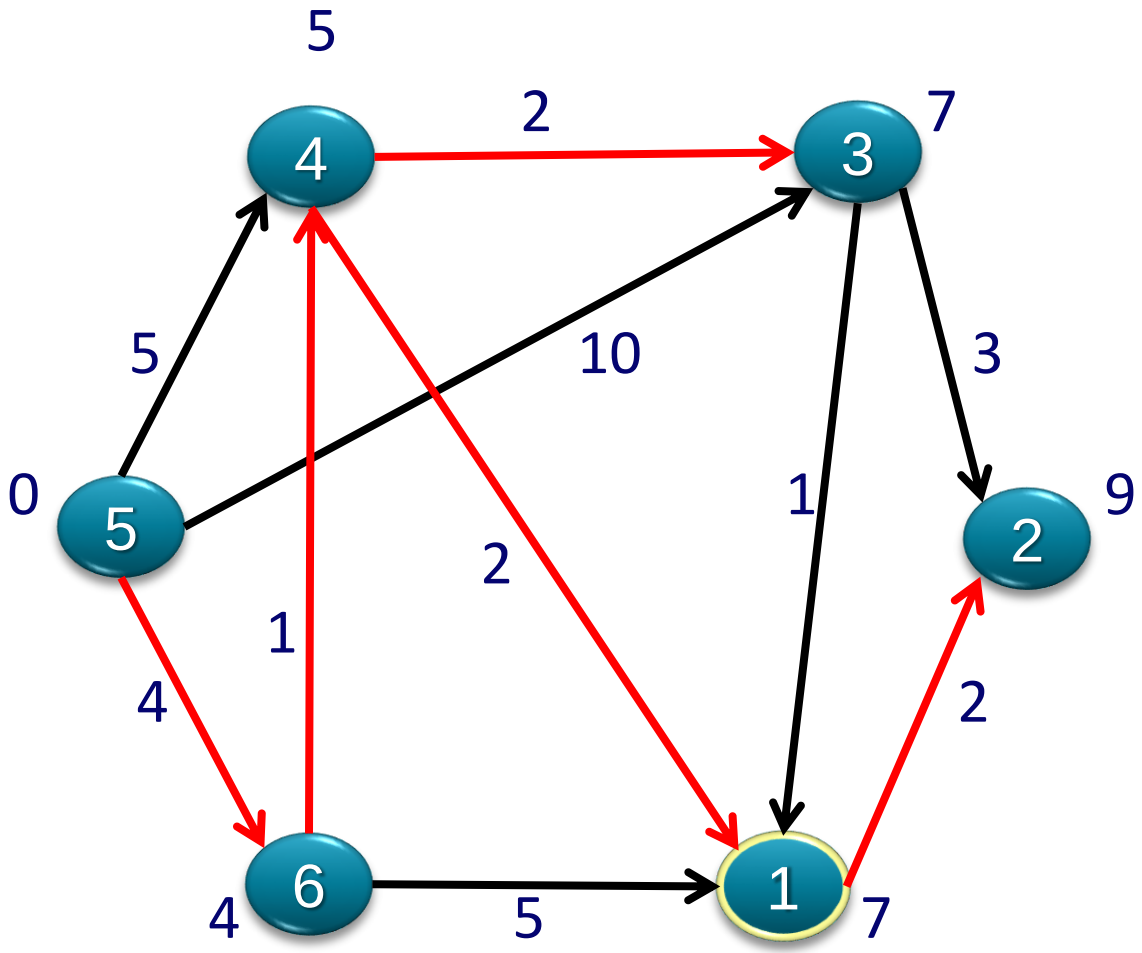
Iter	S	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
1	Ø	∞	∞	∞	∞	∞	∞
2	{5}	∞	∞	10	5	0	4
3	{5,6}	9	∞	10	5	0	4

