ALGORITMO GULOSO

Greedy algorithm

Algoritmo guloso

- Algoritmo guloso;
- Problema do troco;
- Problema do empacotamento;
- Vantagens e desvantagens;
- Árvores de peso mínimo;
- Algoritmo de Prim;
- Algoritmo de Kruskal.

Características básicas

- É um paradigma de programação;
- Aplicado a problemas de otimização, em que se quer encontrar a melhor solução;
- Para cada fase, é visado escolher a melhor opção, sem verificar as consequências;
- Nunca volta atrás após de uma escolha (sem backtracking);
- Nem sempre produz uma solução ótima!

Problema do troco

- O problema do troco é um exemplo fundamental para introdução do algoritmo guloso:
- Se deseja devolver um troco de R$7,88;
- Possuindo apenas moedas de R$1 R$0,50 R$0,25 R$0,10 R$0,05 R$0,01
- Como seria possível entregar o troco utilizando o mínino de moedas possíveis?

Problema do troco

- R$7,88:

- R$7,88 – 7 * R$1,00 = R$0,88
- R$0,88 – 1 * R$0,50 = R$0,38
- R$0,38 – 1 * R$0,25 = R$0,13
- R$0,13 – 1 * R$0,10 = R$0,03
- R$0,03 – 0 * R$0,05 = R$0,03
- R$0,03 – 3 * R$0,01 = R$0,00
- Troco devolvido seguindo algoritmo guloso.

Problema do empacotamento

- O problema do empacotamento também é um exemplo fundamental para introdução do algoritmo guloso:
- Durante uma escavação em uma mina, foram encontrados diversos tipos de metais valiosos, cada um com sua quantidade disponível e valor por quilo.

Metal	Ouro	Prata	Bronze
Quantidade	2Kg	4Kg	10Kg
Valor	$210,00	$100,00	$75,00

- O minerador poderia levar para si somente um carrinho (que pode carregar apenas 8Kg) com metais da mina, como ele poderia arranjar os metais de forma com que conseguisse carregar o maior valor possível?

Problema do empacotamento

- Sempre escolhendo o que tem o melhor benefício por quilo:

Metal	Ouro	Prata	Bronze
Quantidade	2Kg	4Kg	10Kg
Valor	$210,00	$100,00	$75,00

Benefício por quilo	$105,00	$25,00	$7,50

Carrinho: 8Kg	2	$105,00	4	$25,00	2	$7,50
		$210,00		$100,00		$15,00

Lucro final:	$325,00

Problema do empacotamento binário

- Um programa de TV sugeriu que o participante de uma prova pudesse levar para casa tudo o que ele conseguisse colocar em um carrinho;
- Foram disponibilizados alguns tipos de eletrodomésticos, cada um deles possuem um valor e ocupam uma quantidade de espaços;

Eletrodoméstico	TV 4k	Walkman	Máquina de lavar
Espaço ocupado	6	2	10
Valor	$1000,00	$100,00	$2000,00

- Como seria possível colocar aproveitar ao máximo o espaçamento do carrinho, sabendo que ele só possui 12 espaços?
- Obs: Binário pois não se tem valor um produto cortado ao meio, logo ou se pega ele todo ou não.

Problema do empacotamento binário

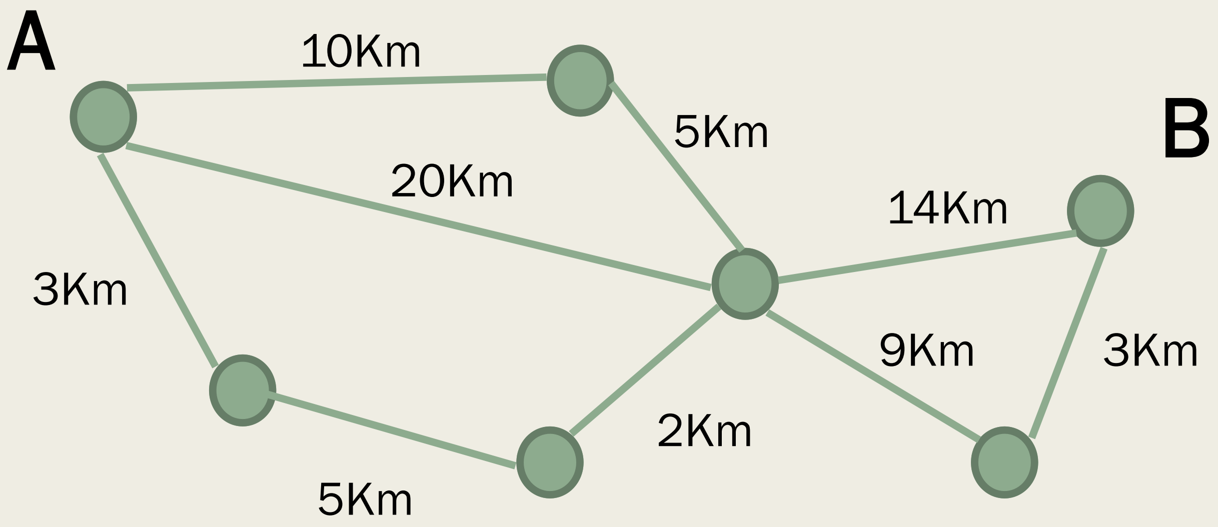
Eletrodoméstico	TV 4k	Walkman	Máquina de lavar
Espaço ocupado	6	2	10
Valor	$1000,00	$100,00	$2000,00

Carrinho: 12 espaços	0	$0,00	1	$100,00	1	$2000,00

Lucro final:	$2100,00

Problema da viagem

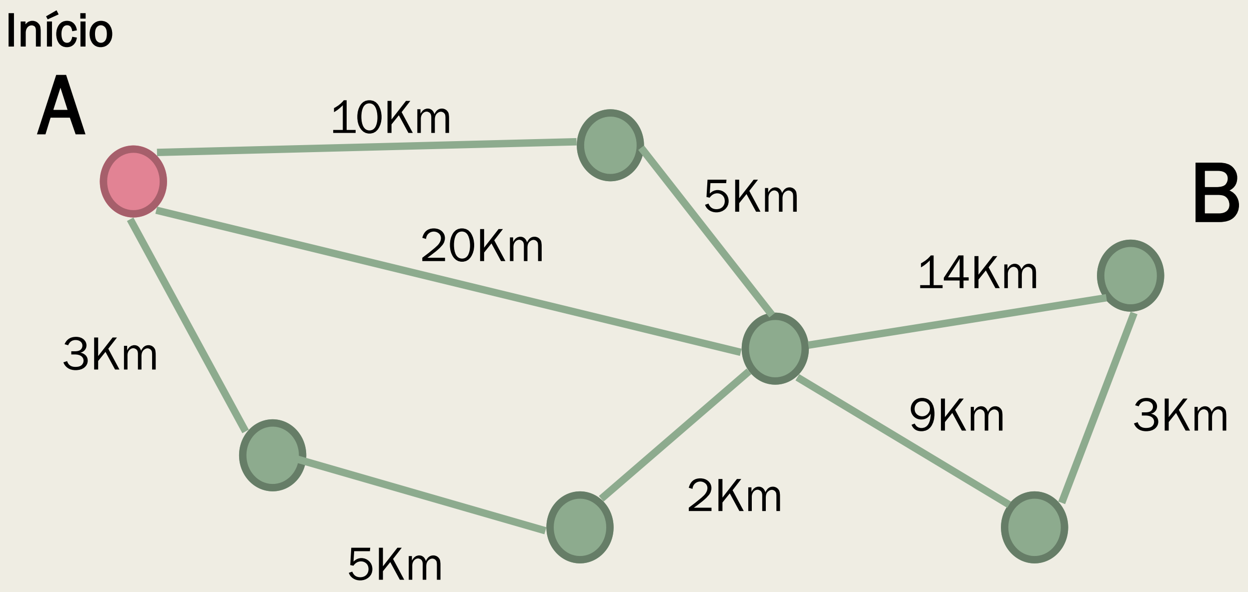
- Como percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Segundo o algoritmo guloso, seria escolhido o menor caminho localmente.

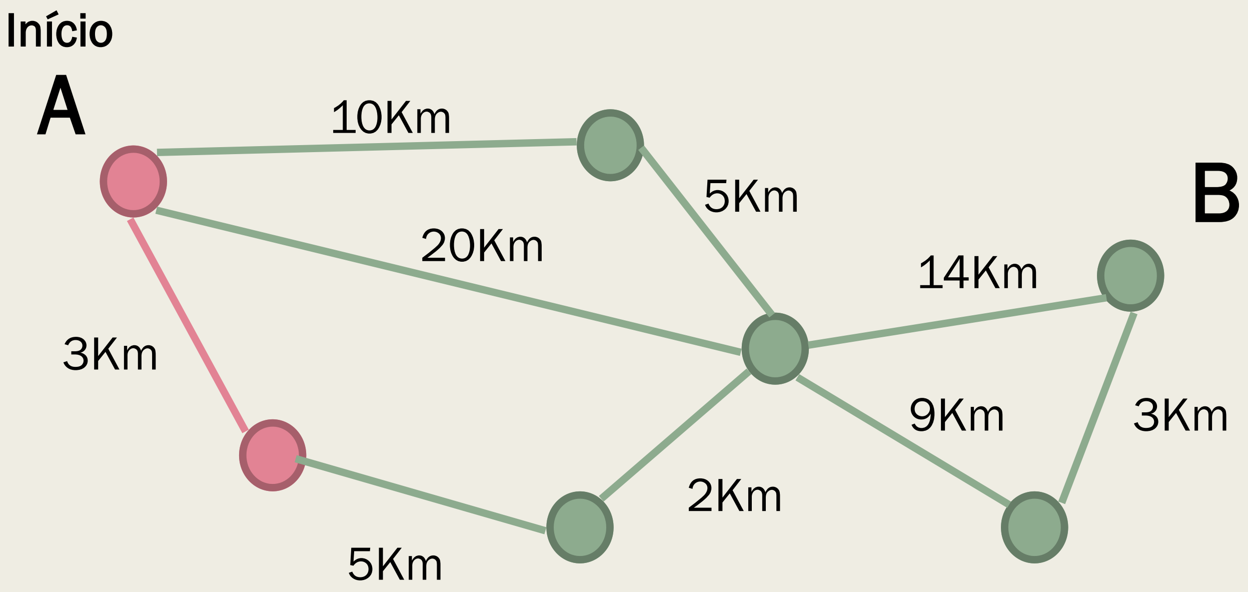
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



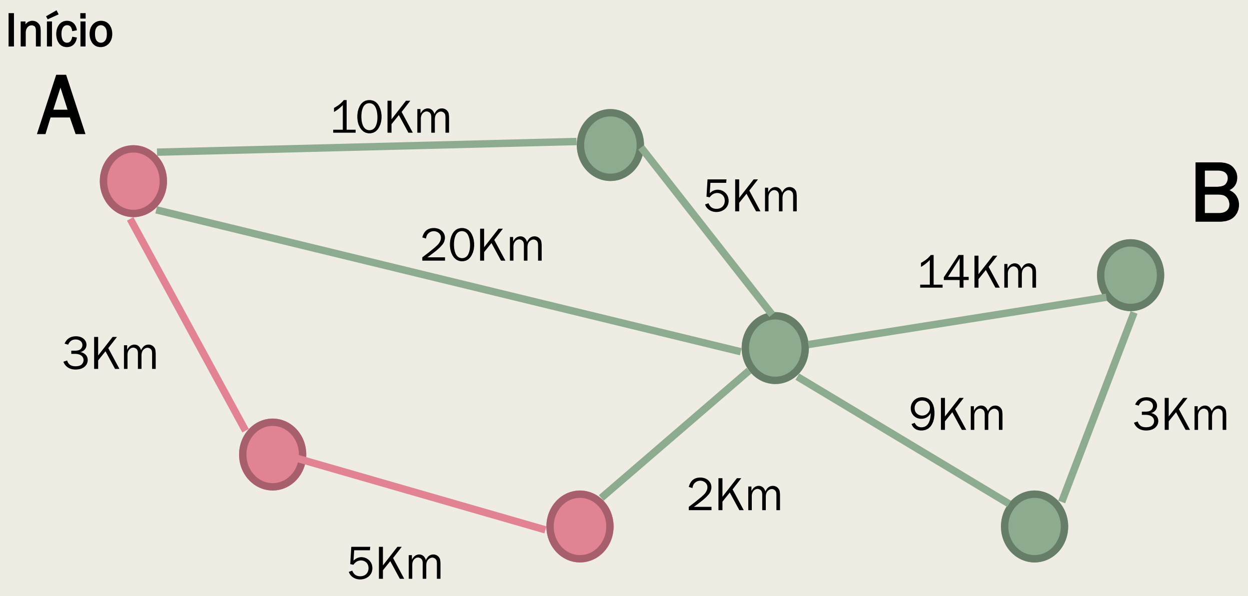
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



