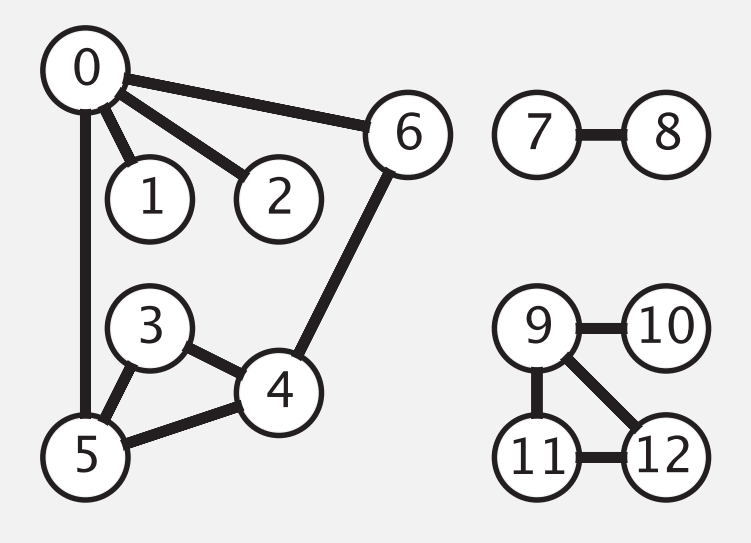
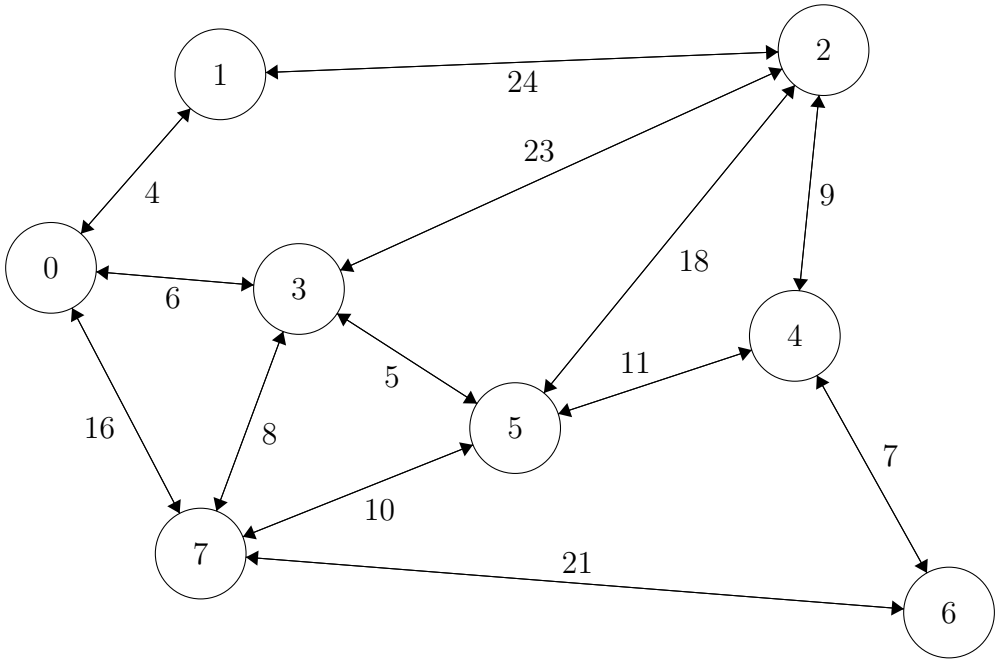
1 Graphs

1.1 Whatever-first search



Perform WhateverFirstSearch (page 200) on the graph above, starting with $S = 0$ and processing nodes in numerical order in case of deciding what to take next, with the following "bag" variants:

