Universidade do Minho
Escola de Engenharia
Departamento de Produção e Sistemas

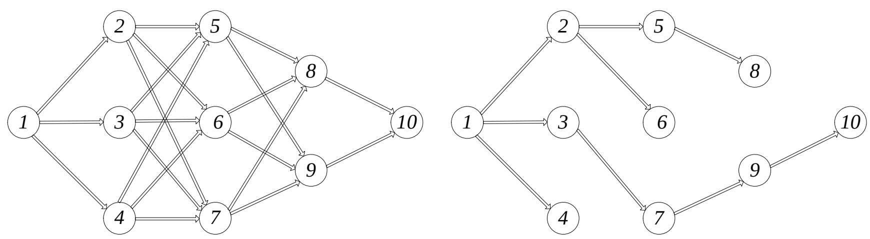
Árvore dos caminhos mais curtos

Filipe Alvelos
falvelos@dps.uminho.pt

Março 2014
Fevereiro, 2016

Árvore dos caminhos mais curtos

- O problema de determinar o caminho mais curto entre um nodo e todos os restantes tem como solução óptima um conjunto de arcos que formam uma árvore de suporte (ignorando a orientação dos arcos)
- Uma árvore é um conjunto de $n - 1$ arcos (em que n é o número de nodos) que não tem ciclos
- Exemplo de uma rede e de uma possível árvore de caminhos mais curtos com raiz em 1:



Árvore dos caminhos mais curtos

- Abordagens para resolver o problema da árvore dos caminhos mais curtos
  - Programação linear / inteira
    - Fácil de implementar (e.g. solver do excel)
    - Fácil de incluir restrições adicionais (e.g. restrição temporal) ou estender para outros problemas (e.g. caminhos disjuntos)
    - Mais pesado computacionalmente do que os algoritmos específicos
  - **Algoritmos específicos**
    - Algoritmo para redes acíclicas
    - Algoritmos para redes que podem ter ciclos
      - Redes sem custos negativos (Dijsktra)
      - Redes com custos arbitrários (Bellman-Ford) – [não abordado nesta UC]
    - Inclusão de restrições ou extensão para outros problemas não são triviais

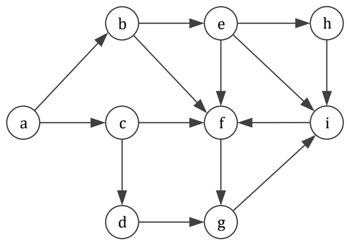
Árvore dos caminhos mais curtos - redes acíclicas

- Para aplicar o algoritmo para redes acíclicas é necessário associar índices aos nodos de forma a que a origem de qualquer arco tenha um índice inferior ao seu destino: $i < j, \forall ij \in A$, i.e. a rede tem de estar ordenada topologicamente
- Só é possível ordenar uma rede topologicamente se a rede não tiver ciclos

```
// Algoritmo de ordenação topológica
// Vector index fica com o novo índice para cada nodo
j=1
Enquanto existirem nodos com grau de entrada = 0
     Seleccionar um nodo i com grau de entrada = 0
     index[i]=j
     Remover da rede todos os arcos com origem em i
     Remover da rede o nodo i
     j=j+1
Se rede está vazia, ordenação topológica concluída
Se não, existe um ciclo (orientado)
```

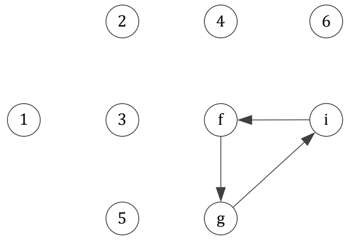
Ordenação topológica

- Exemplo



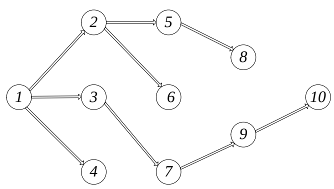
Índice	Nodo	Arcos removidos
1	a	ab, ac
2	b	be, bf
3	c	cd, cf
4	e	eh, ei, ef
5	d	dg
6	h	hi
7	-	-