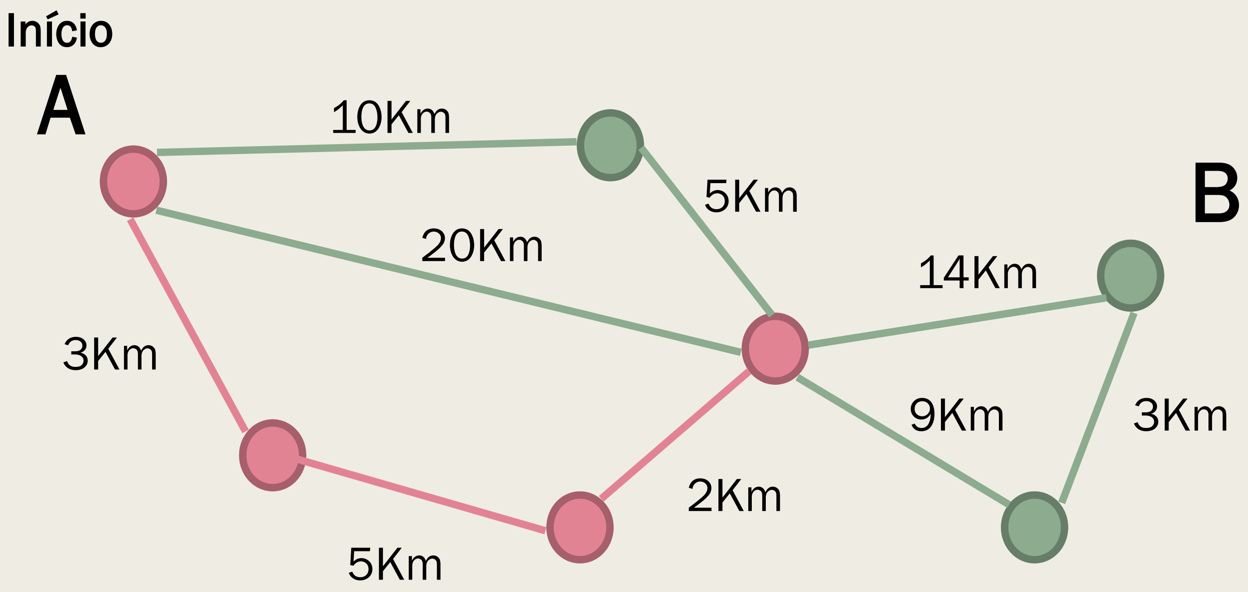
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



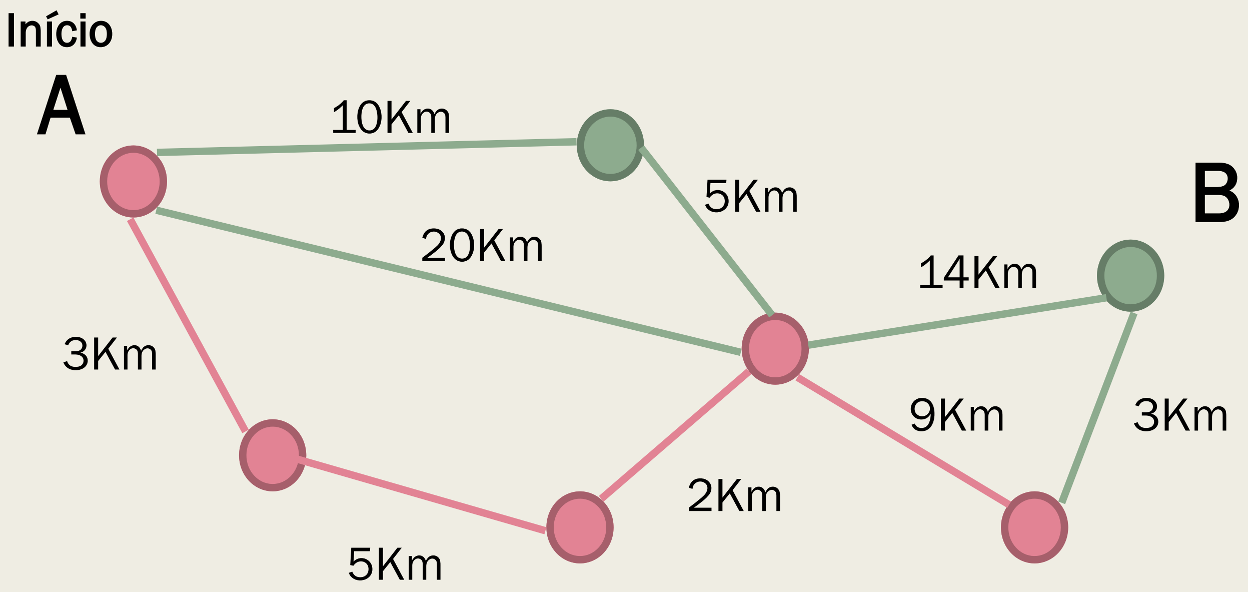
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



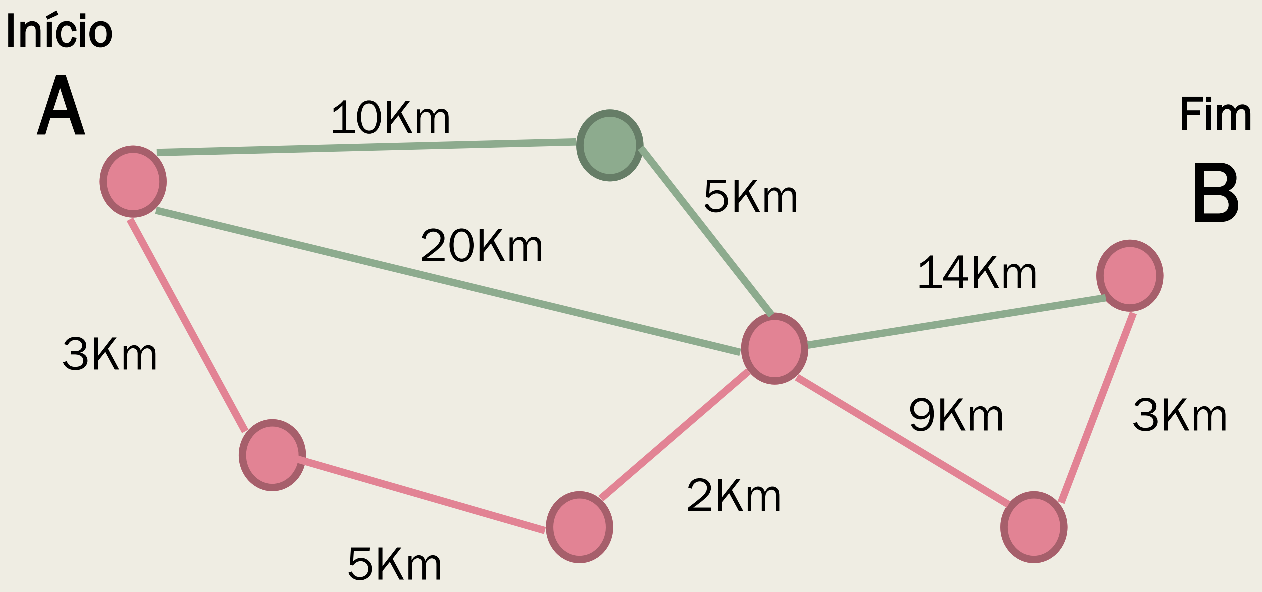
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Custo total e melhor caminho: 3 + 5 + 2 + 9 + 3 = 22Km.

Algoritmo guloso

Vantagens

- Implementação simples;
- Algoritmos de rápida execução;
- Há chance de obter a melhor solução.

Desvantagens

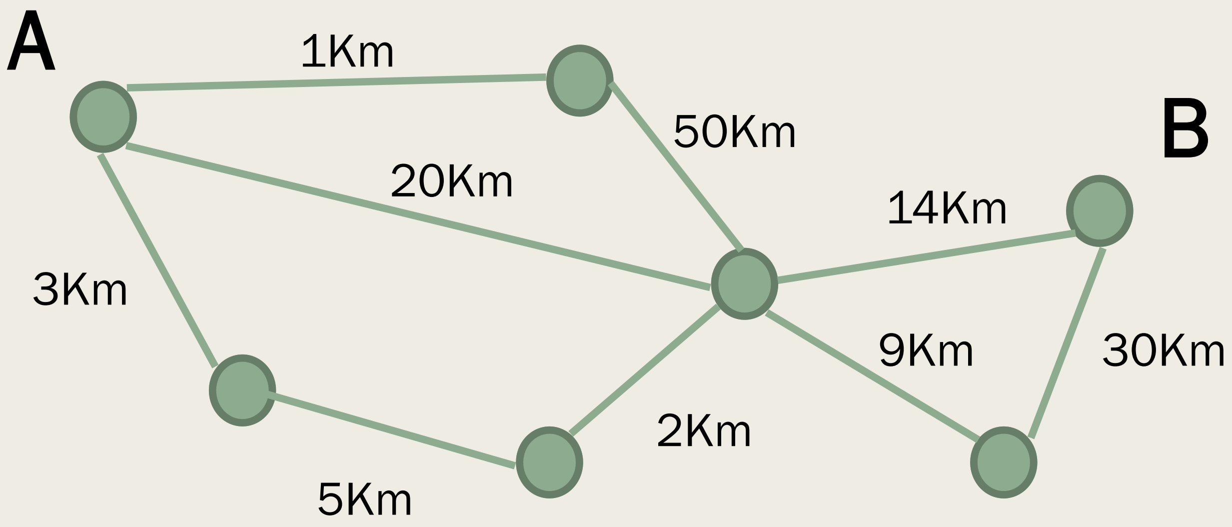
- Por vezes não obtém a melhor solução global;
- Escolhe o melhor caminho localmente;
- Há chance de entrar em loop;
- Pode desenvolver um caminho infinito.

Problema da viagem

- Foi possível conseguir o melhor caminho na situação em que se encontrava o mapa (grafo);
- Porém, como se comportaria o algoritmo caso os caminhos internos fossem maiores, uma vez que ele não prevê esses casos? E se fosse seguido caminho que são inalcançáveis?

Problema da viagem: Desvantagem 1

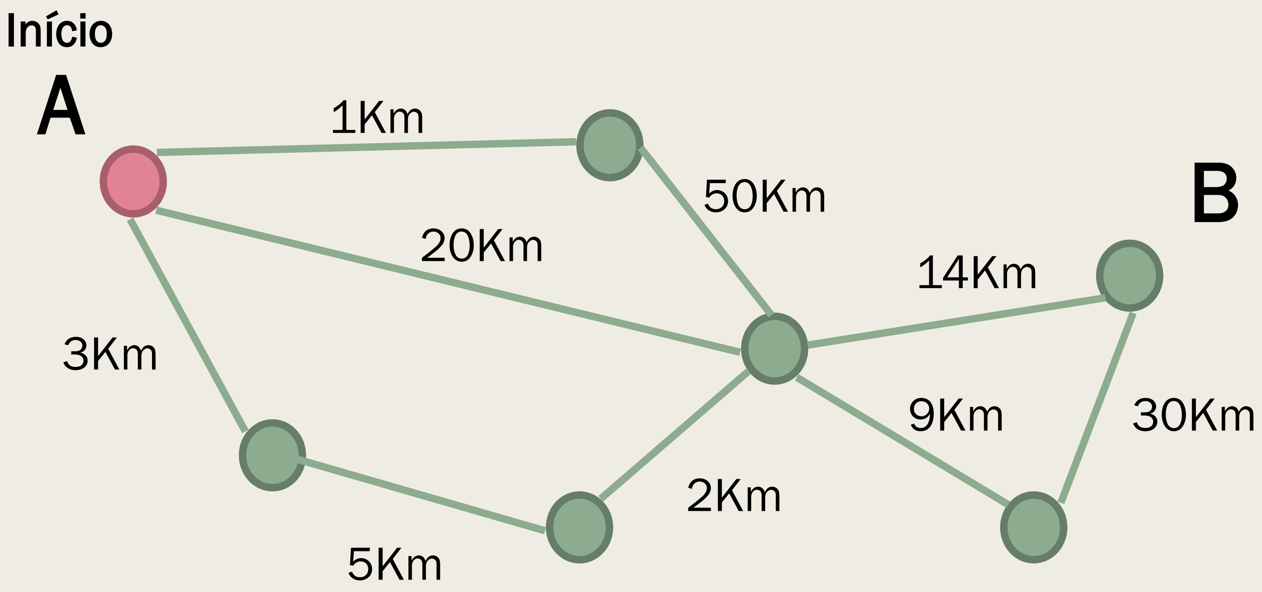
- Como percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Segundo o algoritmo guloso, seria escolhido o menor caminho localmente.

