Pseudocode: Depth-First Search

Reachability in graphs

- What are the vertices reachable from node `i`?

Reachability in graphs

- What are the vertices reachable from node `i`?
- Start from `i`, visit a neighbour `j`

Reachability in graphs

- What are the vertices reachable from node `i`?
- Start from `i`, visit a neighbour `j`
- Suspend the exploration of `i` and explore `j` instead

Reachability in graphs

- What are the vertices reachable from node `i`?
- Start from `i`, visit a neighbour `j`
- Suspend the exploration of `i` and explore `j` instead
- Continue till you reach a vertex with no unexplored neighbours

Reachability in graphs

- What are the vertices reachable from node `i`?
- Start from `i`, visit a neighbour `j`
- Suspend the exploration of `i` and explore `j` instead
- Continue till you reach a vertex with no unexplored neighbours
- Backtrack to nearest suspended vertex that still has an unexplored neighbour

Reachability in graphs

- What are the vertices reachable from node `i`?
- Start from `i`, visit a neighbour `j`
- Suspend the exploration of `i` and explore `j` instead
- Continue till you reach a vertex with no unexplored neighbours
- Backtrack to nearest suspended vertex that still has an unexplored neighbour
- Best defined recursively
  - Maintain information about visited nodes in a dictionary `visited`
  - Recursively update `visited` each time we explore an unvisited neighbour

Depth first search

- Maintain information about visited nodes in a dictionary `visited`
- Recursively update `visited` each time we explore an unvisited neighbour
- To explore vertices reachable from `i`
  - Initialize `visited = {}`
  - `visited = DFS(graph,visited,i)`
  - `keys(visited)` is set of nodes that can be reached from `i`

Depth first search

- Maintain information about visited nodes in a dictionary `visited`
- Recursively update `visited` each time we explore an unvisited neighbour
- To explore vertices reachable from `i`
  - Initialize `visited = {}`
  - `visited = DFS(graph,visited,i)`
  - `keys(visited)` is set of nodes that can be reached from `i`

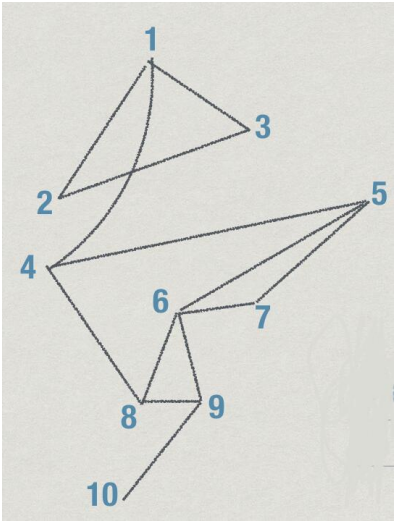
```
Procedure DFS(graph,visited,i)
  visited[i] = True
  foreach j in columns(graph){
   if (graph[i][j] == 1 and
       not(isKey(visited,j))) {
     visited =
       DFS(graph,visited,j)
   }
  }
  return(visited)
End DFS
```

Depth first search

- Maintain information about visited nodes in a dictionary `visited`
- Recursively update `visited` each time we explore an unvisited neighbour
- To explore vertices reachable from `i`
  - Initialize `visited = {}`
  - `visited = DFS(graph,visited,i)`
  - `keys(visited)` is set of nodes that can be reached from `i`
    - If `keys(visited)` includes all nodes, the graph is connected

```
Procedure DFS(graph,visited,i)
  visited[i] = True
  foreach j in columns(graph){
   if (graph[i][j] == 1 and
       not(isKey(visited,j))) {
     visited =
       DFS(graph,visited,j)
   }
  }
  return(visited)
End DFS
```

