Lecture notes for Jan 25, 2023
BFS and applications and DFS

Chun-Hung Liu

January 25, 2023

1 Rooted trees

A *rooted tree* is a tree T with a special vertex $r \in V(T)$. The vertex r is called the root of T.

Let T be a rooted tree with root r.

- For a vertex $v \in V(T)$, we say that a vertex $u \in V(T)$ is an *ancestor (with respect to* (T, r)*)* of v if u is in the unique path in T from r to v. Note that every vertex is an ancestor of itself.
- A *proper ancestor (with respect to* (T, r)*)* of a vertex v is an ancestor (with respect to (T, r)) of v distinct from v.
- A vertex u is a *descendant (with respect to* (T, r)*)* of v if v is an ancestor (with respect to (T, r)) of u.
- A *proper descendant (with respect to* (T, r)*)* of a vertex v is a descendant (with respect to (T, r)) of v distinct from v.
- If v is not the root, then the *parent* of v is the unique proper ancestor adjacent to v.
- For a vertex v, every vertex that is a proper descendant adjacent to v is called a *child* of v.
- Two vertices u, v are *incomparable (with respect to* (T, r)*)* if none of u, v is an ancestor of the other.

Let G be a graph. Let T be a spanning tree of G. Let r be a vertex of G. When treating T as a rooted tree with root r, we say that an edge $e \in E(G) - E(T)$ is

- a *back edge* if one end of e is a proper ancestor of the other end of e, and
- a *crossing edge* if the ends of e are incomparable with respect to (T, r).

2 Distance and breadth-first-search

2.1 Distance in graphs

The *length* of a path is the number of edges of this path. For a graph G and vertices u, v of G, we say that a path P in G is a *shortest path between u and v* if it is a path between u and v, and no path between u and v is shorter than P.

Proposition 1 *Let G be a graph. Let u, v be vertices of G. If P is a shortest path between u and v, then for every vertex w in P, the subpath of P between u and w is a shortest path between u and w.*

Proof. If there exists a path Q between u and w shorter than the subpath of P between u and w, then the union of Q and the subpath of P between w and v is a walk shorter than P, a contradiction. ■

Let G be a graph. The *distance* between two vertices u and v of G, denoted by $d_G(u, v)$, is the length of a shortest path in G between u and v. (If no such a path exists, the distance is defined to be ∞.)

Proposition 2 *If H is a subgraph of G, then for any vertices $u, v \in V(H) \subseteq V(G)$, $d_H(u, v) \geq d_G(u, v)$.*

Proof. Every path between u, v in H is a path in G. So a shortest path between u, v in H is a path in G whose length is at least the length of a shortest path between u, v in G. ■

Note that it is possible $d_H(u, v) = d_G(u, v)$, and it is possible that $d_H(u, v) > d_G(u, v)$.

2.2 Breadth-first-search

Breadth-first-search is an algorithm that helps us compute the distance between two vertices.

==

Breadth-First-Search (BFS)
Input: A connected graph G and a vertex r of G.
Output: A tree rooted at r.
Procedure:

Step 1: Label r as v_1. Set $i = 1$. Set T to be the graph $(\{r\}, \emptyset)$. (That is, T is the graph consisting of the single vertex r and with no edge.)

Step 2: Repeatedly picking one edge e of G incident with v_i and do Step 2-1, until all edges incident with v_i have been seen.

Step 2-1: If e has one end in $V(T)$ and one end not in $V(T)$, then adding e into T and label the end of e not in $V(G)$ as v_j, where j is least positive integer such that no vertex in $V(T)$ has been labelled as v_j.

Step 3: Set i to be $i + 1$. Do Step 2, unless $i > |V(G)|$.

==

Remark:

- It is easy to prove that T is always a tree during the algorithm by induction on i. In particular, T is a tree when the algorithm terminates. The final tree T is called a *breadth-first-search (BFS) tree rooted at* r.

- Note that T is a subgraph of G. Since G is connected, $V(T) = V(G)$. Hence T is a spanning tree of G.

- BFS runs in time $O(|E(G)|)$ since we visit every edge at most twice. When G is disconnected, we can run BFS for each component of G to obtain a spanning forest F in time $O(|V(G)| + |E(G)|)$, where each component of F is a BFS tree of a component of G.

Now we prove nice properties of a BFS tree. For simplicity, until the end of this subsection, G, r, T denotes the graph G, vertex r and tree T mentioned in the BFS algorithm.

Note that during the algorithm, for each vertex $v \neq r$, when it is added into T, it must be added at Step 2-1 and the edge v_iv is added into T for some i, so v_i is the parent of v in T.