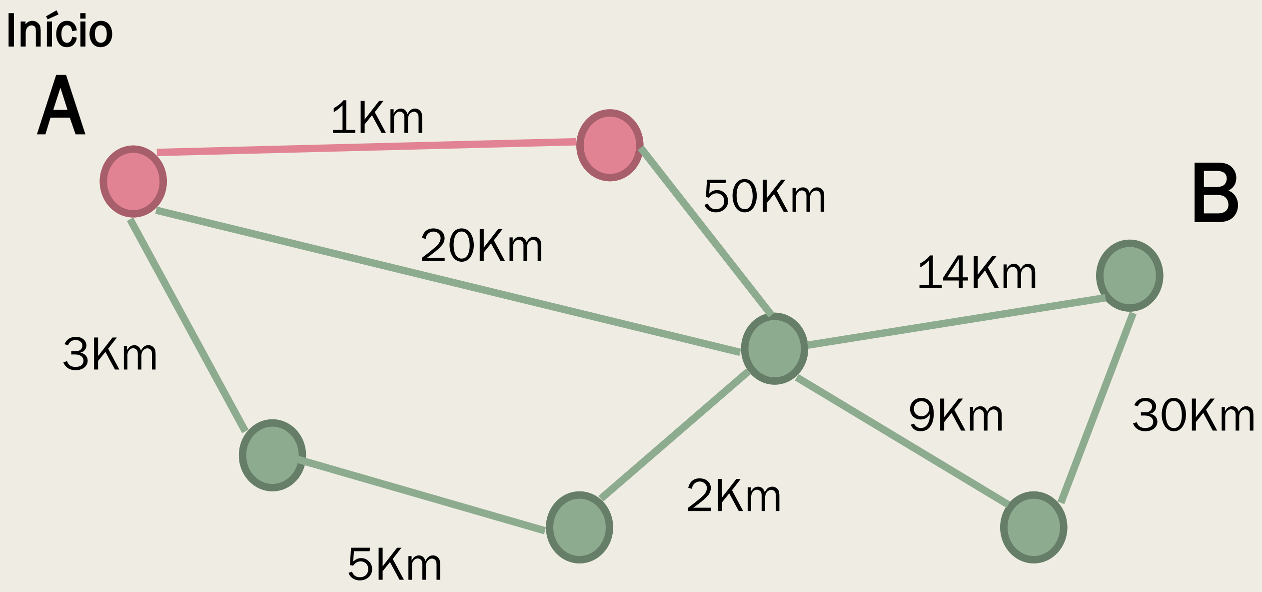
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



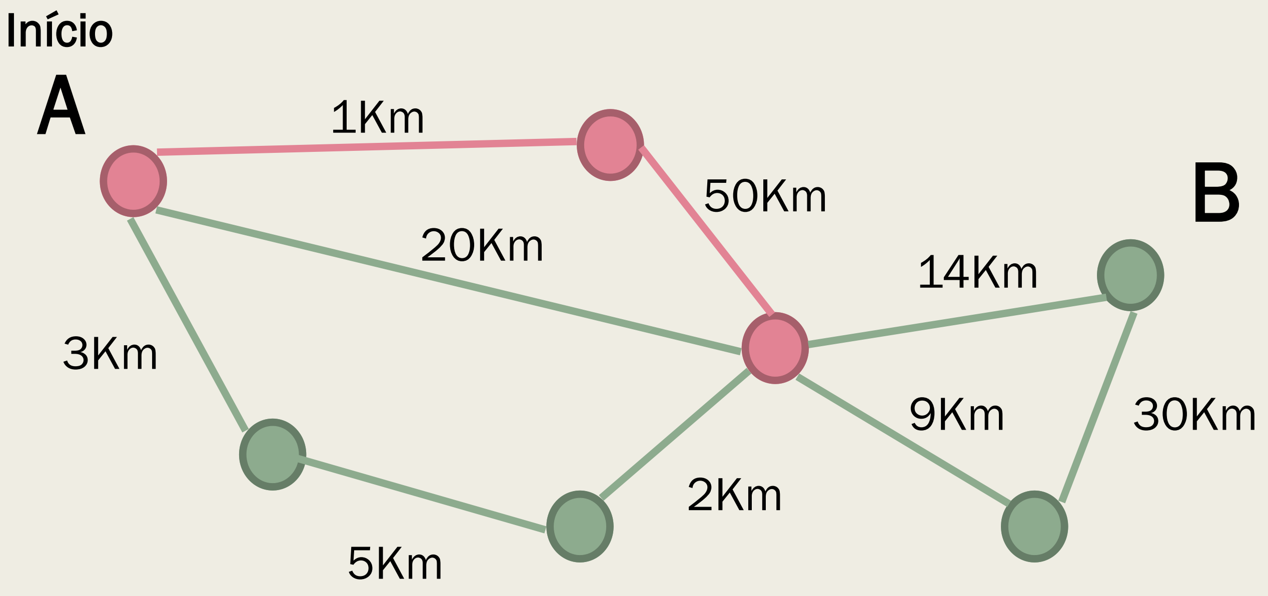
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



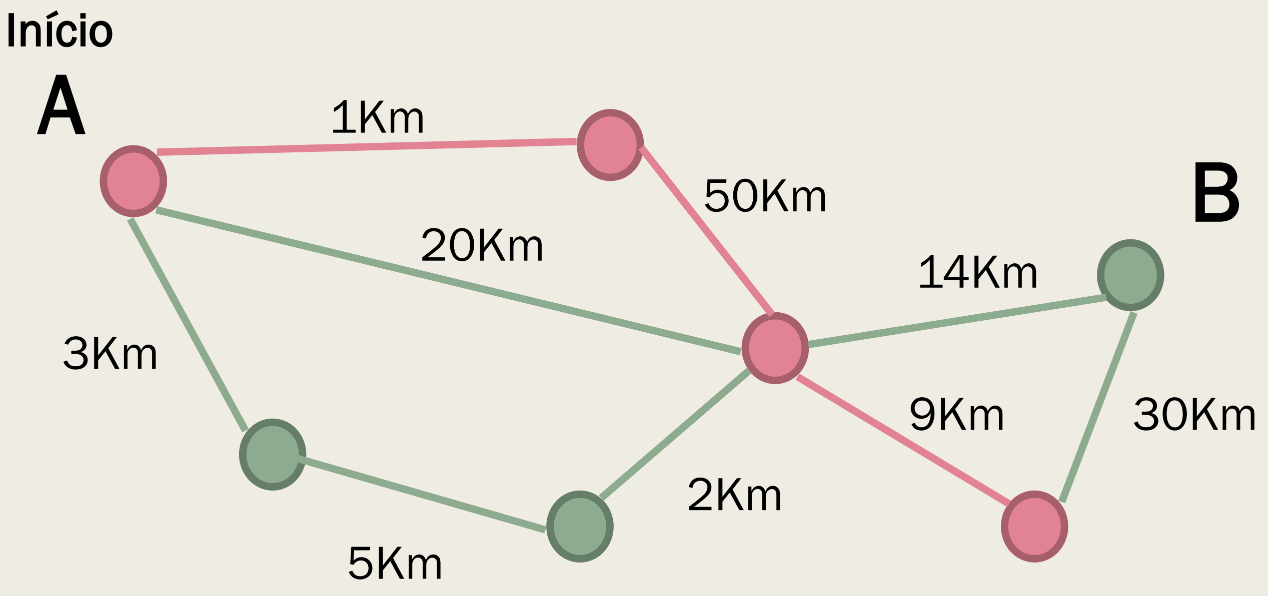
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



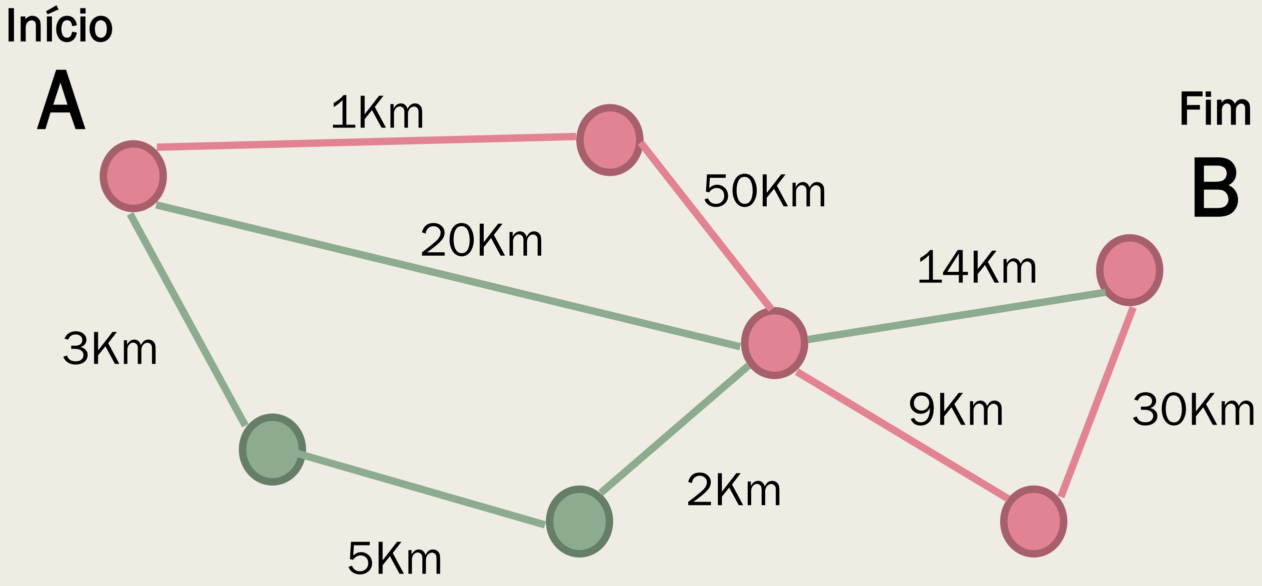
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



Problema da viagem

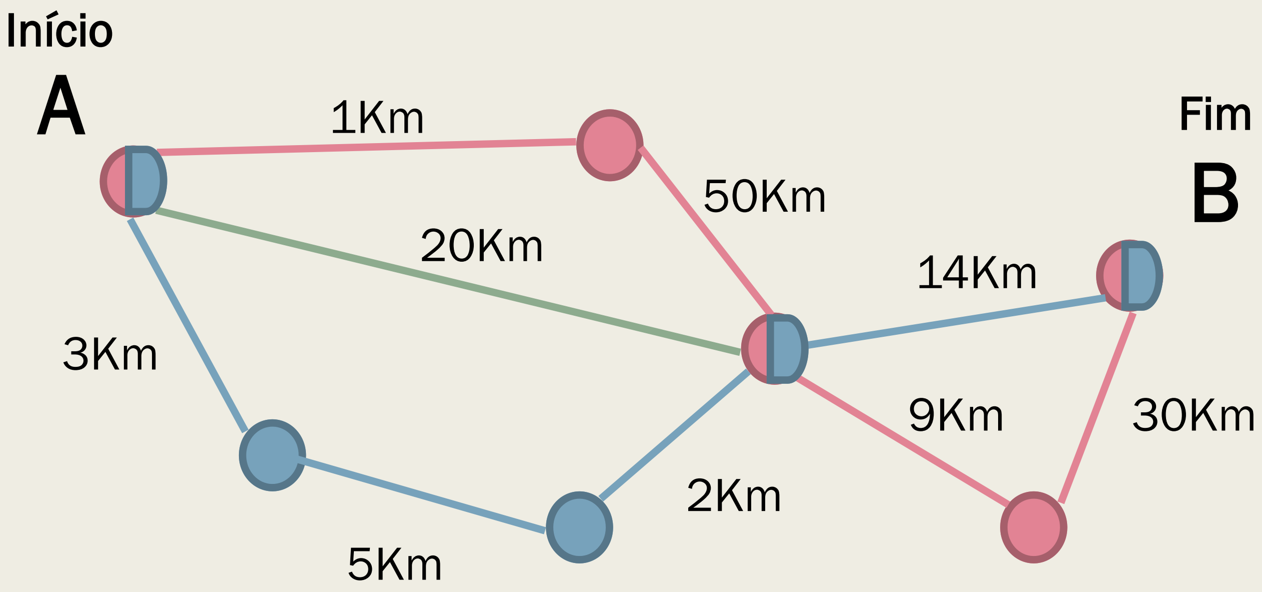
- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Custo total: 1 + 50 + 9 + 30 = 80Km.

Problema da viagem

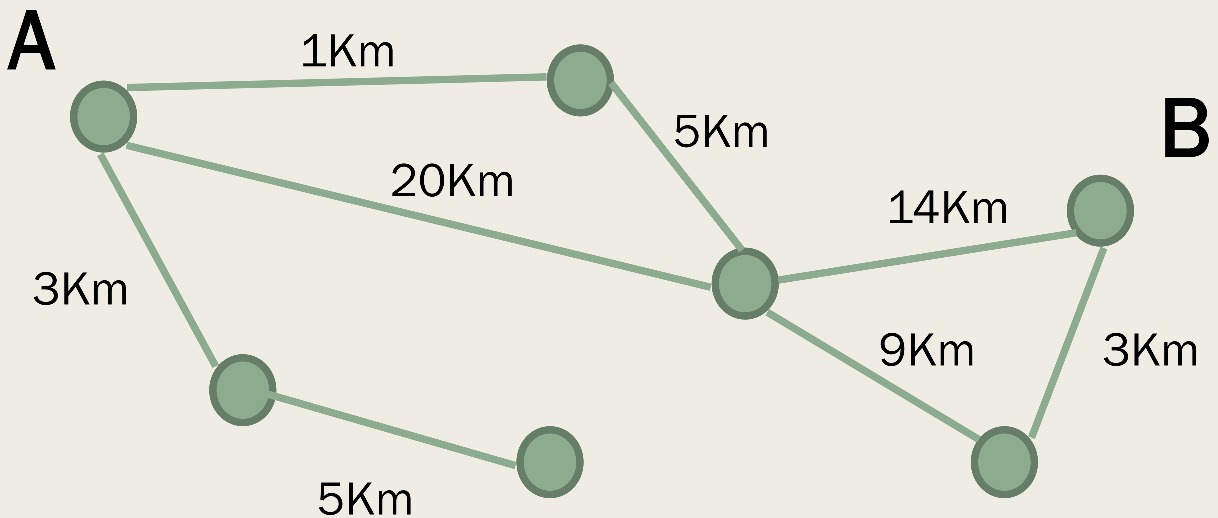
- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Melhor caminho: 3 + 5 + 2 + 14 = 24Km.

