Let \( v.d \) and \( v.f \) denote the discovery and finish times, respectively, of vertex \( v \) in the depth-first search algorithm.

1. Suppose \( v \) is a descendant of \( u \) in a \( G_\pi \)-tree created by the depth-first-search algorithm. How are the discovery and finish times of \( u \) and \( v \) related?

2. Recall that the depth-first search algorithm colors vertices as it progresses. Consider the point when vertex \( v \) is found on the adjacency list of vertex \( u \). At this point, what color is \( v \) if \((u,v)\) becomes back edge in the \( G_\pi \) tree containing \( u \) and \( v \)?

3. In a directed graph, suppose edge \((u,v)\) points from strongly-connected component \( A \) to strongly-connected component \( B \). Which of \( A \) and \( B \) finishes first in a depth-first search? Why?

4. The Bellman-Ford and Dijkstra algorithms both solve the single-source-shortest-paths graph problem. What do the algorithms have in common? Also, how do they differ? For the latter question, answer by giving two distinctions between them — other than having different names. (Use back of page, if necessary.)
Solutions.

1. Suppose $v$ is a descendant of $u$ in a $G_\pi$-tree created by the depth-first-search algorithm. How are the discovery and finish times of $u$ and $v$ related?

Parenthesis Theorem: $u.d < v.d < v.f < u.f$.

2. Recall that the depth-first search algorithm colors vertices as it progresses. Consider the point when vertex $v$ is found on the adjacency list of vertex $u$. At this point, what color is $v$ if $(u, v)$ becomes back edge in the $G_\pi$ tree containing $u$ and $v$?

Back edge $(u, v)$ establishes $v$ as an ancestor of $u$ the $G_\pi$ tree $u$. Therefore, the adjacency list of $v$ is still being scanned in a suspended stack frame underneath the active frame that is scanning the adjacency list of $u$. Consequently, $v$ is gray.

3. In a directed graph, suppose edge $(u, v)$ points from strongly-connected component $A$ to strongly-connected component $B$. Which of $A$ and $B$ finishes first in a depth-first search? Why?

$B$ finishes first.

Suppose $A$ is discovered first via vertex $a \in A$. Then, using link $(u, v)$, there is a white path to every vertex in $B$. Thus every $b \in B$ is a descendant of $a$ in a $G_\pi$ tree. By the parenthesis theorem, we have $a.d < b.d < b.f < a.f$ for every $b \in B$, including the $b$ with the highest finish time. Hence, $B.f < A.f$. That is, $B$ finishes first.

If $B$ is discovered first, say via vertex $b_0 \in B$, then there is a white path from $b_0$ to every vertex in $b$. But, there can be no white path to any element of $A$ because that path, together with the link $(u, v)$ would constitute a cycle among the strongly connected components — a contradiction. So $B$ finishes before $A$ is discovered, and again $B$ finishes first.

4. The Bellman-Ford and Dijkstra algorithms both solve the single-source-shortest-paths graph problem. What do the algorithms have in common? Also, how do they differ? For the latter question, answer by giving two distinctions between them — other than having different names.

Both algorithms invoke the Relax operation on edges. Relax($u, v$) tests condition $v.d > u.d + w(u, v)$. If the condition is true, then $v.d$ is updated to $u.d + w(u, v)$ and $v.\pi$ is set to $u$.

They differ in how they apply the relax operation. Bellman-Ford relaxes all edges $V - 1$ times, which is sufficient to push all $v.d$ values to their optimal values ($\delta(s, v)$). Dijkstra maintains a minHeap of vertices with key $v.d$. When vertex $u$ is removed from the minHeap, edge $(u, v)$ is relaxed for every $v \in u.\text{adj}$.

The algorithms also differ in asymptotic complexity. Bellman-Ford is $O(VE)$, while Dijkstra is $O(E \log V)$.

The algorithms also differ in that Bellman-Ford produces correct results for graphs with negative edges (but no negative cycles), while Dijkstra requires nonnegative weights on all edges.