Tirgul 11

- DFS
- Properties of DFS
- Topological sort

Depth First Search (DFS)

- We will now see another approach to graph traversal – Depth First Search (DFS).
- The strategy that we use in DFS is to go as “deep” as we can in the graph.
- We check the edges that expands from the last vertex we checked, and that wasn’t checked yet.

DFS – cont.

- Like *BFS*, the *DFS* algorithm colors the vertices as it goes. In the beginning of the algorithm, all the vertices are white. In the first time that the algorithm sees a vertex, it is painted in gray. When the algorithms finishes handling a vertex, it is painted in black.
- In addition, each vertex v has two time stamps. The first, v.d, is the time when it was painted in gray (discovered). The second, v.f, is the time when it was painted in black (finished).

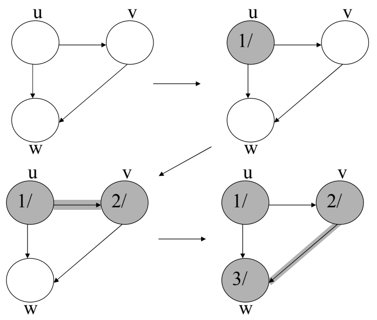
DFS – pseudo code

```
DFS(G)
     //initializing.
     for each vertex u∈V[G] {
          u.color = white;
          u.prev = nil;
     }
     time = 0;
     for each vertex u ∈V[G] {
          if (u.color == white)
               DFS-VISIT(u)
     }
```

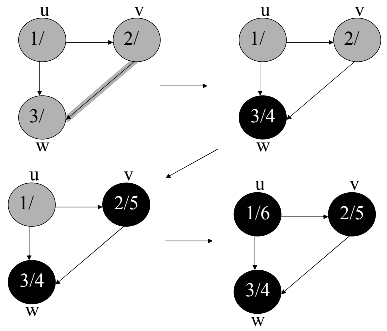
DFS – pseudo code (cont.)

```
DFS-VISIT(u)
     u.color = gray;
     u.d = ++time;
     for each vertex v∈adj[u] {
          if (v.color == white) {
               v.prev = u;
               DFS-VISIT(v);
          }
     }
     u.color = black;
     u.f = ++time;
```

A short example



A short example (cont.)



Running time of DFS :

- What is the running time of DFS ?
- Both loops in the DFS procedure takes O(|V|) time, not including the calls to DFS-VISIT.
- The algorithm calls DFS-VISIT exactly once for each vertex, because it is only called on white vertices. Each DFS-VISIT takes |adj[v]| to finish. Thus, the running time of the second loop is:
- $\sum_{v\in V}|adj[v]| = \Theta(E)$.
- And the total running time is:
- $\Theta(E+V)$.

predecessor subgraph of DFS

- **Definition**: the *predecessor subgraph* of DFS is the graph $G_\pi(V,E_\pi)$ when $E_\pi=\{(v.prev, v) \mid v\in V\}$ (v.prev is defined during the run of DFS).
- The predecessor subgraph of DFS creates a depth-first rooted forest, which consists of several depth-first rooted trees. The coloring of vertices and the fact that we update the prev field only when we reach a white vertex ensures that the trees in the first-depth forest are disjoint.

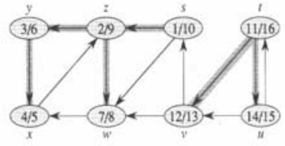
Properties of DFS :

- The parenthesis theorem:
- Let G be a graph (directed or undirected) then after DFS on the graph:
- For each two vertices u, v exactly one of the following is true:
- [u.d, u.f] and [v.d, v.f] are disjoint.
- $[u.d, u.f] \subseteq [v.d, v.f]$ and u is a descendant of v.
- $[v.d, v.f] \subseteq [u.d, u.f]$ and v is a descendant of u.
- Immediate conclusion: a vertex v is a descendant of a vertex u in the first-depth forest iff $[v.d, v.f] \subseteq [u.d, u.f]$.

