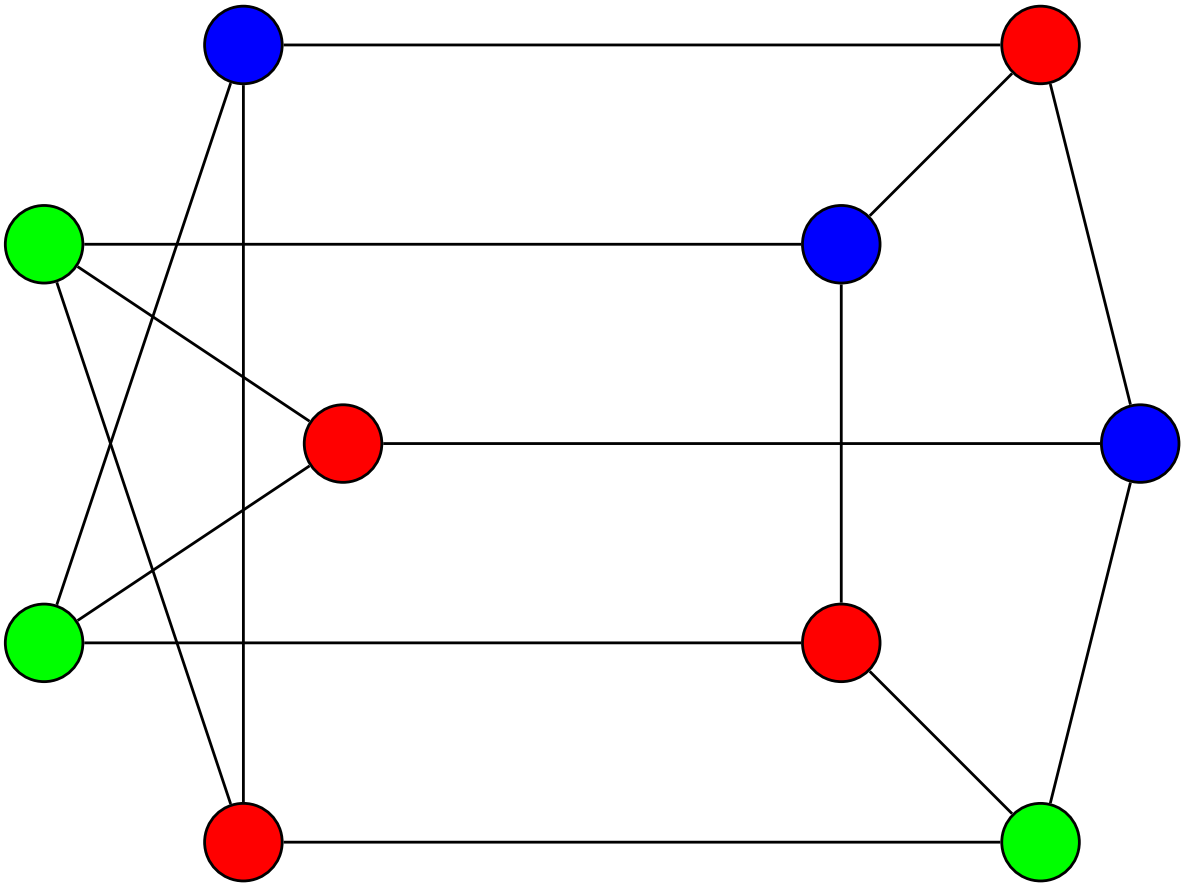
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Profa. Carla Negri Lintzmayer

Notas de aula da disciplina Teoria dos Grafos



Esse material é um compilado dos seguintes:

- Sedgewick, R.. *Algorithms in C, part 5: graph algorithms*. 3rd ed. Addison-Wesley. 2002.
- Bondy, J. A.; Murty, U. S. R.. *Graph Theory*. Graduate Texts in Mathematics. Springer. New York. 2008.
- Lintzmayer, C. N.; Mota, G. O.. *Notas de aulas - Análise de algoritmos e estruturas de dados*. Em construção.

8 Aula 10: árvore geradora mínima (MST - *Minimum Spanning Tree*)

- Seja G um grafo e $T \subseteq G$ um subgrafo de G que é uma árvore. Dizemos que T é uma *árvore geradora de G* se $V(T) = V(G)$.

Teorema 8.1. *Todo grafo conexo contém uma árvore geradora.*

Demonstração. Suponha que nem todo grafo conexo contém uma árvore geradora. Seja G um contra exemplo minimal no número de arestas[a]. Se G não tem ciclos, ele é uma árvore, o que é uma contradição. Então G tem algum ciclo e seja $e = xy$ alguma uma aresta desse ciclo. Note que $G' = G - e$ é conexo, uma vez que ainda existe um xy-caminho em G'. Pela escolha de G, podemos concluir que G' contém uma árvore geradora T'. Mas note que $V(T') = V(G') = V(G)$ e $E(T') \subseteq E(G') \subset E(G)$, o que significa que T' é árvore geradora de G, uma contradição. □

[a]Em outras palavras: Considere o conjunto formato por todas os grafos conexos que não têm uma árvore geradora e tome G como sendo um grafo deste conjunto que não contém nenhum outro grafo deste conjunto.

