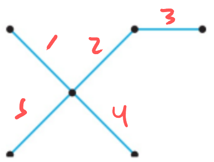
Matemática Discreta

4/10/24

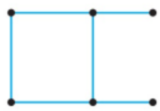
Lista 10

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

(a)



(b)



a) É árvore

G árvore $\Leftrightarrow |E| = n - 1$

6 vértices

5 arestas ✓

b) 6 vértices

6 arestas ✗ Não é

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

$K_{m,n} \rightarrow$ não pode ter ciclos

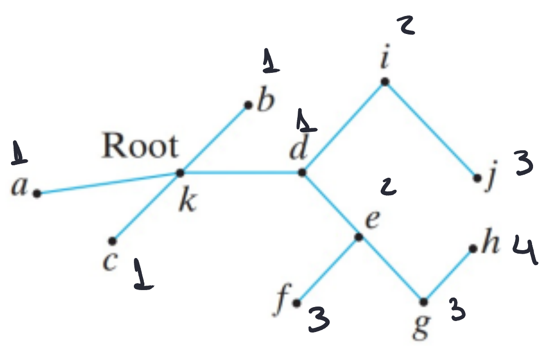
$m = 1, \; n \geq 0$

$n = 1, \; n \geq 0$

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

$n = 1$ e $n = 2$

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



Exercício 5 Encontre a altura da árvore do Exercício 4.

Maior nível possível é 6

Exercício 6 Mostre que uma árvore é um grafo bipartido.

Grafo bipartido $\iff$ 2-colorível

Tomando uma raíz, pintamos os vértices de níveis pares de uma cor e os de nível ímpar de outra

Exercício 7 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.

$\Rightarrow$) Sabemos, por definição, que árvores são conexas e acíclicas. Sabemos que $\forall v_i, v_j \in V$, $\exists! c_1$ que liga v_i e v_j. Logo, se conectarmos v_i ou v_j a v_k, criamos exatamente um ciclo:

$$C_1 = \{v_i, \dots, v_j\} \qquad e = \{v_j, v_k\}$$

$$\text{Ciclo} = \{C_1, e, \underbrace{C_2}_{\{v_k, \dots, v_i\}}\}$$

Existe por definição

$\Leftarrow$) Se ao criar uma aresta entre dois vértices, mais de um ciclo é criado, removendo essa aresta, temos, no mínimo, 2 caminhos de v_i a v_j, logo, o grafo original não pode ser árvore.

Exercício 8 Seja T uma árvore onde todos os vértices têm grau 4, exceto pelas folhas. Seja n o número de vértices de grau 4. Mostre que T possui $(2n+2)$ folhas.

$n-1 = \frac{1}{2}\cdot\sum_{i=1}^{n}\delta(v_i)$ $\longrightarrow$ k = quantidade de folhas

