TEORIA DE GRAFOS E COMPUTABILIDADE

ÁRVORES GERADORAS MÍNIMAS ALGORITMOS DE PRIM E KRUSKAL

Prof. Alexei Machado

PUC MINAS

CIÊNCIA DA COMPUTAÇÃO

Árvores geradoras mínimas

- **Árvore Geradora Mínima** (AGM) é a árvore geradora de menor peso em G

- Problema: Dado um grafo G com pesos associados às arestas, encontrar uma árvore geradora mínima de G

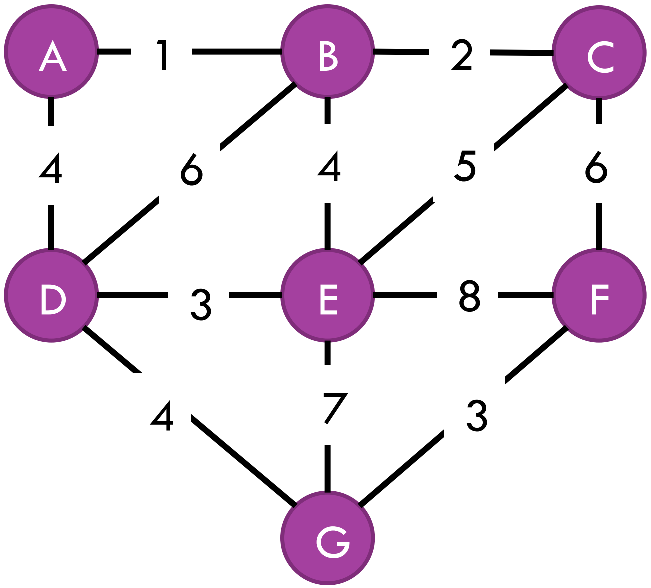
Algoritmo de Prim

- Comece com uma árvore vazia
- A cada passo, adicione um vértice para crescer a árvore. Este vértice deve se conectar à árvore já existente

Algoritmo de Prim

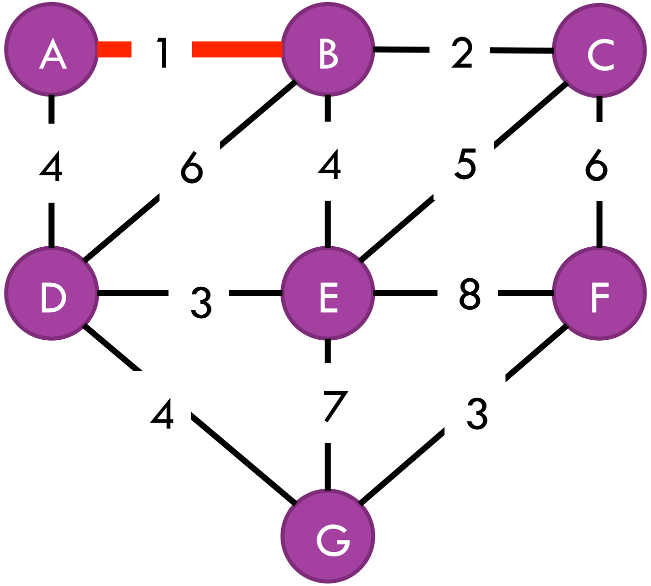
- Iniciar o conjunto T com um vértice v arbitrário
- A cada passo, selecionar a aresta de menor peso que toca T
- Acrescentar a T o vértice ligado por esta aresta
- Continuar até T obter todos os vértices de G

