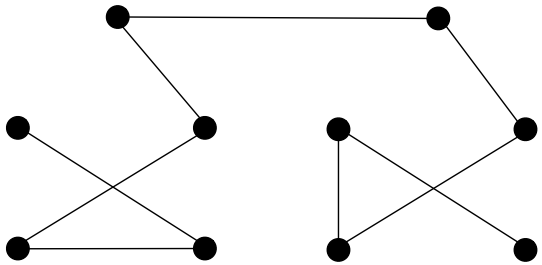
Teoria dos Grafos

Edson Prestes

Teoria dos Grafos

Árvores

Uma árvore é um grafo conexo aciclico, ou seja, um grafo conexo sem ciclos.



Uma folha é um vértice de grau 1.

Uma floresta é um grafo que não contém ciclos.

Uma árvore é uma floresta conexa.

Todo componente de uma floresta é uma árvore.

Teoria dos Grafos

Árvores

Teorema: Para um grafo G=(V,A) de n-vértices (n>0), as seguintes afirmações são verdadeiras (e caracterizam uma árvore com n vértices).

a) G é conexo e não possui ciclos

b) G é conexo e tem n-1 arestas

c) G tem n-1 arestas e nenhum ciclo

d) Para dois vértices u,v tem exatamente um caminho entre u e v.

Trazer a prova na próxima aula

Teoria dos Grafos

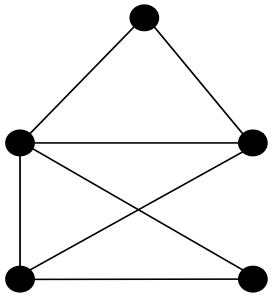
Árvores

Cada aresta de uma árvore é uma aresta de corte.

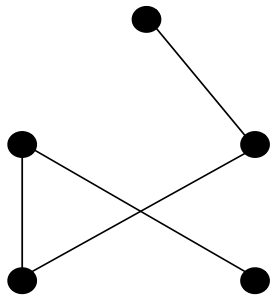
A adição de uma aresta em uma árvore forma exatamente um ciclo.

Cada grafo conexo **que é árvore** contém exatamente uma spanning tree. Uma spanning tree de um Grafo é *um subgrafo que é árvore e que contém todos os nós de G*.

Uma spanning forest é obtida quando o grafo não é conexo. Ela consiste em uma floresta de spanning tree.



Grafo G



Spanning Tree de G

Teoria dos Grafos

Árvores

O diâmetro de uma árvore é calculado de forma similar ao de um grafo que nãoa é árvore. Ele corresponde a maior distância entre qualquer par de vértices, ou seja,

$$diam(G) = \max_{u,v \in V(G)} d(u,v)$$

A excentricidade de um vértice u é a maior distância entre u e qualquer vértice de G, ou seja,

$$\epsilon(u) = \max_{v \in V(G)} d(u,v)$$

O raio de um Grafo G é denotado por

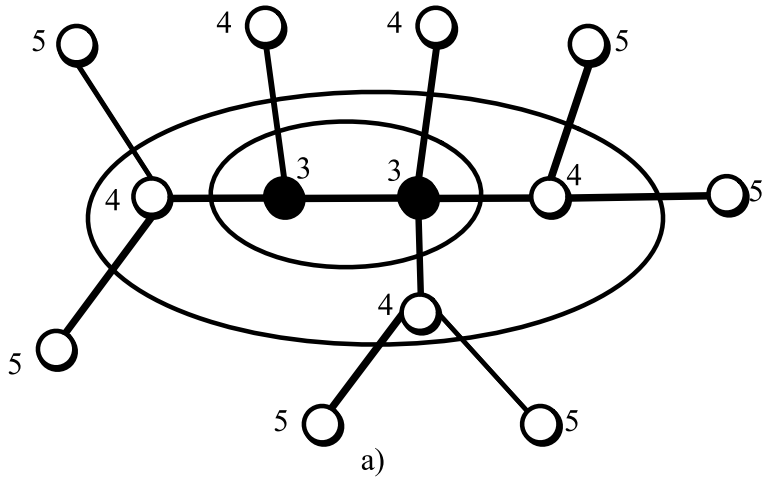
$$\min_{u \in V(G)} \epsilon(u)$$

Teoria dos Grafos

Árvores

O centro de um grafo G é o subgrafo induzido pelos vértices de excentricidade mínima. O centro de uma árvore é um vértice ou uma aresta.

