CES-11

Algoritmos e Estruturas de Dados

Carlos Alberto Alonso Sanches
Juliana de Melo Bezerra

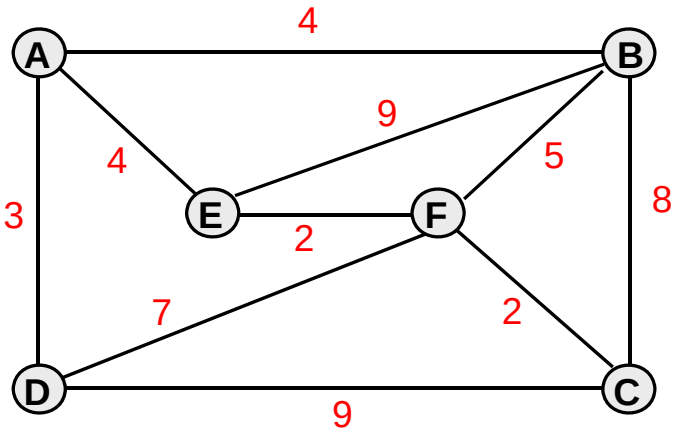
CES-11

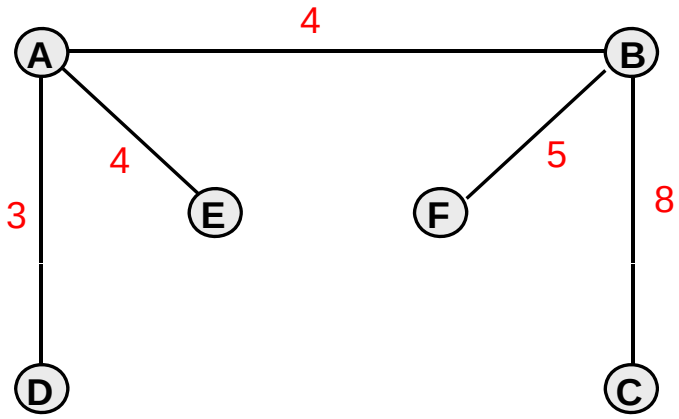
- Grafos
  - Conceitos gerais e representações
- Algoritmos em grafos
  - Exploração sistemática em largura
  - Caminhos mais curtos
  - Exploração sistemática em profundidade
  - Teste de aciclicidade
  - Ordenação topológica
  - Componentes fortemente conexos
  - Vértices e arestas de corte
  - Árvore geradora de custo mínimo

Árvore geradora de custo mínimo

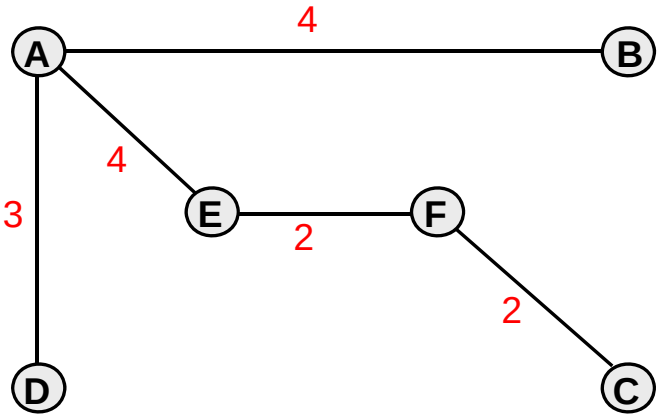
- Dado um grafo $G=(V,E)$, conexo e ponderado, com custo associado $c(e)$, $e \in E$, deseja-se encontrar um subgrafo T tal que:
  - seja gerador de G (isto é, possua todos os vértices);
  - seja acíclico e conexo (isto é, uma árvore);
  - tenha custo total $c(T) = \Sigma_{e \in E} c(e)$ que seja mínimo.
- T também costuma ser chamado de *árvore de espalhamento de custo mínimo.*
- Este conceito poderia ser generalizado para um grafo desconexo...