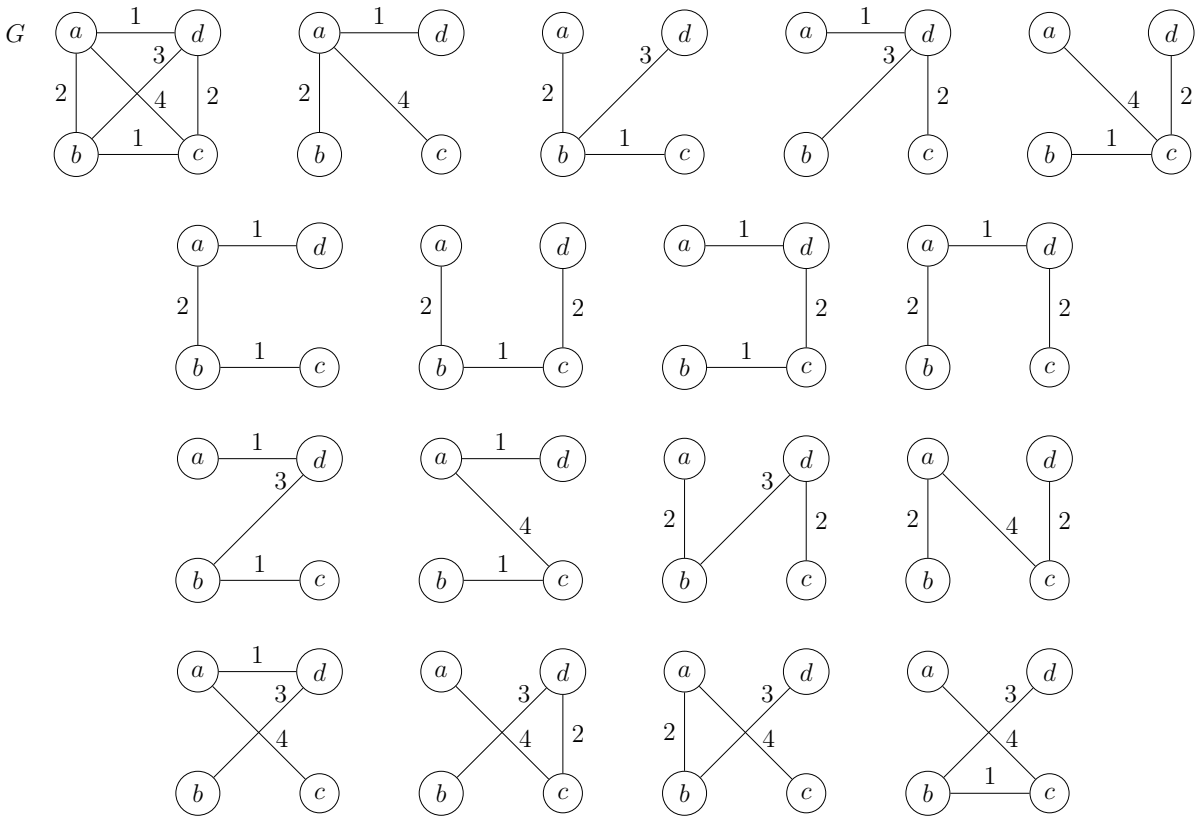
Proposição 8.2. *Sejam T e T' duas árvores geradoras de G. Para cada $e \in E(T) \setminus E(T')$, existe $f \in E(T') \setminus E(T)$ tal que $T - e + f$ e $T' - f + e$ são árvores geradoras de G.*

- Um problema clássico em otimização combinatória é o *problema da Árvore Geradora Mínima*:

  Entrada: grafo (conexo) G e $\omega \colon E(G) \to \mathbb{R}$.

  Objetivo: encontrar árvore geradora T de G tal que $\omega(T) := \sum_{e \in E(T)} \omega(e)$ é mínimo dentre todas as árvores geradoras de G.

- Exemplo: o grafo a seguir tem 16 possíveis árvores geradoras, sendo que duas delas são mínimas.