- Na iteração 7 é detectado o ciclo f-g-i-f logo o algoritmo das redes acíclicas não pode ser aplicado (seria aplicado outro não abordado nesta UC)



Algoritmo para redes acíclicas

- A cada nodo são associdos dois valores que vão sendo actualizados ao longo da execução do algoritmo
  - $d(j)$ – comprimento do caminho desde a raiz até ao nodo *j* com menor comprimento (encontrado até ao momento)
  - $pred(j)$ – nodo que precede o nodo *j* no caminho desde a raiz até *j* com menor comprimento (encontrado até ao momento)
- Condição que tem de se verificar em todos os arcos para os comprimentos corresponderem a caminhos mais curtos
  - $d(j) \leq d(i) + c_{ij}$
- Para os arcos que fazem parte do caminho mais curto, verifica-se
  - $d(j) = d(j) + c_{ij}$



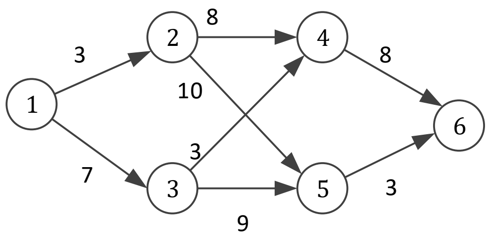
Algoritmo para redes acíclicas

- Algoritmo

```
// Algoritmo para determinar árvore dos caminhos mais curtos em redes
        acíclicas
// Inicialização
Para todos os nodos j
    dist(j)=INF    // INF constante muito elevada
    pred(j)=0      // não há predecessores
dist(1)=0
// Ciclo
Para i=1 até n
    Para todos os arcos ij
        Se dist(j)>dist(i)+custo(ij)
            dist(j)=dist(i)+custo(ij)
            pred(j)=i
```


Algoritmo para redes acíclicas

- Exemplo



i=1, j=2, c_ij=3
$d(j) \leq d(i) + c_{ij}$
$\infty \leq 0+3$

i=3, j=4, c_ij=3
$d(j) \leq d(i) + c_{ij}$
$11 \leq 7+3$

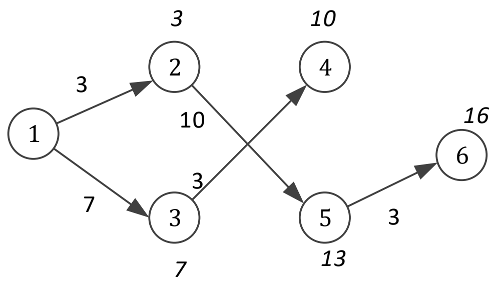
		Nodo (dist,pred)					
		1	2	3	4	5	6
Iteração	0	(0,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
	1		(3,**1**)	(7,**1**)			
	2				(11,2)	(13,**2**)	
	3				(10,**3**)	~~(16,3)~~	
	4						(18,4)
	5						(16,**5**)
	6						

$$d(j) \leq d(i) + c_{ij}$$

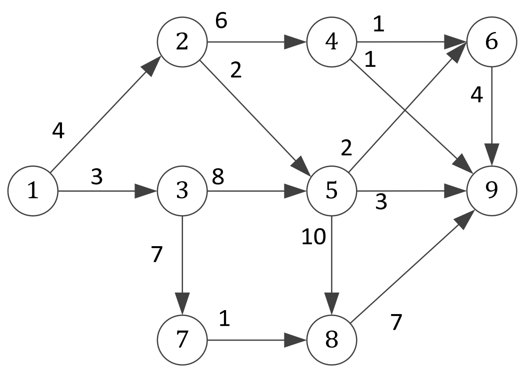
Algoritmo para redes acíclicas

- Solução

Nodo (dist,pred)					
1	2	3	4	5	6
(0,-)	(3,1)	(7,1)	(10,3)	(13,2)	(16,5)