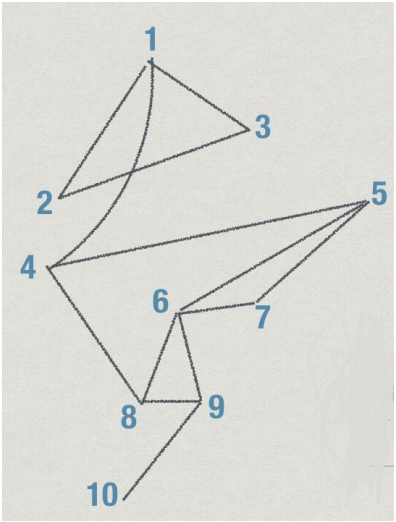
Example



- `visited= {`

  `}`
- `DFS(graph,visited,4)`

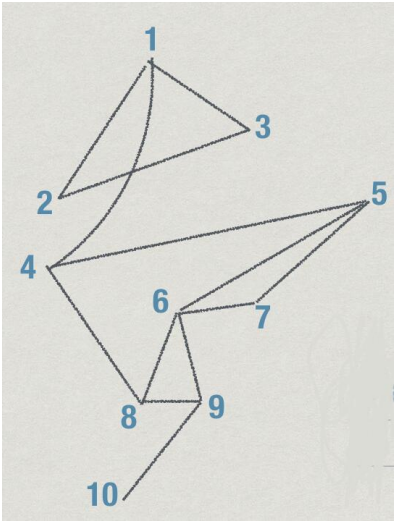
Example



- `visited= { 4:True`

  `}`
- `DFS(graph,visited,4)`
- `DFS(graph,visited,1)`

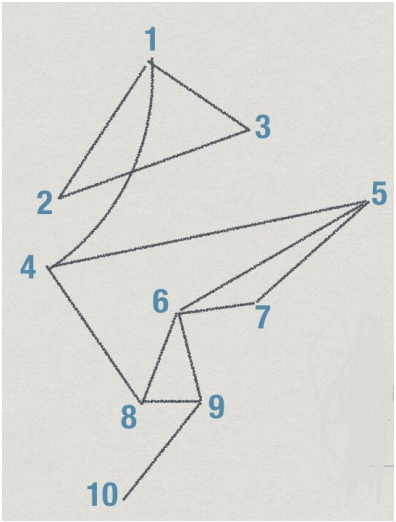
Example



- `visited= {  4:True, 1:True`

  `}`
- `DFS(graph,visited,4)`
- `DFS(graph,visited,1)`
- `DFS(graph,visited,2)`

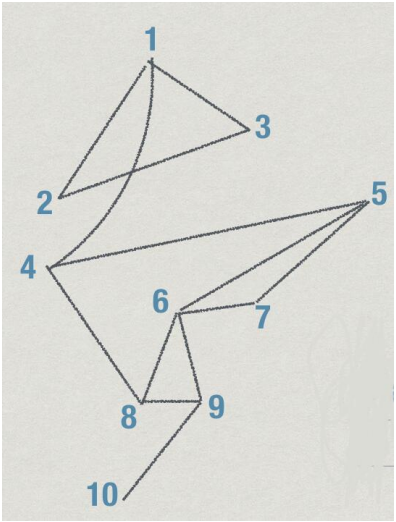
Example



- `visited= {  4:True,  1:True, 2:True`

  `}`
- `DFS(graph,visited,4)`
- `DFS(graph,visited,1)`
- `DFS(graph,visited,2)`
- `DFS(graph,visited,3)`

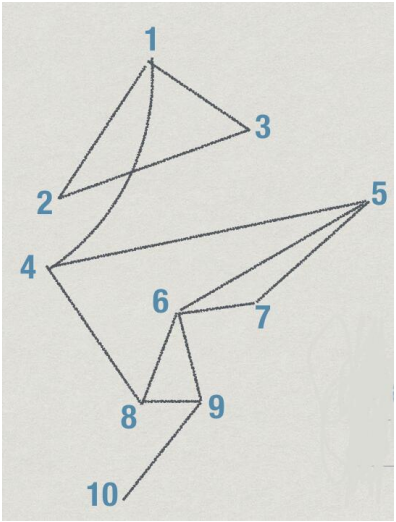
Example



- `visited= { 4:True, 1:True, 2:True, 3:True`

  `}`
- `DFS(graph,visited,4)`
- `DFS(graph,visited,1)`
- `DFS(graph,visited,2)`
- `DFS(graph,visited,3)`