n = vértices de grau 4

$2n-2 = 4n-k \longrightarrow \boxed{k = 2n+2}$

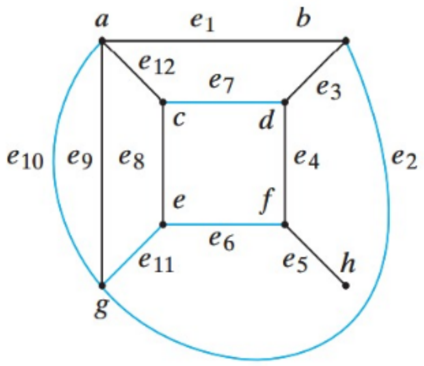
$\downarrow$

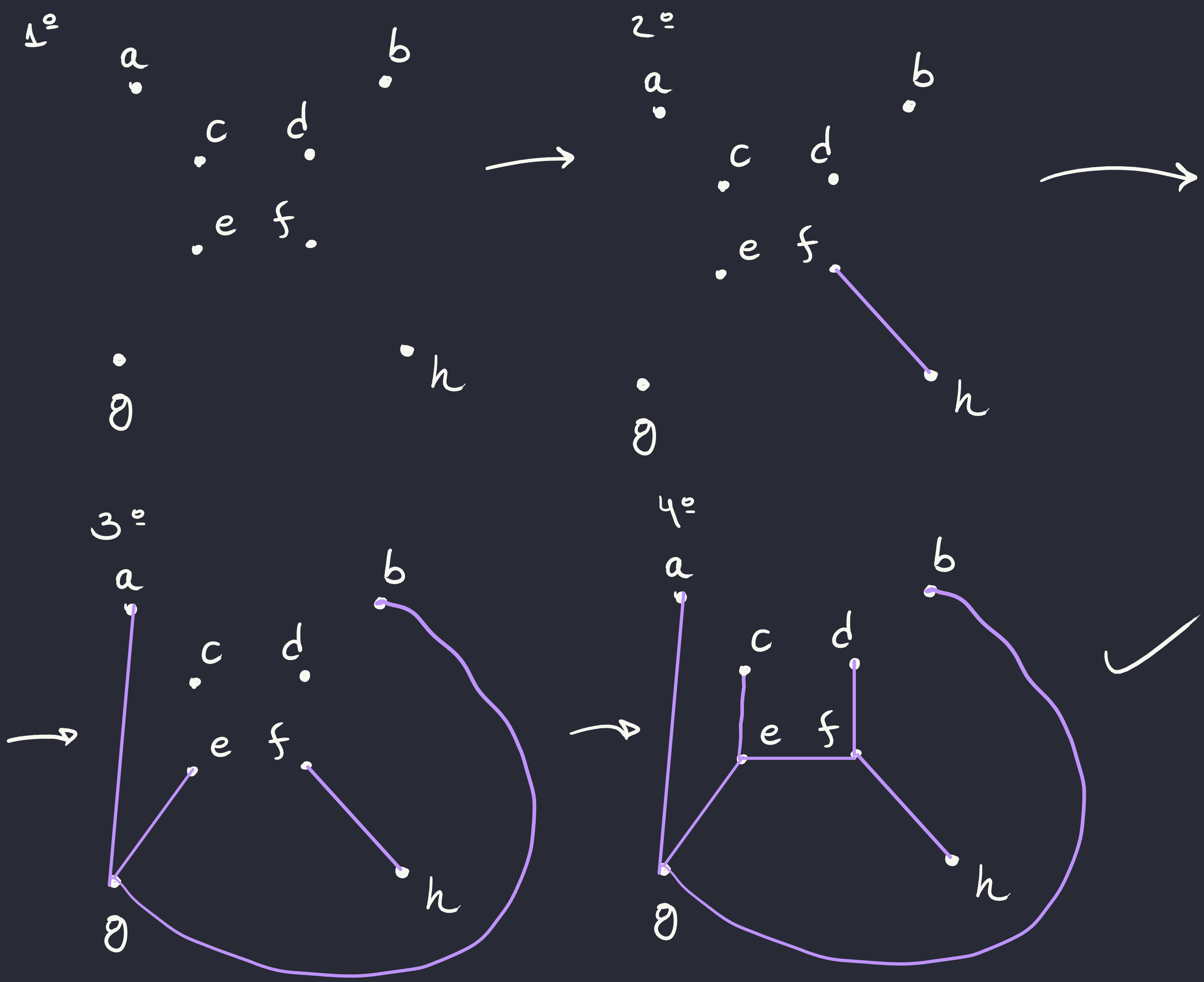
Repetições

Exercício 9 Seja T uma árvore onde todos os vértices adjacentes a uma folha têm grau pelo menos 3. Mostre que T possui pelo menos um par de folhas com um vizinho em comum.

