Grafos – Parte 1

Aleardo Manacero Jr.

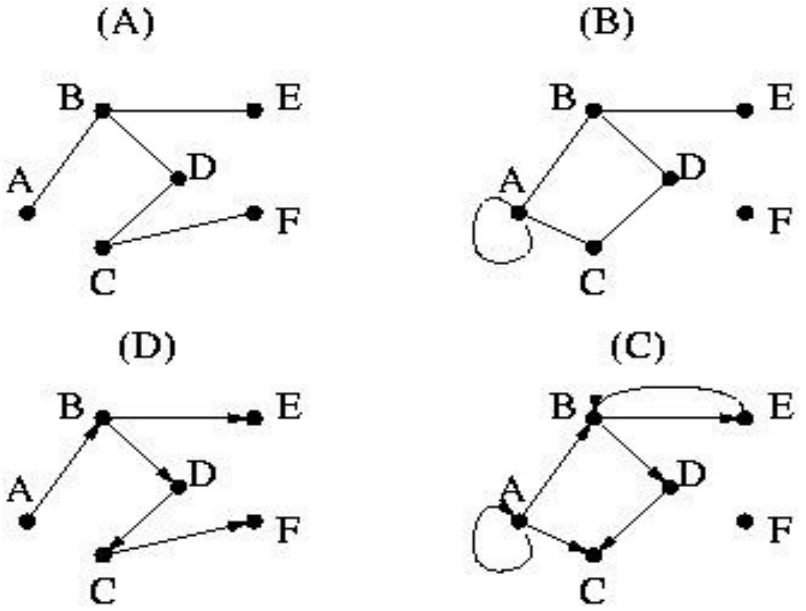
Uma breve introdução

- Grafos são estruturas bastante versáteis para a representação de diversas formas de sistemas e/ou problemas

- Na realidade, árvores e listas podem ser entendidas como especializações de grafos, de forma a atender certas restrições

Uma breve introdução

- Um grafo pode ser definido como uma coleção de vértices (nós) e possivelmente arestas (arcos) ligando tais vértices, como em:



Algumas definições

- Grau de um vértice
  - número de arestas que incidem no vértice
- Circuito
  - é definido como o caminho fechado que sai de um vértice e retorna a ele mesmo
- Relação entre grau e número de arestas
  - num grafo qualquer a soma dos graus de seus vértices é igual ao dobro do número de arestas

Algumas definições

- Grafos conexos
  - são grafos em que não existem vértices isolados
- Grafos dirigidos (dígrafos)
  - são grafos em que as arestas estabelecem uma relação de ordem entre os vértices que unem
- Grafos bipartidos
  - são grafos em que se pode separar os vértices em dois conjuntos de forma que as arestas sempre ligam vértices de conjuntos distintos

Algumas definições

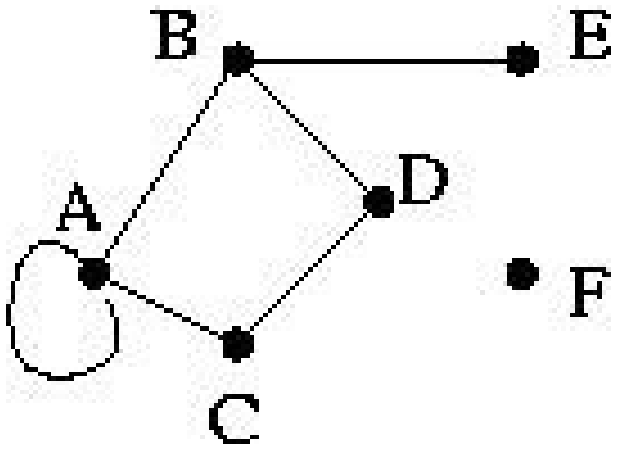
- Grafos planares
  - são grafos em que se pode desenhar sua estrutura de forma plana, sem que ocorra cruzamentos de arestas
- Árvores
  - são grafos conexos em que não se formam circuitos a partir de qualquer vértice

Algumas definições

- Isomorfismo
  - é a propriedade observada entre dois grafos em que a estrutura topológica observada nos dois é idêntica
- Grafos ponderados
  - são grafos em que as arestas possuem pesos, usados no estabelecimento de caminhos