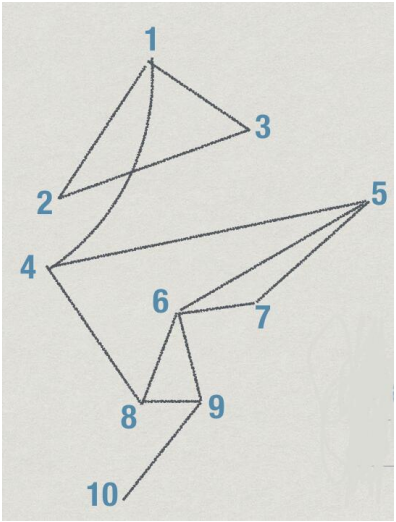
Example



- visited= { 4:True, 1:True, 2:True, 3:True,

  }
- DFS(graph,visited,4)
- DFS(graph,visited,1)
- DFS(graph,visited,2)
- DFS(graph,visited,3)
- DFS(graph,visited,5)

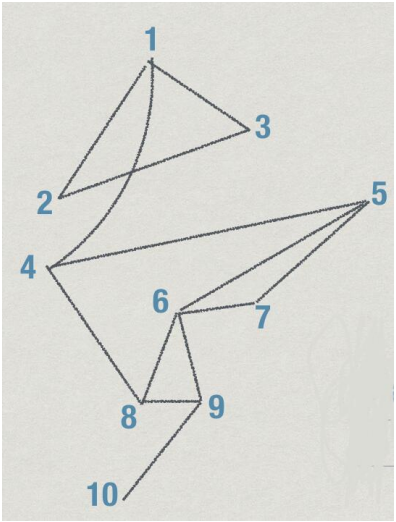
Example



- `visited= { 4:True, 1:True, 2:True, 3:True, 5:True`

  `}`
- `DFS(graph,visited,4)`
- `DFS(graph,visited,1)`
- `DFS(graph,visited,2)`
- `DFS(graph,visited,3)`
- `DFS(graph,visited,5)`
- `DFS(graph,visited,6)`

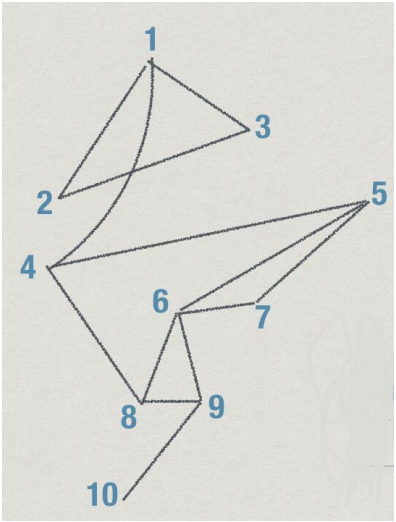
Example



- `visited= { 4:True, 1:True, 2:True, 3:True, 5:True, 6:True`

  `}`
- `DFS(graph,visited,4)`
- `DFS(graph,visited,1)`
- `DFS(graph,visited,2)`
- `DFS(graph,visited,3)`
- `DFS(graph,visited,5)`
- `DFS(graph,visited,6)`
- `DFS(graph,visited,7)`