Encontre a excentricidade de cada vértice, o raio e o centro do grafo abaixo.



a)

$$diam(G) = \max_{u,v \in V(G)} d(u,v)$$

$$\epsilon(u) = \max_{v \in V(G)} d(u,v)$$

$$\text{Raio(G)} = \min_{u \in V(G)} \epsilon(u)$$

Teoria dos Grafos

Árvores

Sabemos que com um ou dois vértices apenas uma árvore pode ser formada.

Entretanto com três vértices podemos formar três árvores.

Com quatro vértices temos quatro estrelas e doze caminhos (eliminando os automorfismos) totalizando 16 árvores.

Se tivermos cinco vértices temos 125 árvores.

Cayley demonstrou que para um conjunto de n vértices distintos existem

n^{n-2} árvores associadas.

Cada uma das árvores pode ser codificada por uma lista de comprimento n-2, chamada Código Prüfer.

Esta lista permite-nos determinar de forma unívoca a árvore em questão.

Teoria dos Grafos

Árvores – Código Prüfer

Este algoritmo recebe como entrada uma árvore T com um conjunto S de n vértices.

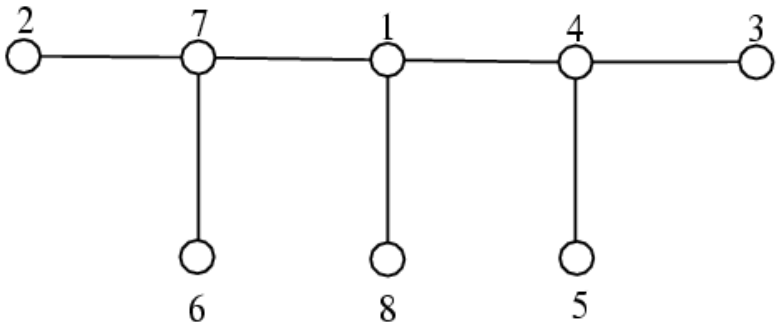
A cada passo, o algoritmo remove a folha b_i com menor rótulo e armazena o seu vizinho, a_i.

Isto é feito até restar apenas uma única aresta.

Ao final do processo, teremos uma (n-2)-upla com os nós não folhas de T.

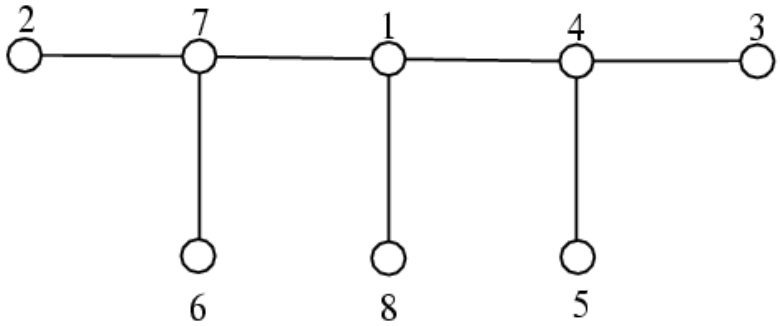
A partir desta tupla e do conjunto S é possível recuperar a árvore T.

Calcule o código Prüfer da árvore abaixo



Teoria dos Grafos

Árvores – Código Prüfer



A cada passo, o algoritmo remove a folha b_i com menor rótulo e armazena o seu vizinho, a_i

Iteração	S	Sel	Viz
0	$\{1,2,3,4,5,6,7,8\}$	$\emptyset$	$\emptyset$
1	$\{1,3,4,5,6,7,8\}$	$\{2\}$	$\{7\}$
2	$\{1,4,5,6,7,8\}$	$\{2,3\}$	$\{7,4\}$
3	$\{1,4,6,7,8\}$	$\{2,3,5\}$	$\{7,4,4\}$
4	$\{1,6,7,8\}$	$\{2,3,5,4\}$	$\{7,4,4,1\}$
5	$\{1,7,8\}$	$\{2,3,5,4,6\}$	$\{7,4,4,1,7\}$
6	$\{1,8\}$	$\{2,3,5,4,6,7\}$	$\{7,4,4,1,7,1\}$

O código Prüfer da árvore acima é (7,4,4,1,7,1)

Teoria dos Grafos

Árvores

A recuperação da árvore a partir do código prüfer é como segue.

