Dupla 23: Antônio F. de C. Neto e Lucas Melo dos Santos

Grafos 2

Exercícios

10. Let $G = (V, E)$ be an (undirected) graph with costs $c_e \geq 0$ on the edges $e \in E$. Assume you are given a minimum-cost spanning tree T in G. Now assume that a new edge is added to G, connecting two nodes $v, w \in V$ with cost c.

(a) Give an efficient algorithm to test if T remains the minimum-cost spanning tree with the new edge added to G (but not to the tree T). Make your algorithm run in time $O(|E|)$. Can you do it in $O(|V|)$ time? Please note any assumptions you make about what data structure is used to represent the tree T and the graph G.

(b) Suppose T is no longer the minimum-cost spanning tree. Give a linear-time algorithm (time $O(|E|)$) to update the tree T to the new minimum-cost spanning tree.

a) Podemos utilizar o algoritmo de Prim para obter a MST, **porém**, quando chegarmos nos nós “v” ou “w”, iremos comparar as arestas que saem desses nós com a nova aresta inserida. A partir daí temos 3 casos:
Caso 1)

- O custo “c” da nova aresta é **menor** que o custo da menor aresta de “v” ou “w”

- Então a nova aresta é uma possível candidata a fazer parte da nova MST, **mas** antes devemos verificar se ela não forma um ciclo.

- Se a nova aresta não forma um ciclo, então temos uma nova MST T (antiga T não existe mais)

```
If c < c_menorAresta then:
        If node is not visited then:
                return 1 # altera a MST
        else:
                return 0 #mesma MST
```

Caso 2)

- O custo “c” da nova aresta é **igual** que o custo da menor aresta de “v” ou “w”

- Então a nova aresta é uma possível candidata a fazer parte da nova MST, **mas** antes devemos verificar se ela não forma um ciclo.

- Se a nova aresta não forma um ciclo, então temos uma nova MST T (agora temos duas MST’s)

```
If c == menorAresta then:
    If node is not visited then:
        return 2 # duas MST
    else:
        return 0 #mesma MST
```

Caso 3)

- O custo “c” da nova aresta é **maior** que o custo da menor aresta de “v” ou “w”

- Então a nova aresta não irá fazer parte da MST T. (mesma T)

```
If “c” > menorAresta then:
    return 0 # mesma MST
```

Note que o algoritmo roda em O(E) pois percorre todas as arestas do grafo, e quando faz a comparação com a nova aresta, o algoritmo acaba e retorna uma resposta. O algoritmo pode rodar em O(V) se os nós “v” e “w” forem encontrados rapidamente.

b) Se T não é mais a MST, então a nova aresta entre “v” e “w” formou uma nova arvore geradora mínima. Partindo desse ponto, podemos utilizar o algoritmo de Prim para achar a nova MST, partindo de “v” ou “w” :
1 – Partindo de “v”, setamos as arestas candidatas em um heap.
2 – Verificamos qual a menor aresta dentre as candidatas
3 – Incluimos o nó alcançado no cut (marca como visitado) e e inclui a aresta na MST
4 - A partir do nó alcançado, verificamos as arestas canditadas e incluímos no heap (arestas que geram um ciclo são descartadas)
5 – Pegamos a menor aresta da fila de prioridade (heap) e repetimos o processo com o próximo nó alcançado.
6 – Quando todos os nós forem alcançados, terminamos o algoritmo de Prim e obtemos a T atualizada.

11. Suppose you are given a connected graph $G = (V, E)$, with a cost c_e on each edge e. In an earlier problem, we saw that when all edge costs are distinct, G has a unique minimum spanning tree. However, G may have many minimum spanning trees when the edge costs are not all distinct. Here we formulate the question: Can Kruskal's Algorithm be made to find all the minimum spanning trees of G?

Recall that Kruskal's Algorithm sorted the edges in order of increasing cost, then greedily processed edges one by one, adding an edge e as long as it did not form a cycle. When some edges have the same cost, the phrase "in order of increasing cost" has to be specified a little more carefully: we'll say that an ordering of the edges is *valid* if the corresponding sequence of edge costs is nondecreasing. We'll say that a *valid execution* of Kruskal's Algorithm is one that begins with a valid ordering of the edges of G.

For any graph G, and any minimum spanning tree T of G, is there a valid execution of Kruskal's Algorithm on G that produces T as output? Give a proof or a counterexample.

Para apresentar a prova, primeiro devemos definir o conceito de ciclo:

1) Conceito de Ciclo
   - Supomos um ciclo C dentro de um grafo G, e A é uma aresta dentro de C. Se a custo(A) > custo(X), sendo X todas as arestas dentro de C, então A **não** pertence a arvore geradora mínima.

2) Prova por mudança de argumento:
   - Vamos supor que A pertence a uma arvore geradora mínima T
   - Se deletarmos A de T, então temos **dois** componentes conectados, pois qualquer aresta que retiramos de uma MST gerará dois SCC's
   - Como A pertence ao ciclo C, então temos outra aresta, que iremos chamar de B, que conecta os dois SCC's.
   - Como já definimos que A tem o maior custo dentro de C, então custo(A) > custo(B)
   - Portanto, temos uma outra árvore geradora T* = (T - {A}) U {B}
   - E se custo(A) > custo(B), então custo(T) > custo(T*)
   - Por fim, chegamos a uma **contradição**, pois não é possível termos 2 MST's se elas não possuem o mesmo custo. Concluimos que na verdade T* é a arvore geradora mínima.

   - Ademais, se deletarmos B de T*, geramos dois SCC's
   - Se considerarmos uma aresta P que conecta os dois SCC's, tal qual custo(P) = custo(B)
   - E supormos T** = (T* - {B}) U {P}
   - Então custo(T**) = custo(T), pois custo(P) = custo(B)
   - Portanto, encontramos outra MST no grafo G