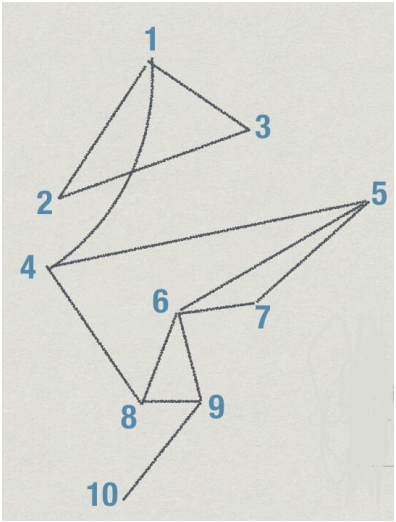
Example



- visited= { 4:True, 1:True, 2:True, 3:True, 5:True, 6:True, 7:True

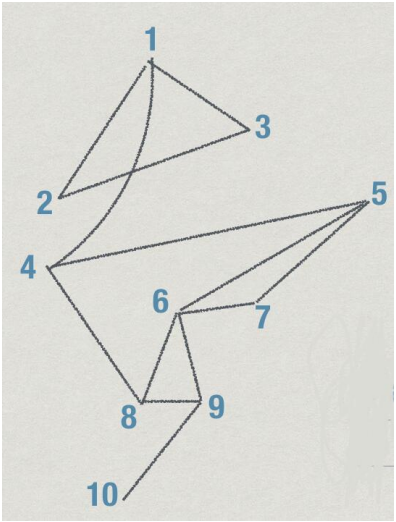
  }
- DFS(graph,visited,4)
- DFS(graph,visited,1)
- DFS(graph,visited,2)
- DFS(graph,visited,3)
- DFS(graph,visited,5)
- DFS(graph,visited,6)
- DFS(graph,visited,7)

Example



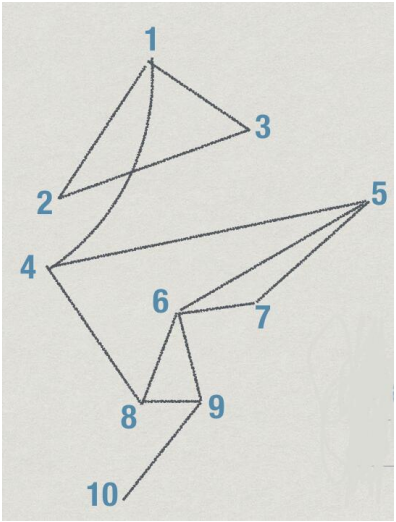
- visited= { 4:True, 1:True, 2:True, 3:True, 5:True, 6:True, 7:True, 8:True }
- DFS(graph,visited,4)
- DFS(graph,visited,1)
- DFS(graph,visited,2)
- DFS(graph,visited,3)
- DFS(graph,visited,5)
- DFS(graph,visited,6)
- DFS(graph,visited,7)
- DFS(graph,visited,8)

Example



- `visited= { 4:True, 1:True, 2:True, 3:True, 5:True, 6:True, 7:True, 8:True, 9:True }`
- `DFS(graph,visited,4)`
- `DFS(graph,visited,1)`
- `DFS(graph,visited,2)`
- `DFS(graph,visited,3)`
- `DFS(graph,visited,5)`
- `DFS(graph,visited,6)`
- `DFS(graph,visited,7)`
- `DFS(graph,visited,8)`
- `DFS(graph,visited,9)`

Example



- visited= { 4:True, 1:True, 2:True, 3:True, 5:True, 6:True, 7:True, 8:True, 9:True, 10:True }
- DFS(graph,visited,4)
- DFS(graph,visited,1)
- DFS(graph,visited,2)
- DFS(graph,visited,3)
- DFS(graph,visited,5)
- DFS(graph,visited,6)
- DFS(graph,visited,7)
- DFS(graph,visited,8)
- DFS(graph,visited,9)
- DFS(graph,visited,10)

Summary

- Depth first search is a systematic procedure to explore a graph
- Recursively visit all unexplored neighbours
- Keep track of visited vertices in a dictionary
- Can discover properties of the graph — for instance, is it connected?