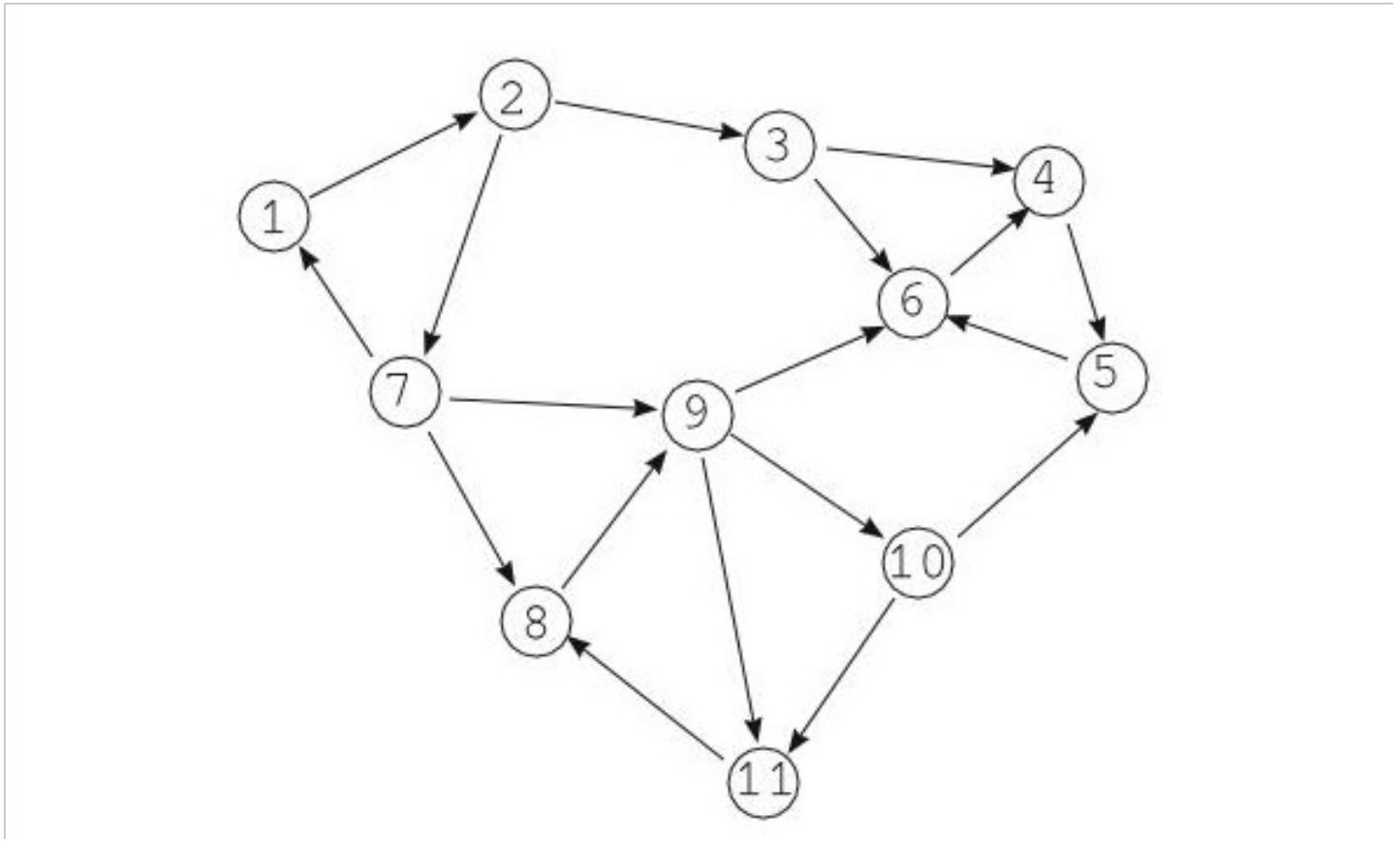
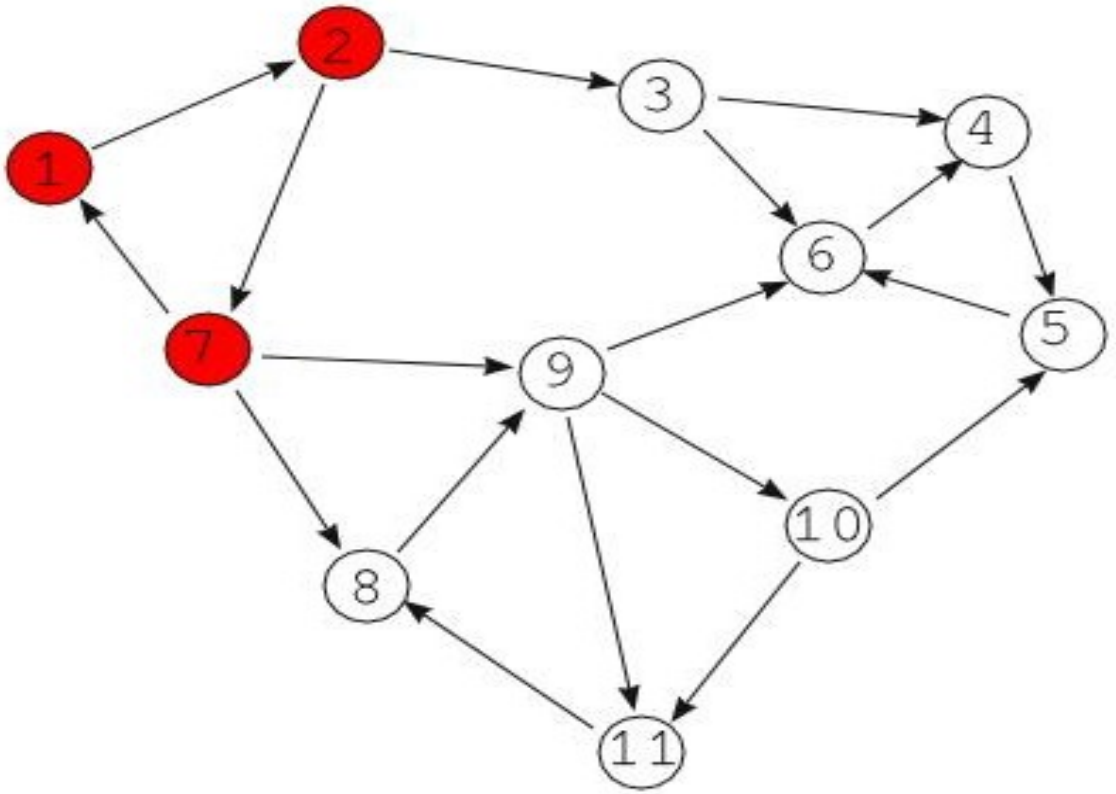
Parallel approach at 2g.