Representação

- Lista de adjacências



a – a, b, c

b – a, d, e

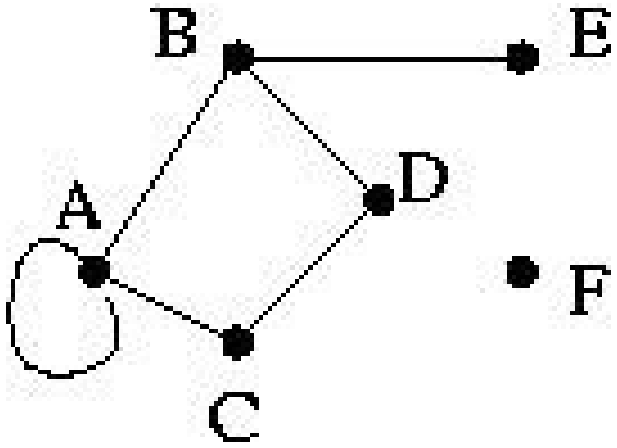
c – a, d

d – b, c

e – b

f –

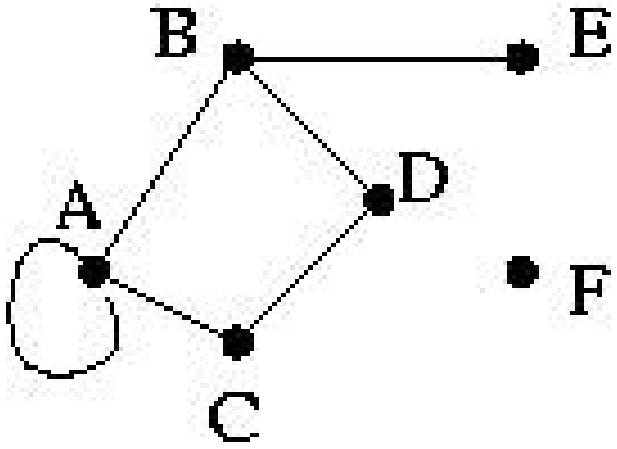
Representação



- Matriz de adjacências

	a	b	c	d	e	f
a	\	\	\	.	.	.
b	\	.	.	\	\	.
c	\	.	.	\	.	.
d	.	\	\	.	.	.
e	.	\	.	.	.	.
f	.	.	.	.	.	.

Representação



- Matriz de incidências

	a-a	a-b	a-c	b-d	b-e	c-d
a	\	\	\	.	.	.
b	.	\	.	\	\	.
c	.	.	\	.	.	\
d	.	.	.	\	.	\
e	.	.	.	.	\	.
f	.	.	.	.	.	.

Percursos em grafos

- Um dos problemas clássicos em grafos é o de determinação de caminhos, em especial o de caminhos mínimos
- Existem diversos algoritmos para a determinação de caminhos mínimos
- Examinaremos aqui os de Dijkstra e o de Floyd

Algoritmo de Dijkstra

- Determina o caminho de um vértice aos demais vértices de um dígrafo com arestas ponderadas
- Usa uma estratégia gulosa (*greedy*) na determinação do caminho
- Detalhes do algoritmo podem ser vistos em livros de ED, Grafos e Redes de Computadores

Algoritmo de Dijkstra

- Parte-se de um vértice com distância conhecida (a origem, com d=0) e determinam-se as distâncias dele aos seus vizinhos

- Isso é repetido para cada vizinho, atualizando-se os valores de distância caso encontre-se caminho menor

Algoritmo de Dijkstra

- Exemplo

	a	b	c	d	e	f
a	3	4	1	N	N	N
b	N	N	N	1	5	N
c	N	N	N	5	N	N
d	N	N	N	N	N	N
e	N	N	N	N	N	N
f	N	N	N	N	N	N

Algoritmo de Floyd

- Trata-se de um algoritmo para determinação de caminhos mínimos entre quaisquer par de vértices

- Trabalha com a acumulação de distâncias a cada iteração, através da matriz de adjacências

