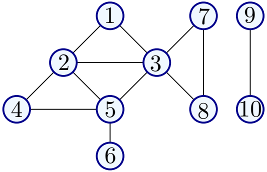
15.3
Computing connected components in undirected graphs using basic graph search

Basic Graph Search in Undirected Graphs

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

```
Explore(G, u):
    Visited[1 .. n] ← FALSE
    // ToExplore, S: Lists
    Add u to ToExplore and to S
    Visited[u] ← TRUE
    while (ToExplore is non-empty) do
        Remove node x from ToExplore
        for each edge xy in Adj(x) do
            if (Visited[y] = FALSE)
                Visited[y] ← TRUE
                Add y to ToExplore
                Add y to S
    Output S
```

Example



Properties of Basic Search

Proposition

Explore(G, u) *terminates with* $S = con(u)$.

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 $v \in S \Rightarrow v \in \text{con}(u)$
- 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 $\text{con}(u)$.
- Thus $S = \text{con}(u)$ at termination.

□

Properties of Basic Search

Proposition

Explore(G, u) *terminates with* $S = con(u)$.

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 $v \in S \Rightarrow v \in \text{con}(u)$
- 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 $\text{con}(u)$.
- Thus $S = \text{con}(u)$ at termination.

□

Properties of Basic Search

Proposition

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

Proof: easy exercise

BFS and **DFS** are special case of BasicSearch.

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*

Properties of Basic Search

Proposition

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

Proof: easy exercise

BFS and **DFS** are special case of BasicSearch.

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***

Search Tree

One can create a natural search tree T rooted at u during search.

```
Explore(G,u):
    array Visited[1..n]
    Initialize:  Visited[i] ← FALSE for i = 1,...,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 x as its parent
    Output S
```

T is a spanning tree of $\text{con}(u)$ rooted at u

Finding all connected components

Exercise: Modify Basic Search to find all connected components of a given graph G in $O(m+n)$ time.

THE END

...

(for now)