Key concept: directed graph



Key concept: chain

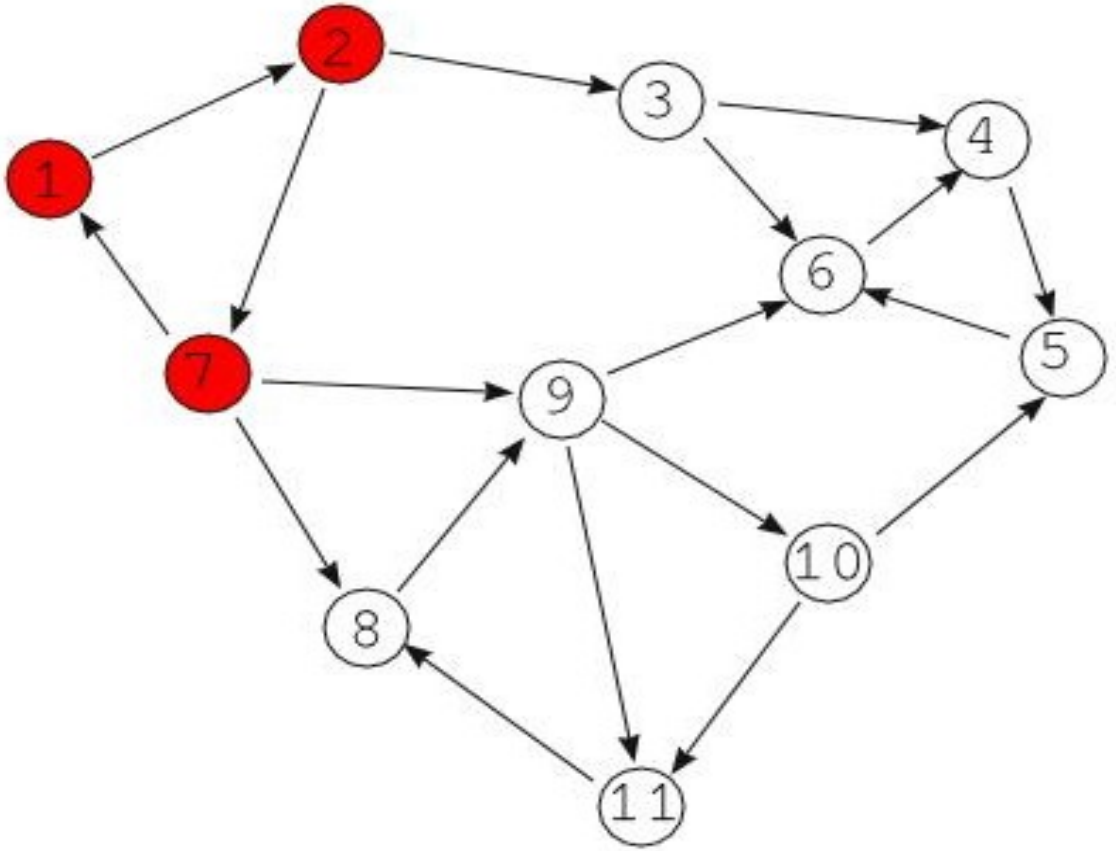
A path that starts in vertex A and finishes in vertex A, visiting every vertex on its way exactly once.