Algoritmo de Prim



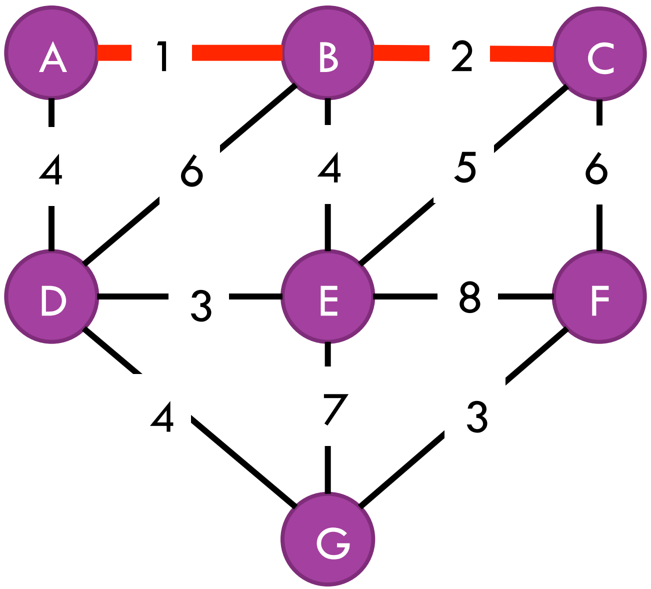
- $T=\{A\}$
- $AG=\varnothing$

Algoritmo de Prim



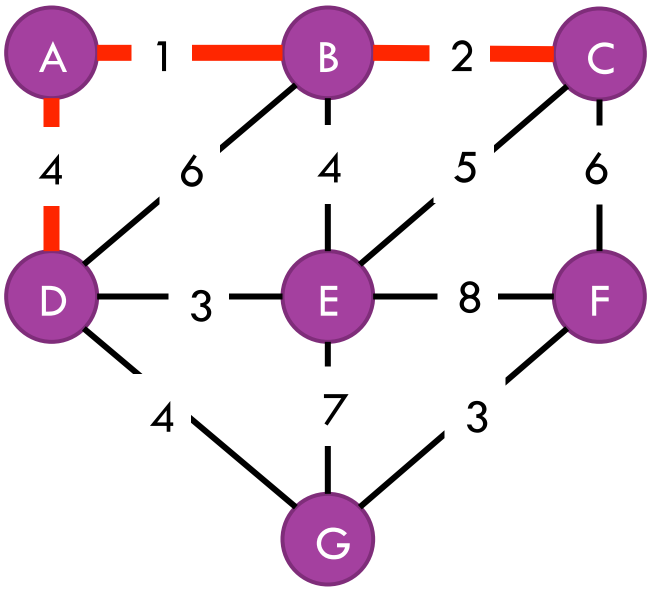
- T={A, B}
- AG={AB}

Algoritmo de Prim



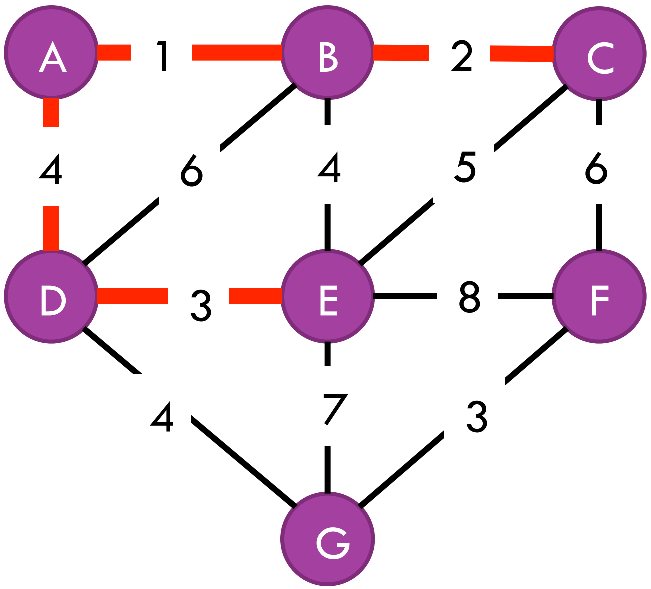
- T={A, B, C}
- AG={AB, BC}

Algoritmo de Prim



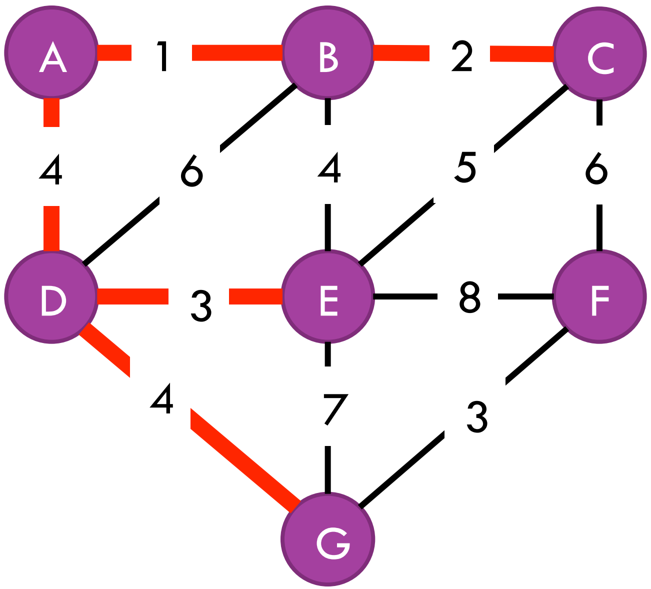
- T={A, B, C, D}
- AG={AB, BC, AD}

Algoritmo de Prim



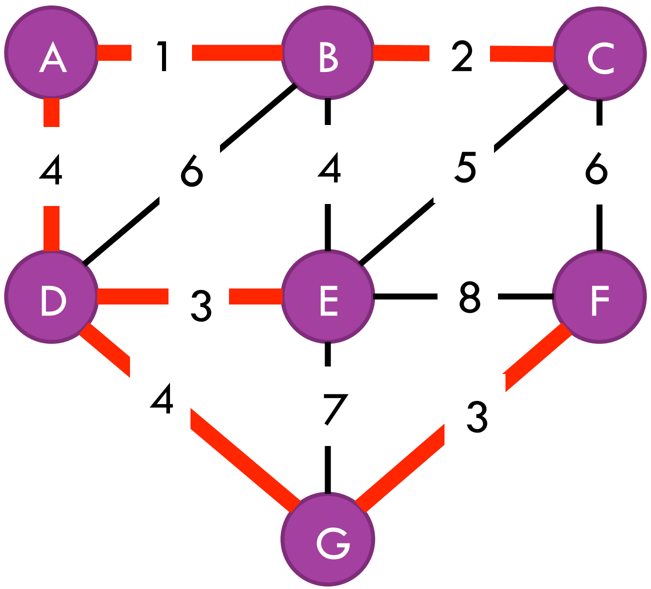
- T={A, B, C, D, E}