- Atenção: queremos uma árvore geradora que tenha peso mínimo, já que toda árvore geradora vai ter $V - 1$ arestas.
- Diversas aplicações diretas em projeto de redes, agrupamento de pontos (*clustering*) e taxonomia.
- A restrição de ser árvore existe por causa de pesos negativos.
  - Caso fossem todos positivos, bastaria exigir apenas algum "subgrafo gerador conexo de peso total mínimo".
- Algoritmos clássicos famosos que *resolvem* o problema:
  - Boruvka: inventado em 1926 por Otakar Borůvka, tempo $O(E \log V)$;
  - Prim: inventado em 1930 por Vojtěch Jarník e redescoberto por Prim em 1957 e Dijkstra em 1959, tempo $O(E \log V)$;
  - Kruskal: inventado em 1956 por Joseph Kruskal, tempo $O(E \log V)$.
- Têm caráter guloso: escolhem arestas para construir a árvore e o critério para a escolha se baseia nos custos das arestas "localmente".
- O algoritmo mais rápido conhecido foi inventado em 2000 por Bernard Chazelle, e tem tempo $O(E\,\alpha(V, E))$, onde α é a função inversa de Ackermann.

- Problemas relacionados:
  - Árvore Geradora Máxima: encontrar árvore geradora cuja soma dos pesos das arestas seja máxima;
  - Árvore Geradora Folhuda: encontrar árvore geradora com maior número de folhas possível;
  - Árvore de Steiner: dado conjunto R de vértices, encontrar árvore que passe por todos os vértices de R e tenha menor custo possível.

8.1 Princípios genéricos de algoritmos para MST

- Seja G um grafo e $X \subseteq E(G)$, com $X \neq \emptyset$.
  - Chamamos a partição $(X, V(G) \setminus X)$ de *corte* de G.
  - O conjunto $\partial_G(X)$, é definido como sendo o conjunto de arestas $\{uv \in E(G) \colon u \in X \text{ e } v \notin X\}$.
  - As arestas em $\partial_G(X)$ *cruzam* o corte.
  - Uma aresta $e \in \partial_G(X)$ é *mínima* ou *leve* para esse corte se $\omega(e) = \min\{\omega(f) \colon f \in \partial_G(X)\}$.
  - O corte *respeita* um conjunto $F \subseteq E(G)$ se $F \cap \partial_G(X) = \emptyset$ (nenhuma aresta de F cruza o corte).
- O lema a seguir nos diz que uma aresta que está sozinha em um corte não pode participar de ciclos.

Lema 8.3. *Sejam H um grafo e C um ciclo de H. Se $e \in E(H)$ pertence a C e $e \in \partial_H(X)$ para algum $X \subseteq V(H)$, então existe $f \in \partial_H(X)$, com $f \neq e$, que pertence a C tal que $f \in \partial_H(X)$.*

Demonstração. Seja $e = uv$ uma aresta pertencente a um ciclo C de H tal que $u \in X$ e $v \in V(H) \setminus X$ para algum $X \subseteq V(H)$. Assim, podemos escrever $C = (u, v, x_1, \ldots, x_k, u)$. Note que C pode ser dividido em dois caminhos distintos entre u e v. Um desses caminhos é a própria aresta $e = uv$ e o outro caminho, $(v, x_1, \ldots, x_k, u)$, necessariamente contém uma aresta $f \in \partial_H(X)$, uma vez que u e v estão em lados distintos do corte. □

- O lema a seguir nos mostra quais arestas são *seguras* para serem escolhidas por um algoritmo para o MST.

Teorema 8.4. *Sejam G um grafo conexo e $\omega\colon E(G) \to \mathbb{R}$. Seja $F \subseteq E(G)$ um conjunto de arestas que está contido em uma árvore geradora mínima de G. Sejam $(S, V(G) \setminus S)$ um corte que respeita F e $e \in \partial_G(S)$ uma aresta mínima que cruza esse corte. Vale que $F \cup \{e\}$ está contido em uma árvore geradora mínima de G.*

Demonstração. Sejam G um grafo conexo e $\omega\colon E(G) \to \mathbb{R}$ uma função de pesos sobre as arestas de G. Considere uma árvore geradora mínima T de G tal que $F \subseteq E(T)$ e seja $S \subseteq V(G)$ um conjunto tal que o corte $(S, V(G) \setminus S)$ respeita F.

Seja $e = uv \in E(G)$ uma aresta mínima que cruza o corte $(S, V(G) \setminus S)$, isto é, $\omega(e) = \min_{f \in \partial_G(S)}\{\omega(f)\}$. Suponha, para fins de contradição, que $F \cup \{e\}$ não está em nenhuma árvore geradora mínima de G.

