Depth-First Search

- Review breadth-first search (BFS)
- Present depth-first search (DFS)
- Present topological sort algorithm

Next Time: Garbage Collection

Tuesday: Review
\textbf{BFS} (V source)

\begin{itemize}
  \item \textbf{Initialization}
    \begin{itemize}
      \item For each \( u \in V \)
        \begin{itemize}
          \item \texttt{u.discovered} = \texttt{false}
          \item \texttt{u.dist} = \infty
          \item \texttt{u.parent Edge} = \texttt{null}
        \end{itemize}
      \item \texttt{Source.discovered} = \texttt{true}
      \item \texttt{Source.dist} = 0
      \item \texttt{queue.enqueue(Source)}
    \end{itemize}
  \item While (\texttt{!queue.isEmpty()})
    \begin{itemize}
      \item \( u = \texttt{queue.dequeue()}
      \item For each edge \( e \) in outgoing edges from \( u \)
        \begin{itemize}
          \item \( v = \texttt{e.dest} \)
          \item If (\texttt{!v.discovered})
            \begin{itemize}
              \item \texttt{v.discovered} = \texttt{true}
              \item \texttt{v.dist} = \texttt{u.dist + 1}
              \item \texttt{v.parent Edge} = \texttt{e}
              \item \texttt{queue.enqueue(v)}
            \end{itemize}
        \end{itemize}
    \end{itemize}
\end{itemize}
Called breadth-first search because of how search proceeds.

In general, short wide tree.
depth-first search

1. Use stack vs queue

2. Continue (re-starting if needed) until all vertices are visited

3. Keep "time" when each vertex discovered & finish processing all outgoing edges

4. Replace boolean discovered by a 3-valued variable, color
false (for discovered)

color white — not yet discovered (discovery time (not on stack yet))

color gray — on stack but not (finishing time done processing)

color black — done processing (finished)
top-level dfs (in a graph class)

dfs ()

Color all vertices white + set parent to null

time = 1

for each vertex u in graph

  if (u.color == white)
    dfsVisit(u)

  each call here will create on tree in dfs forest

  u is root

  if
dfsVisit (Vertex u)
  u. color = gray
  u. discovery Time = time++  u→v
  for all vertices v reachable from u by an edge
  if (v.color == white)
    v. parent = u
    dfsVisit (v)
  if (v.color == gray)
    what does this tell you? you found a cycle (can be constructed using parent edges)
  u. color = black
  u. finishing Time = time++
Time complexity

$O(m+n)$ \{ with adj list \\

call dfs visit exactly once for each vertex, dominant cost is iterating over outgoing edges