- AG={AB, BC, AD, DE}

Algoritmo de Prim



- T={A, B, C, D, E, G}
- AG={AB, BC, AD, DE, DG}

Algoritmo de Prim



- T={A, B, C, D, E, G, F}
- AG={AB, BC, AD, DE, DG, GF}

Algoritmo de Prim - implementação

- A cada passo, como descobrir, eficientemente, a aresta a ser inserida?

Algoritmo de Prim - implementação

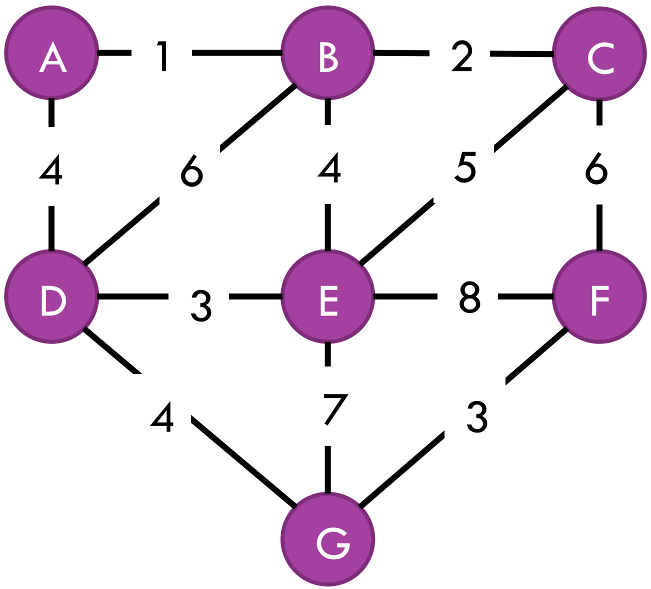
- A cada passo, como descobrir, eficientemente, a aresta a ser inserida?
- Fila de prioridades!