Algoritmo de Floyd


```
Para i=1 até |V|
    Para j=1 até |V|
        Para k=1 até |V|

            Se ( peso[j][k] > peso[j][i]+peso[i][k] )
                peso[j][k] = peso[j][i]+peso[i][k]
```

Algoritmo de Floyd

- Exemplo

	a	b	c	d	e	f
a	٣	٤	١	N	N	N
b	N	N	N	١	٥	N
c	N	N	N	٥	N	N
d	N	N	N	N	N	N
e	N	N	N	N	N	N
f	N	N	N	N	N	N

Grafos com pesos negativos

- O problema é que ao passar por arestas com pesos negativos mudamos completamente o estado da solução
- De certo modo, esse problema é o mesmo que aparece em caminhos mínimos em grafos não dirigidos
- Uma solução é usar o algoritmo de Bellman-Ford

Árvores de espalhamento

- Outro problema importante em grafos é o de árvore de espalhamento (*spanning tree*)
- Aqui o problema é encontrar uma árvore que contenha todos os vértices de um dado grafo (em geral a árvore com profundidade mínima)
- Não existe para grafos desconexos

Árvores de espalhamento

- Existem diversas estratégias para a determinação das árvores de espalhamento mínimo

- Se diferenciam na forma de atuar sobre o grafo, ou mais precisamente, na forma de criação de árvores e ramos das mesmas

Árvores de espalhamento

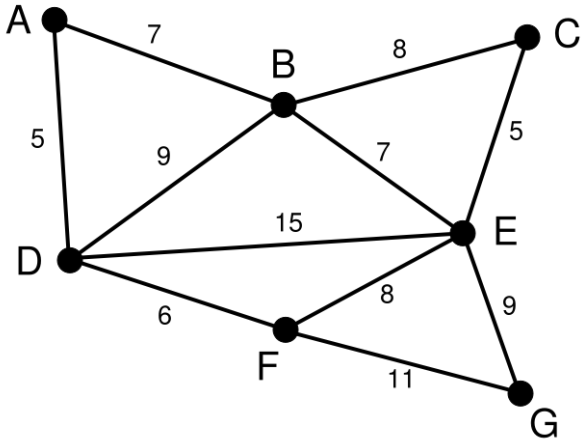
- Os principais algoritmos incluem:
  - Kruskal – expande um conjunto de árvores para formar uma árvore de espalhamento
  - Prim (Jarnik-Prim) – cria e expande apenas uma árvore, adicionando ramos a ela
  - Apaga-Reverso – trabalha de forma inversa ao algoritmo de Kruskal

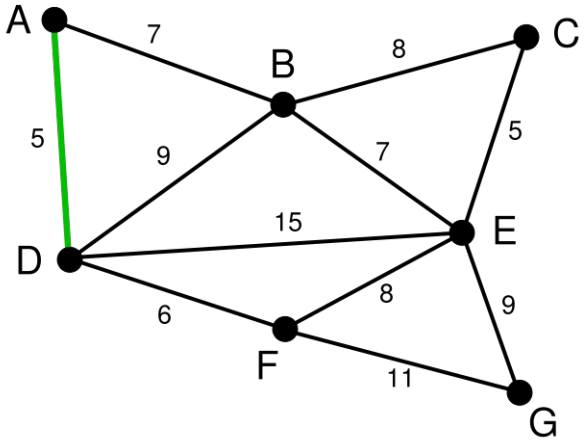
Algoritmo de Kruskal

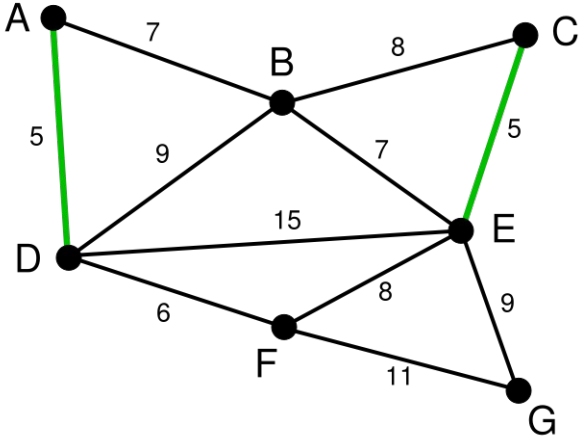
- Inicialmente ordena todas as arestas, segundo seus pesos

