Graph Search

EECS 214, Fall 2018
Questions we might ask about graphs

- Is there a path from \( v \) to \( u \)?
- What’s the shortest path from \( v \) to \( u \)?
- Are there any cycles?
Graph search: basic idea

To answer whether there’s a path (among other things), we can use:

- Depth-first search (DFS): go as far as you can along a path, then go back and try anything you haven’t tried yet
- Breadth-first search (BFS): explore all the successors of a vertex before exploring their successors in turn
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
DFS example
Recursive DFS algorithm (one source)

Procedure DFS($graph$, $start$) is

    $seen$ ← new array (same size as $graph$, filled with false);

    Procedure Visit($v$) is
        if not $seen[v]$ then
            $seen[v]$ ← true;
            for $u$ in Successors($graph$, $v$) do
                Visit($u$)
            end
        end
    end

    Visit($start$);

    return $seen$
end
Recursive DFS algorithm (one source, lifted)

Procedure `Visit(graph, seen, v)` is
   if not `seen[v]` then
      `seen[v] ← true;`
      for `u` in `Successors(graph, v)` do
         `Visit(graph, seen, u)`
   end
end

Procedure `DFS(graph, start)` is
   `seen ← new array (same size as graph, filled with false);`
   `Visit(graph, seen, start);`
   return `seen`
end
Recursive DFS algorithm (1 src., builds tree)

Procedure DFS\((graph, \text{start})\) is
\[
preds \leftarrow \text{new array (same size as graph, filled with false)};
\]
Procedure Visit\((pred, v)\) is
\[
\text{if not } preds[v] \text{ then}
\]
\[
preds[v] \leftarrow pred;
\]
\[
\text{for } u \text{ in Successors}(graph, v) \text{ do}
\]
\[
\text{Visit}(v, u)
\]
\[
\text{end}
\]
\[
\text{end}
\]
Visit\((true, \text{start})\);
return \(preds\)
end
Recursive DFS algorithm (full)

Procedure DFS(graph) is
    preds ← new array (same size as graph, filled with false);
    Procedure Visit(pred, v) is
        if not preds[v] then
            preds[v] ← pred;
            for u in Successors(graph, v) do
                Visit(v, u)
            end
        end
    end
    for v in Vertices(graph) do
        Visit(true, v)
    end
    return preds
end
Iterative DFS algorithm

Procedure $\text{DFS}(\text{graph, start})$ is

- $\textit{preds} \leftarrow \text{new array (same size as graph, filled with false)}$;
- $\textit{todo} \leftarrow \text{new stack}$;

- $\textit{preds}[\textit{start}] \leftarrow \text{true}$;
- $\text{Push}(\textit{todo, start})$;

while $\textit{todo}$ is not empty do

- $v \leftarrow \text{Pop}(\textit{todo})$;
  - for $u$ in $\text{Successors(}\text{graph, v})$ do
    - if not $\textit{preds}[u]$ then
      - $\textit{preds}[u] \leftarrow v$;
      - $\text{Push}(\textit{todo, u})$
    end
  end

  end

return $\textit{preds}$

end
Running DFS on a digraph

tree

back

cross

forward

g ← f ← h

a ← e ← f ← e ← a

c ← d ← b ← c

10
Running DFS on a digraph

tree
back
cross
forward

\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d} \\
\text{e} \\
\text{f} \\
\text{g} \\
\text{h}
\end{array}
\]
Running DFS on a digraph

tree

back

cross

forward

g ← f ← e ← h

c ← b ← a
Running DFS on a digraph
Running DFS on a digraph
Running DFS on a digraph

- tree
- back
- cross
- forward
Running DFS on a digraph

tree
back
cross
forward
Running DFS on a digraph

tree

back

cross

forward

10
Running DFS on a digraph

- Tree
- Back
- Cross
- Forward
Running DFS on a digraph

tree
back
cross
forward
Running DFS on a digraph

tree
back

cross
forward
Running DFS on a digraph

tree

back

cross

forward
Running DFS on a digraph
Running DFS on a digraph
Running DFS on a digraph
Running DFS on a digraph

tree
back
cross
forward
Running DFS on a digraph

tree
back
cross
forward
Running DFS on a digraph

tree

back

cross

forward
A DFS tree
DFS for cycle detection

Procedure FindCycle(graph) is
  started ← new array (same size as graph, filled with false);
  finished ← new array (same size as graph, filled with false);

Procedure Visit(v) is
  if not finished[v] then
    if started[v] then
      | we found a cycle!
    end
    started[v] ← true;
    for u in Successors(graph, v) do
      | Visit(u)
    end
    finished[v] ← true;
  end

for v in Vertices(graph) do
  Visit(v)
end
Breadth-first search

Procedure \( \text{BFS}(\text{graph}, \text{start}) \) is

\[ \text{preds} \leftarrow \text{new array (same size as graph, filled with false)}; \]
\[ \text{todo} \leftarrow \text{new queue}; \]

\[ \text{preds}[\text{start}] \leftarrow \text{true}; \]
\[ \text{Enqueue}(\text{todo}, \text{start}); \]

\[ \text{while } \text{todo} \text{ is not empty do} \]
\[ \quad \text{v} \leftarrow \text{Dequeue}(\text{todo}); \]
\[ \quad \text{for } u \text{ in } \text{Successors}(\text{graph}, v) \text{ do} \]
\[ \quad \\quad \quad \text{if not } \text{preds}[u] \text{ then} \]
\[ \quad \\quad \quad \quad \text{preds}[u] \leftarrow v; \]
\[ \quad \\quad \quad \quad \text{Enqueue}(\text{todo}, u) \]
\[ \quad \text{end} \]
\[ \text{end} \]

\[ \text{return } \text{preds} \]

end
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Running BFS on a digraph
Generic graph search

If \( todo \) is a stack we get DFS; if \( todo \) is a queue we get BFS:

Procedure \( \text{Search}(\text{graph}, \text{start}) \) is

\[ \text{preds} \leftarrow \text{new array (same size as graph, filled with false)}; \]
\[ \text{todo} \leftarrow \text{new collection}; \]
\[ \text{preds}[\text{start}] \leftarrow \text{true}; \]
\[ \text{Add} (\text{todo}, \text{start}); \]

\[ \text{while todo is not empty do} \]
\[ \quad \text{v} \leftarrow \text{Remove} (\text{todo}); \]
\[ \quad \text{for } u \text{ in Successors} (\text{graph}, \text{v}) \text{ do} \]
\[ \quad \quad \text{if not } \text{preds}[u] \text{ then} \]
\[ \quad \quad \quad \text{preds}[u] \leftarrow \text{v}; \]
\[ \quad \quad \quad \text{Add} (\text{todo}, u) \]
\[ \quad \text{end} \]
\[ \text{end} \]
\[ \text{return } \text{preds} \]
\[ \text{end} \]
Next time (after the exam): shortest paths