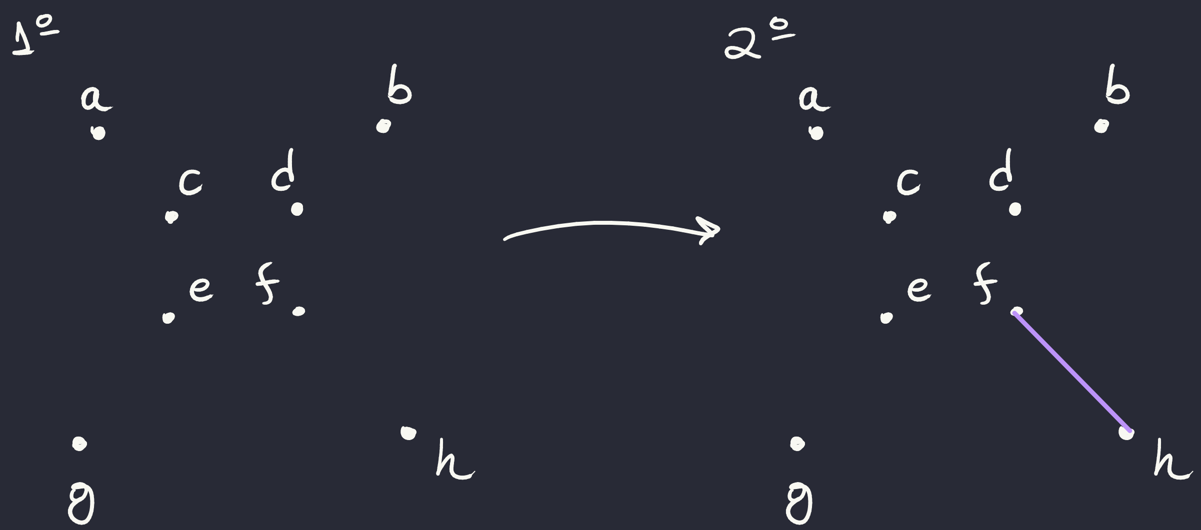
Dada uma árvore T e uma raíz V_0, denotamos k como a altura de T. Sabemos que todo vértice de nível k é uma folha, do contrário, k não seria a altura de T. Defina $U(v_i) :=$ nível do vértice v_i. Escolha v_i T.Q $U(v_i) = k$, como v_i é folha sabemos que $\delta(v_{i-1}) \geq 3$. Pegamos então a subárvore T' de T com raíz v_{i-1}. Como $U(v_{i-1}) = k-1$ em T, se a altura de $T' > 1$, k não será altura de T, o que é absurdo, logo, todo filho de v_{i-1} é uma folha também, logo, v_i tem, pelo menos, uma folha como vizinho.

Exercício 10 Use o *breadth-first search* com a ordem de vértices *hgfedcba* para encontrar uma árvore geradora para o grafo G abaixo.

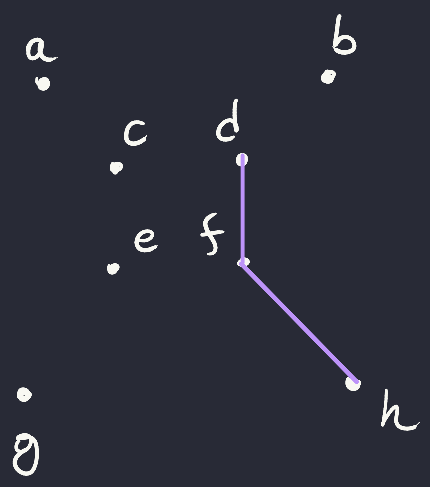




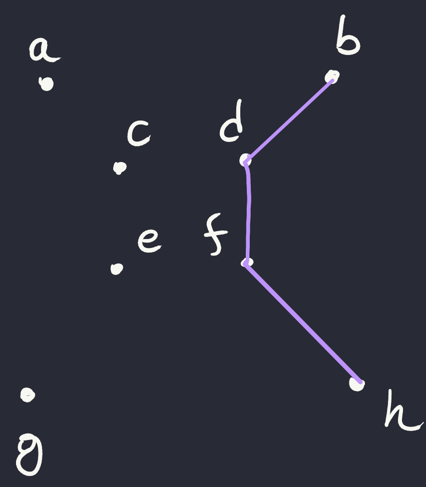
Exercício 11 Use o Algoritmo Depth-First Search com ordenação de vértices $hgfedcba$ para encontrar uma árvore geradora para o grafo do Exercício 11.