Lemma 3 *If $v \neq r$ and v is a child of u, then $d_T(r,v) = d_T(r,u) + 1$.*

Proof. Note that there exists a unique path P in T from r to v. According to the algorithm, the edge $\{u,v\}$ is in T and hence in P. So P passes through r,u,v in the order listed. So $d_T(r,v) = d_T(r,u) + 1$. ■

Lemma 4 *Let x, y be integers with $2 \leq x < y$. If v_x is a child of v_α and v_y is a child of v_β, then $\alpha \leq \beta$.*

Proof. If $\alpha > \beta$, then when we do step 2-1 for $i = \beta$, v_y must be added into T, but v_x is not added into T until $i = \alpha > \beta$, contracting $x < y$. ■

Lemma 5 *$d_T(r,v_1) \leq d_T(r,v_2) \leq d_T(r,v_3) \leq ... \leq d_T(r,v_n)$.*

Proof. We shall prove $d_T(r,v_1) \leq d_T(r,v_2) \leq d_T(r,v_3) \leq ... \leq d_T(r,v_k)$ by induction on k. When $k=1$, there is nothing to prove. When $k=2$, v_2 is a neighbor of $v_1 = r$, so $d_T(r,v_1) = d_T(r,r) = 0 < 1 = d_T(r,v_2)$. So we may assume that $k \geq 3$ and $d_T(r,v_1) \leq d_T(r,v_2) \leq d_T(r,v_3) \leq ... \leq d_T(r,v_{k-1})$.

Since $k \geq 3$, $k-1 \geq 2$. So v_{k-1} is a child of v_α and v_k is a child of v_β for some $\alpha \leq \beta$ by Lemma 4. By the induction hypothesis, $d_T(r,v_\alpha) \leq d_T(r,v_\beta)$. By Lemma 3, $d_T(r,v_{k-1}) = d_T(r,v_\alpha) + 1 \leq d_T(r,v_\beta) + 1 = d_T(r,v_k)$. This proves the lemma. ■

Lemma 6 *For every nonnegative integer j, let $L_j = \{v \in V(G) = V(T) : d_T(v,r) = j\}$. Then no edge of G has one end in L_α and one end in L_β for some integers α, β with $\beta \geq \alpha + 2$. In other words, if xy is an edge of G, then $|d_T(r,x) - d_T(r,y)| \leq 1$. In particular, there exists no back edge.*

Proof. Suppose there exists an edge e of G between a vertex u in L_α and a vertex v in L_β for some $\beta \geq \alpha + 2$. Since $v \in L_\beta$, v is a child of a vertex w in $L_{\beta-1}$ by Lemma 3. Since $\alpha < \beta - 1$, we know $d_T(r,u) = \alpha < \beta - 1 = d_T(r,w)$. By Lemma 5, u joins T earlier than w. Since $uv \in E(G)$, v should be added into T before all neighbors of u are seen in the algorithm, a contradiction. ■

Theorem 7 *For every vertex $v \in V(G)$, $d_G(r, v) = d_T(r, v)$.*

Proof. We shall prove this theorem by induction on $d_G(r, v)$.

When $d_G(r, v) = 0$, $r = v$, so $d_T(r, v) = 0 = d_G(r, v)$. So we may assume that $d_G(r, v) \geq 1$ and $d_G(r, v') = d_T(r, v')$ for every vertex v' with $d_G(r, v') < d_G(r, v)$.

Note that since every path in T is a path in G, $d_G(r, v) \leq d_T(r, v)$. So it suffices to prove that $d_G(r, v) \geq d_T(r, v)$.

Let P be a shortest path in G from r to v. Note that P contains at least 2 vertices since $r \neq v$. Let u be the neighbor of v in P. By Proposition 1, $d_G(r, v) = d_G(r, u) + 1$. So $d_G(r, u) < d_G(r, v)$. By the induction hypothesis, $d_G(r, v) = d_G(r, u) + 1 = d_T(r, u) + 1$. By Lemma 6, $|d_T(r, u) - d_T(r, v)| \leq 1$, so $d_T(r, v) \leq d_T(r, u) + 1 = d_G(r, v)$. This proves the theorem. ■

2.3 Applications of BFS

Corollary 8 *Let G be a connected graph. Let r be a vertex. Then in linear time, we can compute $d_G(r, v)$ and a shortest path in G from r to v for all vertices v of G at once.*

Proof. Find a BFS tree rooted at r. We can compute $d_T(r, v)$ for every $v \in V(G)$ during BFS. And $d_G(r, v) = d_T(r, v)$ by Theorem 7. Moreover, the unique path in T from r to v is a shortest path in G from r to v. ■

The *diameter* of a graph G is the largest distance between two vertices in G. That is, the diameter of G equals $\sup_{u,v \in V(G)} d_G(u, v)$. Note that if G is connected, then this supremum is actually a maximum; if G is disconnected, then the diameter is infinite.

