Spanning Trees

A *spanning tree* of a simple graph $G$ is a subgraph that is a tree and contains every vertex.

A simple graph is connected if and only if it has a spanning tree.

Depth-first search and breadth-first search can be used to find spanning trees.

Depth-first and breadth-first search can be implemented in $O(n+m)$ time where $n$ is the number of vertices and $m$ is the number of edges.

---

**procedure** depth_first_search

$(G$: connected simple graph$)$

$stack :=$ arbitrary vertex in $G$

$T :=$ empty tree

**while** stack is not empty

$u :=$ top of stack

*if* there is an edge $\{u, v\}$ s.t. $v$ is not in $T$

*then* add $\{u, v\}$ to $T$ and push $v$ on stack

*else* pop stack

**end while**

return $T$
procedure breadth_first_search
    \((G: \text{ connected simple graph})\)
    \(queue := \text{ arbitrary vertex in } G\)
    \(T := \text{ empty tree}\)
    \textbf{while} \(queue\) is not empty
        \(u := \text{ beginning of } queue\)
        \textbf{if} there is an edge \(\{u, v\}\) s.t. \(v\) is not in \(T\)
            \textbf{then} add \(\{u, v\}\) to \(T\) and enqueue \(v\) on \(queue\)
        \textbf{else} dequeue \(queue\)
    \textbf{end while}
    \textbf{return} \(T\)

---

Minimum Spanning Trees

A \textit{minimum spanning tree} of a weighted simple graph \(G\) is a spanning tree with the minimum sum of weights.

Prim’s algorithm and Kruskal’s algorithm can be used to find minimal spanning trees.

Prim’s algorithm and Kruskal’s algorithm can be implemented in \(O(m \log m)\) time where \(m\) is the number of edges.
procedure Prim  
\( \text{(G: weighted connected simple graph)} \)
\[ T := \text{smallest edge in } G \]
\[ n := \text{number of vertices in } G \]
for \( i := 1 \) to \( n - 2 \)
\[ e := \text{smallest edge in } G \text{ with one vertex in } T \text{ and the other vertex not in } T \]
\[ \text{add } e \text{ to } T \]
end for
return \( T \)
end procedure

procedure Kruskal  
\( \text{(G: weighted connected simple graph)} \)
\[ T := \text{smallest edge in } G \]
\[ n := \text{number of vertices in } G \]
for \( i := 1 \) to \( n - 2 \)
\[ e := \text{smallest edge in } G \text{ making no circuits with } T \]
\[ \text{add } e \text{ to } T \]
end for
return \( T \)
end procedure