Exemplo





Árvore geradora
com custo 24

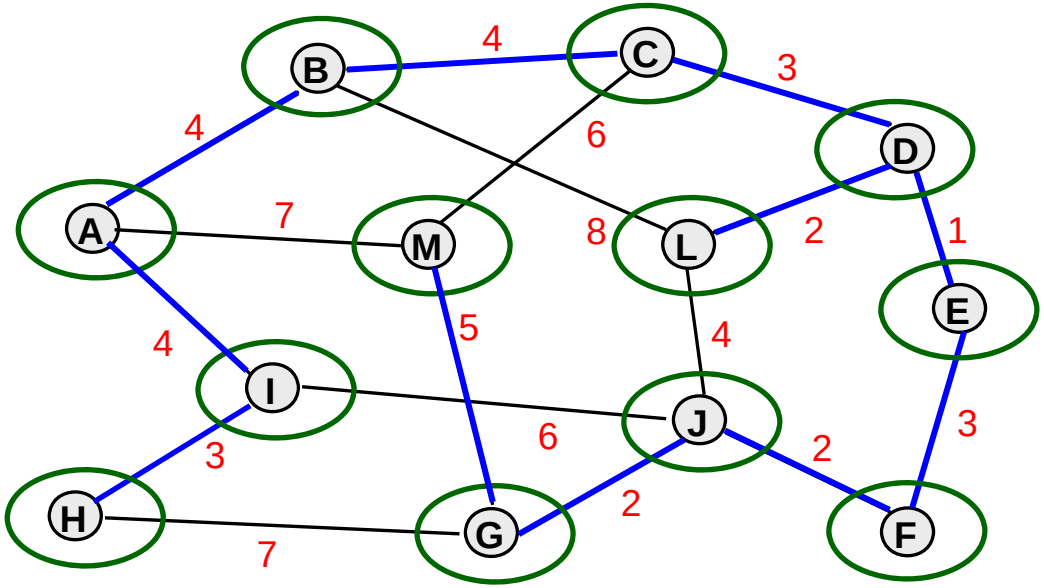


Árvore geradora
com custo 15

Ideia de Prim (1957)

- Inicialmente, T será um vértice arbitrário de G.
- Critério de inclusão de vértices e arestas em T:
  - Dentre todas as arestas de G incidentes em T, escolhe-se a de menor custo.
  - Essa nova aresta e seu vértice adjacente serão incluídos em T somente se esse novo vértice ainda não estiver em T.
  - O processo termina quando T ficar com n vértices.
- É preciso utilizar uma estrutura de dados que armazene em ordem crescente de custo os vértices ainda não incluídos na árvore.

Exemplo



c(T) = 33

Algoritmo de Prim

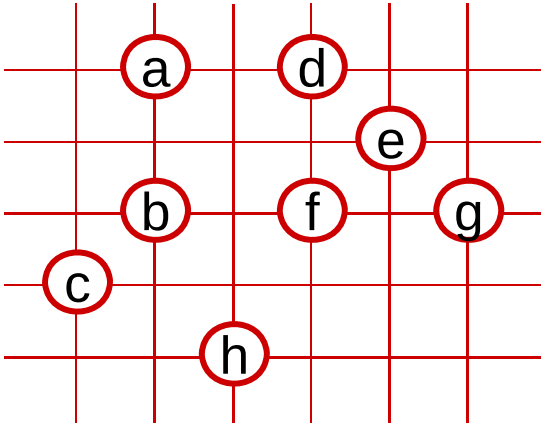
```
Prim(r) {
        T ← ∅;          \\ árvore de espalhamento de custo mínimo
        U ← {r};        \\ vértices que já estão na árvore
        while (U ≠ V) {
                <u,v> = aresta de custo mínimo | u∈U e v∈V-U;
                T ← T ∪ {<u,v>}; \\ aqui, T contém só arestas
                U ← U ∪ {v};
        }
}
```

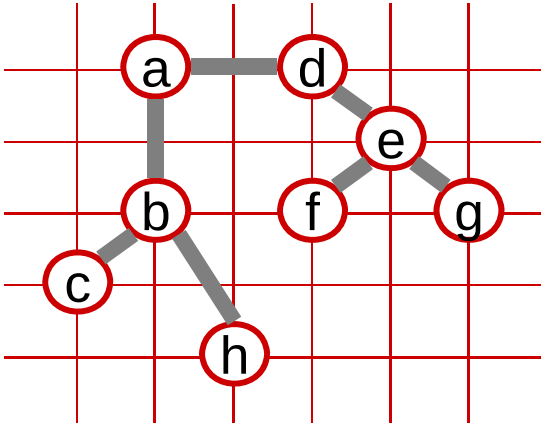
- Com *heap*, pode ser implementado em tempo $O((n+m).\log n)$:
  - O *heap* possuirá inicialmente todos os vértices com distância ∞ para a árvore em construção (com exceção do vértice r, com distância nula).
  - Quando um vértice é retirado do *heap* de mínimo, modificam-se as distâncias que seus vizinhos têm em relação à árvore.
  - No total, são realizadas n extrações de mínimo e m modificações de valor (será preciso manter um vetor auxiliar que armazena a posição corrente que cada vértice ocupa no *heap*).