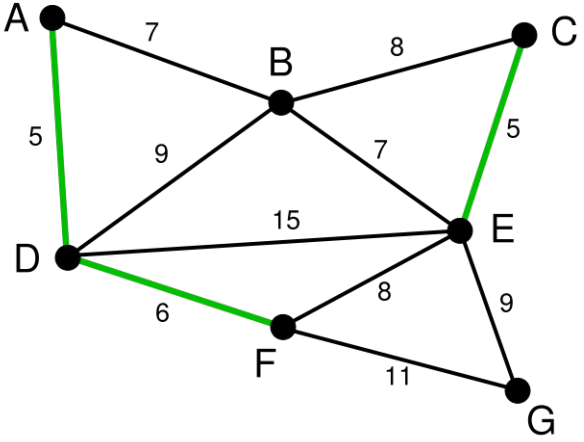
- Para cada aresta verifica se ela pode ser adicionada à árvore em construção, ou seja, se ela não introduz um ciclo na árvore

Algoritmo de Kruskal

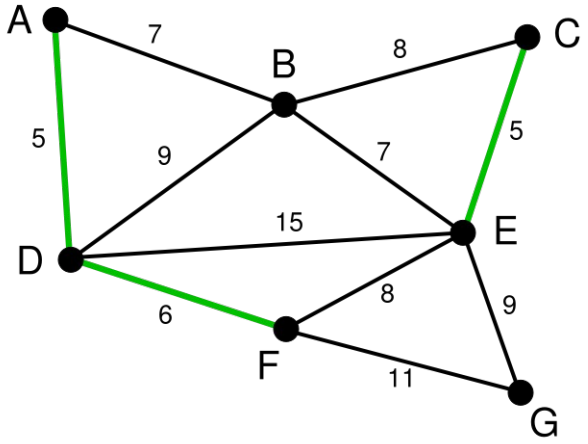


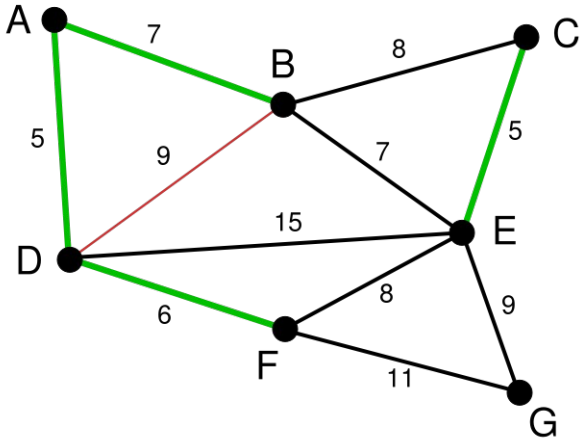


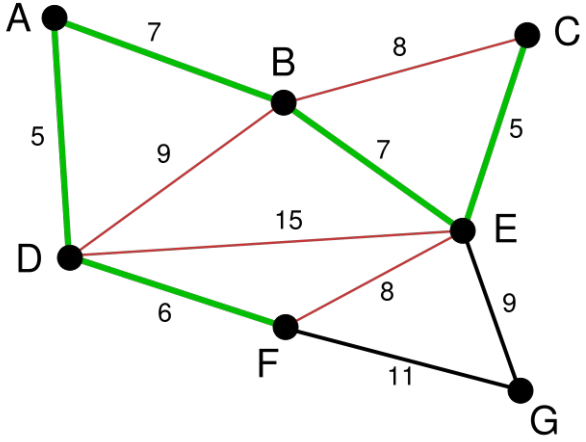


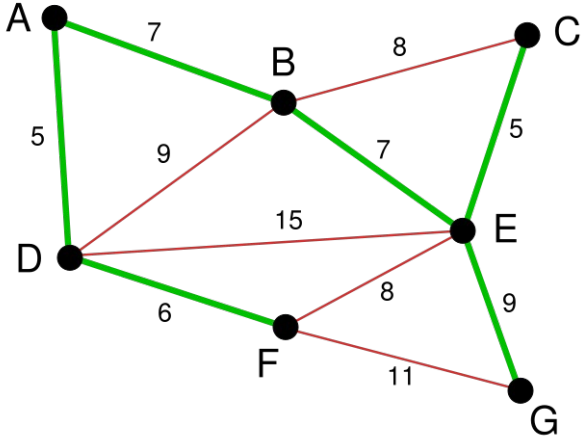


Algoritmo de Kruskal







