Lecture 7: Depth-First Search

CLRS, section 22.3

Outline of this Lecture

- The depth-first search algorithm.
  Given for digraphs but can easily be modified for undirected graphs

- Running time analysis of DFS.

Background

Graph Traversal Algorithms: Graph traversal algorithms visit the vertices of a graph, according to some strategy.

Example: The BFS is an example of a graph traversal algorithm that divides graph up into connected components and traverses each component separately. It traverses the vertices of each component in increasing order of the distances of the vertices from the 'root' of the component.
Can be thought of us processing 'wide' and then 'deep'.

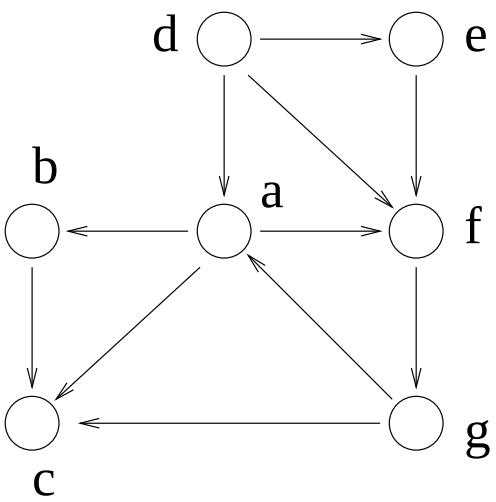
DFS will process the vertices first deep and then wide. After processing a vertex it recursively processes *all* of its descendants

DFS

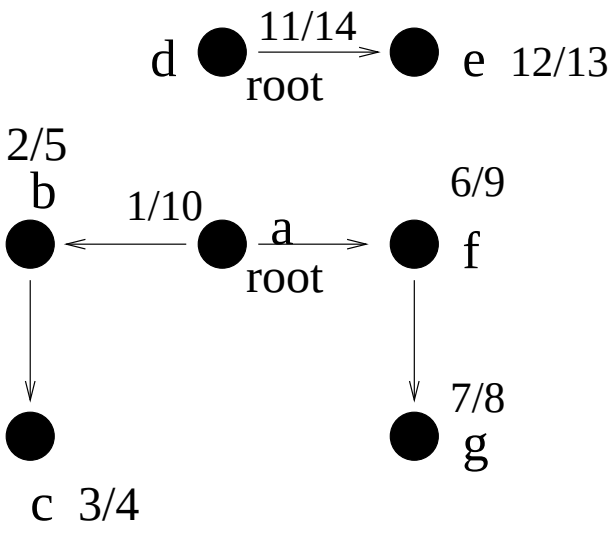
Graph is $G = (V, E)$. The algorithm works in discrete time steps. Each vertex v is given a "discovery" time $d[v]$ when it is first processed and a "finish" time, $f[v]$ when all of its descendants are finished.

The output is a collection of trees. As well as $d[v]$ and $f[v]$ each node points to $\mathsf{pred}[v]$, its parent in the forest.

Example:



original graph



Two source vertices a, d

What Does DFS Do

Given a digraph $G = (V, E)$, it traverses all vertices of G and

- constructs a forest (a collection of rooted trees), together with a set of source vertices (the roots); and

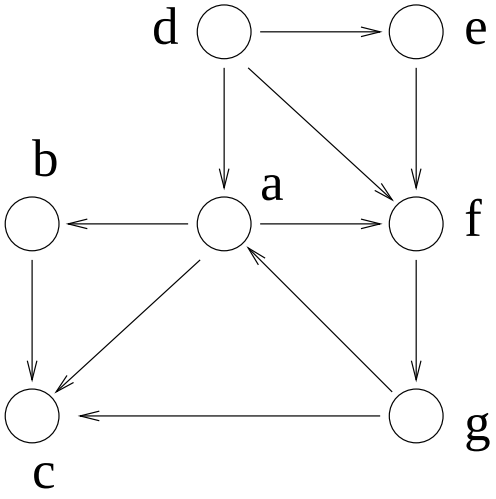
- outputs two arrays, $d[v]/f[v]$, the two time units.

Note: Forest is stored in pred[] array with pred[v] pointing to parent of v in the forest. pred[] of a root node is *Nil.*

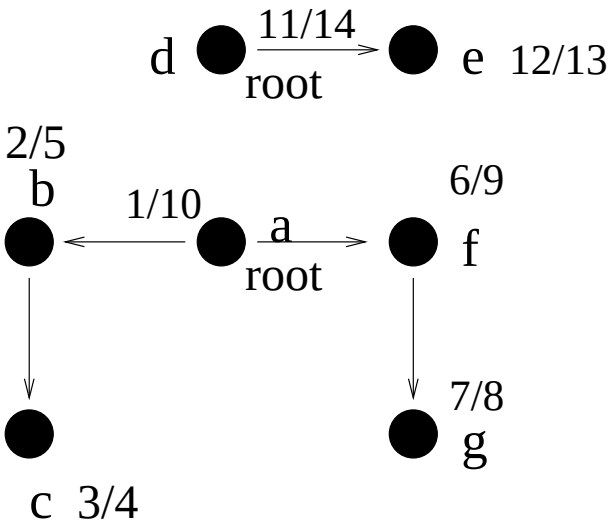
The DFS Forest

DFS Forest: DFS creates a forest $F = (V, E_f)$, a collection of rooted trees, where

