CSE525 Lec14
Graph: Reductions and DFS

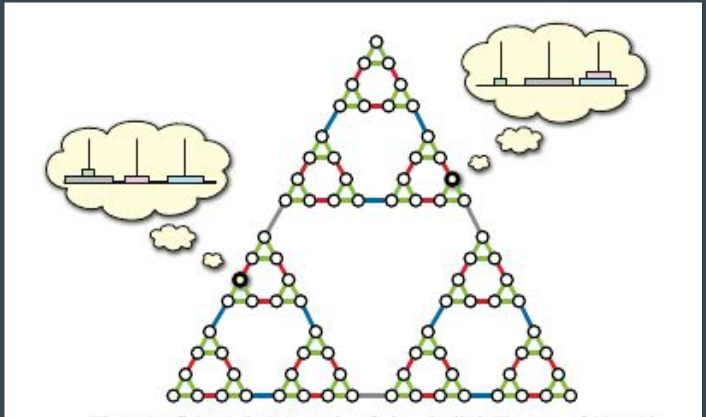
Debajyoti Bera (M20)

https://sites.google.com/a/iiitd.ac.in/cse525-m20

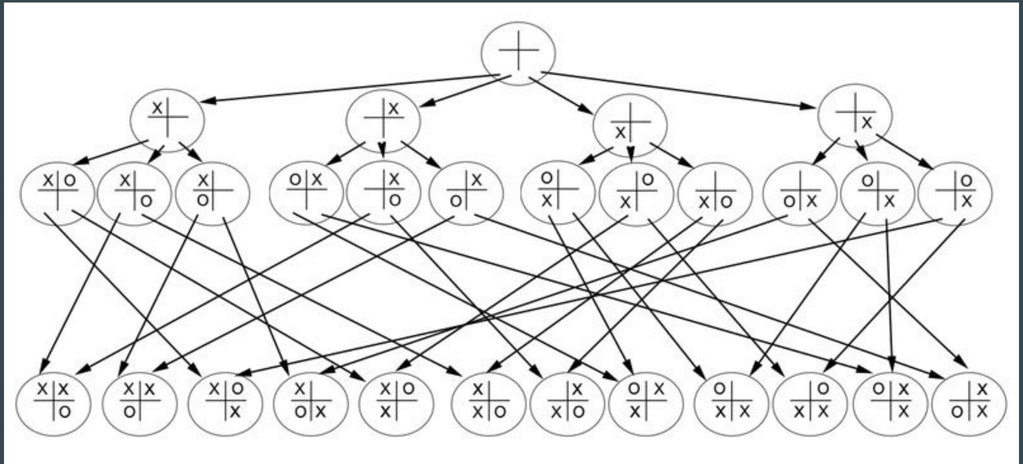
Configuration graph

Node: configuration

Edge(u -> v): configuration v can be obtained from configuration u



The configuration graph of the 4-disk Tower of Hanoi.



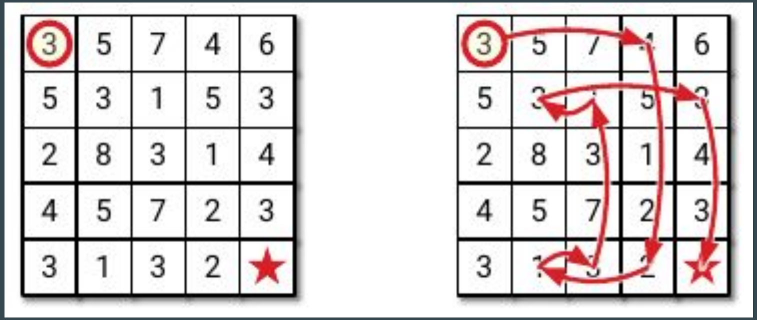
Reduce to graph traversal problems

Given a problem, show to reduce it to a graph traversal problem.

- How to construct a graph?
  - What do vertices represent? How many vertices are there?
  - What do edges represent (when would there be a u -> v edge)? How many edges are there?
  - What is the time necessary to construct the graph (in terms of the problem input size)?
- What traversal algorithm should be used on the graph?
  - What is the net complexity <u>in terms of the problem input size</u> (not that of the graph)?

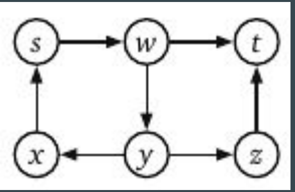
Applications

You are given n rectangular boxes of different lengths, breadths, heights. Find a way to determine if there is a way to pack each box into another so that we are left with only one box; note that only a smaller box can fit in a larger box.



Number maze: Move from top-left to bottom-right using the fewest number of moves.

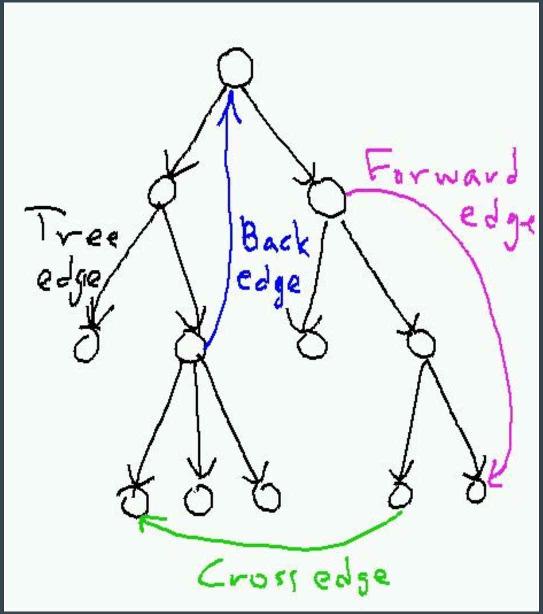
Paths of length 3k



Suppose you are given a directed graph G = (V, E) and two vertices s and t. Describe and analyze an algorithm to determine if there is a walk in G from s to t (possibly repeating vertices and/or edges) whose length is divisible by 3.