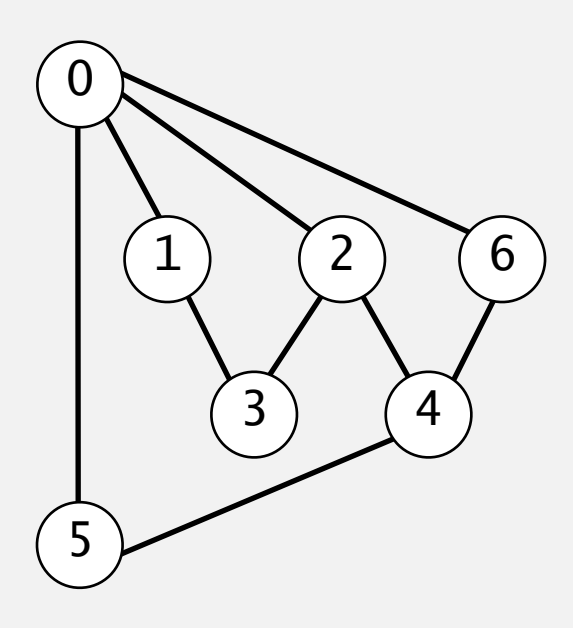
- Stack (what specialized search is this?)
- Queue (what specialized search is this?)
- Priority Queue, on this edge-weighted graph:



1.2 Connected components

Trace CountAndLabel (page 204) on the graph of 13 vertices on the preceding page, using DFS.

1.3 Graph-processing challenges



How difficult?

- Any programmer could do it.
- Typical diligent algorithms student could do it.
- Hire an expert.
- Intractable.
- No one knows.
- Impossible.

1.3.1 Is a graph bipartite?

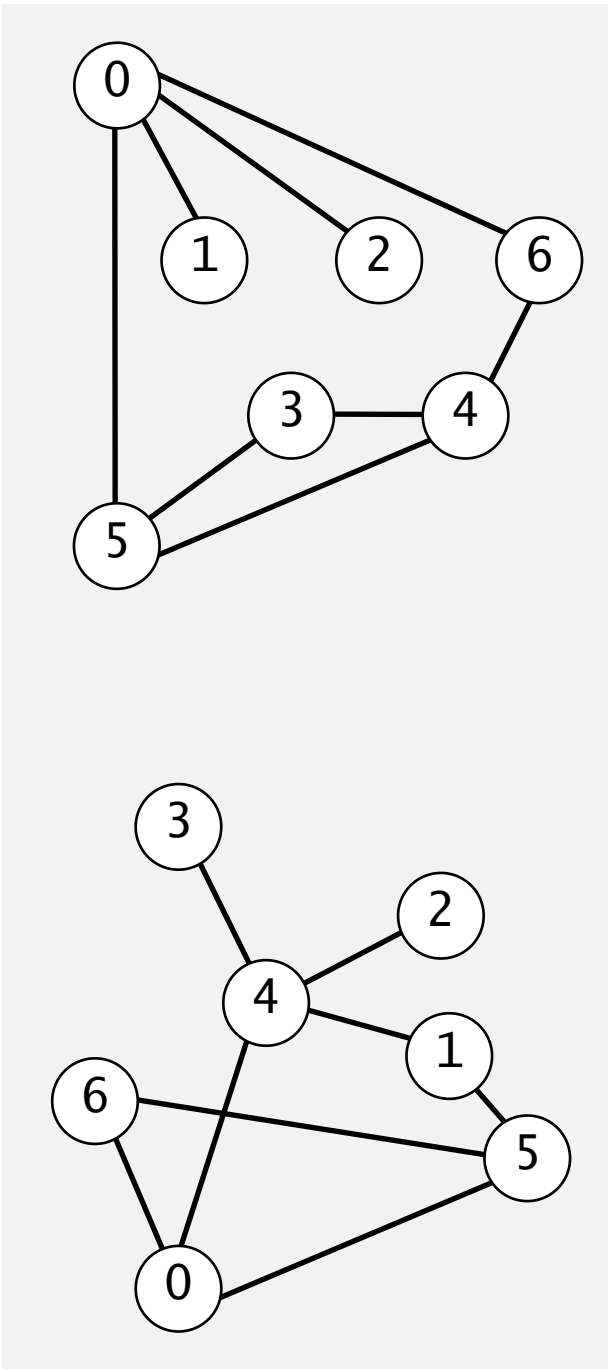
1.3.2 Find a cycle.