Corollary 9 *The diameter of an input graph G can be computed in time $O(|V(G)|^2 + |V(G)||E(G)|)$.*

Proof. For each vertex r of G, we can compute $\sup_{v \in V(G)} d_G(r, v)$ in linear time by Corollary 8. So $\sup_{u,v \in V(G)} d_G(u, v) = \sup_{u \in V(G)} \sup_{v \in V(G)} d_G(u, v)$ can be computed in $O(|V(G)| \cdot (|V(G)| + |E(G)|)) = O(|V(G)|^2 + |V(G)||E(G)|)$. ■

A graph G is *bipartite* if $V(G)$ can be partitioned into two (possibly empty) sets A, B such that every edge of G has one end in A and one end in B. Such a partition $\{A, B\}$ is called a *bipartition* of G. Note that a graph is bipartite if and only if it is 2-colorable.

Corollary 10 *Given the input graph G, we can either find a bipartition of G or an odd cycle in G in linear time.*

Proof. Find a BFS tree T rooted at an arbitrary vertex r. Let $A = \{v \in V(G) : d_T(r,v) \text{ is even}\}$ and let $B = \{v \in V(G) : d_T(r,v) \text{ is odd}\}$.

We can check whether there is an edge of G with both ends in A or with both ends in B in linear time. If there exists no such an edge, then $\{A, B\}$ is a bipartition of G and we output it.

So we may assume that there exists an edge e with both ends in C for some $C \in \{A, B\}$, say $e = uv$. For $x \in \{u, v\}$, let P_x be the unique path in T from r to x. Note that $P_u \cap P_v$ is a path R in T from r to a common ancestor w of u and v. And $P_u \cup P_v$ is the union of R and the unique path P in T from u to v, where P and R are edge-disjoint, so $|E(P)| = |E(P_u)| + |E(P_v)| - 2|E(R)|$. Since both u, v are in C, $|E(P_u)| + |E(P_v)|$ is even, so $|E(P)| = |E(P_u)| + |E(P_v)| - 2|E(R)|$ is even. Therefore, the cycle $P + uv$ is an odd cycle O, and we output O. Note that P_u, P_v, w, R, P can be found in linear time. So O can be found in linear time. ■

Corollary 11 *A graph G is bipartite if and only if G does not contain an odd cycle (as a subgraph).*

Proof. ($\Rightarrow$) Since every odd cycle is not bipartite, no bipartite graph can contain an odd cycle.

($\Leftarrow$) If G has no odd cycle, then the algorithm in Corollary 10 must find a bipartition of G, so G is bipartite. ■

Corollary 12 *Given a graph G, in linear time, we can find a bipartition of G to correctly conclude that G is bipartite if G is bipartite, and output an odd cycle in G to correctly conclude that G is not bipartite if G is non-bipartite. In particular, 2-COLORABILITY is in* **P**.

3 Depth-first-search

==

Depth-First-Search (DFS)
Input: A connected graph G and a vertex r of G.
Output: A tree rooted at r.
Procedure:

Step 1: Set T to be the rooted tree $(\{r\}, \emptyset)$ rooted at r. Set all edges of G as "unmarked". Set S to be the sequence (r).

Step 2: Terminate the algorithm if S has no entry. Say the last entry of S is v. If there exists no unmarked edge of G incident with v, then remove v from S and repeat Step 2. Otherwise, pick an unmarked edge e of G incident with v and mark e. If e is not a loop and the end u of e other than v is not in $V(T)$, then add e into T, add u into S as the last entry, and repeat Step 2; otherwise, just repeat Step 2.

==

Remark:

- Clearly, T is a tree during the entire process, and the final tree T is a spanning tree of G rooted at r. We call the final tree T a *depth-first-search (DFS) tree rooted at* r.

- During the process, if a vertex v is removed from S, then all edges incident with v have been marked, so v cannot be added into S in the future.

- So Step 2 is executed at most $|V(G)| + |E(G)| + O(1)$ times. We can implement the algorithm so that finding an unmarked edge incident with v can be done in $O(1)$ time. Hence the algorithm runs in time $O(|V(G)| + |E(G)|)$.

- Clearly, during the entire process, the entries of S always form a path in T from r to the last entry of S passing all entries in the order listed.

Like having no back edge is a key feature of every BFS tree, a key feature of every DFS tree is that it has no crossing edge.

Lemma 13 *If T is a DFS tree of a connected graph G, then there exists no crossing edge.*

Proof. Suppose to the contrary that e is a crossing edge. Let x, y be the ends of e, where x is added into T earlier than y. When x leaves S, e is already marked. Let z be the last entry of S when we mark e. So $z \in \{x, y\}$ and S contains both x and z at this moment. Hence x is an ancestor of z in T. If $z = x$, then e must be added into the tree T, a contradiction. If $z = y$, then since x is an ancestor of $z = y$, e is a back edge, a contradiction. ■