Um exemplo de uso: caixeiro viajante

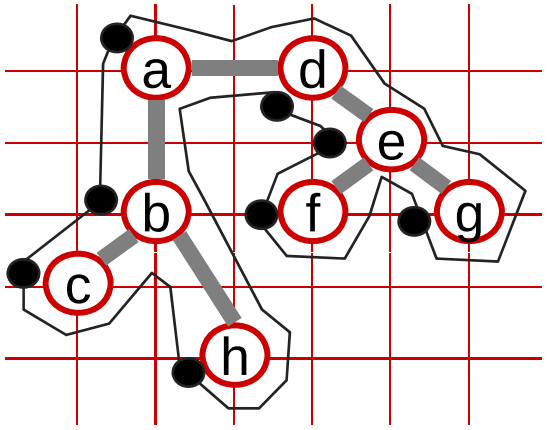
- Dado um grafo ponderado G, deseja-se encontrar um ciclo de custo mínimo que passe uma única vez em cada vértice.
- Este problema é *intratável* : não se conhece nenhum algoritmo polinomial capaz de resolvê-lo.
- É conhecido um *algoritmo aproximativo* (solução quase ótima) para um caso especial do problema: quando o grafo é *completo* e os custos das arestas são *distâncias euclidianas*.
- Ideia desse algoritmo:
  - Encontrar a árvore T de espalhamento de custo mínimo
  - Fazer o percurso pré-ordem em T
  - Transformar esse percurso num ciclo

Exemplo

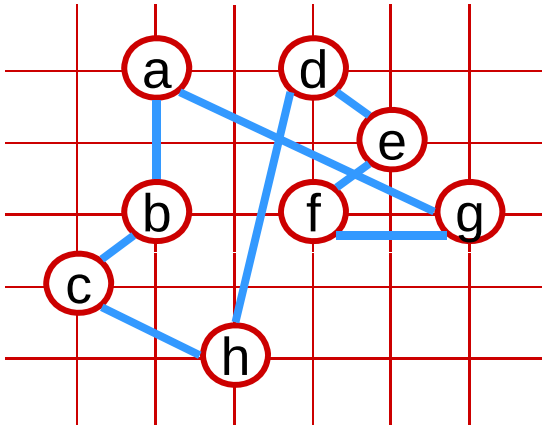




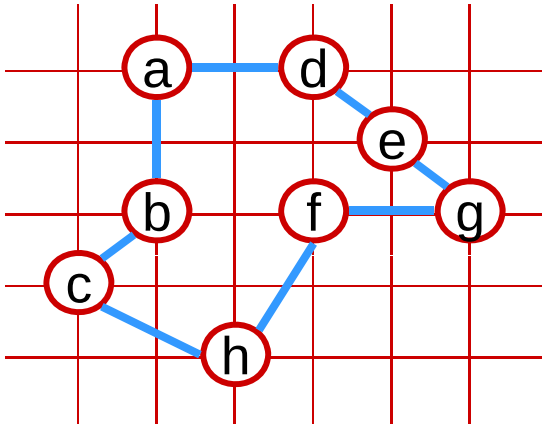
Árvore T: espalhamento de custo mínimo



Ciclo C' (com repetição de arestas)



Solução C quase ótima



Solução C* ótima

Comparação com a solução ótima

- Lembrando:
  - T: árvore de espalhamento de custo mínimo
  - C': ciclo ao redor de T, com repetição de arestas
  - C: ciclo baseado em C', sem repetição de arestas
  - C^*: ciclo de custo mínimo
- Seja $c(G)$ o custo associado a um grafo G.
- Se removermos uma aresta do ciclo mínimo C^*, obteremos uma árvore de espalhamento. Portanto, $c(T) < c(C^*)$.
- No ciclo C', cada aresta de T ocorre exatamente 2 vezes. Logo, $c(C) \leq c(C') = 2.c(T)$.
- Portanto, $c(C) < 2.c(C^*)$.
- Este algoritmo é eficiente e encontra uma solução cujo custo é menor que o dobro da ótima.

Exercícios

- Simular em diversos grafos:
  - Algoritmo de Dijkstra
  - Algoritmo de Tarjan para classificação de arcos
  - Variante do algoritmo de Tarjan para ordenação topológica
  - Variante do algoritmo de Tarjan para determinação de componentes fortemente conexos
  - Variante do algoritmo de Tarjan para encontrar vértices e arestas de corte
  - Algoritmo de Prim