Problema da viagem: Desvantagem 2

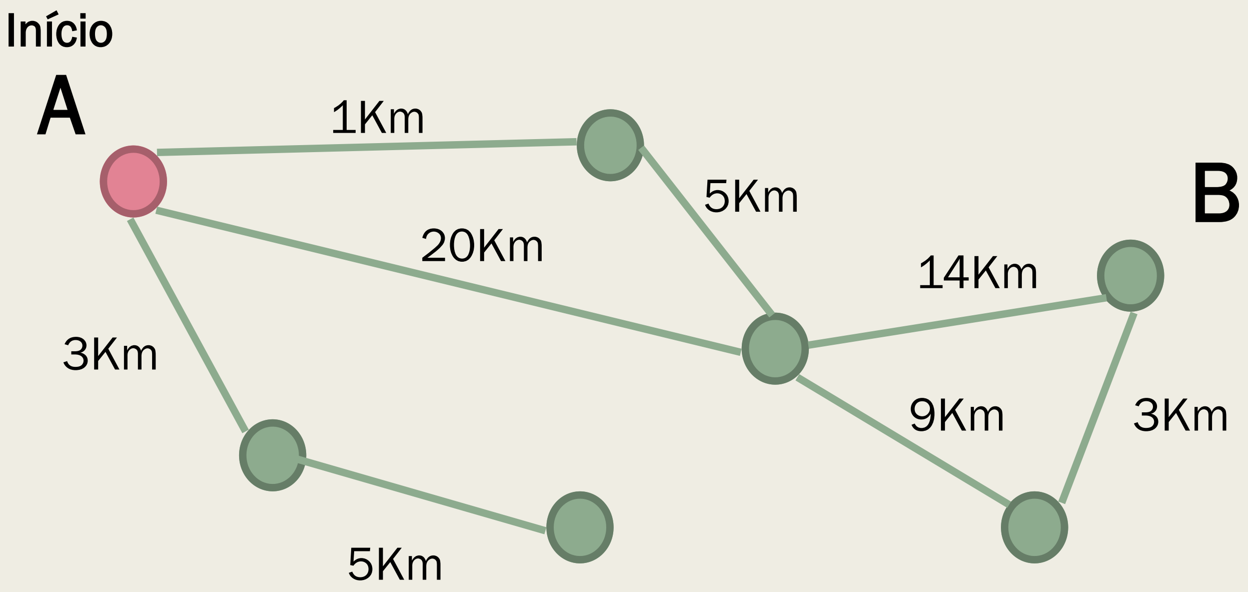
- Como percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Segundo o algoritmo guloso, seria escolhido o menor caminho localmente.

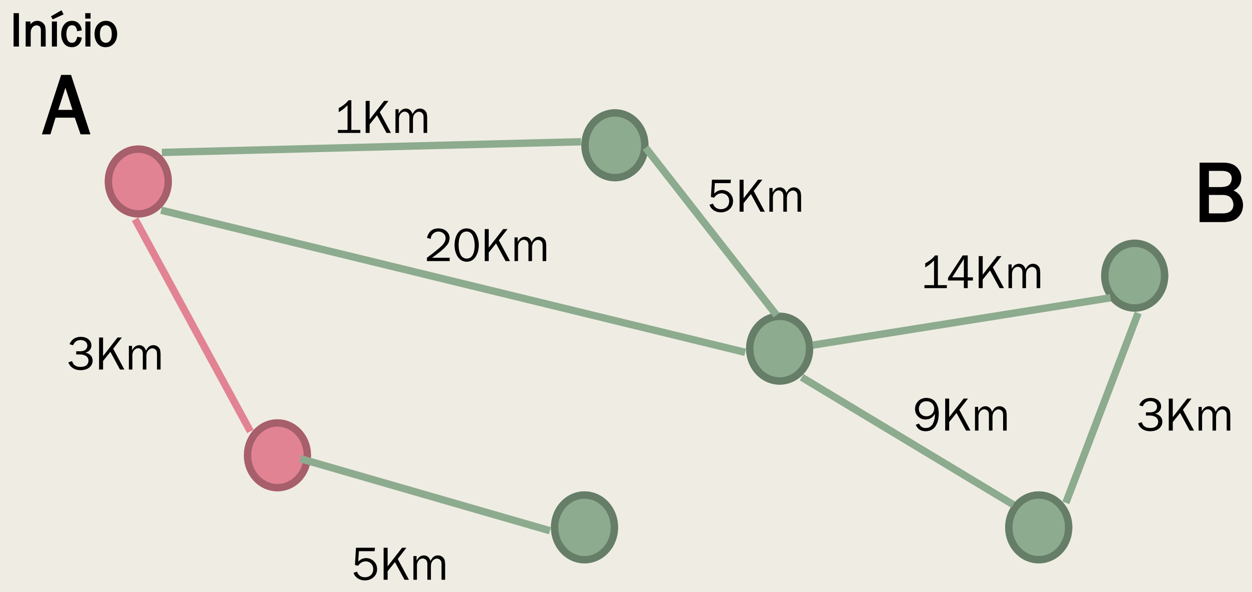
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



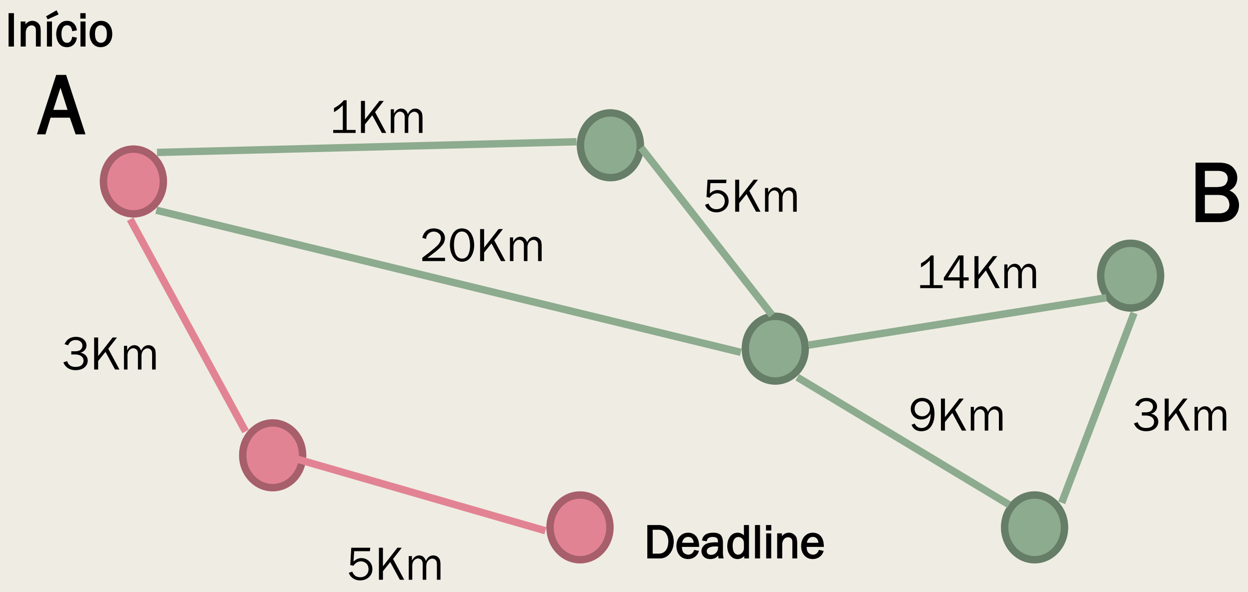
Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



Problema da viagem

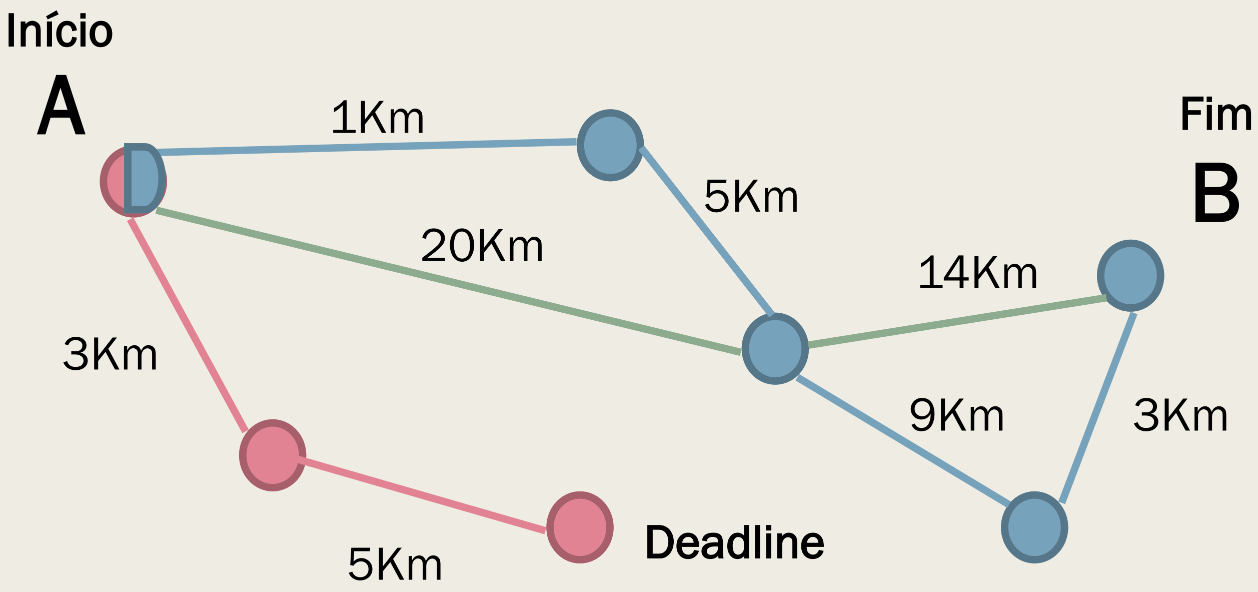
- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Melhor caminho: Objetivo não alcançado!

Problema da viagem

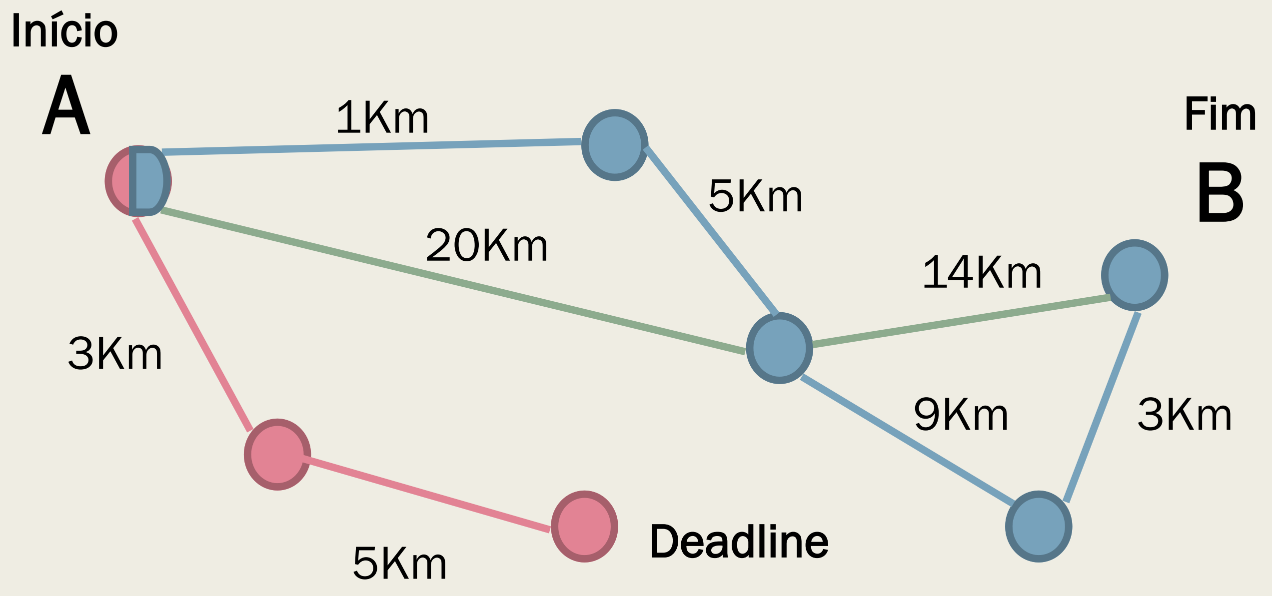
- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Melhor caminho: 1 + 5 + 9 + 3 = 18Km.

Problema da viagem

- Percorrer de uma cidade A até uma cidade B com o menor custo possível?