Exemplo 2



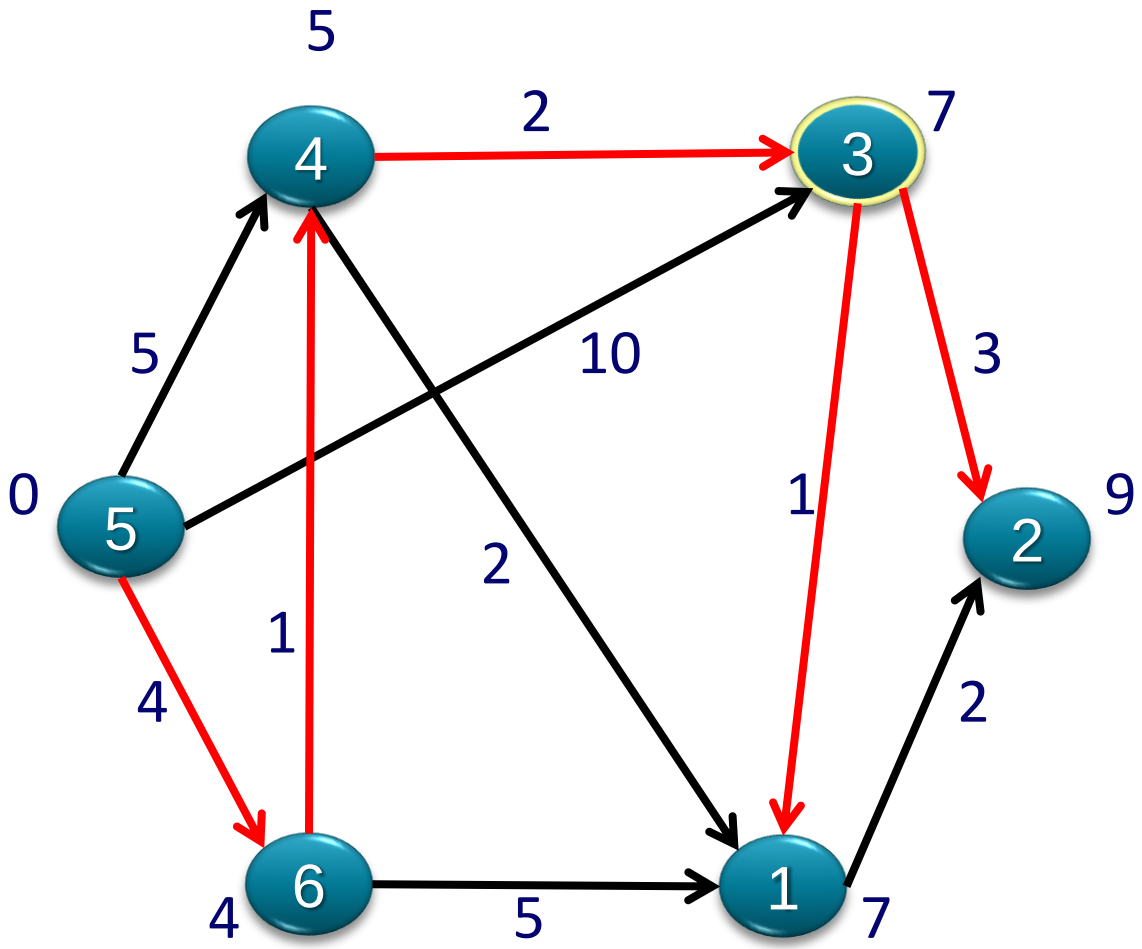
Iter	S	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
1	Ø	∞	∞	∞	∞	∞	∞
2	{5}	∞	∞	10	5	0	4
3	{5,6}	9	∞	10	5	0	4
4	{5,6,4}	7	∞	7	5	0	4

