CS412 Algorithms: Design & Analysis

Spring 2024

Dhanani School of Science and Engineering

Habib University

Practice Problems

Week 7

1. Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

   Solution: Since it seems as though the list for the neighbors of each vertex v is just an undecorated list, to find the length of each would take time $O(\text{out-degree}(v))$. So, the total cost will be P $v \in V$ $O(\text{outdegree}(v)) = O(|E| + |V|)$. Note that the $|V|$ showing up in the asymptotics is necessary, because it still takes a constant amount of time to know that a list is empty. This time could be reduced to $O(|V|)$ if for each list in the adjacency list representation, we just also stored its length.

   To compute the in degree of each vertex, we will have to scan through all of the adjacency lists and keep counters for how many times each vertex has appeared. As in the previous case, the time to scan through all of the adjacency lists takes time $O(|E| + |V|)$.

2. The transpose of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(v, u) \in V \times V : (u, v) \in E\}$. Thus, G^T is G with all its edges reversed. Describe efficient algorithms for computing G^T from G, for both the adjacencylist and adjacency-matrix representations of G. Analyze the running times of your algorithms.

   Solution: For the adjacency matrix representation, to compute the graph transpose, we just take the matrix transpose. This means looking along every entry above the diagonal, and swapping it with the entry that occurs below the diagonal. This takes time $O(|V|^2)$.

   For the adjacency list representation, we will maintain an initially empty adjacency list representation of the transpose. Then, we scan through every list in the original graph. If we are in the list corresponding to vertex v and see u as an entry in the list, then we add an entry of v to the list in the transpose graph corresponding to vertex u. Since this only requires a scan through all of the lists, it only takes time $O(|E| + |V|)$.

3. Most graph algorithms that take an adjacency-matrix representation as input require time $\Omega(V^2)$, but there are some exceptions. Show how to determine whether a directed graph G contains a universal sink-a vertex with in-degree $|V| - 1$ and out-degree 0 - in time $O(V)$, given an adjacency matrix for G.

   Solution: Start by examining position $(1, 1)$ in the adjacency matrix. When examining position (i, j), if a 1 is encountered, examine position $(i + 1, j)$. If a 0 is encountered, examine position $(i, j + 1)$. Once either i or j is equal to $|V|$, terminate. I claim that if the graph contains a universal sink, then it must be at vertex i. To see this, suppose

that vertex k is a universal sink. Since k is a universal sink, row k in the adjacency matrix is all 0's, and column k is all 1's except for position (k, k) which is a 0. Thus, once row k is hit, the algorithm will continue to increment j until $j = |V|$. To be sure that row k is eventually hit, note that once column k is reached, the algorithm will continue to increment i until it reaches k. This algorithm runs in $O(V)$ and checking whether or not i in fact corresponds to a sink is done in $O(V)$. Therefore the entire process takes $O(V)$.

4. Give an example of a directed graph $G = (V, E)$, a source vertex $s \in V$, and a set of tree edges $E_\pi \subseteq E$ such that for each vertex $v \in V$, the unique simple path in the graph $(V, E_\pi$ from s to v is a shortest path in G, yet the set of edges E_π cannot be produced by running BFS on G, no matter how the vertices are ordered in each adjacency list.

Solution: Let G be the graph shown in the first picture, $G' = (V, E_\pi)$ be the graph shown in the second picture, and 1 be the source vertex. Let's see why E_π can never be produced by running BFS on G. Suppose that 2 precedes 5 in the adjacency list of 1. We'll dequeue 2 before 5, so $3.\pi$ and $4.\pi$ must both equal 2. However, this is not the case. Thus, 5 must have preceded 2 in the adjacency list. However, this implies that $3.\pi$ and $4.\pi$ both equal 5, which again isn't true. Nonetheless, it is easily seen that the unique simple path in G' from 1 to any vertex is a shortest path in G.