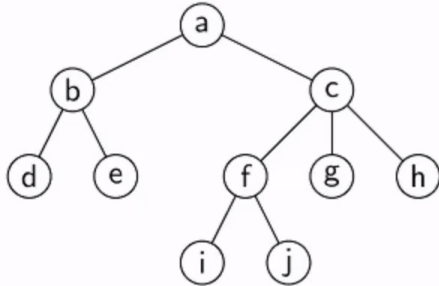# Idea

DFS explores the graph, starting at the last visited
vertex having unvisited neighbors.

- **Special case**: $G$ is a tree $\Rightarrow$ DFS-Order $=$ pre-order



- Maintain Stack ST
  that contains all
  visited but not yet
  satured nodes.

- Rest similar to BFS

**Question:** What is the pre-order for this tree?

$$a\ b\ d\ e\ c\ f\ i\ j\ g\ h$$

## Pseudo-Code

```
DFS(G)        /* G given as adjacency list F */
for all u ∈ V
     state(u)=new
     pre(u)=nil
for all u ∈ V      /* loop not necessary for connected graphs */
     if state(u)==new
          DEPTH(u)


DEPTH(u)
state(u)=visited; write(u)
for all v ∈ F[u]       /* test all neighbors of u */
     if state(v)==new
          pre(v)=u
          DEPTH(v)          ← recursion can be replaced by stack
state(u)=saturated
```
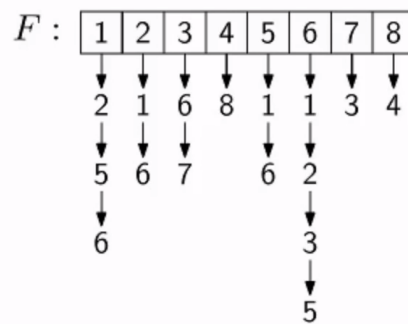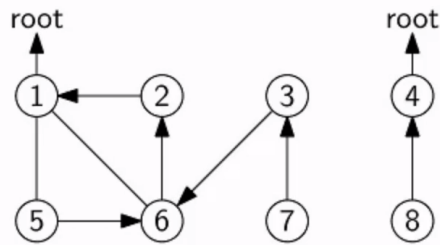
**Example:**



$F:$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 2 | 1 | 6 | 8 | 1 | 1 | 3 | 4 |
| 5 | 6 | 7 |   | 6 | 2 |   |   |
| 6 |   |   |   |   | 3 |   |   |
|   |   |   |   |   | 5 |   |   |

**Further Observations:**

- The pre-pointers form a set of trees (DFS-forest);
- every call of DEPTH in the main programm (not in the recursion) results in a new root and tree.
- For connected graphs there is only one root and tree.

**Properties**

- time complexity
  - DEPTH ist called exactly once per node (only for new nodes, that are immediately marked as "visited").
  - A call of DEPTH($v$) takes $O(\text{degree}(v))$ time
  - $\Rightarrow \Theta(n + m)$ time in total
- space complexity
  - $\Theta(n + m)$ **space in total**

  - correctness
    - vertex that is set to visited:
      - put on stack
      - when removed from stack, all neighbors are considered
    - $\Rightarrow$ every vertex set to visited exactly once
    *