Capítulo V

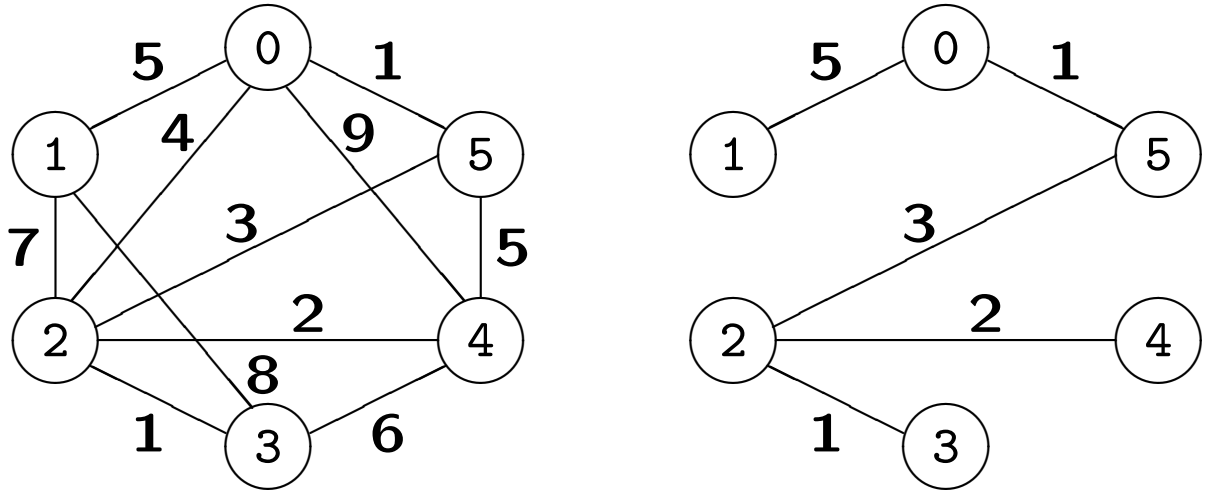
Árvore Mínima de Cobertura (num grafo não orientado)

—

Algoritmo de Kruskal

Problema

Como ligar um dado equipamento, minimizando o comprimento total da ligação?

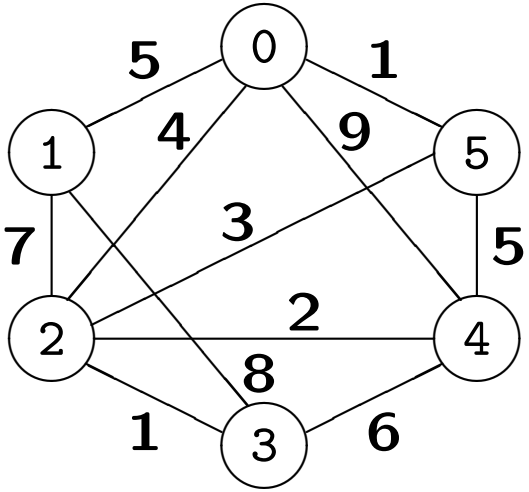


Árvore de Cobertura (sub-grafo acíclico e conexo com todos os vértices) de custo Mínimo (nenhuma árvore de cobertura tem custo menor).

Dado um grafo **não orientado e conexo**, como encontrar uma

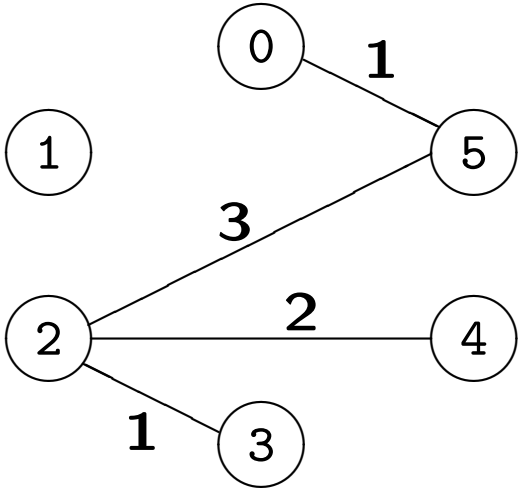
Árvore Mínima de Cobertura?