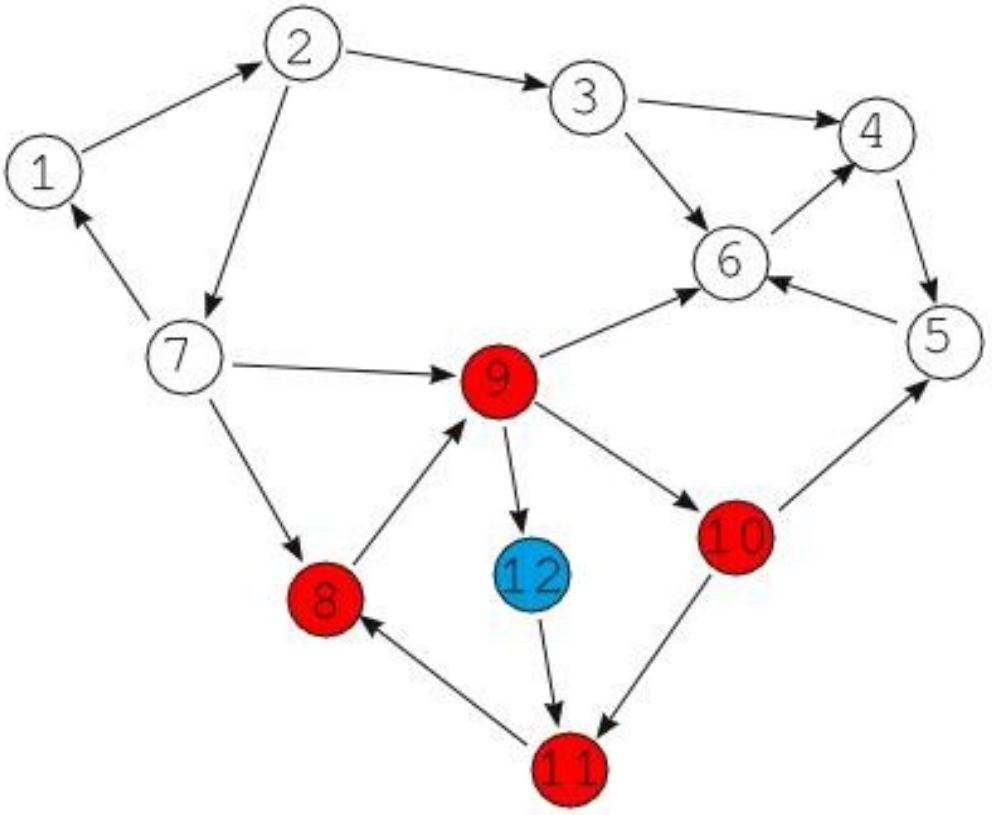
Simpler, more general problem

- Key concept: Strongly Connected Component.
- A strongly connected component is such a maximal set of vertices that for every vertex A and B there is a path from A to B and from B to A.

Can a chain be an SCC?



Why is SCC more general then a chain?



How are SCCs helpful?

- Can a chain live in two different SCCs?

