15.4.2
Graph exploration in directed graphs
Basic Graph Search in Directed Graphs

Given $G = (V, E)$ a directed graph and vertex $u \in V$. Let $n = |V|$.

```
Explore(G, u):
   array Visited[1..n]
   Initialize: Set $Visited[i] \leftarrow$ FALSE for $1 \leq i \leq n$
   List: $ToExplore$, $S$
   Add $u$ to $ToExplore$ and to $S$, $Visited[u] \leftarrow$ TRUE
   Make tree $T$ with root as $u$
   while ($ToExplore$ is non-empty) do
      Remove node $x$ from $ToExplore$
      for each edge $(x, y)$ in $Adj(x)$ do
         if ($Visited[y] = \text{FALSE}$)
            $Visited[y] \leftarrow$ TRUE
            Add $y$ to $ToExplore$
            Add $y$ to $S$
            Add $y$ to $T$ with edge $(x, y)$
   Output $S$
```
Example
Example
Example
Example
Example
Example

![Diagram with nodes B, A, C, E, F, D, G, H connected by arrows]

52 / 73
Example
Example
Example
Example
Properties of Basic Search

Proposition

\( \text{Explore}(G, u) \) terminates with \( S = \text{rch}(u) \).

Proof Sketch.

- Once \( \text{Visited}[i] \) is set to \( \text{TRUE} \) it never changes. Hence a node is added only once to \( \text{ToExplore} \). Thus algorithm terminates in at most \( n \) iterations of while loop.
- By induction on iterations, can show \( v \in S \Rightarrow v \in \text{rch}(u) \)
- Since each node \( v \in S \) was in \( \text{ToExplore} \) and was explored, no edges in \( G \) leave \( S \). Hence no node in \( V - S \) is in \( \text{rch}(u) \). Caveat: In directed graphs edges can enter \( S \).
- Thus \( S = \text{rch}(u) \) at termination.
Properties of Basic Search

**Proposition**

\[
\text{Explore}(G, u) \text{ terminates in } O(m + n) \text{ time.}
\]

**Proposition**

\[T \text{ is a search tree rooted at } u \text{ containing } S \text{ with edges directed away from root to leaves.}\]

Proof: easy exercises

**BFS** and **DFS** are special case of Basic Search.

1. **Breadth First Search (BFS):** use queue data structure to implementing the list \( \text{ToExplore} \)
2. **Depth First Search (DFS):** use stack data structure to implement the list \( \text{ToExplore} \)
Exercise

Prove the following:

Proposition

Let $S = \text{rch}(u)$. There is no edge $(x, y) \in E$ where $x \in S$ and $y \not\in S$.

Describe an example where $\text{rch}(u) \neq V$ and there are edges from $V \setminus \text{rch}(u)$ to $\text{rch}(u)$. 
Directed Graph Connectivity Problems

1. Given $G$ and nodes $u$ and $v$, can $u$ reach $v$?
2. Given $G$ and $u$, compute $rch(u)$.
3. Given $G$ and $u$, compute all $v$ that can reach $u$, that is all $v$ such that $u \in rch(v)$.
4. Find the strongly connected component containing node $u$, that is $SCC(u)$.
5. Is $G$ strongly connected (a single strong component)?
6. Compute all strongly connected components of $G$.

First five problems can be solved in $O(n + m)$ time by via Basic Search (or BFS/DFS). The last one can also be done in linear time but requires a rather clever DFS based algorithm.
Directed Graph Connectivity Problems

1. Given $G$ and nodes $u$ and $v$, can $u$ reach $v$?
2. Given $G$ and $u$, compute $rch(u)$.
3. Given $G$ and $u$, compute all $v$ that can reach $u$, that is all $v$ such that $u \in rch(v)$.
4. Find the strongly connected component containing node $u$, that is $SCC(u)$.
5. Is $G$ strongly connected (a single strong component)?
6. Compute all strongly connected components of $G$.

First five problems can be solved in $O(n + m)$ time by via Basic Search (or BFS/DFS). The last one can also be done in linear time but requires a rather clever DFS based algorithm.
THE END

... (for now)