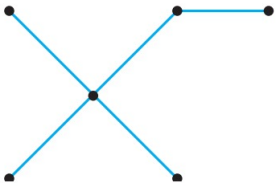
Matemática Discreta 2022
Escola de Matemática Aplicada, Fundação Getulio Vargas
Professora Maria Soledad Aronna
Monitor Felipe Vieira Costa 7 de novembro de 2022

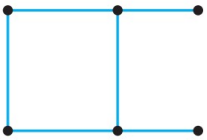
Lista 10

Exercício 1 Quais dos grafos a seguir são árvores? Explique.

(a)



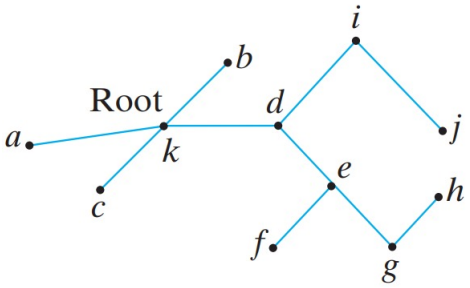
(b)



Exercício 2 Para quais valores de m e n o grafo completo bipartido de m e n vértices é uma árvore?

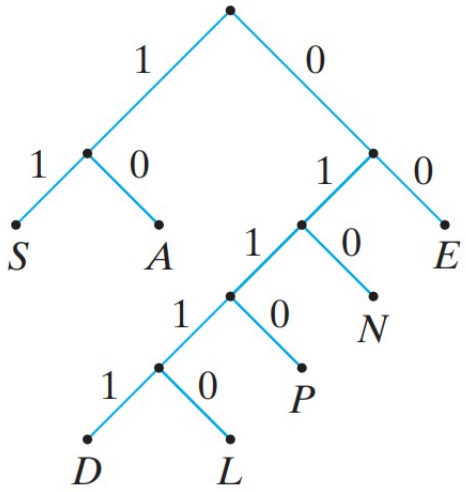
Exercício 3 Para quais valores de n o grafo completo de n vértices é uma árvore?

Exercício 4 Encontre o nível de cada vértice na árvore abaixo.



Exercício 5 Encontre a altura da árvore do Exercício 4.

Exercício 6 Decodifique cada sequência de bits usando o código de Huffman dado.



(a) 011000010

(b) 01111001001110

(c) 01110100110

(d) 1110011101001111

Exercício 7 Codifique cada palavra abaixo usando o mesmo código de Huffman acima.

(a) DEN

(b) NEED

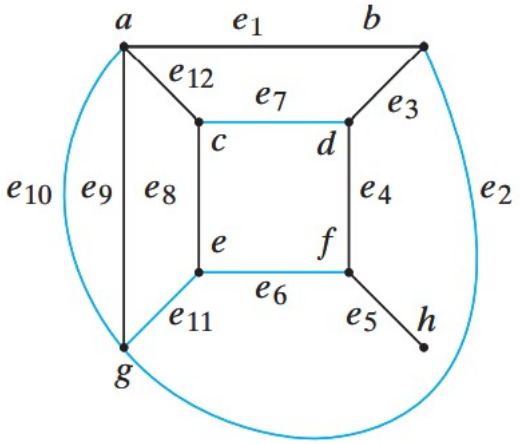
Exercício 8 Construa um código de Huffman ótimo para o conjunto de letras da tabela abaixo.

Letter	*Frequency*	*Letter*	*Frequency*
α	5	δ	11
β	6	ε	20
γ	6		

Exercício 9 Mostre que uma árvore é um grafo bipartido.

Exercício 10 Prove que T é uma árvore se, e somente se, T é conexo e quando uma aresta é adicionada entre quaisquer dois vértices, exatamente um ciclo é criado.

Exercício 11 Use o *breadth-first search* com a ordem de vértices $hgfedcba$ para encontrar uma árvore geradora para o grafo G abaixo.



Definição (Algoritmo Depth-First Search para Árvores Geradoras)**.** Este algoritmo encontra uma árvore geradora usando o método *depth-first search.*

Input: Um grafo conexo G com vértices ordenados $v_1, v_2, ..., v_n$

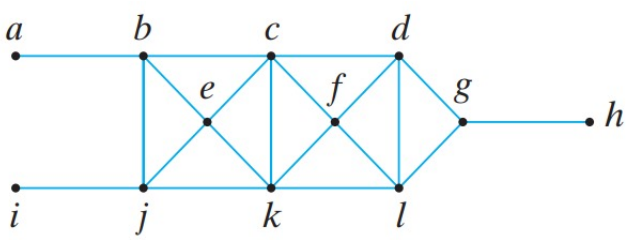
Output: Uma árvore geradora T.

$dfs(V, E)$ {
  $V' = \{v_1\}$ (V' = vértices da árvore geradora T)
  $E' = \varnothing$ (E' = arestas da árvore geradora T)
  $w = v_1$ (v_1 é a raiz da árvore geradora)
  while ($true$) {
    while (existe uma aresta (w, v) que não cria um ciclo quando adicionada a T) {
      escolha a aresta (w, v_k) com o menor k tal que quando adicionada a T não cria um ciclo em T
      add (w, v_k) em E'
      add v_k em V'
      $w = v_k$
    }
    if ($w == v_1$)
      return T
    w = pai de w em T
  }
}

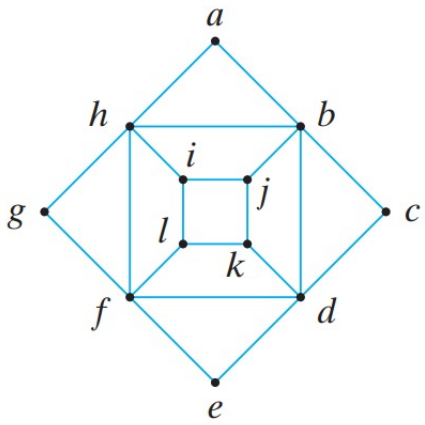
Exercício 12 Use o Algoritmo Depth-First Search com ordenação de vértices $hgfedcba$ para encontrar uma árvore geradora para o grafo do Exercício 11.