- Melhor caminho: 1 + 5 + 9 + 3 = 18Km.

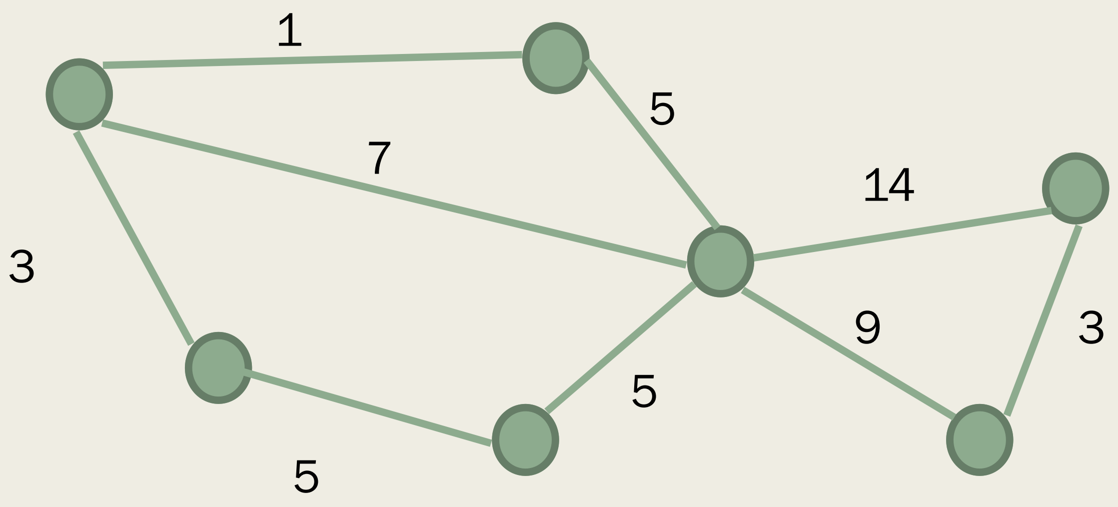
Guloso vs Programação dinâmica

- Quando aplicar um algoritmo guloso?
  - ***Propriedade de escolha gulosa:*** *uma solução ótima global pode ser obtida a partir de uma escolha ótima (gulosa) local;*
  - *A programação dinâmica necessita verificar as soluções dos subproblemas.*
  - ***Subestrutura ótima:*** *uma solução ótima do problema contém uma solução ótima para os subproblemas.*
- Algoritmos gulosos nem sempre produzem uma solução ótima.
- Algoritmos gulosos x programação dinâmica
  - *Em comum: subestrutura ótima*
  - *Diferença: propriedade da escolha gulosa*
  - *Programação dinâmica pode ser utilizada se o algoritmo guloso não der a solução ótima.*

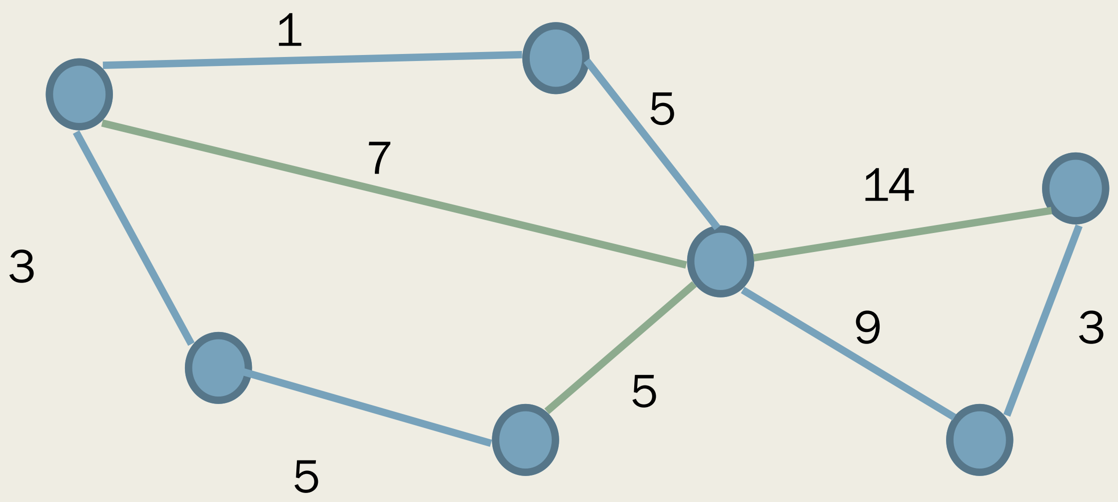
Árvore de peso mínimo

- Seja o grafo G: Uma árvore de peso mínimo desse grafo é um subgrafo sem ciclos que conecta todos os vértices.
- Um grafo pode ter diversas árvores de peso, porém a árvore de peso mínimo escolhe as arestas de menor peso;

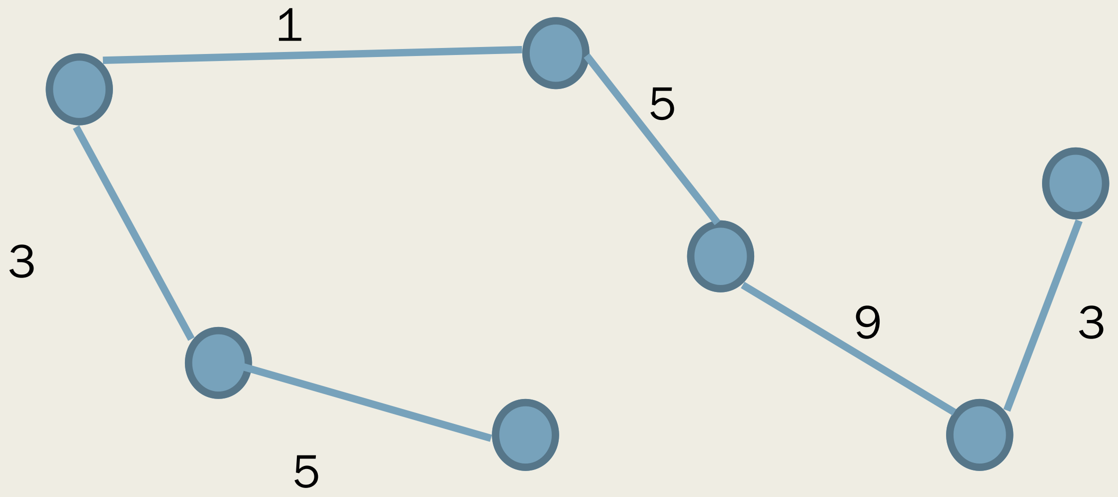
Árvore de peso mínimo: Exemplo



Árvore de peso mínimo: Exemplo



Árvore de peso mínimo: Exemplo

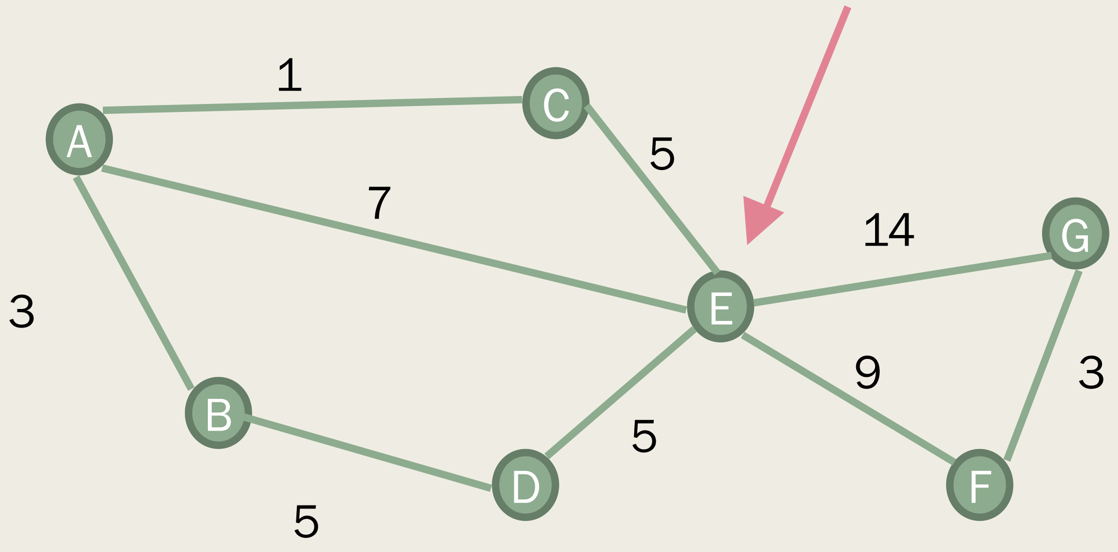


Algoritmo de Prim

- Seja o grafo conectado G: O algoritmo de Prim visa buscar uma árvore de peso mínimo.

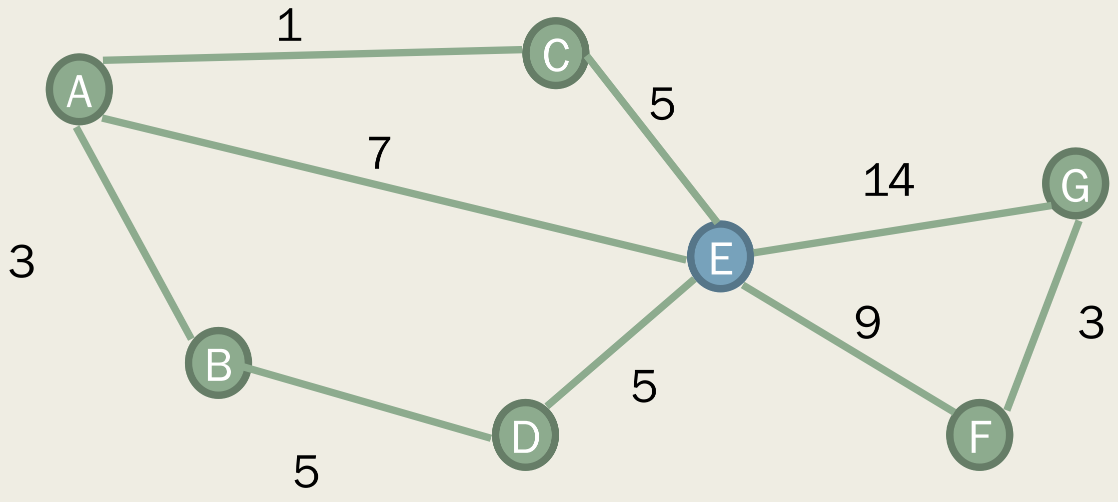
- É escolhido um vértice pertencente a G, e colocado em uma fila todos os outros vértices que fazem ligação com ele em uma fila;
- Escolhe-se o vértice de menor peso dentre os vizinhos da fila;
- Ao escolher o menor, repete o primeiro passo para ele, e assim segue recursivamente.

Algoritmo de Prim: Exemplo



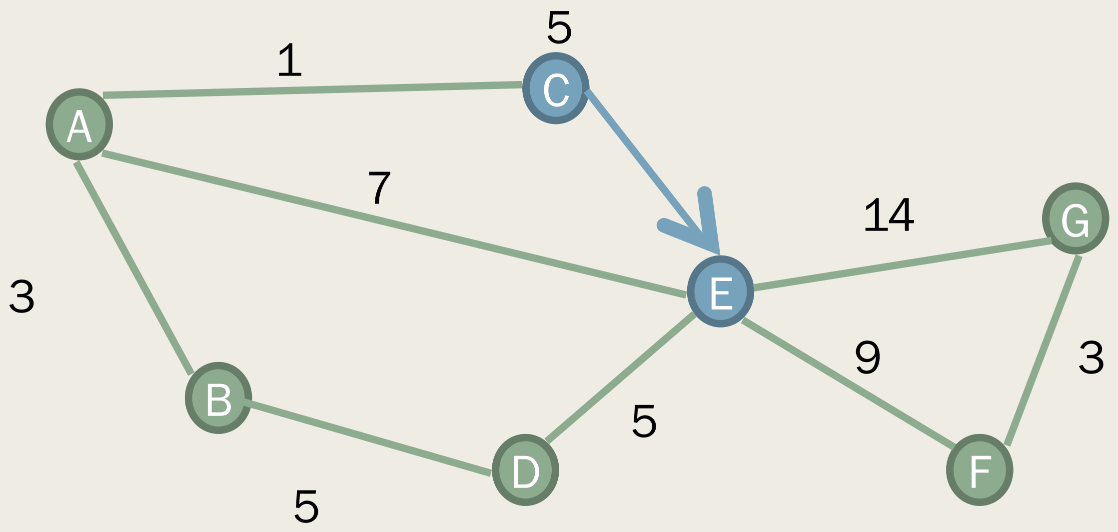
- Fila = {}.