Algoritmo de Prim - implementação

- Inicialização: fila de prioridades recebe todas as arestas do vértice inicial
- A cada passo: retirar a menor aresta da fila de prioridades
  - Se ambos os vértices já estão em T, descarte
  - Senão: 1- insira o vértice não pertencente a T

    2- insira na fila todas as arestas deste vértice

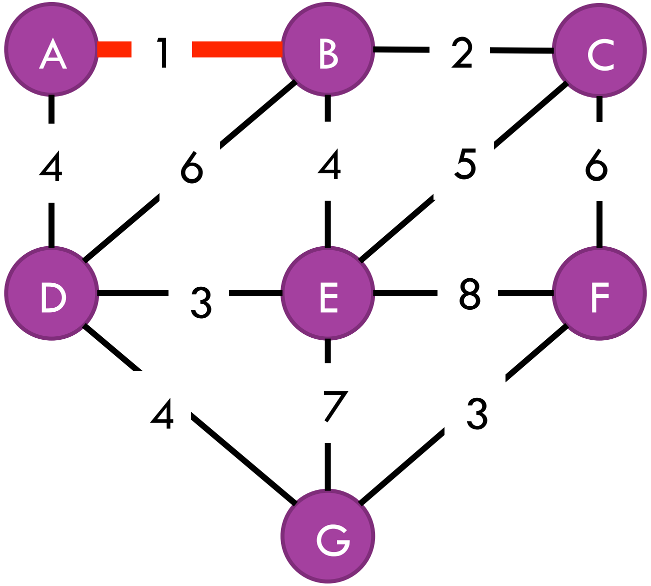
Prim com fila de prioridades



Algoritmo de Kruskal

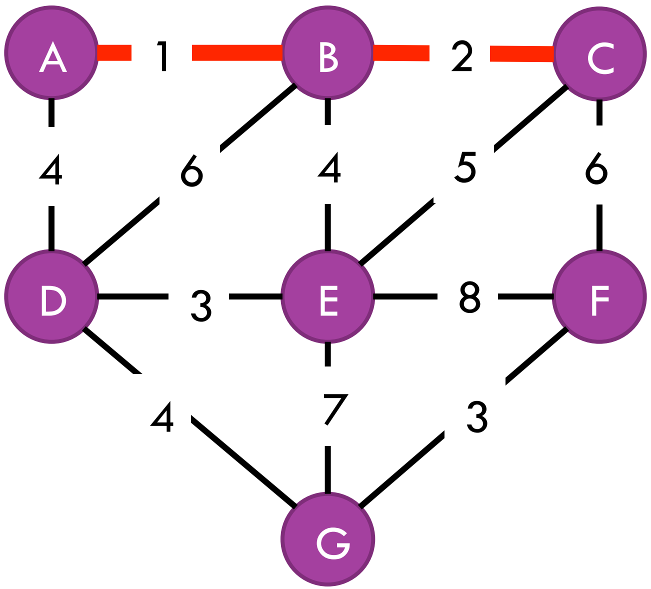
- Inicia com uma AGM vazia
- A cada passo, inclui na AGM a aresta de menor custo que não forme ciclo
- Termina com exatamente n-1 passos (arestas)