Inicialmente são criadas uma seqüência e um conjunto de vértices: a seqüência S que representa o código Prüfer e o conjunto V dos vértices que não aparecem no código Prüfer.

Em seguida contruimos uma floresta com todos os vértices da árvore em questão. A cada passo, pegamos o primeiro elemento, a_1, de S e o menor elemento m de V e criamos a aresta (a_1,m).

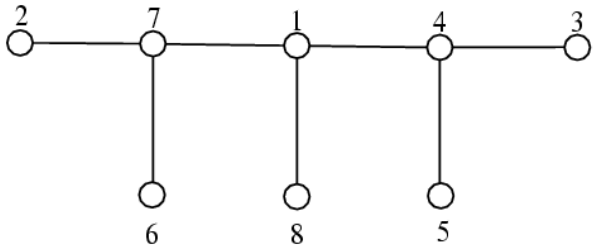
Removemos tanto a_1 quanto m de seus locais de origem. Se após este processo a_1 não aparecer mais em S, então o incluimos em V. O processo é repetido até que S seja o conjunto vazio. Quando S for o conjunto vazio unimos os dois vértices que sobraram em V.

Recupere a árvore associada ao código Prüfer (7,4,4,1,7,1)

Teoria dos Grafos

Árvores

Iteração	S	V	Arestas
0	$\{7,4,4,1,7,1\}$	$\{2,3,5,6,8\}$	$\emptyset$
1	$\{4,4,1,7,1\}$	$\{3,5,6,8\}$	$(7,2)$
2	$\{4,1,7,1\}$	$\{5,6,8\}$	$(7,2),(4,3)$
3.a	$\{1,7,1\}$	$\{6,8\}$	$(7,2),(4,3),(4,5)$
3.b	$\{1,7,1\}$	$\{6,8,4\}$	$(7,2),(4,3),(4,5)$
4	$\{7,1\}$	$\{6,8\}$	$(7,2),(4,3),(4,5),(1,4)$
5.a	$\{1\}$	$\{8\}$	$(7,2),(4,3),(4,5),(1,4),(7,6)$
5.b	$\{1\}$	$\{8,\mathbf{7}\}$	$(7,2),(4,3),(4,5),(1,4),(7,6)$
6.a	$\emptyset$	$\{8\}$	$(7,2),(4,3),(4,5),(1,4),(7,6)$ $(1,7)$
6.b	$\emptyset$	$\{8,\mathbf{1}\}$	$(7,2),(4,3),(4,5),(1,4),(7,6)$ $(1,7)$
6.c	$\emptyset$	$\{8,1\}$	$(7,2),(4,3),(4,5),(1,4),(7,6)$ $(1,7)(\mathbf{1},\mathbf{8})$



Teoria dos Grafos

Árvores – Código Prüfer

Dado um conjunto de inteiros positivos $d_1,d_2,\ldots,d_n$ totalizando 2n-2 (*2n-2, deve-se ao fato de que uma árvore com n vértices possui exatamente n-1 arestas.Logo, a soma dos graus de cada vértice é igual a 2(n-1)*) então existem

$$\frac{(n-2)!}{\Pi(d_i-1)!}$$

árvores com um conjunto de n vértices tal que o grau do vértice i é d_i

Prova: Observe que cada vértice não folha x é registrado no código prüfer exatamente d_x-1 vezes no código prüfer, pois ele tem d_x vértices vizinhos e após a remoção de seus d_x-1 vizinhos "folha", ele será folha de seu último vizinho. Logo ele aparecerá d_x-1 vezes.