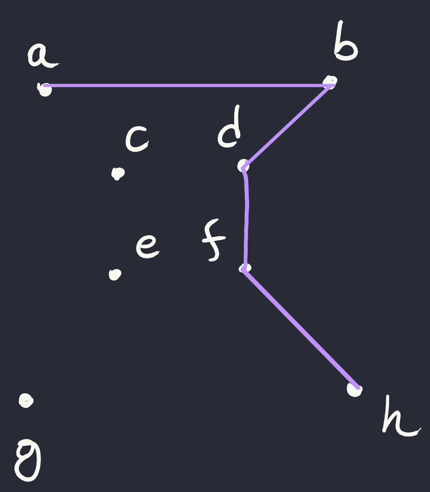
3º



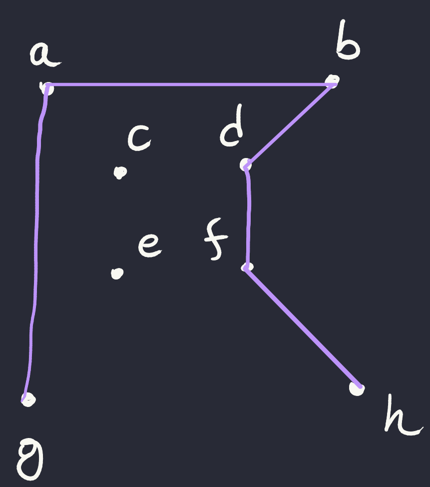
4º



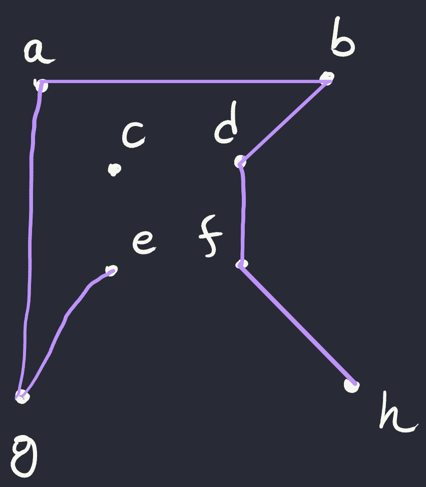
5º



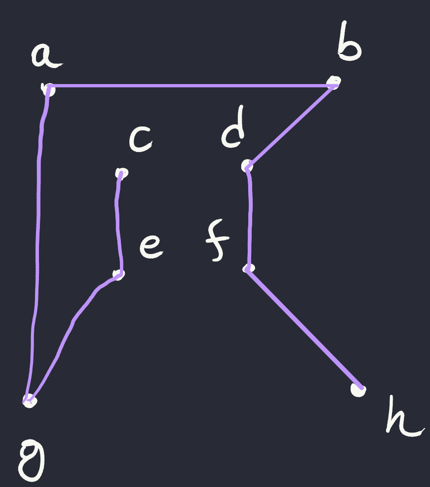
6º



7º

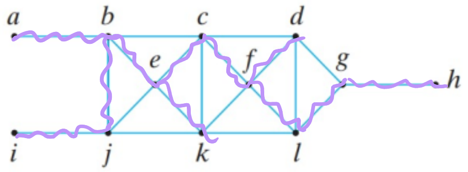


8º

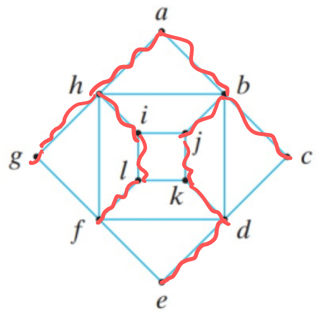


Exercício 12 Nos itens a seguir, encontre uma árvore geradora para cada grafo.

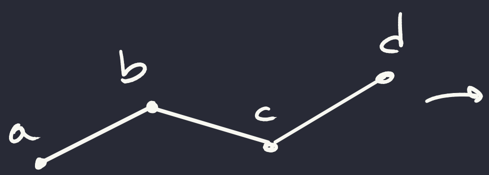
(a)



(b)



Exercício 13 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.



ordenação:
abcd = dcba

Exercício 14 Prove que o Algoritmo Breadth-First Search está correto.

```
bfs(V, E) {
   // V = vertices ordered v1, ..., vn; E = edges
   // V' = vertices of spanning tree T; E' = edges of spanning tree T
   // v1 is the root of the spanning tree
   // S is an ordered list
1  S = (v1)
2  V' = {v1}
3  E' = ∅
4  while (true) {
5     for each x ∈ S, in order,
6        for each y ∈ V − V', in order,
7           if ((x, y) is an edge)
8              add edge (x, y) to E' and y to V'
9     if (no edges were added)
10       return T
11    S = children of S ordered consistently with the original vertex ordering
   }
}
```