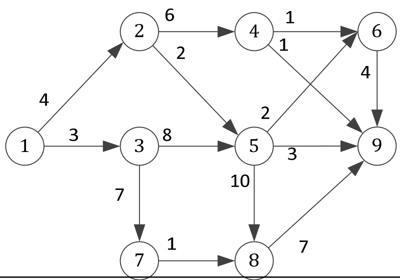


Algoritmo para redes acíclicas



Algoritmo para redes acíclicas

- Exemplo

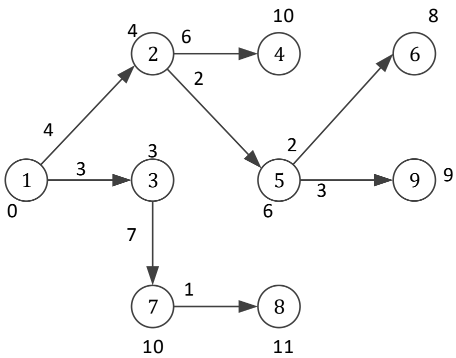


		Nodo (dist,pred)								
		1	2	3	4	5	6	7	8	9
Iteração	0	(0,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
	1		(4,1)	(3,1)						
	2				(10,2)	(6,2)				
	3					~~(11,3)~~		(10,3)		
	4						(11,4)			(11,4)
	5						(8,5)		(16,5)	(9,5)
	6									~~(12,6)~~
	7								(11,7)	
	8									~~(18,8)~~
	9									

Algoritmo para redes acíclicas

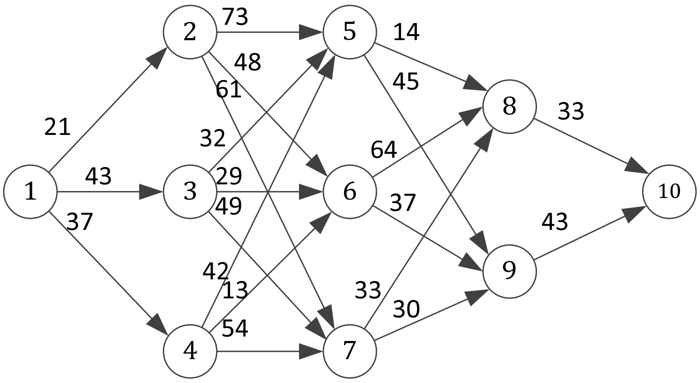
- Solução óptima: pred(9)=5, pred(8)=7, pred(7)=3, pred(6)=5, pred(5)=2, pred(4)=2, pred(3)=1, pred(2)=1

Nodo (dist,pred)								
1	2	3	4	5	6	7	8	9
	(4,1)	(3,1)	(10,2)	(6,2)	(8,5)	(10,3)	(11,7)	(9,5)



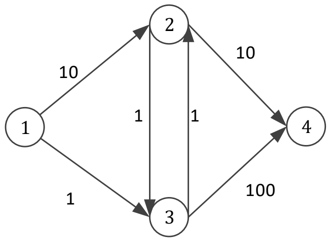
Exercício

- Determine a árvore de caminhos mais curtos com raiz em 1 na rede da figura.



Algoritmo para redes acíclicas

- Algoritmo não funciona para redes ciclícas porque a existência de arcos de um nodo com índice superior para nodo com índice inferior não permite que o caminho mais curto até um nodo *i* seja conhecido na iteração *i*
- No exemplo abaixo, o caminho mais curto entre 1 e 4 é 1-2-3-4 com comprimento 12 e não o obtido! Notar que a rede tem um ciclo e portanto o algoritmo para redes acíclicas não funciona. Notar aidna que o arco 3-2 não respeita a condição *i*<*j*.



