6ª Série de exercícios – Teoria dos Grafos

Árvores Geradoras Mínimas (MST's) e os Algoritmos de Kruskal e Prim

1) Um problema relevante em diversas aplicações computacionais consiste obter uma MST a partir de um grafo G. Existem diversos algoritmos para solucionar esse problema. Responda:

a) O que é uma árvore geradora mínima (MST)? Explique

b) Com base no pseudo-código a seguir, explique o funcionamento do algoritmo de Kruskal. Descreva qual o critério adotado para se definir que uma aresta é segura, explicando as primitivas básicas utilizadas no algoritmo. Sua solução é única, ou seja, todas as execuções gerarão a mesma MST? Explique.

```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

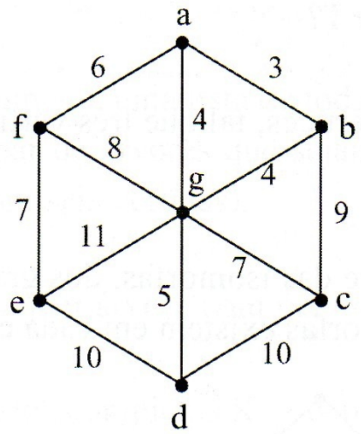
c) Prove que o algoritmo MST-Kruskal sempre retorna uma MST de G.

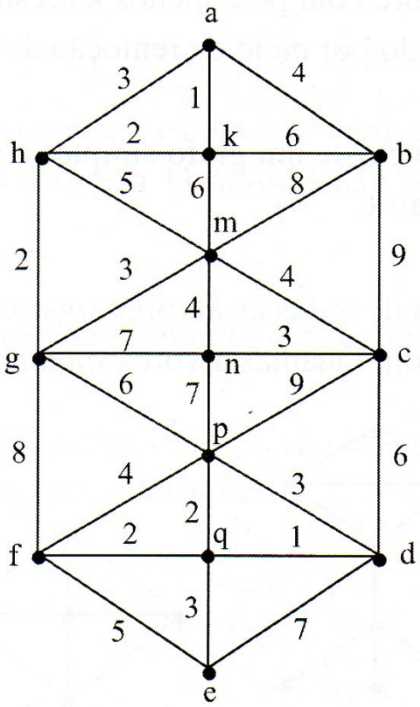
d) Com base no pseudo código a seguir, explique o funcionamento do algoritmo de Prim. O que garante a escolha de arestas seguras? Podemos afirmar que a MST obtida será única, ou seja, em todas as execuções o algoritmo retornará a mesma solução? Explique

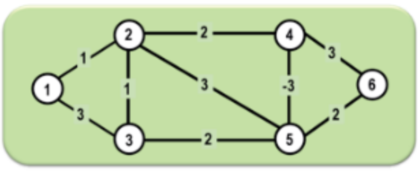
```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

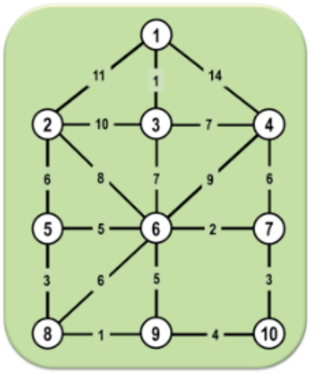
e) Prove que o algoritmo MST-Prim sempre retorna uma MST de G.

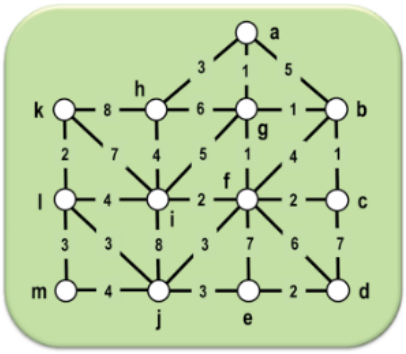
2) As questões a seguir fazem referência aos grafos a abaixo.

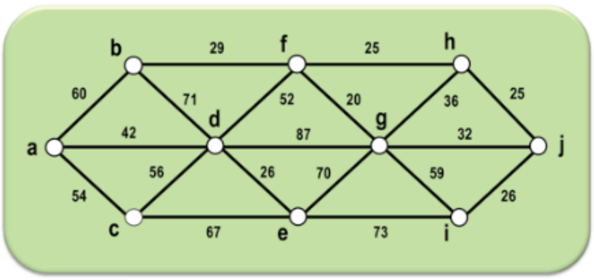


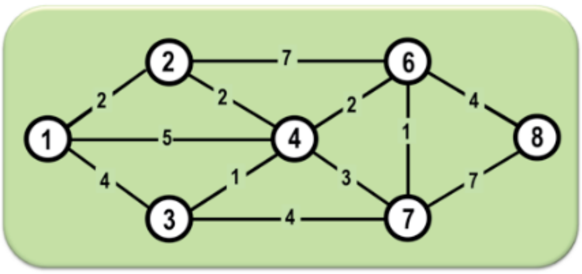


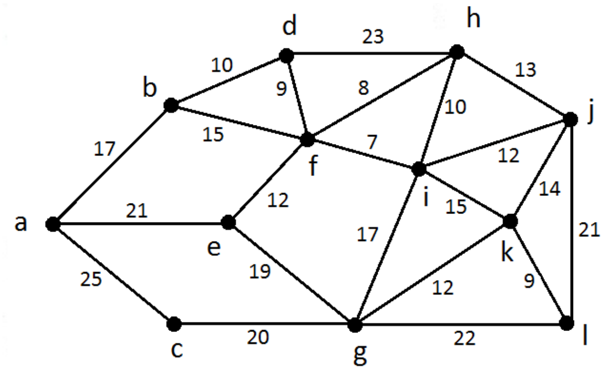












a) Encontre a árvore geradora minima (MST) para cada um dos grafos conectados ponderados usando os algoritmos de Kruskal e Prim. (Realize o trace completo dos algoritmos). Desenhe as árvores resultantes em cada um dos casos. Qual é o peso das MST's obtidas?

b) Codifique as árvores obtidas no item anterior utilizando o código de Prufer.

3) A descrição a seguir é a de um terceiro algoritmo para encontrar uma árvore geradora mínima de um grafo G conectado e ponderado com n vértices: remova uma por uma as arestas de G com os maiores pesos, de maneira que cada remoção não implique um grafo desconectado, até que sobrem apenas n-1 arestas. O subgrafo resultante é uma árvore geradora mínima de G. Simule um exemplo desse método num dos grafos acima e verifique o resultado obtido. Compare o peso com as árvores obtida por Prim.

4) Aplique o algoritmo de Kruskal no grafo direcionado a seguir. Que problemas podemos encontrar?