Algoritmo de Prim: Exemplo



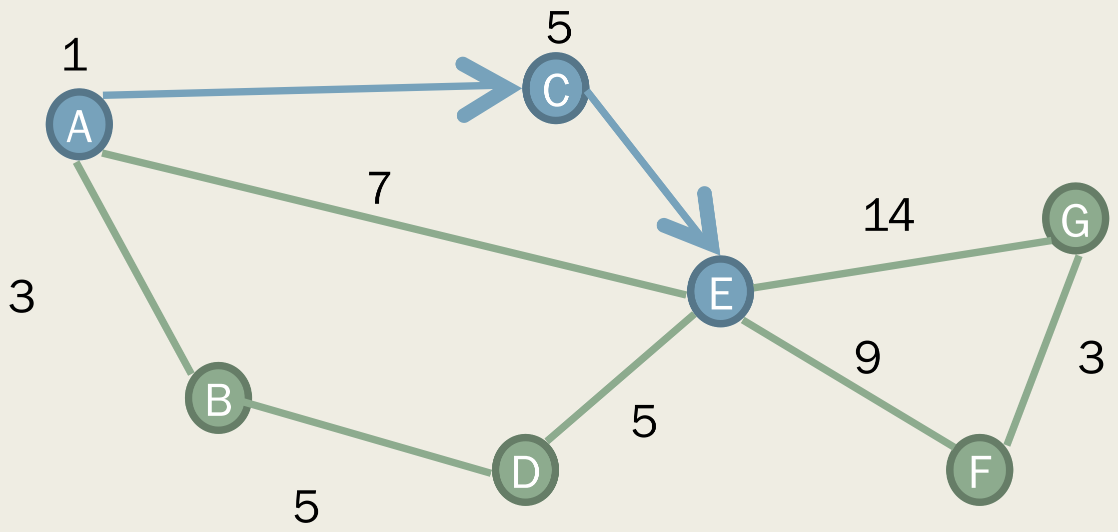
- Fila = {C, D, F, G}.

Algoritmo de Prim: Exemplo



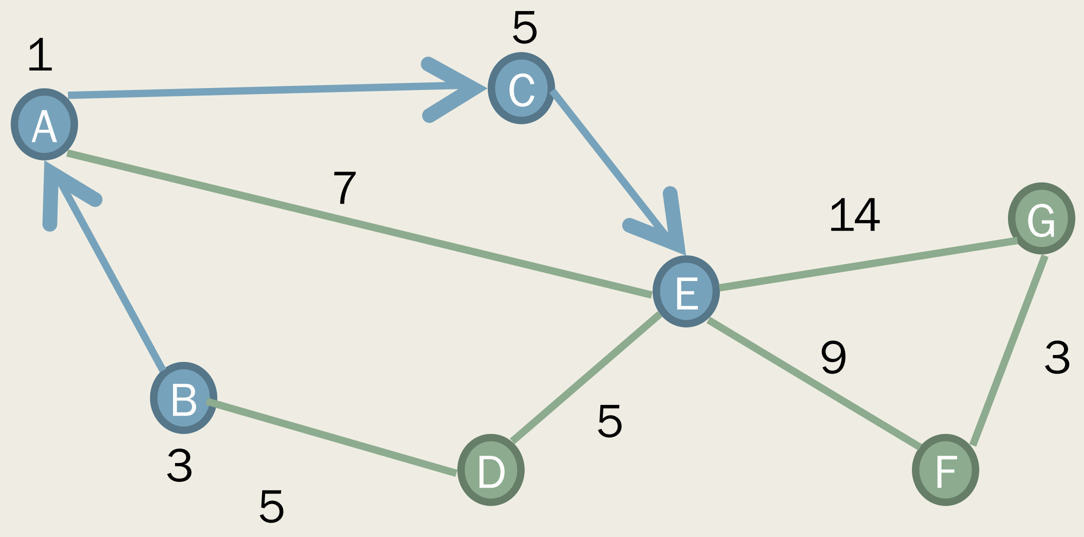
- Fila = {A, D, F, G}.

Algoritmo de Prim: Exemplo



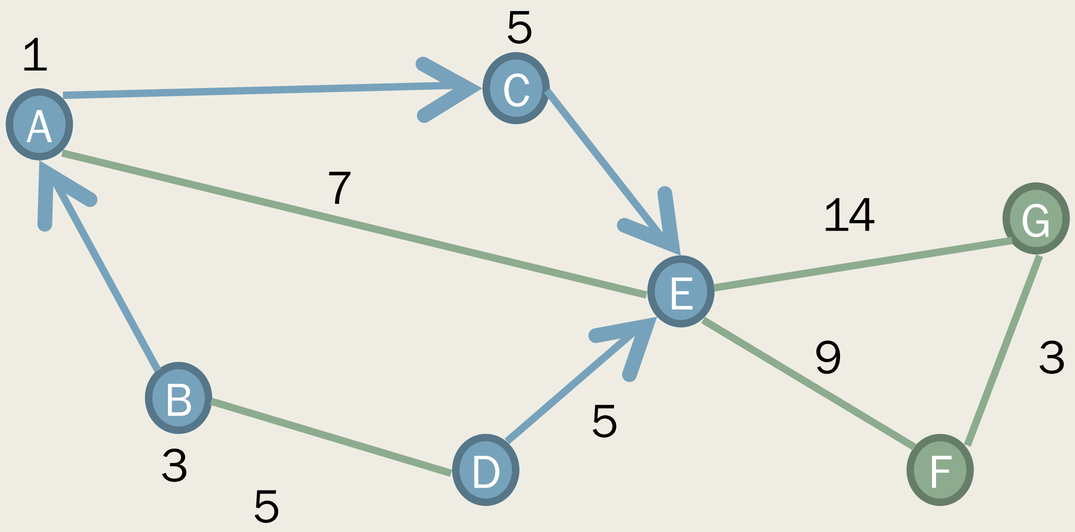
- Fila = {B, D, F, G}.

Algoritmo de Prim: Exemplo



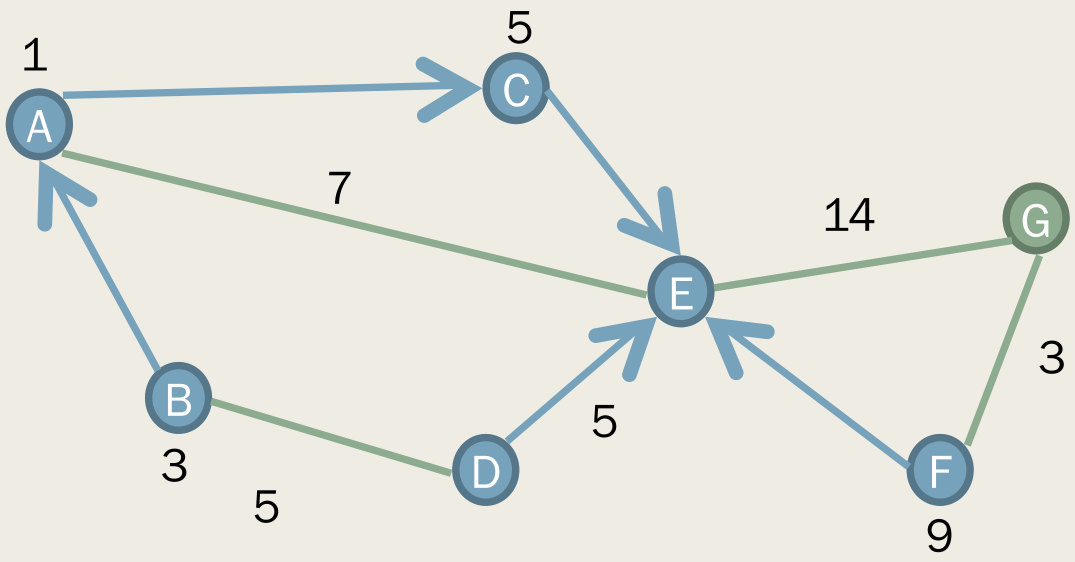
- Fila = {D, F, G}.

Algoritmo de Prim: Exemplo



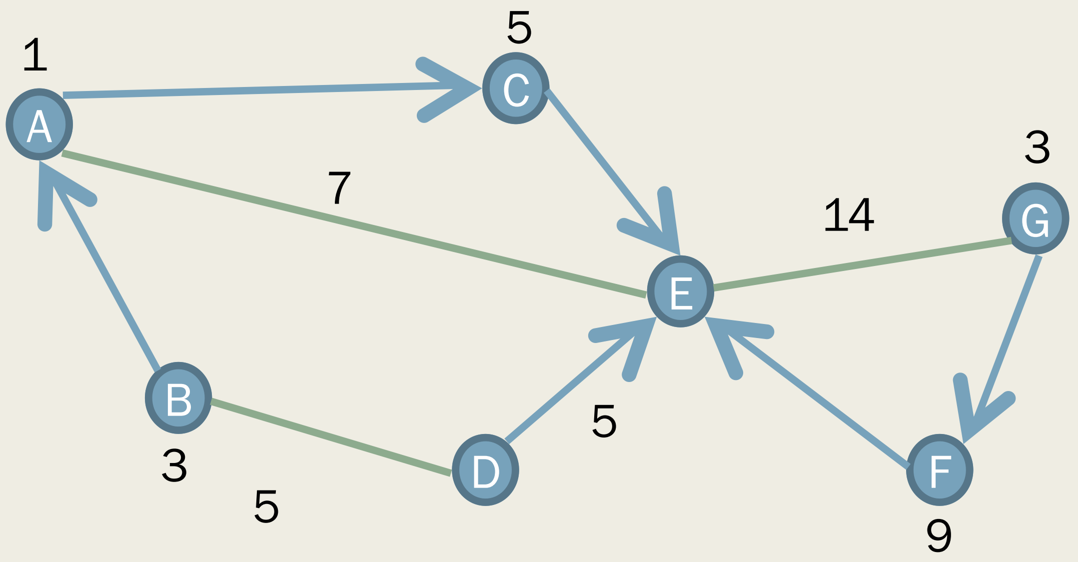
- Fila = {F, G}.

Algoritmo de Prim: Exemplo



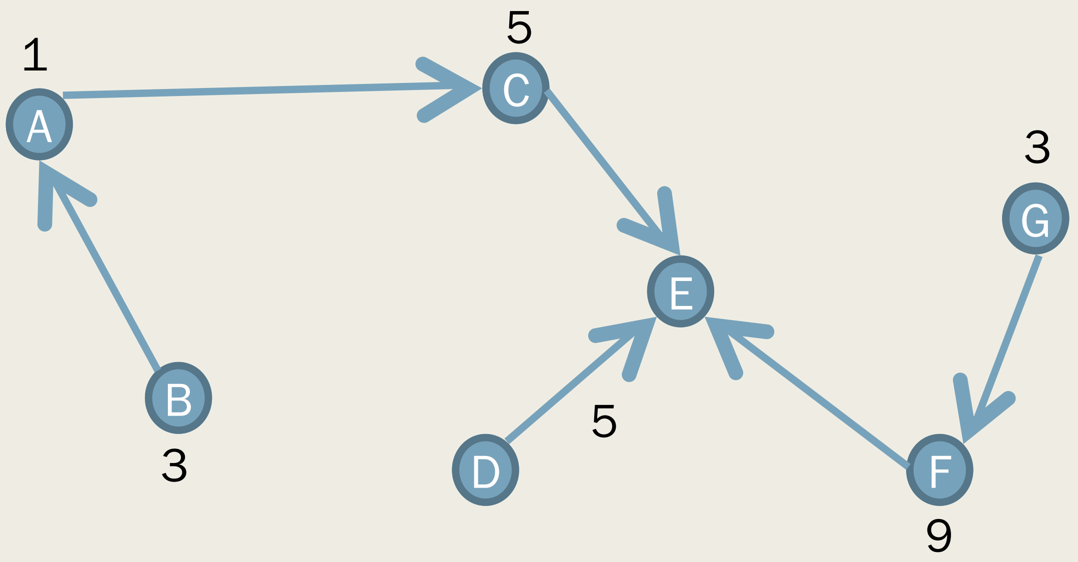
- Fila = {G}.

Algoritmo de Prim: Exemplo



- Fila = {}.

Algoritmo de Prim: Exemplo



Algoritmo de Prim: Análise

- Tempo: $\Theta(\text{Vértices}) * T_{\text{extração mínimo}} + \Theta(\text{Arestas}) * T_{\text{chave decremental}}$

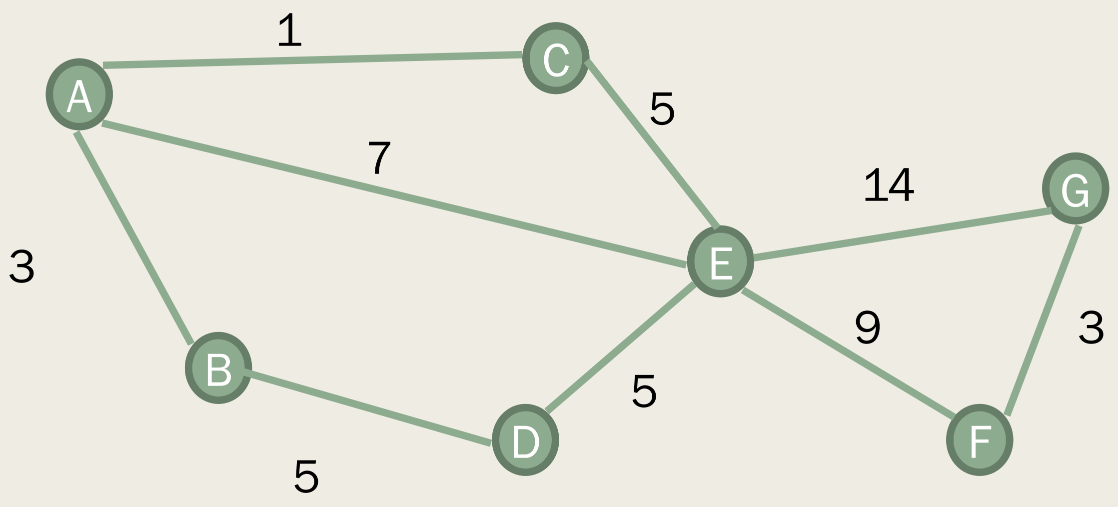
- Utilizando um array como entrada:
  - $T_{\text{extração mínimo}} = O(V)$
  - $T_{\text{chave decremental}} = O(1)$
  - $Tempo = \Theta(V) * O(V) + \Theta(E) * O(1) = O(V^2)$

Algoritmo de Kruskal

- Seja o grafo conectado G: O algoritmo de Kruskal visa buscar uma árvore de peso mínimo.

