DAGS: DIRECTED ACYCLIC GRAPHS
(Notes by Henri Mertens)
DAG: DIRECTED ACYCLIC GRAPH

USES:
- PARTIAL ORDERS (≤, ≤)
- ARITHMETIC EXPRESSIONS, COMMON SUBEXPRESSIONS
- PREREQUISITE TREE
- PARTITIONS OF REGIONS
- PERT NETWORKS (JOB PLANNING)
- CRITICAL PATHS
- GAME OF NIM
EXAMPLE: PARTIAL ORDER OF SETS

\[ \{1, 2, 3, 4\} \]

\[ \{1, 3\} \]

\[ \{2, 3\} \]

\[ \{1, 4\} \]

\[ \{2, 3\} \]

\[ \{1, 4\} \]

\[ \{2, 4\} \]

\[ \{1, 2, 4\} \]

\[ \{1, 2, 3\} \]

\[ \{1, 2, 3, 4\} \]

plus many multi-level edges.
EXAMPLE: EXPRESSION DAG

Graph:

```
    *   -
   / \
  +   /
 /   /
*   /
/   /
*   /
/
```

Represents:

\[
(\alpha \cdot (3 + \gamma) + \delta) \cdot \\
\left(\frac{\alpha}{(3 + \gamma)} - 8\right)
\]

Use: In optimizing compilers.
There exists a linear ordering consistent with all directed edges.

**TOPOLOGICAL SORTING**
**CONSISTENT LABELING**

All days can be represented like this:
**Proof.**

We exhibit an algorithm that produces a linear ordering: DFS, with nodes listed in order of $f[v]$-values.

We need: If $(u, v) \in E$, then $f[u] < f[v]

Consider the unique time when $v$ is processed during DFS.

1. $v$ gray: $v$ ancestor of $u$ → impossible.
2. $v$ black: $f[v] < f[u]$
3. $v$ white: $v$ descendant of $u$, so $u$ by nesting
COMPLEXITY: $O(1V + 1E)$ (DFS) started anywhere?
REVERSING A DAG:

Empty the adjacency lists.

vertex
1 → 19, 4, 8, 7
2 → 5, 9, 3
3 → 8, 6, 10
4 → 11, 5, 12
7 → 8, 9, 11, 10
11 → 12

adjacency lists

Note: 5, 6, 8, 9, 10, 12 leaves

vertex
1 → 2
2 → 1
3 → 2
4 → 1
5 → 2, 4
6 → 3
7 → 2
8 → 1, 3, 7
9 → 2, 7
10 → 3, 7
11 → 4, 7
12 → 1, 4, 11

Note:

Time $O(|E| + |V|)$

Note: 1, 7 roots.
APPLICATION: ACTIVITY or PERT NETWORKS
CRITICAL PATHS

Given: A dag. O - crucial moment in time.
\( v \) = activity between moments \( v, v' \), taking time \( T[v, v'] \).

One starting node

One leaf node (job finished)

Critical path
Find minimum time to complete the job. Note that for all modes $v$:

$$\text{Time}[v] = \max_{(u,v) \in E} (\text{Time}[u] + T[u,v])$$

**Algorithm**

1. $\text{Time}[^{\text{root}}{}_{}] \leftarrow 0$
2. For all $v$, in linear order, do (*).
3. Return $\text{Time}[^{\text{leaf}}{}_{}]$

**Critical path** — path on which any delay causes a project delay.

**Exercise**: Find a critical path from root to leaf.
The game is based on an integer vector, where each integer represents a number of sticks.

For example, (1, 3, 5, 7) corresponds to four piles of sticks:

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
```

Players alternate. In one move, they can remove any number of sticks from one pile. The player who is forced to take the last stick loses.

Question: Given the game vector, can we determine if the first player will win or lose?
THE NIM GAME DAG.

Example: $(1, 3, 5)$ game

BAD \[\longrightarrow\] (11) (2) (2) (4) (5) (12) (13) (14) (15)

GOOD \[\longrightarrow\] (22) (111)

BAD \[\longrightarrow\] (23) (24) (25) (122) (112) (113) (114) (115)

GOOD \[\longrightarrow\] (33) (123)

BAD \[\longrightarrow\] (34) (35) (124) (125) (134) (135)

GOOD \[\longrightarrow\] (1)

One move from GOOD to BAD suffices. All moves from BAD to BAD must lead to GOOD.
So, we linearly order the positions and determine for all vertices whether they are good (G) or bad (B).

Data Structure: \( \text{Kind}[u] \in \{ \text{"BAD", "GOOD"} \} \)

Algorithm:

1. Topologically sort all positions, and place them in a queue \( Q \)
2. While \( |Q| > 0 \):
   - \( u \leftarrow \text{DEQUEUE}(Q) \)
   - \( \text{Kind}[u] \leftarrow \text{"BAD"} \)
   - For all \( v \) that can be reached from \( u \) by one move do:
     - If \( \text{Kind}[v] = \text{"BAD"} \) then \( \text{Kind}[v] = \text{"GOOD"} \)

Complexity: \( O(|E| + |V|) \)
**QUESTIONS**

**Research:** For a fixed game, say \((a_1, a_2, \ldots, a_m)\), is there a simple way to determine, formulaically, if it is "BAD" or "GOOD"? 

*(Yes: Guy and Berlekamp)*

**CHOMP** (game invented by David Gale, U.C. Berkeley)

```
  O  O  O  O  O
  O  O  O  O  O
  O  O  O  O  O
  O  O  O  O  O
  O  O  O  O  O
```

Each player in turn takes a rectangular bite. The poison pill eater loses.

"4x5 CHOMP"
DECOMPOSITION OF DIRECTED GRAPHS

Two modes \( u, v \) are equivalent \( U \equiv V \), if there is a path from \( u \rightarrow v \) and from \( v \rightarrow u \).

This creates equivalence classes, also called strongly connected components.

THE DAG structure of strongly connected components.
ALGORITHM TO DETERMINE ALL S.C. COMPONENTS

Given $G = (V, E)$

Compute $G^T = (V, E^T)$ — all edges of $E$ are reversed in $E^T$

Step 1: Perform DFS on $G$. Record all $f[u]$, $u \in V$.

Step 2: Perform DFS on $G^T$, considering nodes from large $f$-value to small.

Note: The trees in the Depth First Forest after Step 2 are the strongly connected components. (and the roots are the forefathers)

Complexity: $O(|V| + |E|)$
In step 2, DFS started at 2, must visit all of S

$\forall u:\ d[2] \leq d[u] \leq f[u] \leq f[c]$ (white path theorem)
Recall: edges reversed
Forefathers considered from large to small
\[ f[x_1] > f[x_2] > \ldots > f[x_k] \].

DFS at \( x_1 \) halts when \( S_1 \) visited.
DFS at \( x_2 \) " " " \( S_2 \) " " .
DFS at \( x_k \) " " " \( S_k \) " " .