Directed Acyclic Graph

O(V+E) time

Strongly Connected

∀v, w ∈ V : if v → w and w → v for all u ≠ v

Directed graphs
reachability isn't symmetric

DFS(v):

mark v
PREVISIT(v)
if w is unmarked
parent(w) = v
DFS(w)
POSTVISIT(v)

DFSALL(G):

Preprocessing (G)
For all vertices v
unmark v
For all vertices v
if v unmarked
DFS(v)
DAG

DFS(v):
- mark v
- PREVISIT(v)
  - For each edge v -> w
    - if w is unmarked
      - parent[w] = v
      - DFS(w)
- POSTVISIT(v)

Memoize(x):
- if value(x) is undefined
  - init value(x)
  - for all subproblems y of x
    - Memoize(y)
    - update value(x) to value(y)
- finalize value(x)
  - return value(x)

Memoized recursion IS DFS in DAG
Dynamic programming IS top sort + iterative traversal
**DFS(v):**
- `mark v`
- For each edge `v→w`
  - if `w` is unmarked
  - `parent(w) = v`
  - `DFS(w)`
  - `v.done ← clock`
  - `clock ← clock + 1`

**DFSALL(G):**
- `Clock ≤ 0`
- For all vertices `v`
  - `unmark v`
  - `Add vertex s`
  - `for all v ∈ V`
  - `add s → v`
  - `DFS(s)`

**Lemma:** After DFSALL, if `G` is a dag for all `v→w`
- `v.done > w.done`

**Proof:** Let `t` be the vertex s.t. `t.done = 0`

**Claim:** `t` has no outedges. t is a sink.
- Suppose `t → z`
  - `DFS(t)` calls `DFS(z)` unless `z` is marked
  - `z` is not already marked
    - `call DFS(z) → z.done < ?`
    - `t.done < ? + 1 > z.done ≥ 0`
    - `t.done > 0` ✗
  - `z` is already marked, and `z.done` undefined.
    - `z` can reach `t`
    - `t` can reach `z`
    - cycle! ✗
Longest Path in a DAG

**Input:** DAG $G=(V,E)$ $l : E \rightarrow \mathbb{R}$ edge length

**Want:** total length of longest path in $G$

$V_0 \rightarrow V_1 \rightarrow \ldots \rightarrow V_k$

**Alg:**
1. Top sort DAG

$LLP(i) = \text{length of longest path in } G \text{ starting at vertex } i$

$LLP(i) = \begin{cases} \infty, & i \notin V \\ \max \{ l(i, j) + LLP(j) \mid i \rightarrow j \} & \text{o/w} \\ \max \phi = 0 \end{cases}$

$O(V + E)$