O código prüfer possui n-2 elementos e pode ter (n-2)! permutações, entretanto devido a repetição de alguns vértices, teremos várias permutações iguais. Para cada vértice x que aparece no código, teremos (d_x-1)! permutações iguais. Logo a retirada destes elementos iguais é dada por

$$\frac{(n-2)!}{\Pi(d_i-1)!}$$

Teoria dos Grafos

Árvores – Código Prüfer

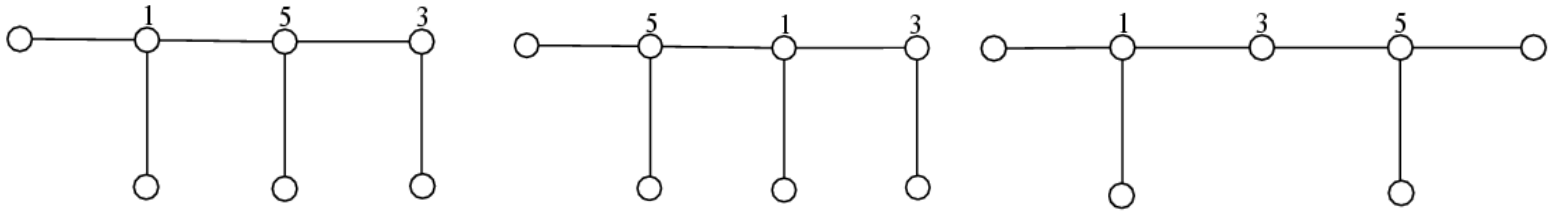
Considere o seguinte conjunto de vértices S={1,2,3,4,5,6,7} e os graus em ordem de cada elemento de S {3,1,2,1,3,1,1}.

Quantas árvores podemos gerar a partir deste conjunto ?

Considerando que temos como vértices não folha os vértices {1,3,5} teremos

$$\frac{(n-2)!}{(d_1-1)!(d_3-1)!(d_5-1)!} = \frac{5!}{2!1!2!} = 30.$$

Árvores. Alguns exemplos podem ser vistos abaixo



Teoria dos Grafos

Árvores – Spanning Tree

Teorema: O número de árvores enraizadas com vértices não distintos é definido pela seguinte função geradora

$$T(x) = x \prod_{r=1}^{\infty} (1 - x^r)^{-T_r}$$

Onde T_r é o número de árvores enraizadas com r vértices

Teorema: O número de árvores não enraizadas com vértices não distintos é definido por

$$t(x) = T(x) - \frac{1}{2}(T^2(x) - T(x^2))$$

Trabalho : calcular a quantidade de árvores enraizadas e não enraizadas com vértices não distintos usando 2,3,4,5 e 6 vértices

Teoria dos Grafos

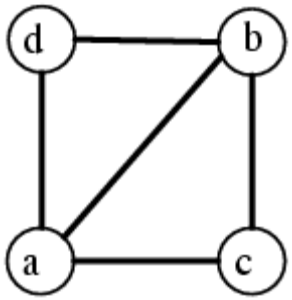
Árvores – Spanning Tree

Teorema: Dado um grafo G simples com um conjunto de vértices $V=\{v_1, v_2, ..., v_n\}$ faça $a_{i,j}$ ser o número de arestas entre os vértices v_i e v_j. Construa uma matriz Q quadrada de ordem **n** de forma que a entrada Q(i,j) seja igual a

$-a_{i,j}$ se $i \neq j$ e igual a $d(v_i)$ se i=j. Se Q* é a matriz obtida removendo a linha **s** e coluna **t** de Q, então o número de spanning trees de G é igual a