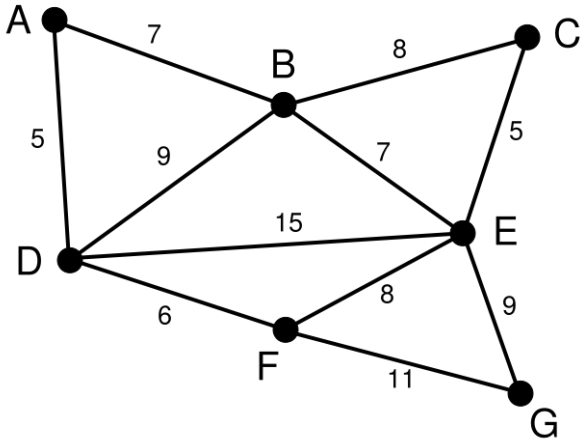


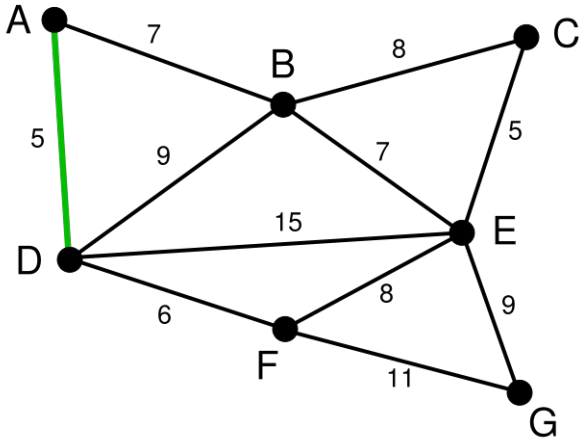
Algoritmo de Prim

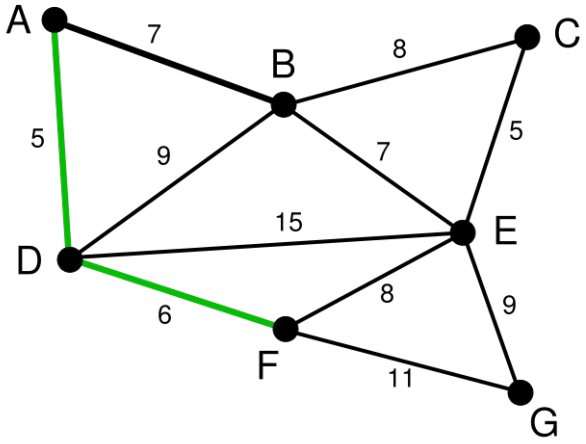
- Inicialmente ordena todas as arestas, segundo seus pesos

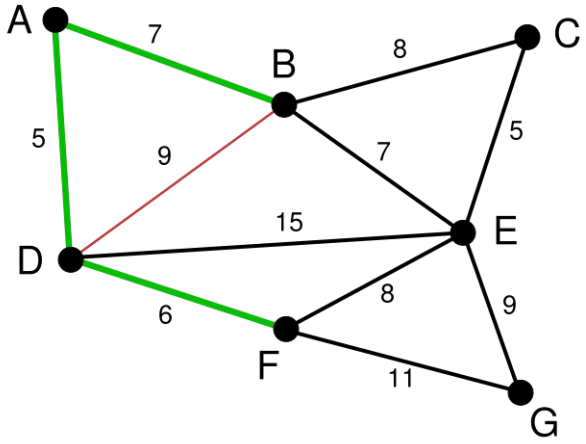
- Para cada aresta verifica se ela pode ser adicionada à árvore em construção, ou seja, se ela não introduz um ciclo na árvore e seja incidente sobre algum vértice já na árvore