  No – this would violate SCC's connectivity.
- How many SCC's are there? Depends on the problem, but Amazon Book Similarity Network had 91K and flickr 2005 crawl showed 0.28M SCCs.
- We can work separately on every SCC.

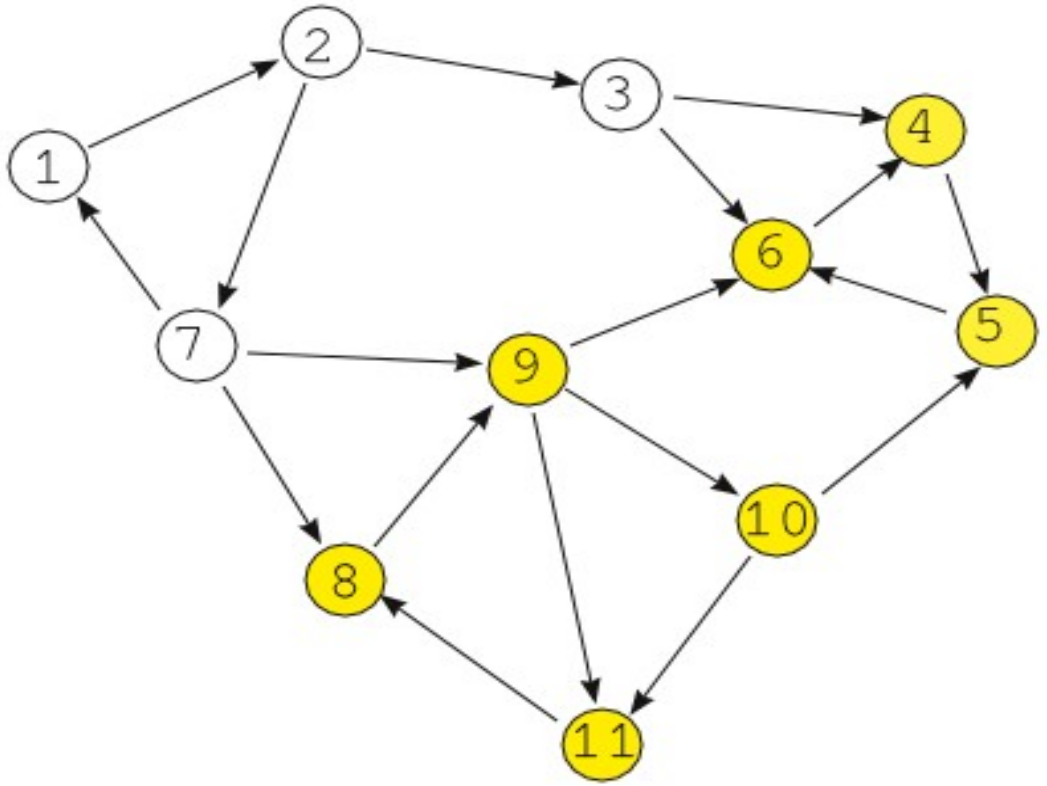
How to look for SCCs?

- Standard approach: Tarjan's algorithm, Kosaraju's algorithm. Good, linear speed, very hard to parallelise.

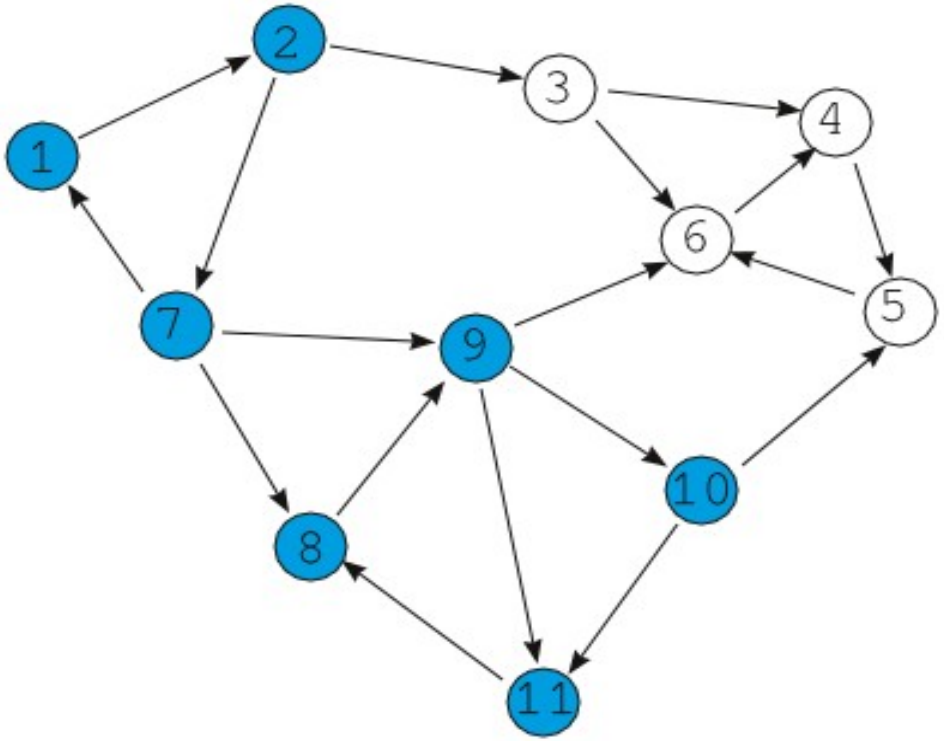
Coppersmith's 2005 algorithm.