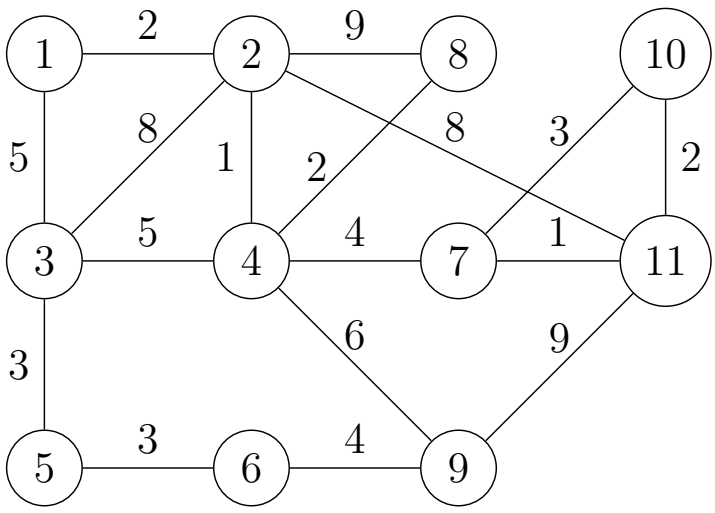
Como T é uma árvore geradora, então existe caminho de u a v em T, o que significa que $T + e$ contém exatamente um ciclo. Assim, pelo Lema 8.3, existe outra aresta $f \in E(T)$ que está no ciclo e cruza o corte $(S, V(G) \setminus S)$. Pela Proposição 8.2, o grafo $T' = T + e - f$ é também uma árvore geradora. Ademais, como $(S, V(G) \setminus S)$ respeita F, vale que $f \notin F$.

Por construção, temos $\omega(T') = \omega(T) - \omega(f) + \omega(e) \leq \omega(T)$, pois $\omega(e) \leq \omega(f)$, o que vale pela escolha de e. Como T é uma árvore geradora mínima e $\omega(T') \leq \omega(T)$, então só podemos ter $\omega(T') = \omega(T)$, isto é, T' é árvore geradora mínima também. Mas note que como $f \notin F$, teremos $F \cup \{e\} \subseteq E(T')$, o que é uma contradição. □

- O teorema anterior nos dá um algoritmo para gerar árvores geradoras mínimas:
  - Comece com um conjunto $F \subseteq E(G)$ que está contido em uma árvore geradora mínima.
  - Enquanto $G[F]$ não for conexo, adicione uma aresta segura a F.
- Esse algoritmo sempre devolve uma árvore geradora mínima basicamente porque mantém a *invariante* que F está sempre contida em uma árvore geradora mínima.
  - Por qual F começar?
  - Como escolher um corte que respeita F?
  - Como escolher uma aresta mínima que cruza um corte?
- O corolário a seguir é uma consequência direta do teorema anterior e responde à segunda pergunta.

Corolário 8.5. *Sejam G um grafo conexo e $\omega\colon E(G) \to \mathbb{R}$. Seja $H \subseteq G$ um subgrafo de uma árvore geradora mínima de G. Se $J \subseteq H$, $J \neq H$, contém uma ou mais componente conexas de H e $uv \in E(G)$ é uma aresta mínima no corte $\partial_G(V(J))$, então uv é uma aresta segura para H.*

- Algoritmo de Prim:
  - Comece com $H \subseteq G$ tal que $V(H) = \{s\}$, $s \in V(G)$ e $E(H) = \emptyset$.
  - Enquanto $V(H) \neq V(G)$:
    - Adicione a $E(H)$ uma aresta segura do corte $\partial_G(V(H))$.
  - Devolva H
- Algoritmo de Kruskal:
  - Comece com $H \subseteq G$ tal que $V(H) = V(G)$ e $E(H) = \emptyset$.
  - Enquanto H tem mais de uma componente conexa:
    - Adicione a $E(H)$ uma aresta segura de um corte $\partial_G(X)$ em que X não separa uma componente conexa de H (isto é, os vértices de qualquer componente conexa de H estão contidos em X ou não)
  - Devolva H
- Exemplo:



- A ideia do Algoritmo de Prim que foi descrita anteriormente pode ser melhor formalizada como a seguir.

```
1:  Função PRIM(G, ω)
2:      Para cada v ∈ V(G) faça
3:          visitado[v] ← 0
4:          pred[v] ← −1
5:      seja s ∈ V(G) qualquer
6:      visitado[s] ← 1
7:      Enquanto há vértice não visitado faça
8:          seja uv uma aresta de menor peso com visitado[u] = 1 e visitado[v] = 0
9:          visitado[v] ← 1
10:         pred[v] ← u
```

- A ideia do Algoritmo de Kruskal que foi descrita anteriormente pode ser melhor formalizada como a seguir.

```
1:  Função KRUSKAL(G, ω)
2:      sejam C[1..E] e F[1..V − 1] dois vetores
3:      copie as arestas de G para C com seus respectivos pesos
4:      ordene C de modo não-decrescente de acordo com o peso das arestas
5:      k ← 1
6:      Para i = 1 até E faça
7:          seja uv a aresta em C[i]
8:          Se u e v estão em componentes conexas distintas de G[F] então
9:              F[k] ← {uv}
10:             k ← k + 1
11:     Devolve F
```