		Nodo (dist,pred)			
		1	2	3	4
Iteração	0	(0,0)	(M,0)	(M,0)	(M,0)
	1		(10,1)	(1,1)	
	2			~~(11,2)~~	**(20,2)**
	3		(2,3)		~~(101,3)~~

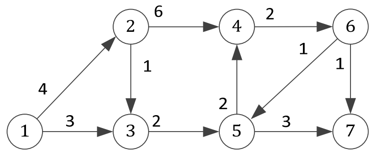
Algoritmo de Dijskstra

- Algoritmo de Dijkstra obtém a árvore de caminhos mais curtos (i.e. o caminho mais curto entre um nodo e cada um dos restantes)
- Aplicável em redes sem custos negativos (podendo ter ciclos)
- Em cada iteração fica determinada a distância entre a fonte e um nodo (algoritmo de rotulação permanente)
- Em cada iteração é tentada a actualização dos caminhos mais curtos que chegam a um conjunto de nodos

Algoritmo de Dijskstra

Algoritmo de Dijkstra para o problema do caminho mais curto
Entrada:
    rede com conjunto de nodos N e conjunto de arcos A
    comprimento de cada arco ij, $c_{ij}, \forall ij \in A$
    nodo fonte s
Saída:
    árvore de caminhos mais curtos representada por $pred(j)$ (nodo predecessor do nodo j) e comprimento de s até j, $d(j)$, $\forall j \in N$.
// Inicialização
Para todos os nodos $j \in N$
    $d(j)=M$
    $pred(j) = 0$
// Para o nodo s fazer
$d(s) = 0$;
$S = \emptyset$
Enquanto $|S| \neq |N|$
    Seleccionar um nodo $i \in S - N$ tal que $d(i) = min\{d(j): j \in S - N\}$
    $S = S \cup \{i\}$;
    Para todos os arcos ij
        Se $d(j) > d(i) + c_{ij}$
            $d(j) = d(i) + c_{ij}$
            $pred(j) = i$

Algoritmo de Dijskstra

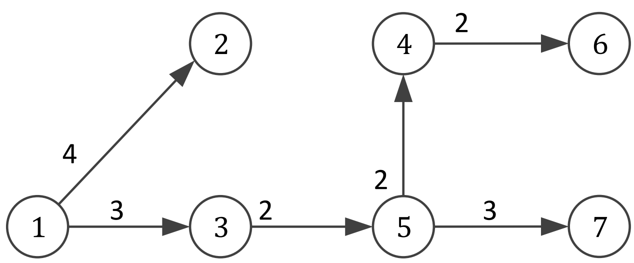


Iteração	i	Nodo (dist,pred)						
		1	2	3	4	5	6	7
0	-	(0,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)
1	1		(4,1)	(3,1)				
2	3					(5,3)		
3	2				(10,2)			
4	5				(7,5)			(8,5)
5	4						(9,4)	
6	7							
7								

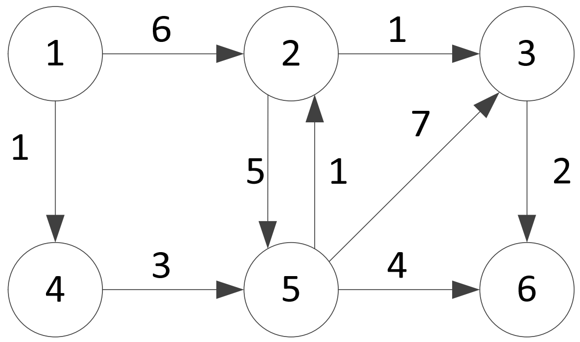
Algoritmo de Dijskstra

- Solução

Nodo (dist,pred)						
1	2	3	4	5	6	7
(0,-)	(4,1)	(3,1)	(7,5)	(5,3)	(9,4)	(8,5)



Algoritmo de Dijskstra



FA, Problemas de caminhos