$E_f = \{(pred[v], v) |$ where DFS calls are made$\}$



original graph



Two source vertices a, d

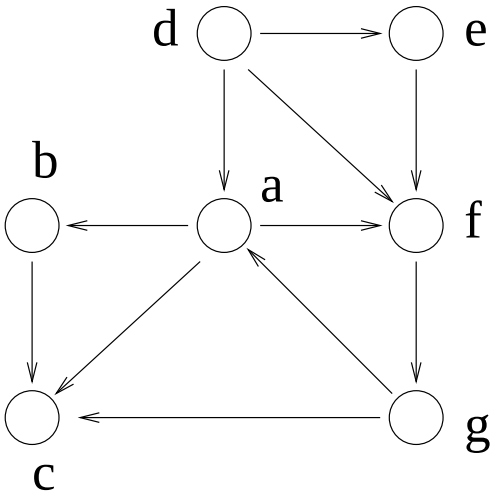
Idea of the DFS Algorithm

- In DFS, edges are explored out of the most recently discovered vertex v. Only edges to unexplored vertices are explored.
- When all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discovered.
- The process continues until we have discovered all the vertices that are reachable from the original source vertex.
- If any undiscovered vertices remain, then one of them is selected as a new source vertex, and the search is repeated from that source vertex.
- This process is repeated until all vertices are discovered.

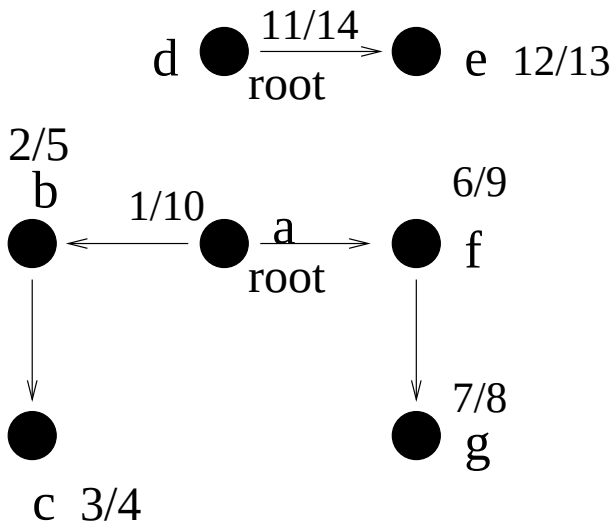
The strategy of the DFS is to search "deeper" in the graph whenever possible.

Idea of the DFS Algorithm by Example

Apply the Idea to the Following Graph:



original graph



Two source vertices a, d

Four Arrays for the DFS Algorithm

To record data gathered during traversal.

- $color[u]$, the color of each vertex visited: white means *undiscovered*, gray means *discovered* but not finished processing, and black means *finished* processing.

- $pred[u]$, the predecessor pointer, pointing back to the vertex that discovered u.

- $d[u]$, the *discovery time*, a counter indicating when vertex u is discovered.

- $f[u]$, the *finishing time*, a counter indicating when the processing of vertex u (and all its descendants) is finished.

Depth-First Search (DFS) Algorithm

```
DFS(G)                                  // Main program
{   for each u in V                     // Initialize
    {   color[u]=W; pred[u]=NIL; }
    time=0;
    for each u in V
        if(color[u]==W) DFSVisit(u); // Start new tree
}

DFSVisit(u)                             // Process vertex u
{   color[u]=G;                         // Vertex discovered
    d[u]=++time;                        // Time of discovery
    for each v in adj[u]
        if(color[v]==W)                 // Visit undiscovered
        {   pred[v]=u;  DFSVisit(v);}//     neighbours
    color[u]=B;                         // Vertex finished
    f[u]=++time;                        // Time of finish
}
```

Running Time Analysis of DFS

```
DFS(G)
{   for each u in V
    {   color[u]=W; pred[u]=NIL; }      2n
    time=0;                             1
    for each u in V
        if(color[u]==W) DFSVisit(u);    n (tests)
}           Sum: 3n + 1

DFSVisit(u)                             1  (call)
{   color[u]=G;                         1
    d[u]=++time;                        2
    for each v in adj[u]
        if(color[v]==W)                 outdeg(u)  (tests)
        {   pred[v]=u; DFSVisit(v); }   ≤ outdeg(u)
    color[u]=B;                         1
    f[u]=++time;                        2
}   Sum: 7 + outdeg(u) ≤ T_u ≤ 7 + 2 outdeg(u)
```

Running Time Analysis of DFS – Continued

The total running time is

$$(3n + 1) + \sum_{u \in V} T_u.$$

We have

$$\sum_{u \in V} T_u \leq \sum_{u \in V} (7 + 2\,\text{outdeg}(u)) = 7n + 2e$$

and

$$\sum_{u \in V} T_u \geq \sum_{u \in V} (7 + \text{outdeg}(u)) = 7n + e.$$

Hence

$$T(n, e) \leq 10n + 2e + 1 = O(n + e),$$

$$T(n, e) \geq 10n + e + 1 = \Omega(n + e).$$

Therefore $T(n, e) = \Theta(n + e)$.

Questions about DFS

Question: Is the DFS forest unique? Why?

Question: How do you prove that the DFS traverses over all vertices of G?

Question: Does the DFS algorithm work for undirected graphs?

On-line Example of DFS

Task: Apply the DFS to this digraph and compare whether your DFS forests are the same! What about the two arrays $d[v]$ and $f[v]$.