Algoritmo de Kruskal



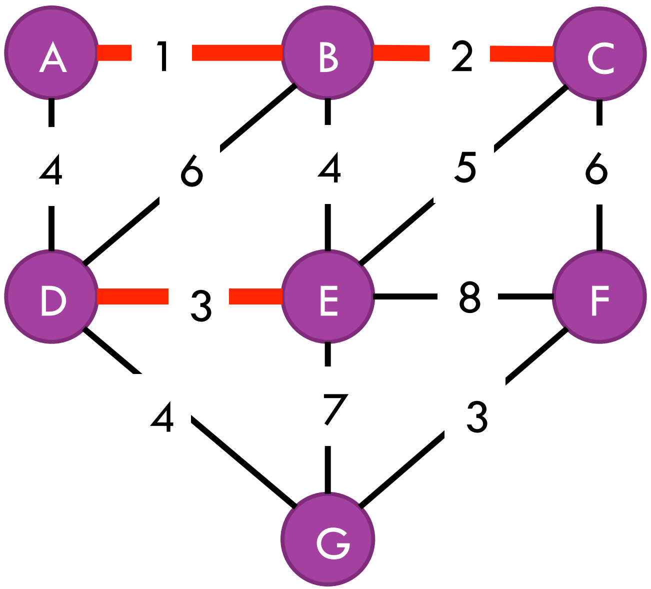
- AG={AB}

Algoritmo de Kruskal



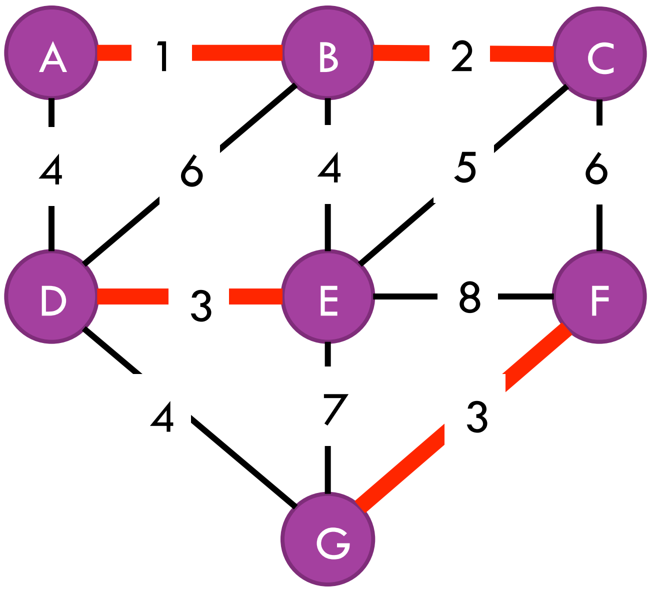
- AG={AB, BC}

Algoritmo de Kruskal



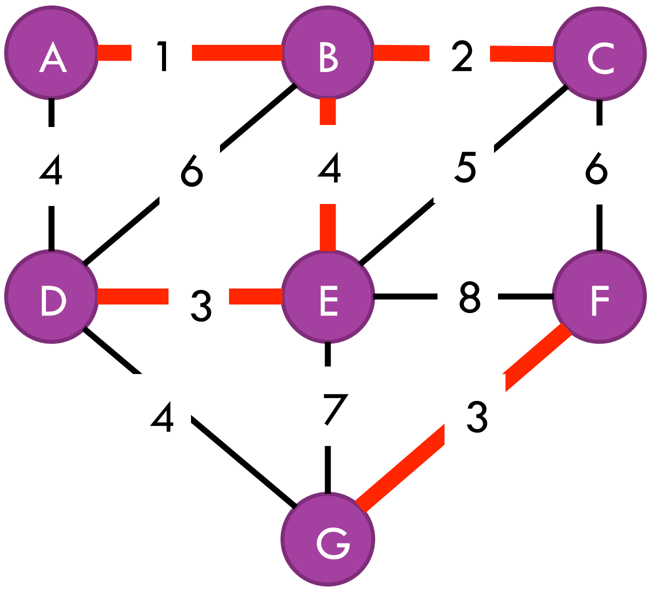
- AG={AB, BC, DE}

Algoritmo de Kruskal



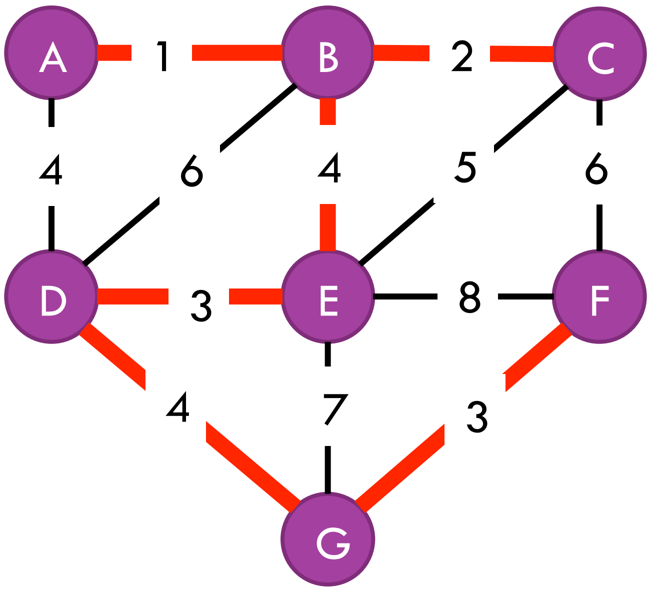
- AG={AB, BC, DE, GF}

Algoritmo de Kruskal



- AG={AB, BC, DE, GF, BE}

Algoritmo de Kruskal

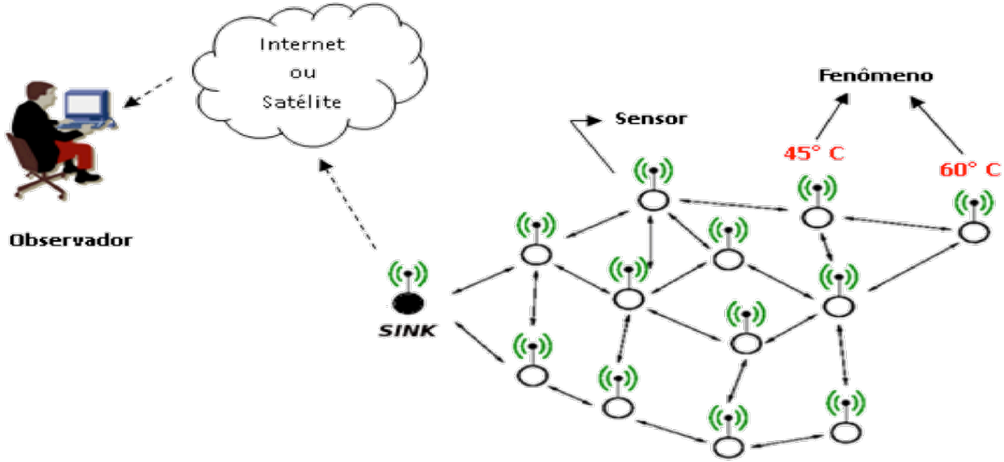


- AG={AB, BC, DE, GF, BE, DG}

Árvore de Steiner

Problema:

Dada uma rede representada por um grafo $G = (V, E)$, onde $V = v_1, \cdots, v_n$ é o conjunto de nós sensores, e E são as arestas que representam as conexões entre eles com um custo associado, o problema está em como construir a árvore de custo mínimo conectando todos os nós sensores $S = s_1, s_2, \cdots, s_m$, $S \subseteq V$ ao nó sink.



Árvore de Steiner

Problema:

- É um problema em otimização combinatória.
  – Consiste em achar a menor árvore que conecta um conjunto de pontos dados.
- Semelhante ao problema da árvore geradora mínima:
  – Dado um conjunto V de pontos (vértices), interligá-las por um grafo de menor tamanho, que é a soma dos tamanhos de todas as arestas.
  – As AGMs conectam um conjunto de pontos dados e possuem custo mínimo, mas requerem que todas as conexões sejam entre estes pontos.
- Diferença para o problema da AGM: vértices intermediários e arestas extras podem ser adicionados ao grafo, a fim de reduzir o comprimento da árvore de expansão.
  – Isso serve para diminuir o comprimento total da conexão (pontos ou vértices de Steiner).
  – Podem-se utilizar pontos extras, de modo que o custo da árvore gerada seja ainda menor do que o custo da AGM.
- Resultado: uma árvore, conhecida como a árvore Steiner.
  – Pode haver várias árvores Steiner para um determinado conjunto de vértices inicial.