1.3.3 Find a closed walk that uses every edge exactly once. (Euler cycle)

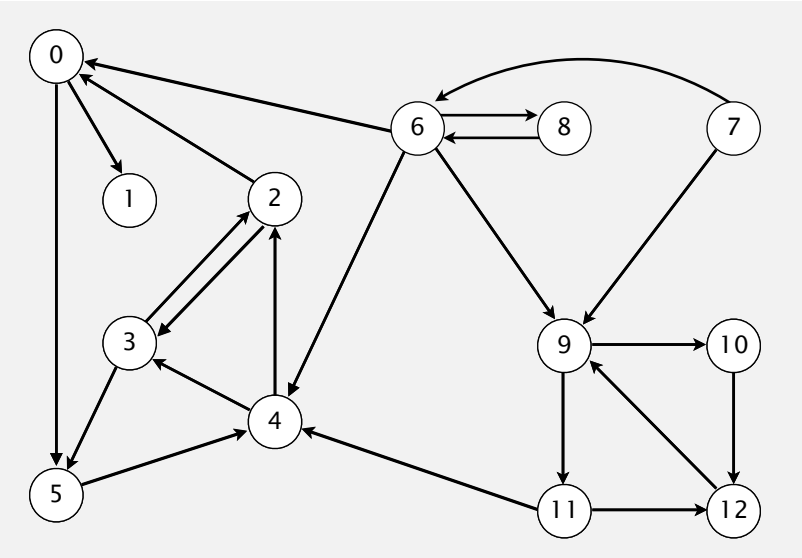
1.3.4 Find a cycle that visits every vertex exactly once. (Hamiltonian cycle)

1.3.5 Are two graphs identical except for vertex names? (see next page)

1.3.6 Lay out a graph in the plane without crossing edges? (planarity)


0
1
2
6
3
4
5
3
2
4
1
6
5
0

1.4 Depth-first search on digraphs

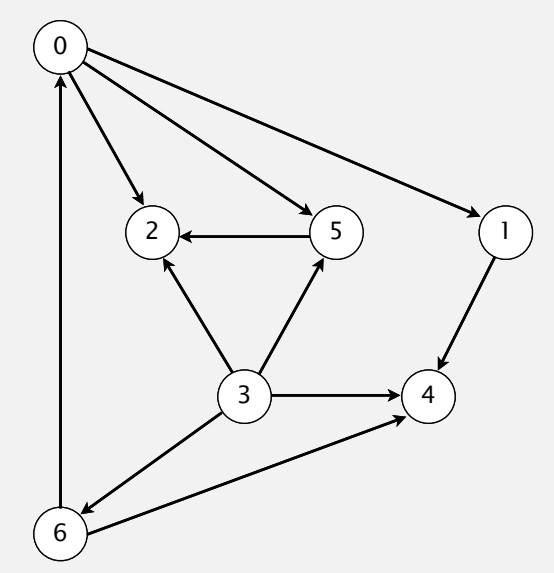


Trace DFSAll (page 228) on this:

Identify different types of edges (tree, forward, back, cross).

1.5 Topological sort

- Run DFS
- Return vertices in reverse postorder



1.6 Strong Components

Kosaraju-Sharir algorithm

Simple (but mysterious) algorithm for computing strong components.

- run DFS on $rev(G)$ to compute reverse postorder.
- run DFS on G, considering vertices in order given by first DFS.