1. Pick any node
2. Find all the ancestors of this node
3. Find all descendants of this node
4. All the nodes which are both ancestors and descendants create an SCC.
5. All other SCCs can be found either in remaining descendants, or remaining ancestors, or in remainding nodes. So repeat the procedure for these 3 sets.

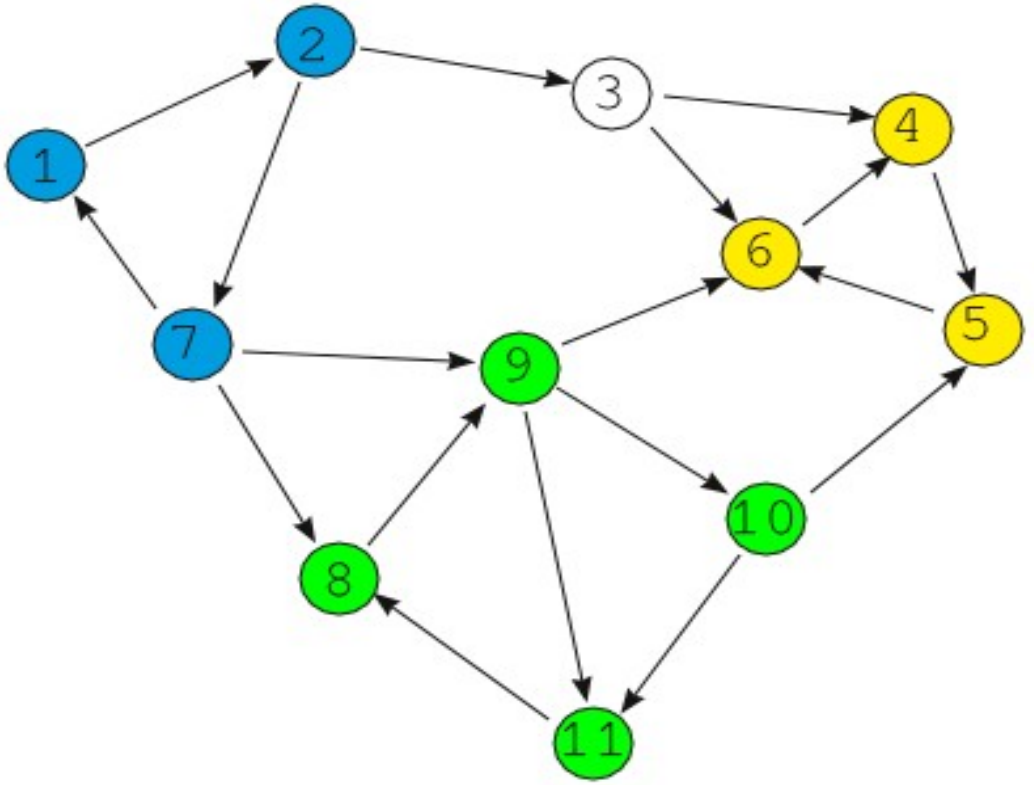
Let's find the descendant's of 9



Let's find the ancestors of 9



Let's merge both graph's



Fin

- Try to apply Coppersmith's algorithm starting from a different vertex, e.g. 6.
- During the workshops, if you want so, you can try to implement Coppersmit's algorithm in java. I will provide you Node.java, DataGraph.java and Kosaraju.java files. Kosaraju.java is to compare your results with Coppersmith's.