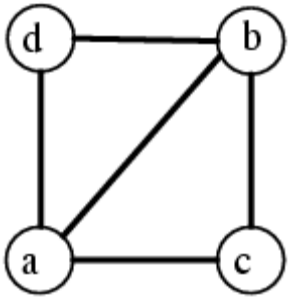
$$(-1)^{s+t} \mid Q^* \mid$$

Determine o número de spanning trees do grafo abaixo



Teoria dos Grafos

Árvores – Spanning Tree



M

	a	b	c	d
a	3	-1	-1	-1
b	-1	3	-1	-1
c	-1	-1	2	0
d	-1	-1	0	2

Removendo linha e coluna c

Q*

	a	b	d
a	3	-1	-1
b	-1	3	-1
d	-1	-1	2

Grau dos vértices

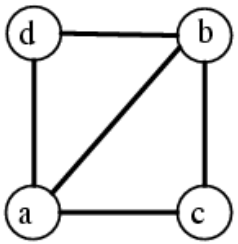
Quantidade de arestas

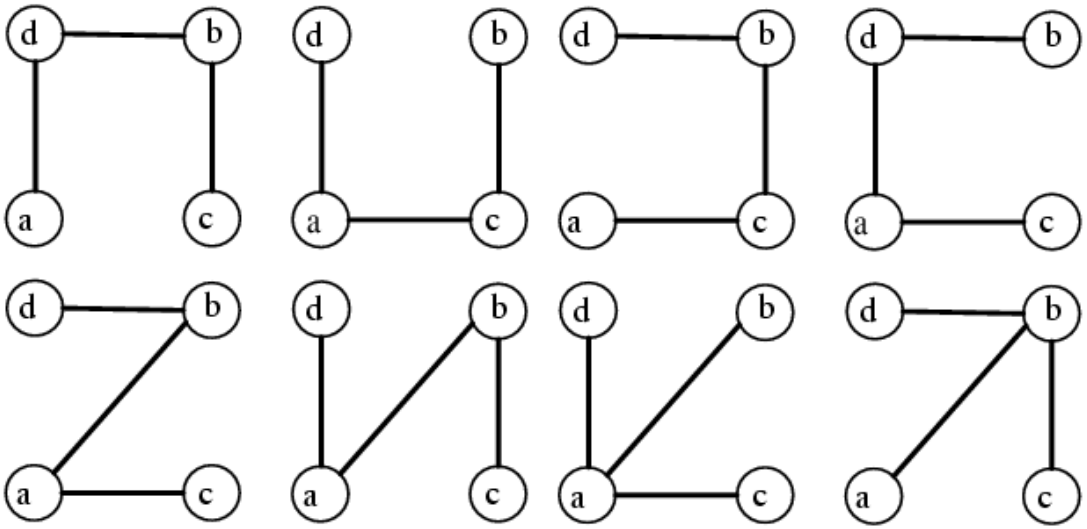
→ $(-1)^{3+3} \cdot |Q^*| = (-1)^{3+3} \cdot (18-1-1-(3+2+3)) = 8$

Teoria dos Grafos

Árvores – Spanning Tree

Liste as 8 spanning trees do grafo.





Teoria dos Grafos

Árvores – Algoritmo de Kruskal

O algoritmo de Kruskal permite determinar a spanning tree de custo mínimo. Este custo corresponde à soma dos pesos (distância, tempo, qualidade, ...) associados a cada aresta do grafo.

O algoritmo recebe como entrada **um grafo G conexo** com pesos e monta um **grafo desconexo G'**, o qual corresponde a uma floresta de árvores composta unicamente pelos vértices de G.

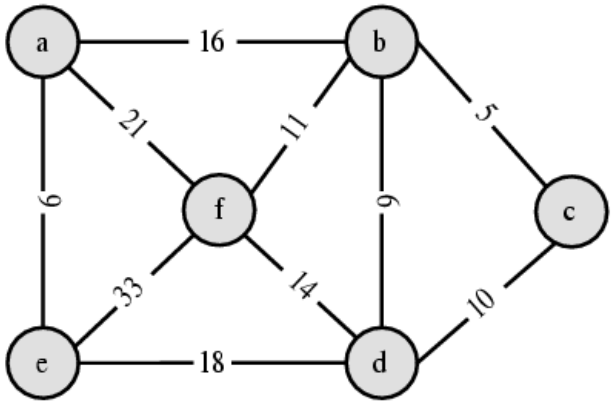
Em seguida, ele ordena **as arestas de G** em ordem crescente e seleciona a cada instante a de menor peso. A aresta selecionada é marcada, para não ser analisada mais tarde, e verificada se pode ser adicionada ao grafo G' de forma a não gerar ciclos.

O processo termina quando G' estiver conexo.