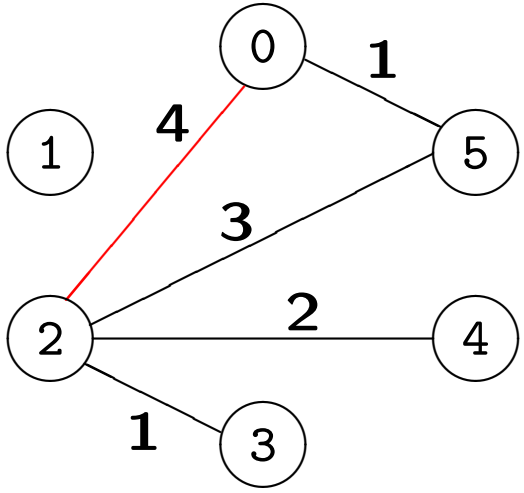
Algoritmo de Kruskal [1956]

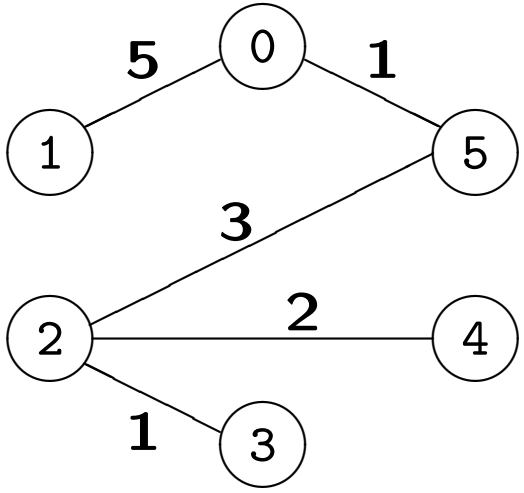


1	$0 \longleftrightarrow 5$	$\surd$
1	$2 \longleftrightarrow 3$	$\surd$
2	$2 \longleftrightarrow 4$	$\surd$
3	$2 \longleftrightarrow 5$	$\surd$
4	$0 \longleftrightarrow 2$	
5	$0 \longleftrightarrow 1$	$\surd$
5	$4 \longleftrightarrow 5$	
6	$3 \longleftrightarrow 4$	
7	$1 \longleftrightarrow 2$	
8	$1 \longleftrightarrow 3$	
9	$0 \longleftrightarrow 4$	

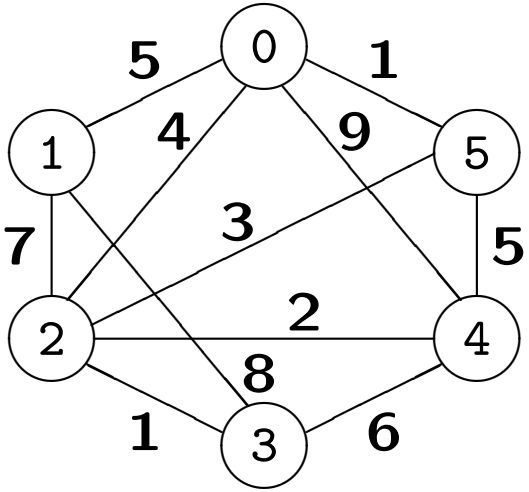
Custo da Árvore: **12**



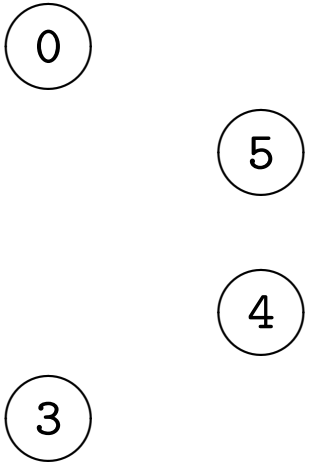




Há ciclo? (1)

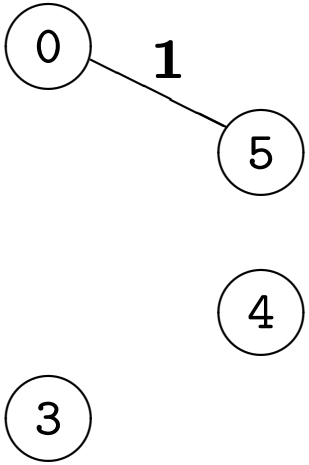


$(0,5)$?