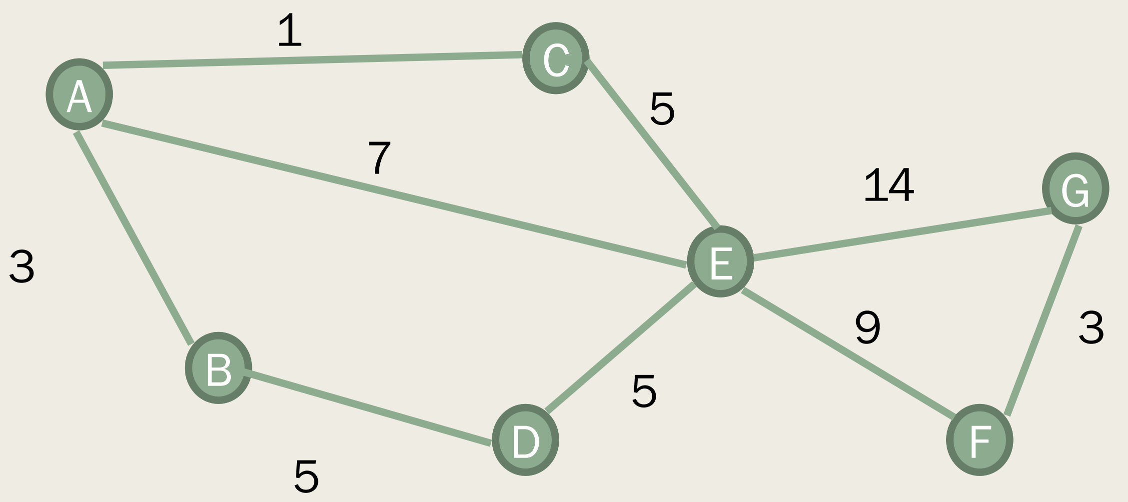
- Todas as arestas pertencentes a G entram em uma lista;
- Essa lista é ordenada em ordem decrescente;
- Pega a menor aresta disponível:
  - *Se formar ciclo, descarta;*
  - *Caso contrário, inclui.*
- Repete o passo anterior até que a lista esteja vazia.

Algoritmo de Kruskal



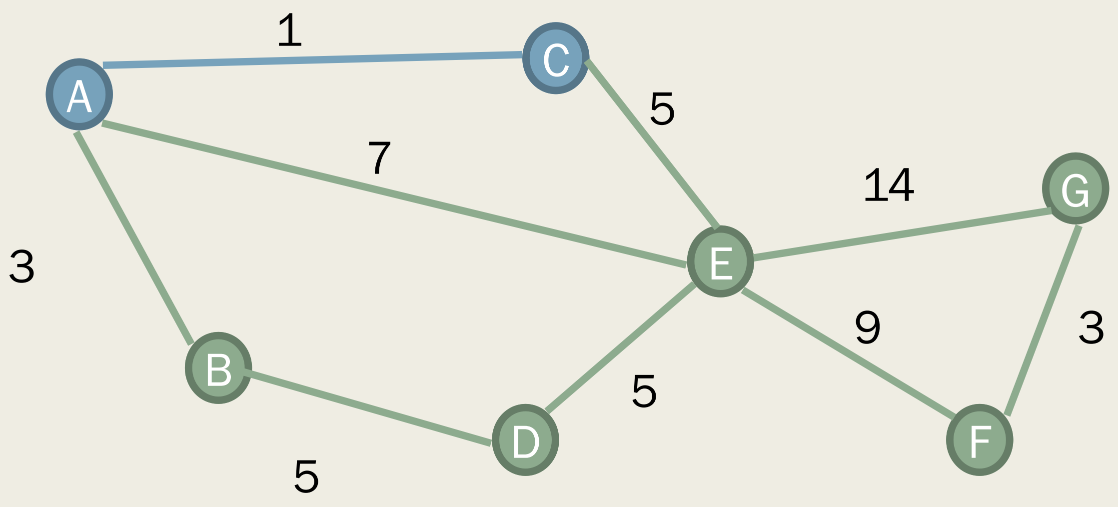
- Fila = { (A,B) , (A,C) , (A,E) , (B,D) , (C,E) , (D,E) , (E,F) , (E,G) , (F,G) }.

Algoritmo de Kruskal



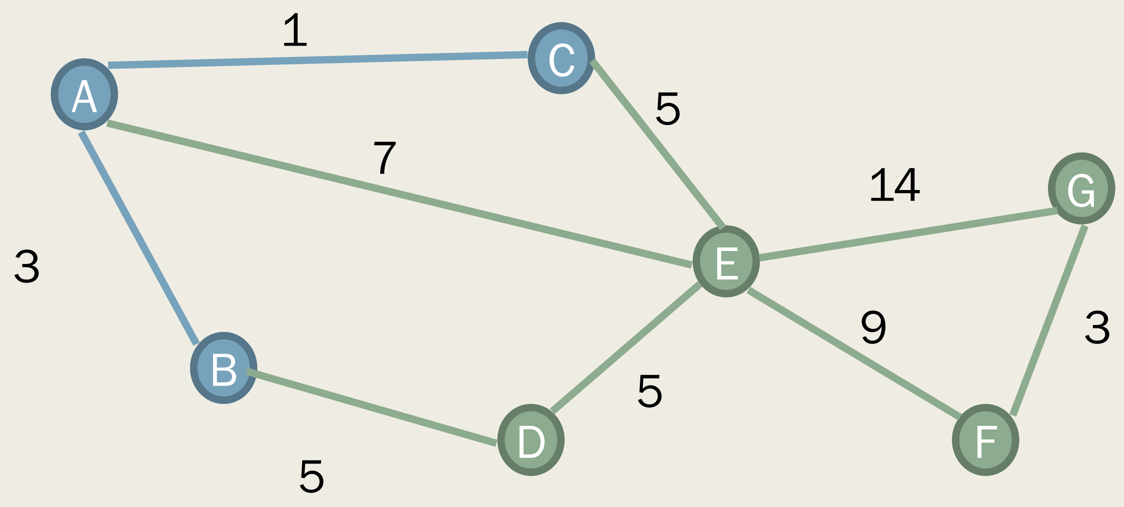
- Fila = { (A,C) , (A,B) , (F,G) , (B,D) , (C,E) , (D,E) , (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



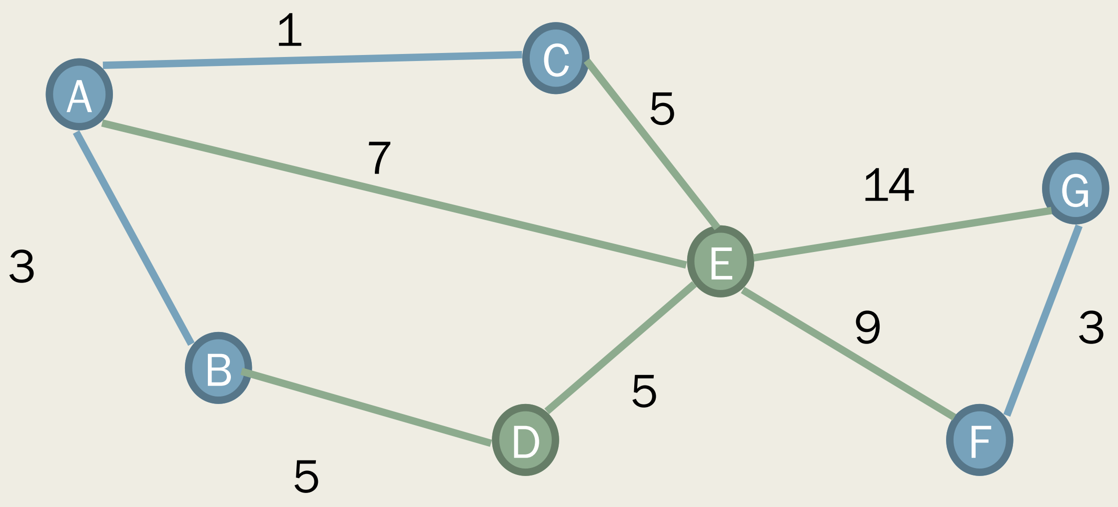
- Fila = { (A,B) , (F,G) , (B,D) , (C,E) , (D,E) , (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



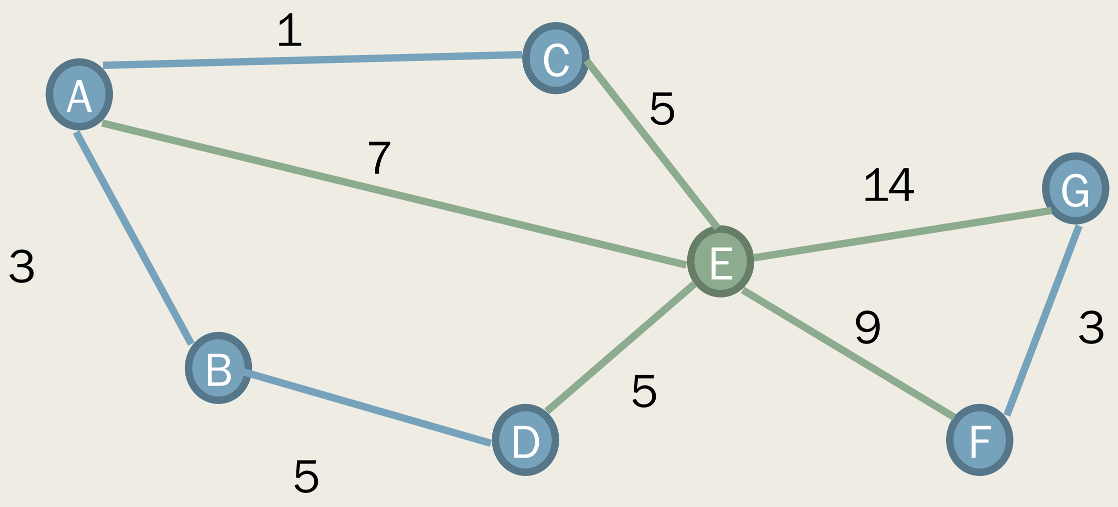
- Fila = { (F,G) , (B,D) , (C,E) , (D,E) , (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



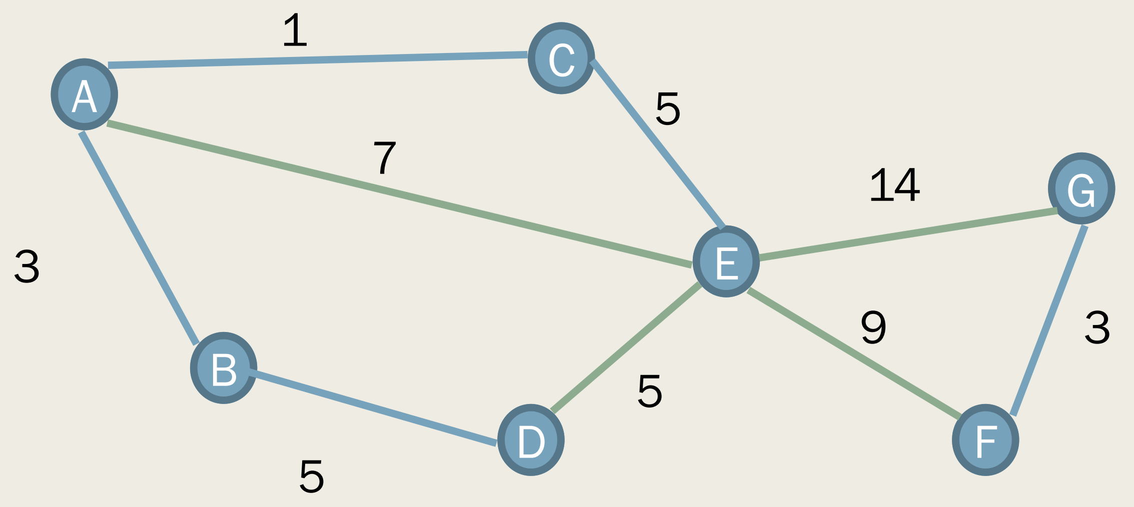
- Fila = { (B,D) , (C,E) , (D,E) , (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