Teoria dos Grafos

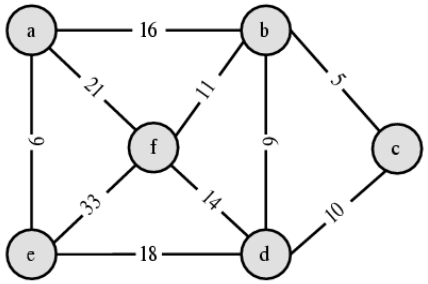
Árvores – Algoritmo de Kruskal

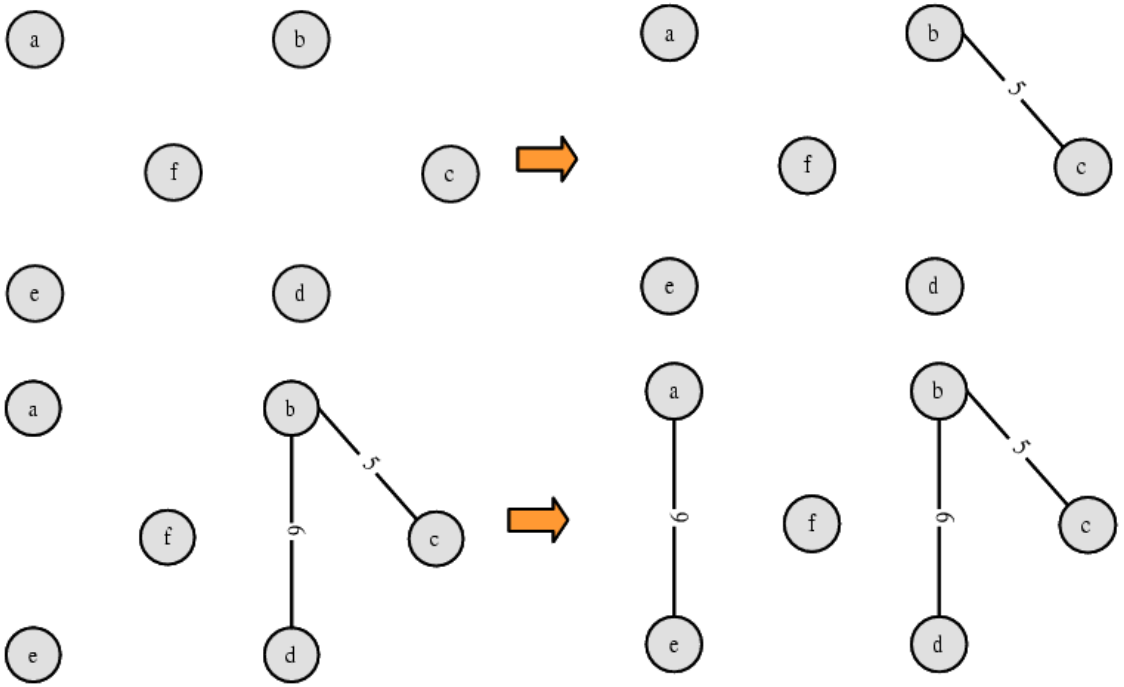
Determine a *spanning tree* de custo mínimo no grafo abaixo usando o algoritmo de Kruskal