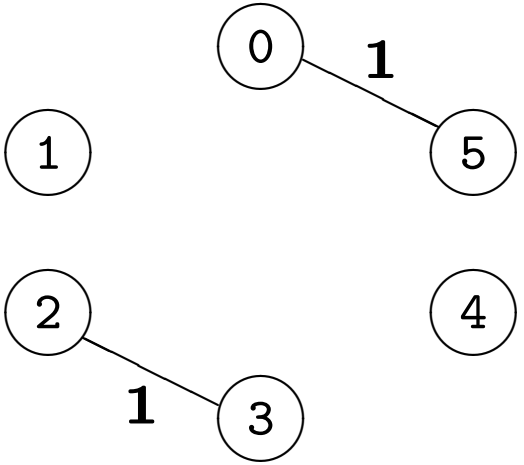


$\{\{0\},\{1\},\{2\},\{3\},\{4\},\{5\}\}$

$(2,3)$?



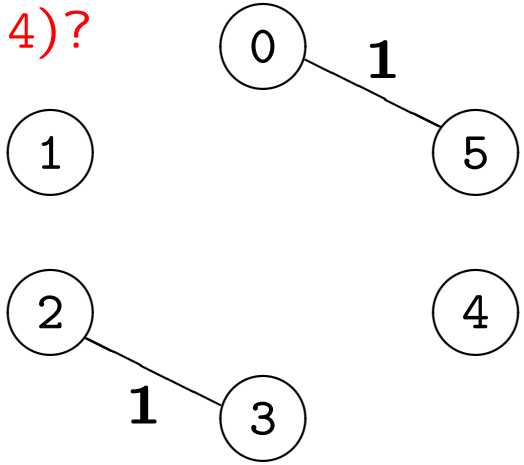
$\{\{0,5\},\{1\},\{2\},\{3\},\{4\}\}$



$\{\{0,5\},\{1\},\{2,3\},\{4\}\}$

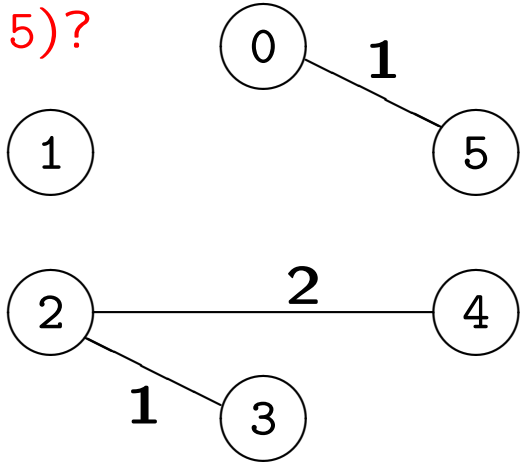
Há ciclo? (2)

$(2,4)$?



$\{\{0,5\},\{1\},\{2,3\},\{4\}\}$

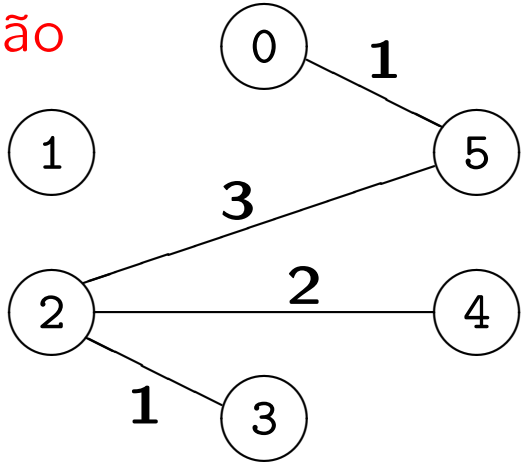
$(2,5)$?