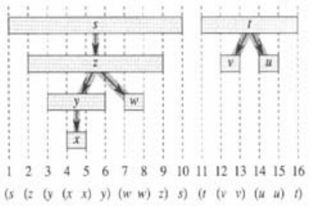
Proof of the parenthesis theorem :

- We will consider the case where u.d < v.d
- If u.f > v.d this means that we first encountered v when u was still gray. Therefore, v is a descendant of u. Furthermore, since v was discovered after u, all the edges that expand from v are checked and it is painted black before u is painted in black. Thus, v.f < u.f and $[v.d, v.f] \subseteq [u.d, u.f]$.
- If u.f < v.d then [u.d, u.f] and [v.d, v.f] are disjoint.
- The case which v.d < u.d is symmetrical, only switch u and v in the above argument.

The DFS results:



- After DFS on directed graph. Each vertex has a time stamp. The edges in gray are the edges in the depth-first forest.



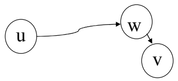
- The first-depth forest of the above graph. There is a correspondence between the discover and finish times of each vertex and the parenthesis structure below.

The white path theorem:

- Theorem: in a depth-first forest of a graph G, a vertex v is a descendant of a vertex u iff in the time u.d, which the algorithm discovers u, there is a path from u to v which consists only of white vertices.
- Proof: ⇒ Assume that u is a descendant of v, let w be a vertex on the path from u to v in the depth-first tree. The conclusion from the parenthesis theorem implies that u.d<w.d and thus w is white in time u.d

The white path theorem (cont.)

- Proof: ⇐



- w.l.o.g. each other vertex along the path becomes a descendant of u in the tree.
- Let w be the predecessor of u in the path.
- According to the parenthesis theorem:
- w.f≤u.f (they might be the same vertex).
- v.d>u.d and v.d<w.f.
- Thus, u.d<v.d<w.f ≤u.f. According to the parenthesis theorem, [v.d, v.f] ⊆ [u.d, u.f], and v must be a descendant of u.

edge classification:

- Another interesting property of depth-first search is that it can be used to classify the edges of the graph. This classification can give important information on the graph.
- We define four types of edges:

□ 1. *tree edges* are edges in G_π.

▨ 2. *back edges* are edges which connects a vertex to it's ancestor in a first-depth tree (a cycle).

■ 3. *forward edges* are edges which are not tree edges but connects a vertex u to a descendant v in a first-depth tree.

4. *cross edges* are all the other edges.

A-cyclic graphs and DFS:

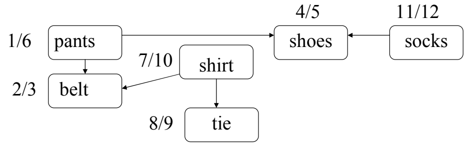
- A directed a-cyclic graph is denoted *DAG*.
- Theorem: A graph is a DAG iff during a DFS run on the graph, there are no back edges.
- Proof: ⇐ Assume that there is a back edge, (u,v). So v is an ancestor of u in the depth first tree and the edge (u,v) completes a cycle.
- ⇒ Assume that G contains a cycle c. Let v be the first vertex that DFS discovers. Let u be the predecessor of v in the cycle. In time v.d, there is a white path from v to u. According to the white path theorem, u is a descendant of v in the depth first tree. Thus, the edge (u,v) is a back edge.

Topological Sort

- A topological sort of a DAG G is a linear ordering of the vertices in G, such that if G contains an edge (u,v), then u appears before v in the ordering. If G contains a cycle, no such ordering exists.
- Topological-Sort(G)
- call DFS on G. As each vertex is finished, insert it onto the front of a linked list.
- What is topological sort good for ?
- DAG's are used in many applications to denote precedence order in a set of events. A Topological Sort of such a graph suggests an order to the events.

Topological Sort - example

- How to dress in the morning ?



- After using DFS in order to compute the finish times we have the vertices, ordered according to the finish times and now we can dress…
- socks --- shirt --- tie --- pants --- shoes --- belts