Exercício 13 Nos itens a seguir, encontre uma árvore geradora para cada grafo.

(a)



(b)



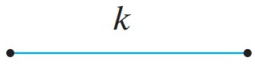
Exercício 14 Mostre, com um exemplo, que o Algoritmo Breadth-First Search pode produzir árvores geradoras idênticas para um grafo conexo G a partir de duas ordenações de vértices distintas de G.

Exercício 15 Prove que o Algoritmo Breadth-First Search está correto.

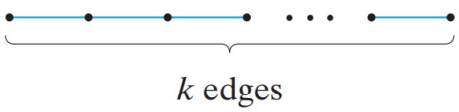
Exercício 16 Sob quais condições uma aresta em um grafo conexo G estará em qualquer árvore geradora de G?

Exercício 17 Sejam T e T' duas árvores geradoras de um grafo conexo G. Suponha que uma aresta x está em T mas não em T'. Mostre que existe uma aresta y em T' mas não em T tal que $(T - \{x\}) \cup \{y\}$ e $(T' - \{y\}) \cup \{x\}$ são árvores geradoras de G.

Exercício 18 Seja G um grafo com pesos no qual o peso de cada aresta é um inteiro positivo. Seja G' o grafo obtido a partir de G substituindo cada aresta



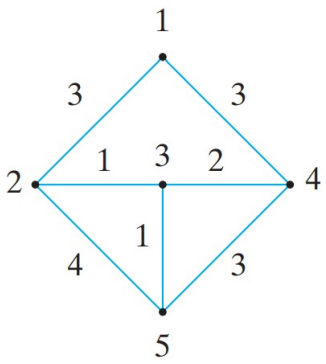
em G de peso k por k arestas sem peso em sequência:



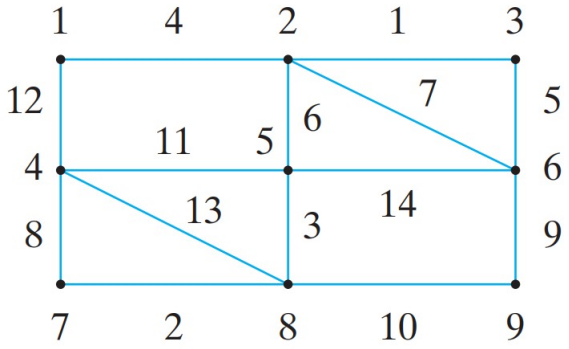
Mostre que o algoritmo de Dijkstra para encontrar o menor comprimento de cada caminho em um grafo com pesos G a partir de um um vértice fixo v para todos os outro vértices e realizar um breadth-first search no grafo sem pesos G começando pelo vértice v são, em efeito, o mesmo processo.

Exercício 19 Nos itens abaixo, encontre a *árvore geradora minimal* dada pelo Algoritmo de Prim para cada grafo.

(a)



(b)



Exercício 20 Mostre que o Algoritmo de Prim examina $O(n^3)$ arestas no pior caso.

Definição (Versão Alternativa do Algoritmo de Prim)**.** Este algoritmo encontra uma árvore geradora minimal em um grafo conexo com pesos G, Em cada passo, alguns vértices têm rótulos temporários e alguns têm rótulos permanentes. O rótulo do vértice i é denotado L_i.

Input: Um grafo conexo com pesos G com vértices $1, ..., n$ e vértice de início s. Se (i, j) é uma aresta, $w(i, j)$ é igual ao peso de (i, j); se (i, j) não é uma aresta, $w(i, j)$ é igual a inf.

Output: Uma árvore geradora minimal T.

prim_alternative(w, n, s) {
  Defina T como o grafo com o vértice s e sem arestas
  for $j = 1$ a n {
    $L_j = w(s, j)$ (esses rótulos são temporários)
    $back(j) = s$
  }
  $L_s = 0$
  torne L_s permanente
  while (existe rótulos temporários) {
    escolha o menor rótulo temporário L_i
    torne L_i permanente
    adicione a aresta $(i, back(i))$ a T
    adicione o vértice i a T
    for each L_k rótulo temporário

```
            if (w(i,k) < L_k) {
               L_k = w(i,k)
               back(k) = i
            }
        }
        return T
    }
```

Exercício 21 Mostre que o algoritmo acima examina $O(n^2)$ arestas no pior caso.

Exercício 22 Prove que o algoritmo anterior está correto; ou seja, que no fim dele, T é uma árvore geradora minimal.

Exercício 23 Seja G um grafo conexo com pesos e seja v um vértice de G e e uma aresta incidente em v com peso mínimo. Mostre que e está contida em alguma árvore geradora minimal.

Exercício 24 Mostre que se todos os pesos de um grafo conexo G são distintos, G contém uma única árvore geradora minimal.