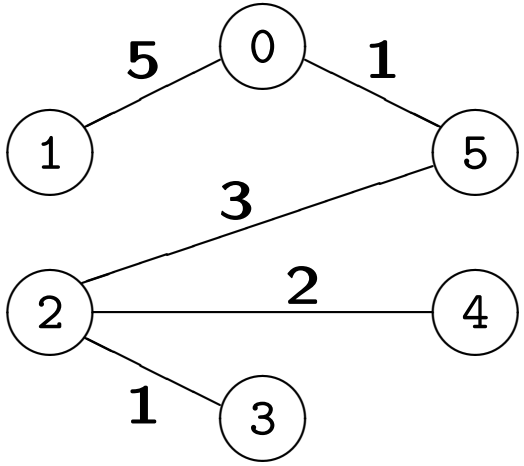
$\{\{0,5\},\{1\},\{2,3,4\}\}$

$(0,2)$? Não

$(0,1)$?



$\{\{0,5,2,3,4\},\{1\}\}$



$\{\{0,1,2,3,4,5\}\}$

TAD Partição (com n elementos)

Os elementos dos conjuntos são $0, 1, 2, \ldots, n-1$. Cada conjunto é identificado por um dos seus elementos, denominado **o representante** do conjunto.

Domínio $= \{0, 1, ..., n-1\}$

// Cria a partição $\{\{0\}, \{1\}, \ldots, \{n-1\}\}$.
Partição **cria**(**int** n);

// Devolve o representante do conjunto ao qual e pertence.
Domínio **representante**(Domínio e);

// Substitui os conjuntos C_e e C_f, cujos representantes são e e f,
// respetivamente, pelo conjunto $C_e \cup C_f$.
// **Pré-condição:** $e \neq f$ (ou seja, $C_e \neq C_f$).
void **união**(Domínio e, Domínio f);

Interface Partição (com n elementos)

```
public interface UnionFind
{
    // Creates the partition {{0},{1},...,{domainSize − 1}}.
    // UnionFind( int domainSize );

    // Returns the representative of the set that contains
    // the specified element.
    int find( int element ) throws InvalidElementException;

    // Removes the two distinct sets S1 and S2 whose representatives
    // are the specified elements, and inserts the set S1 ∪ S2.
    // The representative of the new set S1 ∪ S2 can be any of
    // its members.
    void union( int representative1, int representative2 ) throws
        InvalidElementException, NotRepresentativeException,
        EqualSetsException;
}
```

Construir a Fila com Prioridade de Arcos

```
@SuppressWarnings( "unchecked" )
MinPriorityQueue<L, Edge<L>> buildQueue( UndiGraph<L> graph )
{
    Entry<L, Edge<L>>[] auxArray =
        (Entry<L, Edge<L>>[]) new Entry[ graph.numEdges() ];
    int pos = 0;
    for every Edge<L> e in graph.edges()
        auxArray[pos++] = new EntryClass<L, Edge<L>>(e.label(), e);
    MinPriorityQueue<L, Edge<L>> priQueue =
        new MinHeap<L, Edge<L>>(auxArray);
    return priQueue;
}
```

Árvore Mínima de Cobertura (1)

```
Iterator<Edge<L>> mstKruskal( UndiGraph<L> graph )
{
    MinPriorityQueue<L, Edge<L>> allEdges = buildQueue(graph);

    UnionFind vertPartition =
        new UnionFindInArray( graph.numVertices() );

    List<Edge<L>> mst = new DoublyLinkedList<Edge<L>>();

    int mstFinalSize = graph.numVertices() - 1;
```

Árvore Mínima de Cobertura (2)

```
    while ( mst.size() < mstFinalSize )
    {
        Edge<L> edge = allEdges.removeMin().getValue();
        Vertex[] endPoints = edge.endVertices();
        int rep1 = vertPartition.find( endPoints[0] );
        int rep2 = vertPartition.find( endPoints[1] );
        if ( rep1 != rep2 )
        {
            mst.addLast(edge);
            vertPartition.union(rep1, rep2);
        }
    }

    return mst.iterator();
}
```

Complexidade

Identificação das Operações

criar heap	$\Theta(\|A\|)$
criar partição	?
criar lista ligada	$\Theta(1)$
Ciclo (executado entre $\|V\| - 1$ e $\|A\|$ vezes)	
1 remover mínimo	$O(\log \|A\|)$
2 representante	?
Ciclo (executado $\|V\| - 1$ vezes)	
1 inserir à cauda	$\Theta(1)$
1 união	?