1. Acha um subgrafo de G
→ Fácil de visualizar pela construção do algoritmo

Não Terminado

2. Retorna um grafo acíclico e conexo
→ Linhas 9 e 10 eu adiciono uma aresta na árvore se a aresta interliga um vértice da árvore com um que não está na árvore, logo, sempre há um caminho entre todos os vértices (Conexo)
→ Se existe uma aresta que forma ciclo no grafo retornado, isso quer dizer que em algum momento, o algoritmo conectou dois vértices que já estavam na árvore.

Exercício 15 Sob quais condições uma aresta em um grafo conexo G estará em qualquer árvore geradora de G?

Se, ao remover a aresta, G vira um grafo desconexo

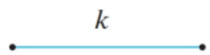
Exercício 16 Sejam T e T' duas árvores geradoras de um grafo conexo G. Suponha que uma aresta x está em T mas não em T'. Mostre que existe uma aresta y em T' mas não em T tal que $(T - \{x\}) \cup \{y\}$ e $(T' - \{y\}) \cup \{x\}$ são árvores geradoras de G.

Escolhida uma raíz w para T e T', e tendo x e y como:

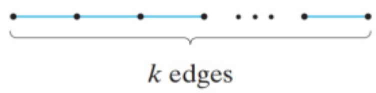
$x = \{x_{i-1}, x_i\}$ $y = \{y_{i-1}, y_i\}$

Sabemos que x está em todo caminho de w até os descendentes de x_i. Escolhemos então y como QUALQUER aresta incidente em x_i ou seus descendentes, desde que ela esteja em T' (É absurdo que nenhuma delas esteja em T', se isso ocorresse, T' não seria árvore). Assim, obtemos $(T-\{x\}) \cup \{y\}$ como árvore geradora e $(T'-\{y\}) \cup \{x\}$ também

Exercício 17 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



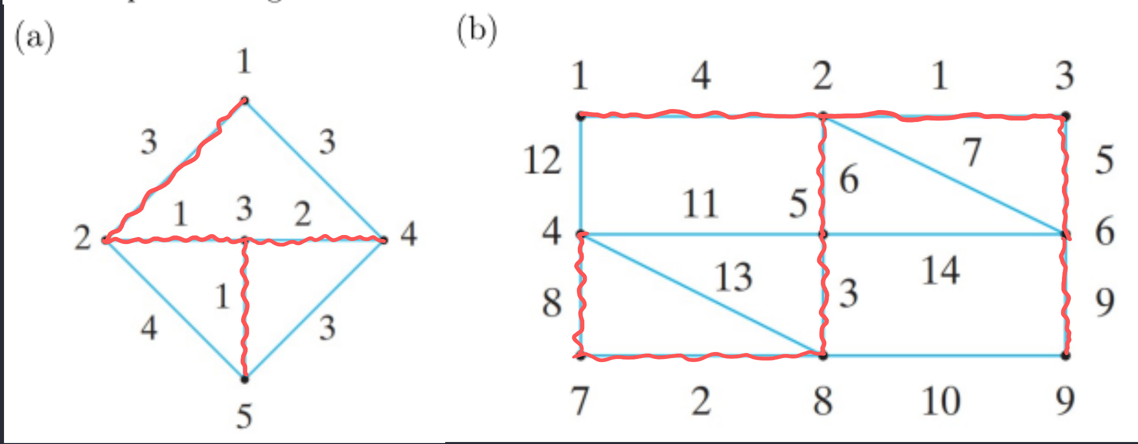
em G de peso k por k arestas sem peso em sequência:



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.

→ Basta mostrar que o breadth-first search, nessas condições, acha o menor caminho.

Exercício 18 Nos itens abaixo, encontre a *árvore geradora minimal* dada pelo Algoritmo de Prim para cada grafo.



