CS 170: Algorithms

Prof David Wagner.
Slides edited from a version created by Prof. Satish Rao.
For UC-Berkeley CS170 Fall 2014 students use only.
Do not re-post or distribute
Cycle in a directed graph?

Fast algorithm for finding out whether directed graph has cycle?
For each edge \((u, v)\) remove, check if \(v\) is connected to \(u\)
\(O(|E|(|E| + |V|))\).
Linear Time (i.e. \(O(|V| + |E|)\))?
Directed graphs.

$G = (V, E)$

vertices $V$.

edges $E \subseteq V \times V$.

Edge: $(u, v)$

From $u$ to $v$.

Source – $u$

Dest – $v$
Introspection: pre/post.

Previsit(v):
2. clock := clock+1

Postvisit(v):
2. clock := clock+1

DFS(G):
0. Set clock := 0.

Clock: goes up to 2 times number of tree edges.
First pre: 0

Property: For any two nodes, u and v, \([pre(u), post(u)]\) and \([pre(v), post(v)]\) are either disjoint or one is contained in other.

Interval is “clock interval on stack.”

Either both on stack at some point (contained) or not (disjoint.)
Directed Acyclic Graphs: Depth First Search

Edge: \((u, v)\)

From \(u\) to \(v\).

Source – \(u\)
Dest – \(v\)

Given a DFS forest, the edge \((u, v)\) of the graph is a

Tree edge – “Direct call tree of explore”, \((u, v) \in T\), \(pre(u) < pre(v) < post(v) < post(u)\).

Forward edge – “Edge to descendant (not in tree), \((u, v) \notin T\), \(pre(u) < pre(v) < post(v) < post(u)\)

Back edge – “Edge to ancestor” \((u, v) \notin T\), \(pre(v) < pre(u) < post(u) < post(v)\)

Cross edge – None of the above: \((u, v) \notin T\), \(pre(v) < post(v) < pre(u) < post(u)\)

\(v\) already explored before \(u\) is visited.

These are all the possible edges.
Directed Acyclic Graph

Directed Graph ...without cycles.
Why?

"Hello" before "Goodbye"
Example.

Acyclic Graph?
Depth first search: directed.

Back edge \((u, v)\): \(\text{int}(v)\) contains \(\text{int}(u)\).

\[ \text{int}(C) = [3, 4] \text{ and } \text{int}(B) = [1, 8]. \]

Back edge \((u, v)\)
edge to ancestor
..........path of tree edges from \(v\) to \(u\).
Back edge means cycle! \(\implies\) not acyclic!
Testing for cycle.

**Thm:** A graph has a cycle if and only if there is a back edge in any DFS.

**Proof:**
We just saw: Back edge \( \iff \) cycle!

In the other direction: Assume there is a cycle

\[ v_0 \rightarrow v_1 \rightarrow v_2 \cdots \rightarrow v_k \rightarrow v_0 \]

Assume that \( v_0 \) is the first node explored in the cycle

( without loss of generality since can renumber vertices. )

When \texttt{explore}(v_0)\ returns all nodes on cycle explored.

All \( \text{int}[v_i] \) in \( \text{int}[v_0] \)!

\( \iff (v_k, v_0) \) is a back edge.

Cycle \( \iff \) back edge!
Fast checking algorithm.

**Thm:** A graph has a cycle if and only if there is back edge.
Algorithm ??

Run DFS.

\[ O(|V| + |E|) \] time.

For each edge \((u, v)\): is \(\text{int}(u)\) in \(\text{int}(v)\).

\[ O(|E|) \] time.

\[ O(|V| + |E|) \] time algorithm for checking if graph is acyclic!
Directed Acyclic Graph

```
Hello
    ↓
Goodbye
```

"Hello" before "Goodbye"

No cycles! Can tell in linear time!

Ohhh...Kayyyy...
Really want to find ordering for build!
Linearize.

**Topological Sort:** For $G = (V, E)$, find ordering of all vertices where each edge goes from earlier vertex to later in acyclic graph.
A linear order:

\[ A, E, F, B, G, D, C \]

In DFS: When is \( A \) popped off stack?

Last! When is \( E \) popped off? second to last. ...