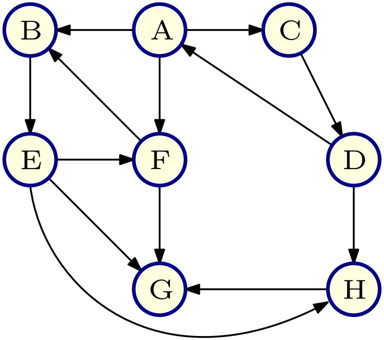
15.4.2
Graph exploration in directed graphs

Basic Graph Search in Directed Graphs

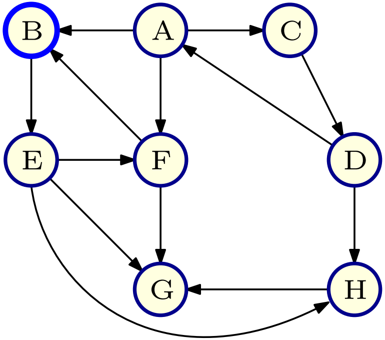
Given $G = (V, E)$ a directed graph and vertex $u \in V$. Let $n = |V|$.

```
Explore(G, u):
    array Visited[1..n]
    Initialize:  Set Visited[i] ← FALSE for 1 ≤ i ≤ n
    List:  ToExplore, S
    Add u to ToExplore and to S, Visited[u] ← 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] = FALSE)
                Visited[y] ← TRUE
                Add y to ToExplore
                Add y to S
                Add y to T with edge (x, y)
    Output S
```

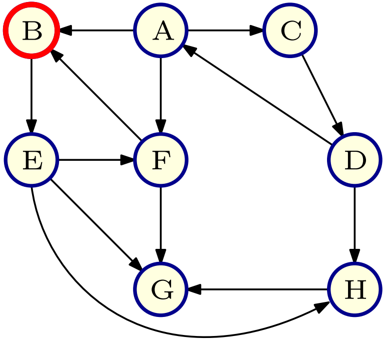
Example



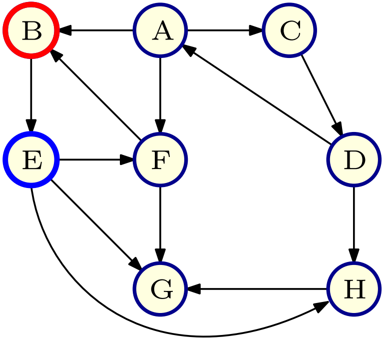
Example



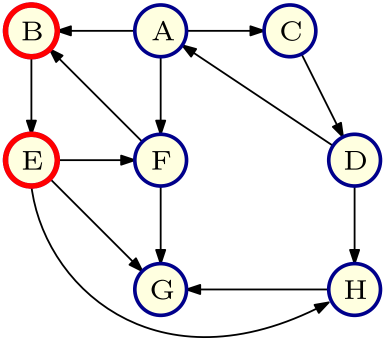
Example



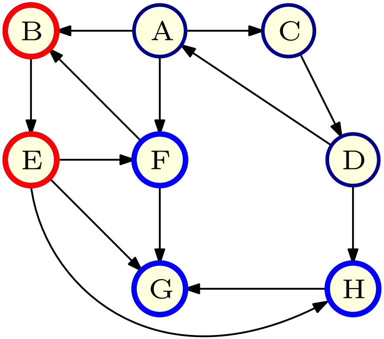
Example



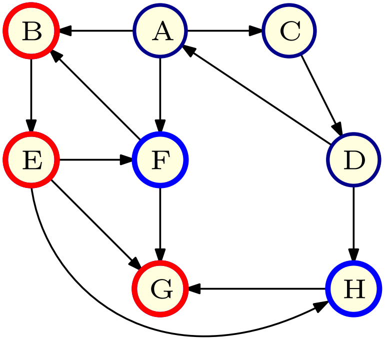
Example



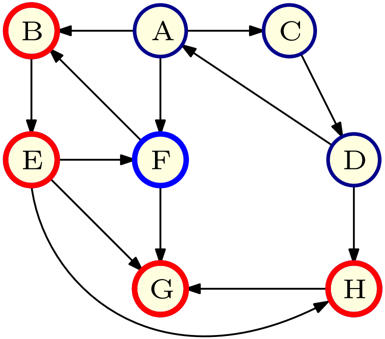
Example



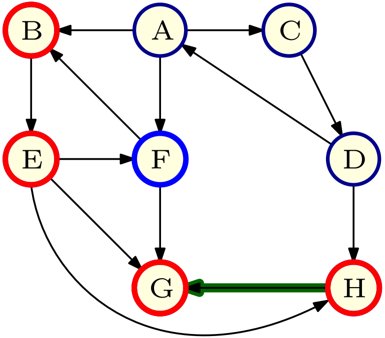
Example



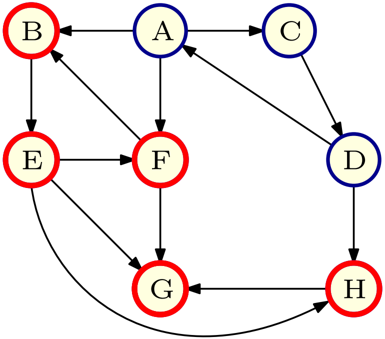
Example



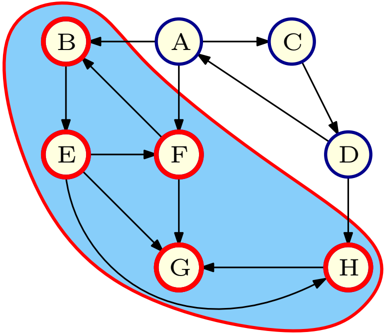
Example



Example



Example



Properties of Basic Search

Proposition

Explore(G, u) *terminates with* $S = \mathbf{rch}(u)$.

Proof Sketch.

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

$\square$

Properties of Basic Search

Proposition

Explore(G, u) *terminates in* $O(m+n)$ *time.*

Proposition

T *is a search tree rooted at* u *containing* S *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 ***ToExplore***
2. Depth First Search (**DFS**): use stack data structure to implement the list ***ToExplore***

Exercise

Prove the following:

Proposition

Let $S = \mathbf{rch}(u)$. There is no edge $(x, y) \in E$ where $x \in S$ and $y \notin S$.

Describe an example where $\mathbf{rch}(u) \neq V$ and there are edges from $V \setminus \mathbf{rch}(u)$ to $\mathbf{rch}(u)$.

Directed Graph Connectivity Problems

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