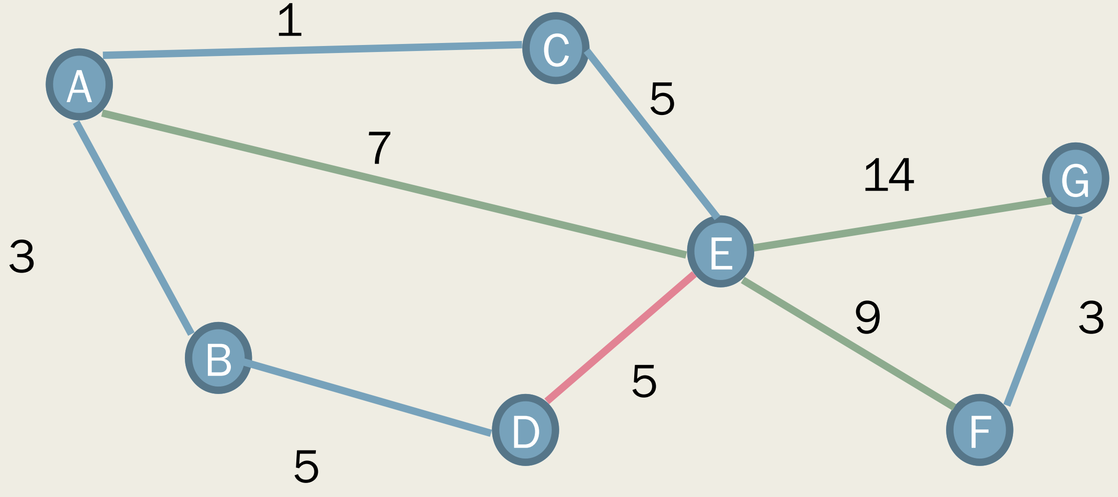
- Fila = { (C,E) , (D,E) , (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



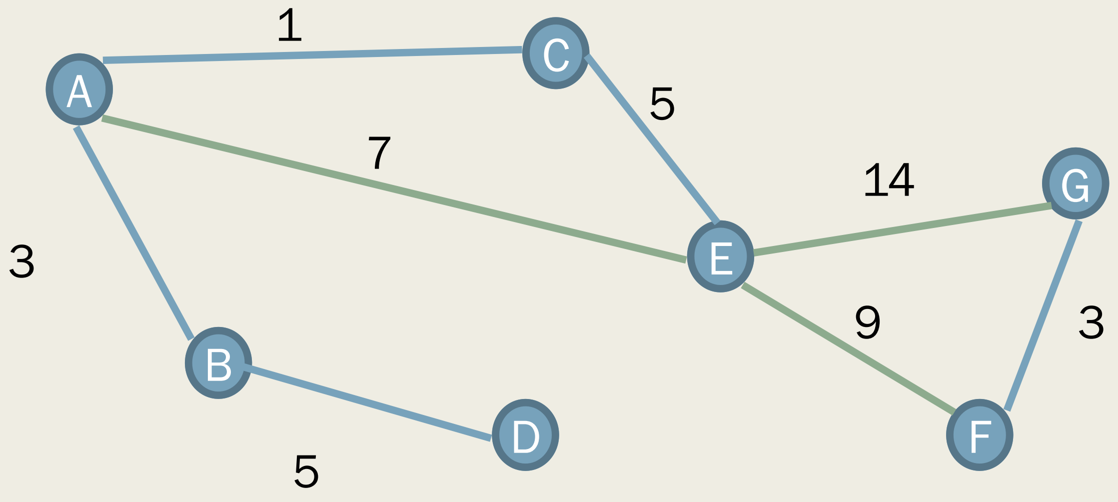
- Fila = { (D,E) , (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



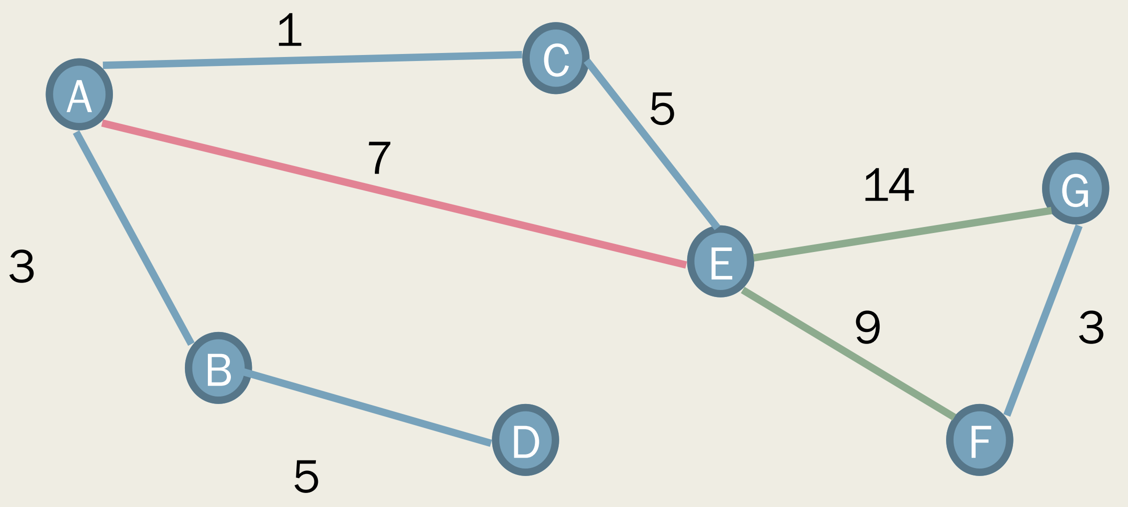
- Fila = { (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



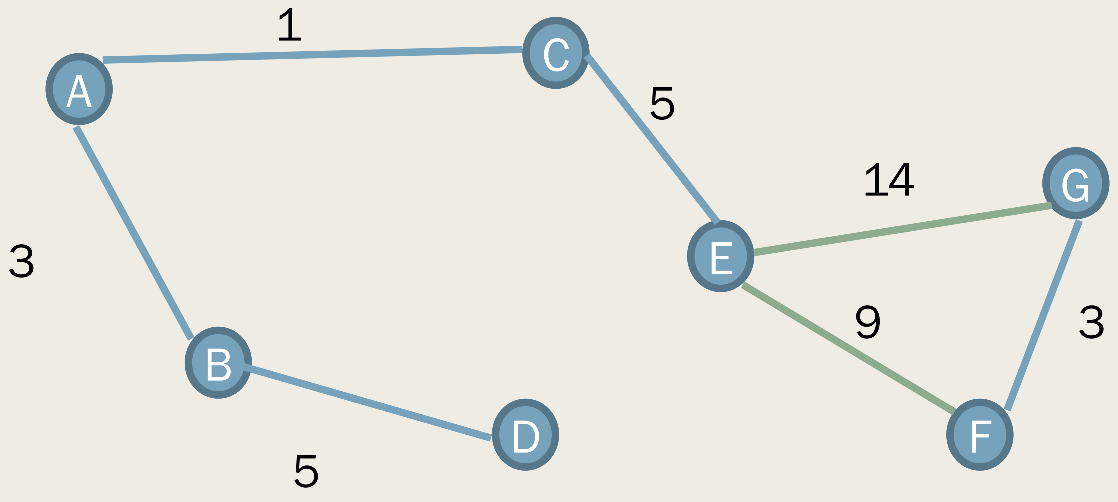
- Fila = { (A,E) , (E,F) , (E,G) }.

Algoritmo de Kruskal



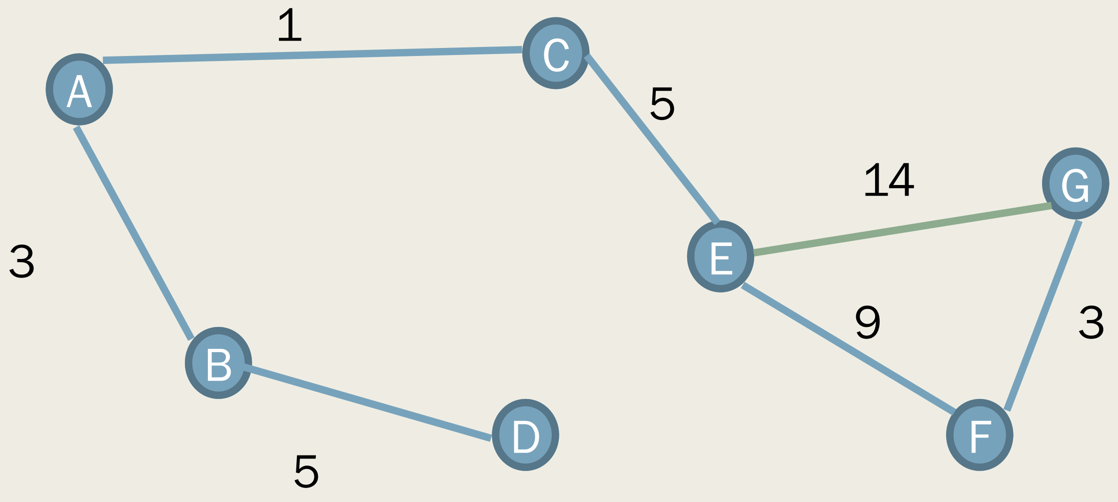
- Fila = { (E,F) , (E,G) }.

Algoritmo de Kruskal



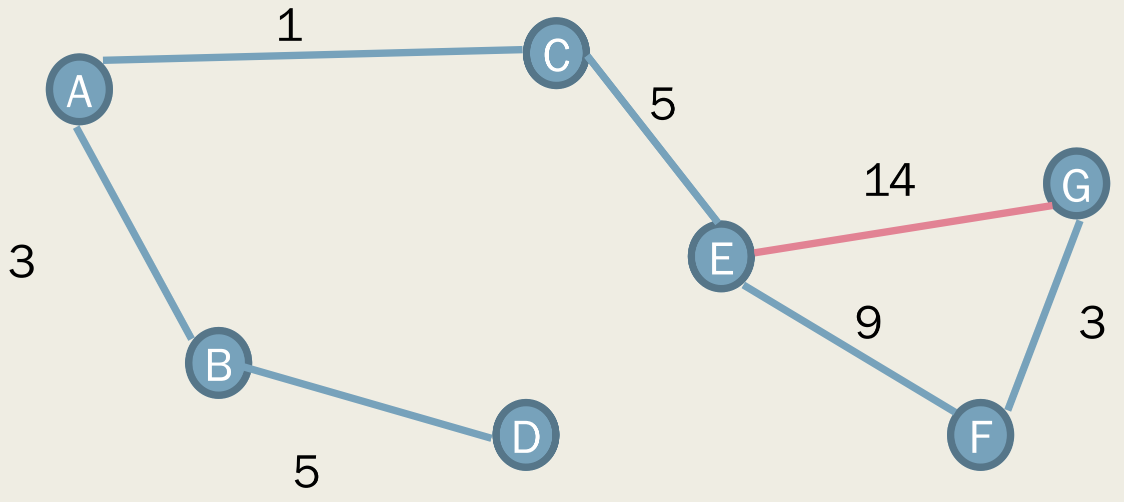
- Fila = { (E,F) , (E,G) }.

Algoritmo de Kruskal



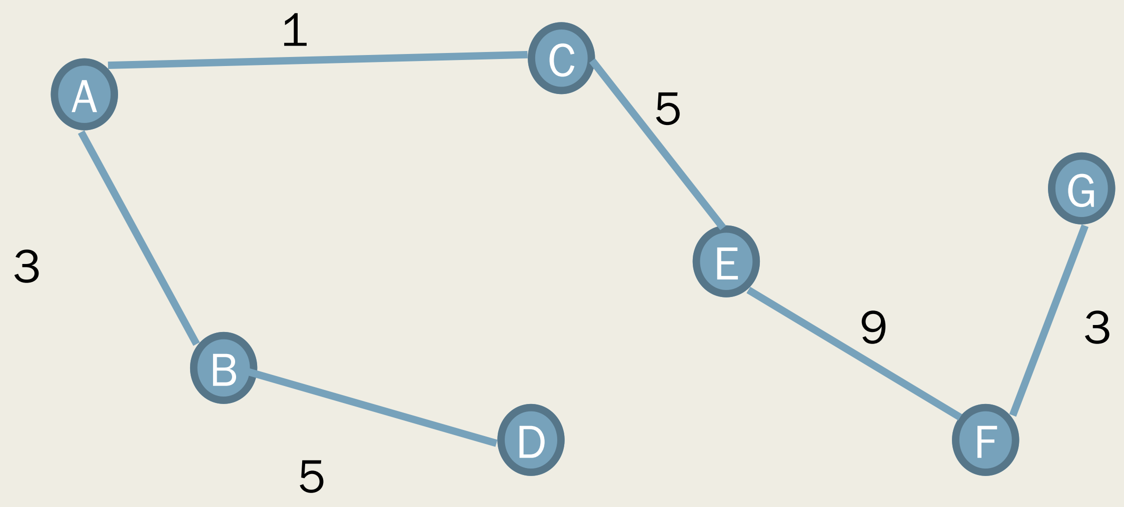
- Fila = { (E,G) }.

Algoritmo de Kruskal



- Fila = { }.

Algoritmo de Kruskal



- Fila = { }.

Algoritmo de Kruskal: Análise

- Tempo: O(Arestas lg Vértices) ou O(Arestas log Arestas), pois <= $|\text{Vértices}^2|$