Distributed Algorithms **M.Tech., CSE, 2016**

Lecture 4: Breadth first search, shortest paths, and spanning tree.

Faculty: K.R. Chowdhary *: Professor of CS*

4.1 Introduction

The next problem we consider is performing breadth-first search (BFS) in a network based on arbitrary strong connected directed graph having identified source node. More precise, we are interested to construct a breadth-first spanning tree for the digraph. The purpose for this is to have broadcast communication. This tree minimizes the communication time for each communication channel.

We define a directed spanning tree of a directed graph $G = (V, E)$ to be a rooted tree that consists entirely of directed edges in E, all edges directed from parents to children, and that consist every vertex in G. Every strongly connected digraph has a breadth-first directed spanning tree.

Let the source is i_0, the BFS algorithm is supposed to output the structure of the BFS directed spanning-tree rooted at i_0. The processes communicate through the directed edges and do not have idea of the diameter of the graph.

4.2 BFS algorithm

> At any point during the execution there is set of processes that are marked, at begin only i_0 is marked. The process i_0 sends out search messages to 1, 2, ..., all its neighbors. At any round, if an unmarked process receives a search message, it marks itself and chooses one of the processes from which the search has arrived as its parents. At the first round after a process gets marked, it sends a search message to all of its outgoing neighbors.

4.3 Complexity

The time complexity of above is at mist $O(b^d) = |V|$, where d is depth (or diameter) of the graph, and b is branching factor. For message communication, the message m can be piggy-backed in the search, and can be sent to all the nodes, using BFS.

Since we are considering directed graph, the reverse communication (some of) may be sent through indirect routs.

In an undirected graph, total time to communicate as a BFS tree is $O(diam)$ and communication complexity is $O(E)$. Since all the communications go in parallel, the total number of messages are $O(diam|E|)$, where $diam$ is diameter the graph. Due to E concurrent communications, there will be $|E|b$ bits in a message, where b is number of bits needed to represent a UID. This yields total $O(diam|E|b \times |E|) = O(diam|E|^2 b)$ bits of communication. But, since only E messages are concurrent, this figure is only $O(|E|^2 b)$.

4.4 Applications

BFS is one of the most basic building blocks of distributed computing. following are some applications.

broadcast: message broadcast can be implemented along with the establishment of BFS tree. The message need to propagated only though edges from parents to children. This allows us to reuse the tree, as many messages can be sent along the same tree. Once the tree is constructed the additional time to send the message is only $O(diam)$ number of message is only $O(n)$, where $n = |V|$.

Global Computation: Another application of BFS tree is collection of information from through out the network or, more generally, the computation of a function is based on distributed inputs. The same can also be used for bidirectional search.

Electing a leader: BFS can be used o elect a leader in a network with UIDs, even when the processes have no knowledge of n or $diam$. For this all the processes can initiate a BFS in parallel. Each process i uses the tree constructed so far, and the globla computation procedure is used to determine the maximum UID, once reached, declares itself the leader.

Computing the diameter: The diameter of the network can be computed by having all the processes initiate BFS in parallel. Each process i uses the tree thereby constructed to determine $max\text{-}dist_i$, defined to be the maximum distance from i to any other process in the network. Each process i then uses its BFS tree for a global computation to discover the maximum of the max-dist values. If the graph is undirected, the time is $O(diam)$ and the number of messages is $O(n.|E|)$.

4.5 Shortest paths

Consider a strongly connected directed graph, with the possibility of unidirectional communication between some pairs of neighbors. Assume that each directed edge $e = (i, j)$ has an associated non-negative real weight, denoted by $weight(e)$ or $weight_{i,j}$. Weight of a path is sum of the weights on its edges.

The problem is to find a Shortest path from the source node i_0 to each other node in the graph. All the shortest path edges are oriented from parent to child.

We assume that every process initially knows the weight of all its incident edges, or, more precisely, that the weight of an edge appears in special $weight$ variables at both of its end point processes. We want that each process should be able to determine its distance fro i_0, and knows its parent node.

The following algorithm finds the shortest path of each process from i_0, for unequal weight of the edges.

Bellman-Ford Algorithm:

> Each process i keeps track of $dist$, the shortest distance from i_0 it knows so far, together with $parent$ - the incoming neighbor that precedes in the path whose weight is $dist$. Initially, $dist_0 = 0$, $dist_i = \infty$ for $i \neq i_0$, and $parent$ components are undefined. At each round, each process sends, its $dist$ to all its outgoing neighbors. Then each process i updates its $dist$ by a relaxation step, in which its takes minimum of its previous $dist$ value and all the values $dist_j + weight_{j,i}$, where j is incoming neighbor. If $dist$ is changed, the $parent$ component is updated accordingly. After $n-1$ rounds, $dist$ contains the shortest distance, and $parent$ the parent in the shortest path tree.

It is easy to realize that, after $n-1$ rounds, the $dist$ values converge to the correct distances. The complexity of above algorithm is $n - 1$ and number of messages are $(n - 1)|E|$

4.6 Minimum Spanning Tree

The minimum spanning tree is minimum weight spanning tree (MST) in an undirected graph network with weighted edges. Again, the main use of for such a tree is as a basis for broadcast communication. A minimum spanning-weight spanning tree minimizes the total cost for any source process to communicate with all other processes in the network.

4.6.1 The problem

A *spanning forest* of an undirected graph $G = (V, E)$ is a forest (i.e., a graph that is but not necessarily connected) that consists entirely of undirected edges in E and that contains every vertex of G that is connected. A *spanning tree* of an undirected graph G is spanning forest of G that is connected. If there are weights associated with edges in E, then the *weight* of any subgraph of G is sum of the weights of its edges. We assume (this time only) that $weight(e) = weight_{i,j} = weight_{j,i}$. Also, we know that every process knows the weights of its incident edges.

The problem is to find minimum weight (undirected) spanning tree for the entire network. The general strategy is to start with spanning forest of all the vertices and then join the edges so that all the vertices (processes) are connected.

General strategy for MST:

> Start with the trivial spanning forest that consists of n individual nodes and no edges. Then repeatedly do the following: Select an arbitrary component C in the forest and an arbitrary outgoing edge e of C having minimum weight among the outgoing edges of C. Combine C with the component at the other end of e, including edge e in the new combined component. Stop when the forest has a single component.

References

[1] Nancy A. Lynch, "Distributed Algorithms," *Elsevier*, 2013.

[2] Allen B. Tucker, Jr., "The computer Science and Engineering Handbook," *CRC Press*, 1997.