Algoritmo de Prim

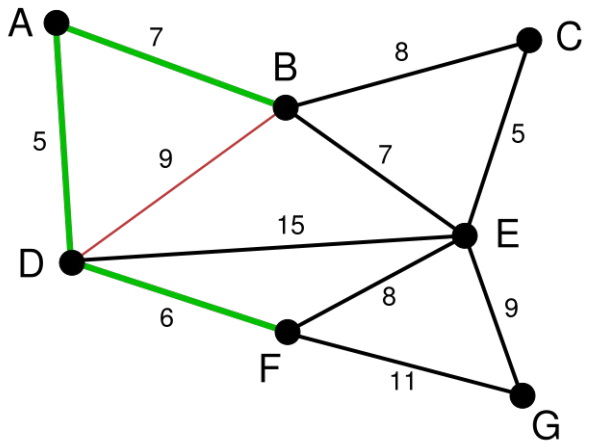


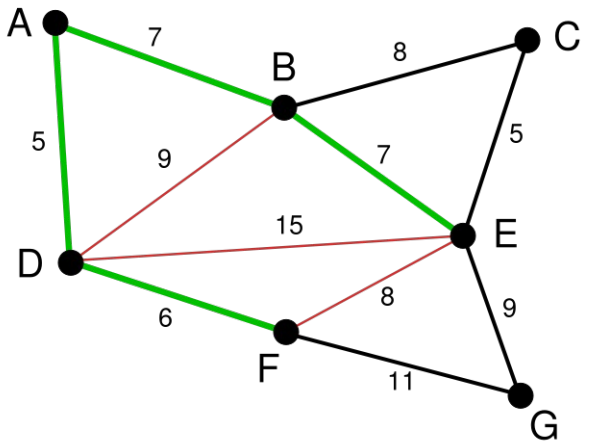


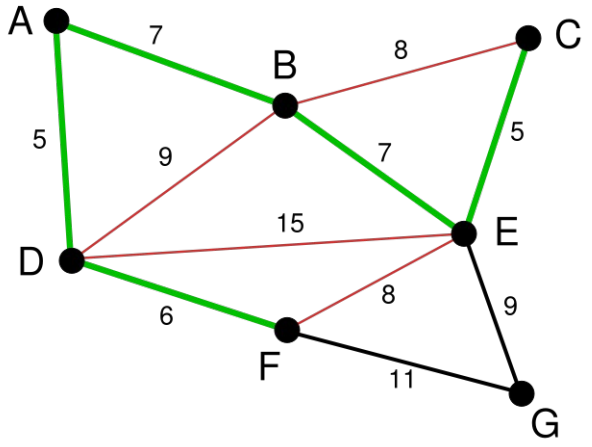


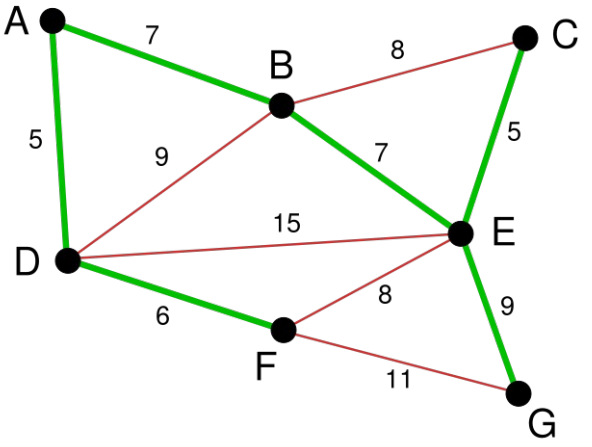


Algoritmo de Prim









Algoritmo Apaga-Reverso

- Parte do grafo completo
- Escolhe então a aresta de maior peso ainda não examinada
- Se a remoção dessa aresta não gerar grafos desconexos, faz sua retirada
- Mantém a aresta no grafo caso sua remoção gere grafos desconexos
- Repete até que nenhuma aresta possa ser retirada