Teoria dos Grafos

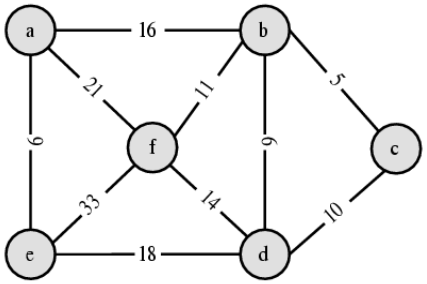
Árvores – Algoritmo de Kruskal

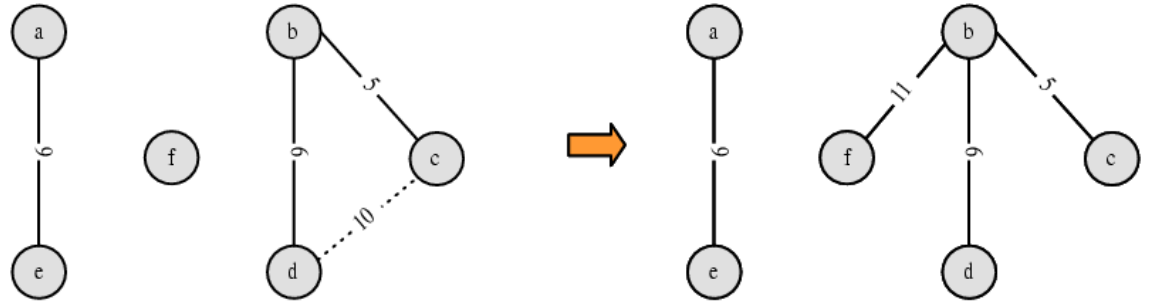


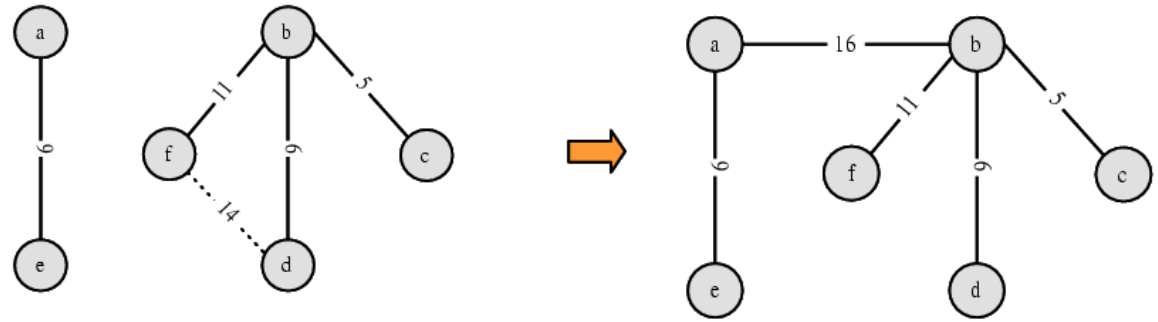


Teoria dos Grafos

Árvores – Algoritmo de Kruskal







Teoria dos Grafos

Árvores – Algoritmo de Dijkstra

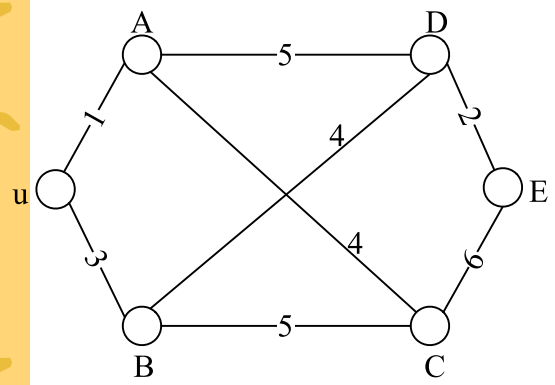
O algoritmo de Dijkstra é usado para determinar a menor rota entre duas posições em um grafo. Ele assume que o caminho entre dois vértices, **u** e **v**, **é composto sempre dos menores caminhos entre dois vértices quaisquer componentes do caminho**.

O algoritmo considera um grafo G (ou dígrafo) com arestas de pesos positivos e um vértice inicial **u**. O peso da aresta formada pelos vértices u e v é **w(u,v)**. Se **u** e **v** não são adjacentes então w(u,v)= ∞

Ele considera que existe um conjunto **S** de vértices tal que o menor caminho a partir de **u** até cada vértice de **S** é conhecido.

Teoria dos Grafos

Árvores – Algoritmo de Dijkstra



Iteração	S	u	a	b	c	d	e
0	$\{u\}$	0	1	3	∞	∞	∞
1	$\{ua\}$	0	1	3	5	6	∞
2	$\{uab\}$	0	1	3	5	6	∞
3	$\{uabc\}$	0	1	3	5	6	11
4	$\{uabcd\}$	0	1	3	5	6	8
5	$\{uabcde\}$	0	1	3	5	6	8

Inicialmente, S={u}, t(u)=0, t(z)=w(u,z) para z $\neq$ u.

A cada iteração seleciona-se um vértice $v = \arg\min_{z \notin S} t(z)$ e adiciona-o a S.

Em seguida, as arestas a partir de **v**, **(v,z)**, são exploradas e para cada z $\notin$ S, a nova distância aproximada t(z) é atualizada com

min{t(z), t(v)+w(v,z)}.

O processo continua até S=V(G) ou até t(z)= ∞ para todo z $\notin$ S.