Deepanshu 2019CS50427

Page :
Date :

Ans ⇒ Proof by induction.

P(n): In a connected graph having n vertices, calling DFS on any vertex would not leave any vertex as gray (unvisited) and all ancestors call return.

Base Case n=2

Let two vertices are v_1 and v_2. Without loss of generality, assume that DFS is called on v_1. Following events will occur (in order)

1. v_1 will be coloured yellow.
2. DFS will be called on v_2 and v_2 will be coloured yellow.
3. v_2 will return from DFS and finally v_1 will return from DFS.

Hence, P(2) is true

Induction Hypothesis

Assume P(n) is true for n=k

for n=k+1

choose any vertex arbitrarily (call it v_1). Now DFS will be called on v_1. Let v_2 be the neighbour vertex of v_1 where DFS will be called next.

Note that calling DFS on v_2 cannot ever call DFS on v_1 since colour of v_1 is changed to yellow before calling DFS on v_2.

Thus, a total of (n+1)-1=n vertices are in consideration.

Problem effectively reduces to calling DFS on n vertices which would not leave any vertex as gray (unvisited) and all ancestors call return, according to induction hypothesis.

Eventually, v_1 will call return.

Hence P(k+1) is true whenever P(k) is true.

Therefore, by principle of Mathematical Induction P(n) is true for all n.

Hence Proved

Deepanshu 2019CSJ0427

Page:
Date:

Ans ⇒ **Prove by contradiction**

Suppose source is in one component (component 1) and Dfs(v_m) is called after calling Dfs(s) and v_m is not reachable from s and is in different component (component 2).

Let Dfs is called in following order
$s, u_1, u_2, u_3 \text{ —— } u_n\ v_1, v_2 \text{ —— } v_m$ where
$s, u_1, u_2 \text{ —— } u_n$ are all vertices in component 1 and
$v_1, v_2 \text{ —— } v_m$ are in component 2.

Here, Dfs is called on v_1 ~~because of~~ after Dfs call on u_n. ~~This~~ This is possible in 2 cases :-

Case-1: v_1 is neighbour of u_n and was colored gray when Dfs(u_n) was called.
This is not possible as u_n and v_1 are in different component.

Case-2: Any one of $s, u_1, u_2 \text{ —— } u_n$ is ancestor of v_1
This is again not possible as v_1 is in different component from $s, u_1, u_2 \text{ —— } u_n$.

There is no third case possible as Dfs is triggered after entering in for loop which require neighbour relation or ancestor relationship.

Here we arrive at a contradiction that Dfs on v_1 cannot be called from $s, u_1, u_2 \text{ —— } u_n$.
Hence, our assumption is wrong and "no recursive calls will be made on nodes not reachable from s".

Hence Proved

Deepanshu 2019CS50427

Page:
Date:

Ans ⇒ Claim: The running time of DFS is $O(|V|+|E|)$.

- All colouring part takes constant time.
- We need to look for neighbour and make recursive calls which take time.

We can say that running time is proportional to number of vertices scanned and number of edges scanned.

Now number of vertices scanned $\leq |V|$ because –

1. We visit only gray nodes.
2. Every time we visit a gray node, we change its colour to green.
3. We never change colour of any node back to gray.

Also note that number of edges scanned is $O(|E|)$ because –

1. Since every vertex is visited at most once, the number of edges scanned is roughly twice the number of vertices. So, total edges scanned is $O(|E|)$.

All other operations like checking colour, changing colour takes constant time.

Hence, in total running time is $O(V+E)$ for DFS.