Complexidade do Algoritmo de Kruskal

União sem Estratégia

Representante sem Efeitos Laterais

criar heap	$\Theta(\|A\|)$
criar partição	$\Theta(\|V\|)$
criar lista ligada	$\Theta(1)$
Ciclo (executado entre $\|V\|-1$ e $\|A\|$ vezes)	
1 remover mínimo	$O(\log \|A\|)$
2 representante	$O(\|V\|)$
Ciclo (executado $\|V\|-1$ vezes)	
1 inserir à cauda	$\Theta(1)$
1 união	$\Theta(1)$
TOTAL	$O(\|A\| \times \|V\|)$

Complexidade do Algoritmo de Kruskal

União sem Estratégia

Representante sem Efeitos Laterais

Complexidade do Primeiro Ciclo

$$O(\quad |A| \times \log |A| \quad + \quad |A| \times (2\mathbf{R}) \quad)$$

$$O(\quad \underbrace{|A| \times \log |A|}_{|A| < |V|^2} \quad + \quad |A| \times |V| \quad)$$

$$O(\quad |A| \times \log |V| \quad + \quad |A| \times |V| \quad)$$

$$O(\quad |A| \times |V| \quad)$$

Complexidade do Algoritmo de Kruskal

União por Altura ou por Tamanho

Representante sem Efeitos Laterais

criar heap	$\Theta(\|A\|)$
criar partição	$\Theta(\|V\|)$
criar lista ligada	$\Theta(1)$
Ciclo (executado entre $\|V\|-1$ e $\|A\|$ vezes)	
1 remover mínimo	$O(\log \|A\|)$
2 representante	$O(\log \|V\|)$
Ciclo (executado $\|V\|-1$ vezes)	
1 inserir à cauda	$\Theta(1)$
1 união	$\Theta(1)$
TOTAL	$O(\|A\| \times \log \|V\|)$

Complexidade do Algoritmo de Kruskal

União por Altura ou por Tamanho

Representante sem Efeitos Laterais

Complexidade do Primeiro Ciclo

$$O(\ \ |A| \times \log |A| \ \ + \ \ |A| \times (2\mathbf{R}) \ \ )$$

$$O(\ \ |A| \times \log |V| \ \ + \ \ |A| \times \log |V| \ \ )$$

$$O(\ \ \ \ \ \ \ \ \ \ |A| \times \log |V| \ \ \ \ \ \ \ \ \ \ )$$

Complexidade do Algoritmo de Kruskal

União por Nível ou por Tamanho

Representante com Compressão do Caminho

criar heap	$\Theta(\|A\|)$
criar partição	$\Theta(\|V\|)$
criar lista ligada	$\Theta(1)$
Ciclo (executado entre $\|V\|-1$ e $\|A\|$ vezes)	
1 remover mínimo	$O(\log \|A\|)$
2 representante	$O(\log \|V\|)$
Ciclo (executado $\|V\|-1$ vezes)	
1 inserir à cauda	$\Theta(1)$
1 união	$\Theta(1)$
TOTAL	$O(\|A\| \times \log \|V\|)$

Complexidade do Algoritmo de Kruskal

União por Nível ou por Tamanho

Representante com Compressão do Caminho

Complexidade do Primeiro Ciclo

$$O(\ |A| \times \log |A| \ + \ \underbrace{|A| \times (2\mathbf{R})}_{2|A| \geq 2(|V|-1) \geq |V|}\ )$$

$$O(\ |A| \times \log |V| \ + \ (2\,|A|)\ \alpha(2\,|A|, |V|)\ )$$

$$O(\ |A| \times \log |V| \ + \ |A|\ )$$

$$O(\ |A| \times \log |V|\ )$$