Sample

Friday, 3 September 2021

QUESTION 2, TUTORIAL 3

Fix a vertex s in G. We will present an algorithm to compute smallest cycle containing s.

Algorithm (Find minimum cycle containing *s*):

1. Compute $T = BFS(s)$.
2. Let $v_1, v_2, \ldots, v_k$ be children of s in T.
3. For ($i = 1 \ \ to \ \ k$): Scan vertices in $T(v_i)$ and for each $x \in T(v_i)$, set $L[x] = v_i$.
   /*Note: For any $a, b \in V(G)$, $s = LCA(a, b)$ iff $L[a] \neq L[b]$ */
4. $Q = \{(a, b) \in E(G) \setminus E(T) \mid L[a] \neq L[b]\}$.
5. (x_0, y_0) = The edge in G for which $dist(s, x_0) + dist(s, y_0)$ is minimum.
6. Output $C_0 = treepath(s, x_0) :: (x_0, y_0) :: treepath(y_0, s)$.

Claim 1: For any edge $(x, y) \in Q$, $C = treepath(s, x) :: (x, y) :: treepath(y, s)$ is a cycle

Proof: Recall Q comprises of those non-tree edges *(a,b)* for which $s = LCA(a, b)$. Thus, C is a closed walk where no vertex is repeated. Hence C is a cycle.

Claim 2: Any cycle containing v must contain an edge from set Q.

Proof: Let C be a cycle containing v. Without loss of generality assume z_1, z_2 are neighbors of v in C. Let (x, y) be first edge on segment $C[z_1, z_2]$ such that $x \in T(z_1)$ and $y \notin T(z_1)$. Such an edge must exists as $z_2 \notin T(z_1)$. Thus, (x, y) is an edge lying in Q. Hence proved.

Claim 3: Let $(x, y) \in Q$ be an edge that lie on some smallest cycle of s, say C. Then, $|C| = 1 + dist(s, x) + dist(s, y)$.

Proof: $|C[s, x]| \geq dist(s, x)$ and $|C[s, y]| \geq dist(s, y)$. Therefore,

$$|C| \geq 1 + dist(s, x) + dist(s, y).$$

Now, by Claim 1, $treepath(s, x) :: (x, y) :: treepath(y, s)$ is a cycle. Since C is cycle of minimum size. We have,

$$|C| \leq 1 + dist(s, x) + dist(s, y).$$

The claim follows by above two inequalities.

Correctness of Algorithm to find minimum length cycle containing *s*:

By Claim 1, C_0 is a cycle, and by Claim 3 and definition of (x_0, y_0), we get that $|C_0| \leq |C|$. This proves that above algorithm correctly computes a smallest cycle containing v.

Finding minimum length cycle in G:

The time complexity to find a minimum length cycle containing a given vertex is $O(m)$.

We can apply the same algorithm to each vertex of the graph to compute minimum length cycle.

So, we get that a cycle of minimum length in an undirected graph can be computed in $O(mn)$ time.