Exemplo 2



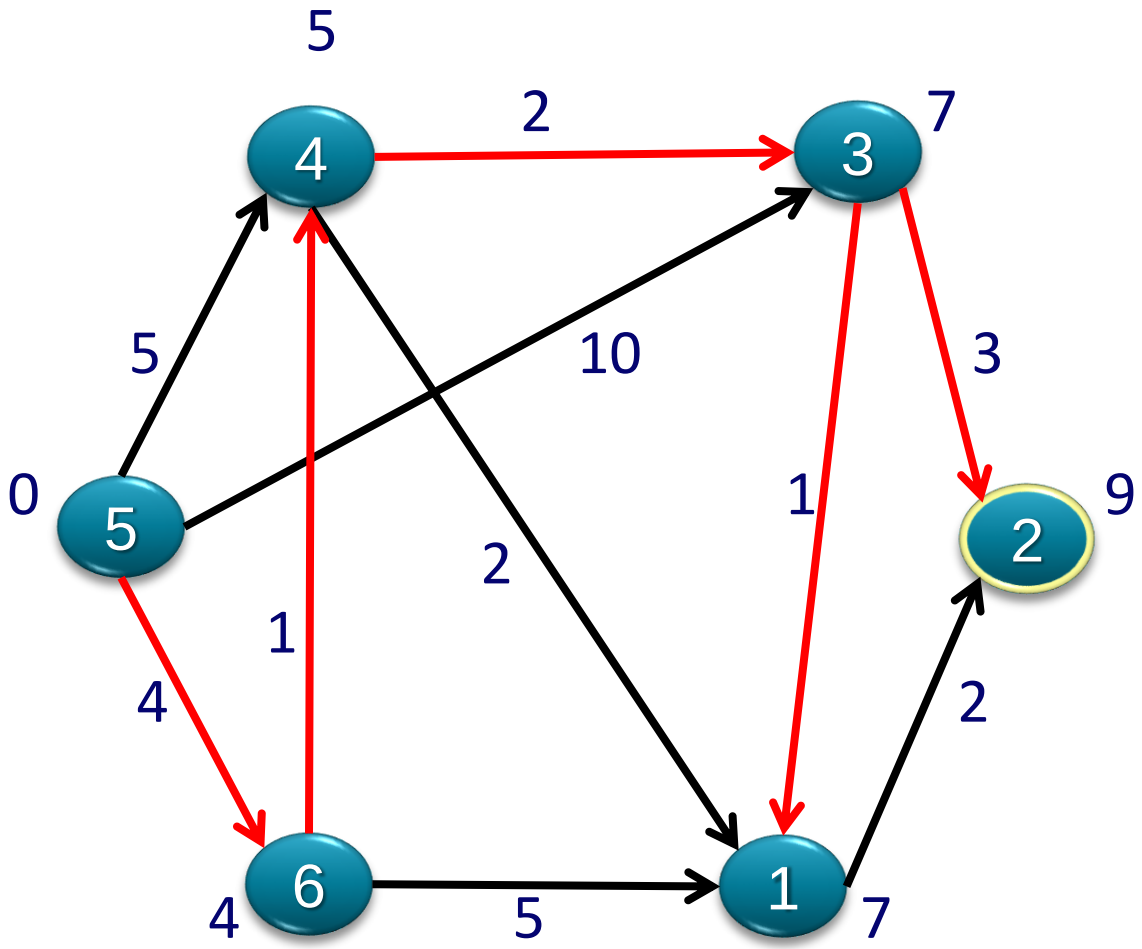
Iter	S	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
1	Ø	∞	∞	∞	∞	∞	∞
2	{5}	∞	∞	10	5	0	4
3	{5,6}	9	∞	10	5	0	4
4	{5,6,4}	7	∞	7	5	0	4
5	{5,6,4,1}	7	9	7	5	0	4

Exemplo 2



Iter	S	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
1	Ø	∞	∞	∞	∞	∞	∞
2	{5}	∞	∞	10	5	0	4
3	{5,6}	9	∞	10	5	0	4
4	{5,6,4}	7	∞	7	5	0	4
5	{5,6,4,1,3}	7	9	7	5	0	4

Exemplo 2



Iter	S	d[1]	d[2]	d[3]	d[4]	d[5]	d[6]
1	Ø	∞	∞	∞	∞	∞	∞
2	{5}	∞	∞	10	5	0	4
3	{5,6}	9	∞	10	5	0	4
4	{5,6,4}	7	∞	7	5	0	4
5	{5,6,4,1,3}	7	9	7	5	0	4
6	{5,6,4,1,3,2}	7	9	7	5	0	4

Custo: 9

Contato

mcastrosouza@live.com

www.geeksbr.com

www.marcoscastro.me

www.twitter.com/mcastrosouza