Exercício 19 Mostre que o Algoritmo de Prim examina $O(n^3)$ arestas no pior caso.

```
prim(w, n, s) {
  // v(i) = 1 if vertex i has been added to mst
  // v(i) = 0 if vertex i has not been added to mst
  for i = 1 to n
    v(i) = 0
  // add start vertex to mst
  v(s) = 1
  // begin with an empty edge set
  E = ∅
  // put n − 1 edges in the minimal spanning tree
  for i = 1 to n − 1 {
    // add edge of minimum weight with one vertex in mst and one vertex
    // not in mst
    min = ∞
    for j = 1 to n
      if (v(j) == 1) // if j is a vertex in mst
        for k = 1 to n
          if (v(k) == 0 ∧ w(j, k) < min) {
            add_vertex = k
            e = (j, k)
            min = w(j, k)
          }
    // put vertex and edge in mst
    v(add_vertex) = 1
    E = E ∪ {e}
  }
  return E
```

O pior caso possível é quando eu necessito percorrer todos os 3 for loops.

1º: $n-1$ iterações

2º: n iterações

3º: n iterações

$n \cdot n(n-1)$

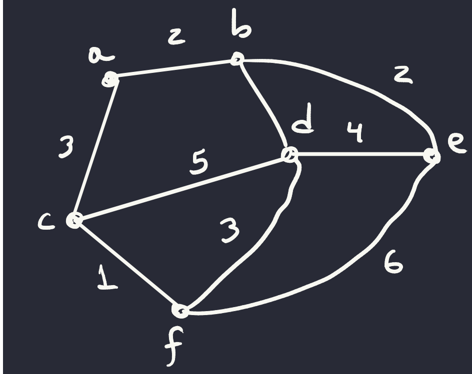
$n^3 - n^2 \Rightarrow \boxed{O(n^3)}$

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)
    $back(j) = s$
  } I
  $L_s = 0$
  torne L_s permanente
  while (existe rótulos temporários) { II
    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 ($w(i, k) < L_k$) {
        $L_k = w(i, k)$
        $back(k) = i$ III
      }
  }
  return T
}



$\boxed{L_b = 2}$ $\quad L_d = \infty$

$L_c = 3$ $\quad L_e = \infty$

$L_f = \infty$

Exercício 20 Mostre que o algoritmo acima examina $O(n^2)$ arestas no pior caso.

Pior caso onde todos os for são percorridos:

$$\underbrace{n}_{I} + \underbrace{n}_{II} \cdot \underbrace{(n)}_{III} = n + n^2 \Rightarrow O(n^2)$$

↳ A cada interação, um rótulo temporário é removido.

Exercício 21 Prove que o algoritmo anterior está correto; ou seja, que no fim dele, T é uma árvore geradora minimal.

Seja T_i o grafo obtido após a i-ésima iteração, T_i é uma árvore, por construção. Vamos mostrar que T_i é uma árvore mínima do subgrafo G com $V' = \{1, ..., i\}$.

Base:

T_1 = Aresta de menor peso entre v_1 e v_2

Indução:

A cada iteração T_i, a aresta que liga um vértice de T_i com um fora de T_i com menor peso.

Continuar (Tentar entender melhor o algoritmo)

Exercício 22 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.

Como e tem peso mínimo, $\forall a \in E$, $w(a) \geq e$

$w(e)$ = peso da aresta. Aplicando o algoritmo de Prim, quando uma das arestas que "e" incide for adicionado na árvore geradora, a aresta de menor peso será escolhida. Se $\nexists e' \in E \,/\, w(e') = w(e)$, então "e" vai ser a aresta escolhida, do contrário, tanto e' quanto e podem ser escolhidas

Exercício 23 Mostre que se todos os pesos de um grafo conexo G são distintos, G contém uma única árvore geradora minimal.

Vamos supor que existe 2 árvores geradoras minimais diferentes T_1 e T_2.

$$w(T_1) = \sum_{i=1}^{n-1} w(i) \qquad w(T_2) = \sum_{j=1}^{n-1} w(j)$$

$\sum_{i=1}^{n-1} w(i) = \sum_{j=1}^{n-1} w(j)$, como o somatório é igual, todos os pesos devem ser iguais, e como $T_1 \neq T_2$, há ao menos 1 aresta diferente, com peso diferente, logo, se os somatórios são iguais, concluimos que $T_1 = T_2$, ou seja, há apenas uma árvore minimal.