Lecture 19: Graph Applications

COSC242: Algorithms and Data Structures

Brendan McCane

Department of Computer Science, University of Otago

DFS, BFS, stacks and queues

BFS keeps track of the vertices in the graph using a queue. BFS adds new vertices (the children of the current vertices) to the back of a queue.

DFS on the other hand, uses a stack. In the previous algorithm the stack was implicit — we used the system stack, but we could just as well have used an explicit stack (S in the algorithm on the next page).

But, as can be seen on the following pages, using an explicit stack often results in more complicated code.

Recursive DFS

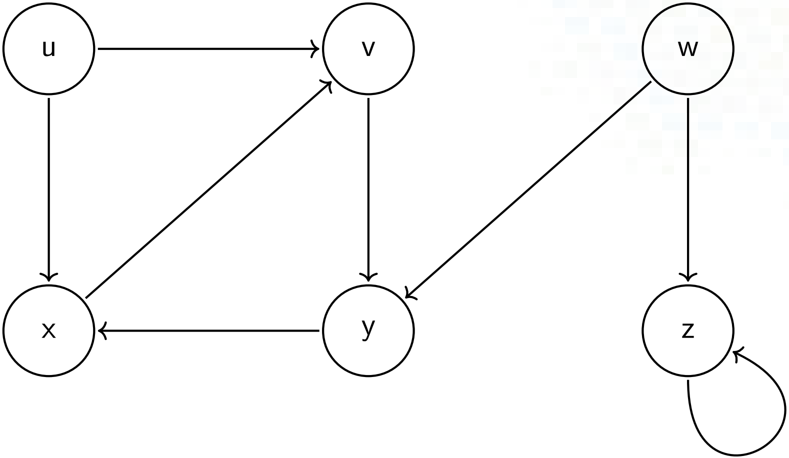
```
procedure DFS(G)
    for each u ∈ V do
        colour[u] ← white
    end for
    time ← 0
    for each u ∈ V do
        if colour[u]=white then
            DFS_visit(u)
        end if
    end for
end procedure
```

```
procedure DFS_VISIT(u)
    colour[u] ← grey
    time ← time + 1
    d[u] ← time
    for each vertex v ∈ adj[u] do
        if colour[v] = white then
            DFS_visit(v)
        end if
    end for
    colour[u] ← black
    time ← time + 1
    f[u] ← time
end procedure
```

DFS with explicit stack

procedure DFS_VISIT(u)
    $colour[u] \leftarrow grey$; $time \leftarrow time + 1$; $d[u] \leftarrow time$; Push(S, u)
    while not empty(S) **do**
        $u \leftarrow$ Top(S); $push \leftarrow false$
        for each vertex $v \in adj[u]$ **do**
            if $colour[v] = white$ **then**
                $push \leftarrow true$; break
            end if
        end for
        if push **then**
            $colour[v] \leftarrow grey$; $time \leftarrow time + 1$; $d[v] \leftarrow time$; Push(S, v)
        else
            $u \leftarrow$ Pop(S); $colour[u] \leftarrow black$; $time \leftarrow time + 1$; $f[u] \leftarrow time$
        end if
    end while
end procedure

DFS example



Application: Topological Sort

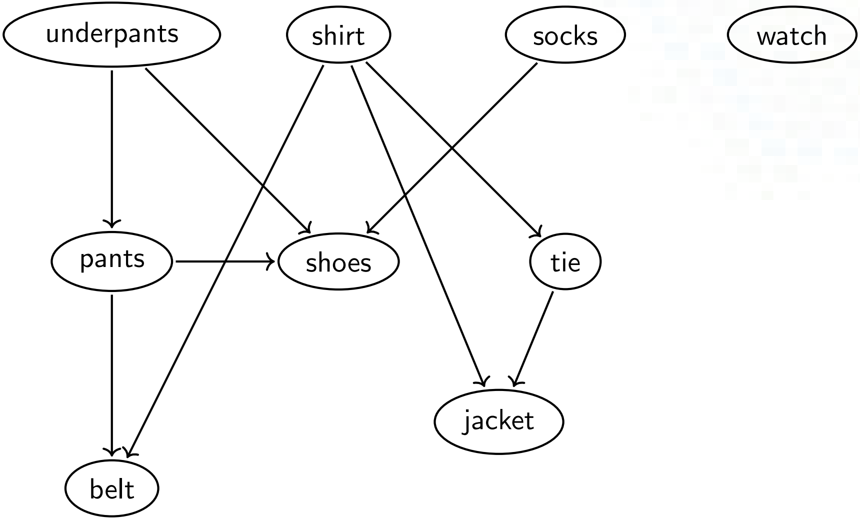
A *directed acyclic graph* (DAG) has no cycles.

A DAG can be used to model a project's dependencies.

A topological sort turns a DAG into a list where later items depend only on earlier items.

We can produce a topological sort using DFS — as each vertex is finished ($f[u]$ is set), add the vertex to the front of the list.

Topological sort example



Application: Path Finding

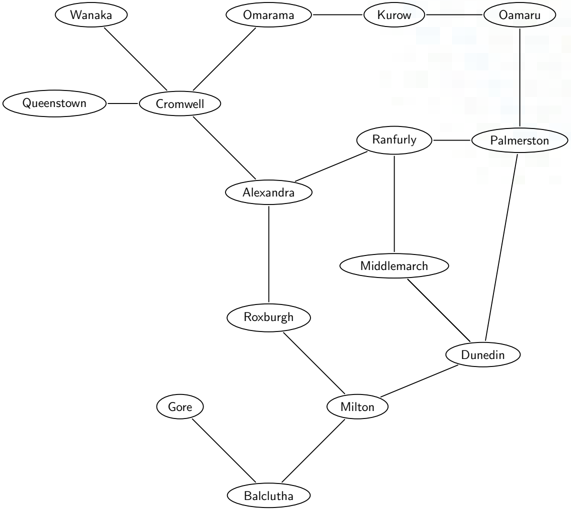
Maps are often represented as graphs. Every location on the graph is a vertex, and edges represent the fact that you can get from vertex a to vertex b in a single "step". This is how Google Maps represents locations on the map. Google Maps probably has many millions of locations (perhaps billions).

You can use DFS to find a path between two locations:

1. call DFS starting at the destination
2. build the DFS tree as you go,
3. halt if you reach the required source (which will be a leaf in the tree). Note, there may be no path.
4. then backtrack up the tree from the source to the leaf, and that gives a path between the two locations.

This path may not be the shortest one — we'll look at shortest path algorithms in a later lecture.

Path finding: Dunedin to Oamaru



Relevant parts of textbook

DFS is discussed in Section 22.3.

Topological sort is discussed in Section 22.4.