Graph Traversals

Dr. Jayantha Lanel

University of Sri Jayawardanapura

February 10, 2020

Outline

Breadth-First Search Algorithm

We now use a FIFO list for L (First In First Out) and choose for u the first element added to L. This is a **breadth first search** (BFS).

Algorithm BreadthFirstSearch(G, s)
mark s; $L := \{s\}$;
while $L \neq \phi$; **do**
u := first(L)
if $\exists(u, v)$ such that v is unmarked **then**
choose (u, v) with v of smallest index;
mark v; $L := L \cup \{v\}$;
else
$L := L \setminus \{u\}$

Depth-First Search Algorithm

Because of the choices, this algorithm allows for different versions. Let us use a LIFO list for L (Last In First Out) and choose for u the last element added to L. This is a **depth first search** (DFS).

Algorithm DepthFirstSearch(G, s)
mark s, $L := \{s\}$;
while $L \neq \phi$; **do**
$u :=$ last(L)
if $\exists(u, v)$ such that v is unmarked **then**
choose (u, v) with v of smallest index;
mark v; $L := L \cup \{v\}$;
else
$L := L \setminus \{u\}$

Kruskal Algorithm

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

Kruskal Algorithm

Remove from a connected graph as many edges as possible while remaining connected; this should yield a tree with $n-1$ edges. This is the minimal spanning tree found by the following algorithm.

Algorithm KruskalMST(G)
$E_{rest} := sort(E); E' := \phi;$
while $\mid E' \mid < n-1$ **do**
$\alpha := first(E_{rest}); E_{rest} := E_{rest} \setminus \{\alpha\};$
if $(V, E' \cup \{\alpha\})$ is acyclic
then
$E' := E' \cup \{\alpha\};$
end if
end while

Prim's Algorithm

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.
The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.

Prim's Algorithm

Now we look at an alternative algorithm with different time complexity. The idea is to pick a random node and then grow a minimal tree from there,

Algorithm PrimMST(G)
Choose $u \in V$; $V' := \{u\}$; $E' := \phi$
for $i = 1 : n - 1$ **do**
$E^{''}$:= edges linking V to V'
choose $e = (u, v) \in E''$ of minimal weight and such that
$(V' \cup \{v\}), E' \cup \{e\})$ is acyclic
end for