Tirgul 11

(and more)

sample questions

Q. Let G = (V,E) be an undirected, connected graph with an edge weight function w : E→R. Let r∈ V . A depth-first search (DFS) of G starting at r will construct a spanning tree for G. That spanning tree is not necessarily a minimum weight spanning tree. DFS examines the neighbors of a vertex in an order that is arbitrarily chosen. Suppose a graph search utilizes the edges incident to a vertex in non decreasing order by weight. Call such a graph search - weight aware.

a. Give the pseudocode for a weight aware version of DFS.

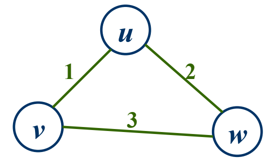
b. Prove or disprove: The depth-first search tree returned by a weight aware DFS is always an MST.

a.

```
DFS-WeightAware(G, w, s)
    ▷ G = (V, E) is an n-vertex, connected, undirected graph
    ▷ w : E → R is the edge weight function
    ▷ s ∈ V is the start vertex
    for u ∈ V
        do color[u] ← WHITE
           π[u] ← NIL
    time ← 0
    ▷ Since G is connected, we only start DFS once
    DFS-Visit-WeightAware(s)

DFS-Visit-WeightAware(u)
    color[u] ← GRAY
    time ← time + 1
    d[u] ← time
    c ← |Adj[u]|          ▷ number of neighbors of u
    N[1..c] ← Adj[u] sorted in nondecreasing order by w(u, v)
    for i ← 1 to c
        do v ← N[i]
           if color[v] = WHITE
              then π[v] ← u
                   DFS-Visit-WeightAware(v)
    color[u] ← BLACK
    time ← time + 1
    f[u] ← time
    return d, π
```

b. The claim is erroneous:

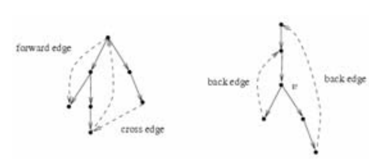


Q. (Cormen 22.3-12) A directed graph $G=(V,E)$ is ***singly-connected*** if there is at most one simple path between any two vertices $u,v \in V$, give an algorithm for deciding whether a graph G is ***singly-connected***

A. Construct the ***Strongly Connected Components (SCC)*** $s_1, s_2, s_3, \ldots, s_k$ of G. For each such ***SCC*** Run DFS from each of its vertices (total of $O(|V\cdot(V+E)|)$). A Forward edge or a cross edge means the graph is not ***singly-connected***. We then run DFS from each vertex in G^{SCC}, since there are no back edges, each DFS will take $O(|V|)$ – a total of $O(|V^2|)$.

So the total complexity is $O(|V\cdot(V+E)|)$

Improvement:



notice the fact that if a ***SCC*** has either a cross edge, a forward edge or double back edges it is not ***singly-connected***

We can do with just one DFS for each and since there are no cross edge, a forward edge and at least one back edge, the complexity is $O(|V|)$

➲ a total of $O(|V^2|)$.

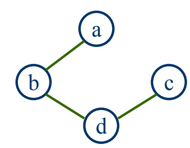
Given a graph G represented using an adjacency matrix M, what is M^2 , M^3 , M^k ?
we know that M holds all paths of length one between two vertices (edges)
remember matrix multiplication:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & & & \\ a_{n1} & & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & & & \\ b_{n1} & & \cdots & b_{nn} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & & & \\ c_{n1} & & \cdots & c_{nn} \end{pmatrix}$$

where $c_{11} = a_{11} \cdot b_{11} + a_{12} \cdot b_{21} + a_{13} \cdot b_{31} + \ldots + a_{1n} \cdot b_{n1}$
$c_{12} = a_{11} \cdot b_{12} + a_{12} \cdot b_{22} + a_{13} \cdot b_{32} + \ldots + a_{1n} \cdot b_{n2}$ ...
$c_{21} = a_{21} \cdot b_{11} + a_{22} \cdot b_{21} + a_{23} \cdot b_{31} + \ldots + a_{2n} \cdot b_{n1}$ ...
...
$c_{nn} = a_{n1} \cdot b_{1n} + a_{n2} \cdot b_{2n} + a_{n3} \cdot b_{3n} + \ldots + a_{nn} \cdot b_{nn}$

Let us look at the following simple graph:

G =



We can represent it using the following matrix:

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

let us look at M^2 :

$$M^2 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 2 \end{pmatrix}$$

Now let us look at how we calculated m_{14}^2 :

$$m_{14}^2 = m_{11} \cdot m_{14} + m_{12} \cdot m_{24} + m_{13} \cdot m_{34} + m_{14} \cdot m_{44}$$

$m_{12} \cdot m_{24}$ equals to one ***iff*** there is an edge (in M) between a and b and another edge between b and d or in other words, a path in length 2 from a to d that goes through b, the same observation is true for all the expressions in the sum

- m_{14}^2 tells us how many roots of length 2 exist between a and d
- M^2 holds all the paths of length 2 in the graph G

Q. (Cormen 22.5-6) Given a directed graph $G=(V,E)$, explain how to create another graph $G'=(V,E')$ such that:

a. G' has the same SCC as G

b. G' has the same components graph as G

c. E' is as small as possible.

Describe an efficient algorithm for computing G'

A. 1. Calculate the SCC of G

2. leave the components graph as is

3. create a cycle from each component (remove redundant edges)