Map this to a traversal problem on some graph H. How does H look? What traversal to use on H? What is the complexity wrt. G?

DFS for directed graphs

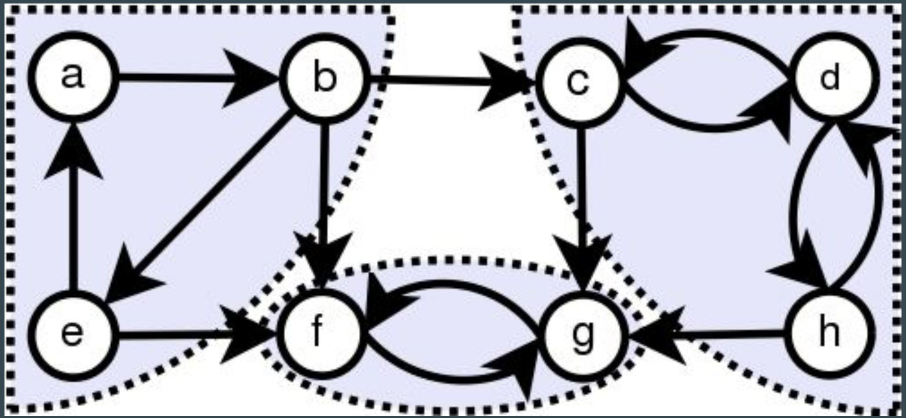


http://www.cs.yale.edu/homes/aspnes/pinewiki/DepthFirstSearch.html

Exercise: Show how to classify edges using pre & post times and parent/children information.

```
DFS(v):
    mark v
    PREVISIT(v)
    for each edge vw
        if w is unmarked
            parent(w) ← v
            DFS(w)
    POSTVISIT(v)
```

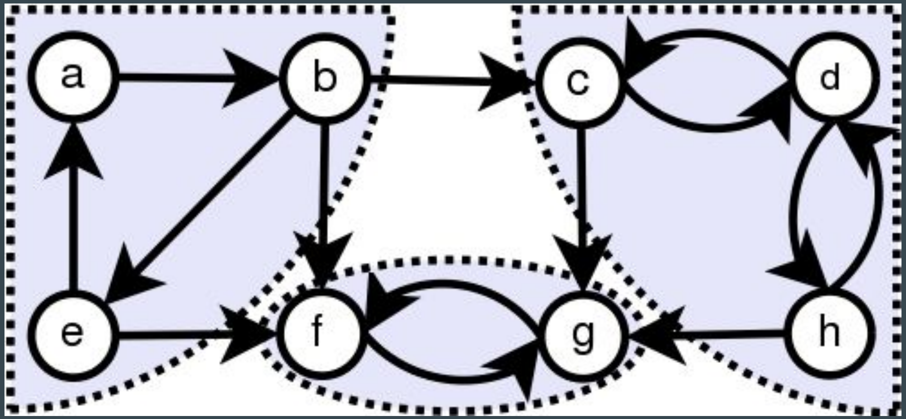
Kosaraju ('78) Sharir ('81) SCC



Source SCC = component with no incoming edge

Sink SCC = component with no outgoing edge

Kosaraju ('78) Sharir ('81) SCC



Source SCC = component with no incoming edge

Sink SCC = component with no outgoing edge

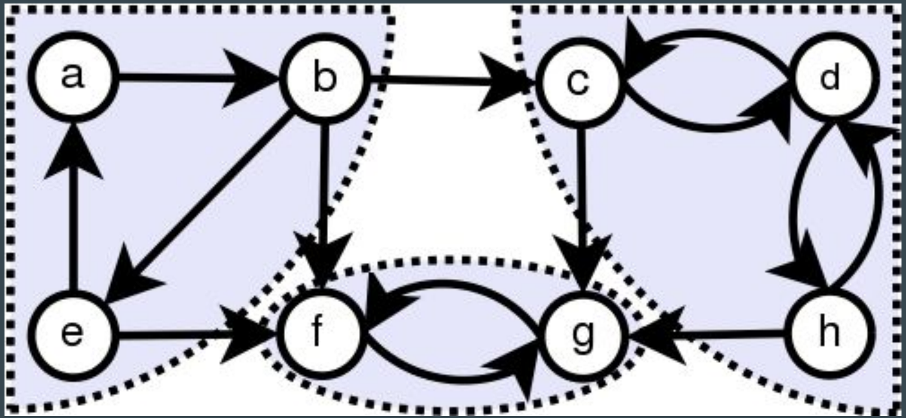
Component graph is acyclic.

Proof:
Let there be cycle, say among some of the components. Without loss of generality, let the cycle be among components C1, C2, C3, ... Ck.

Let u be some vertex in C1. There is an edge from some vertex in C1, say u1, to some vertex in C2. Since every vertex (including u1) in C1 is reachable from u, and u2 is reachable from u1, therefore, u2 is reachable from u. Since every vertex in C2 is reachable from u2, therefore, every vertex in C2 is reachable from u. There is an edge from some vertex in C2 to some vertex in C3.
Applying the same argument as above we get that every vertex in C3 is reachable from u. Continuing this for all the components in C4, C5, ..., we get that all the vertices in Ck is reachable from u.

Let uk from Ck have an edge to some w in C1. So, u has a path to uk. Furthermore, uk has path to w and w has to path to u => uk has a path to u. Thus, u and uk have a path to each other.
So uk must belong to SCC(u) which contradicts the fact that SCC(u) is different from SCC(uk).

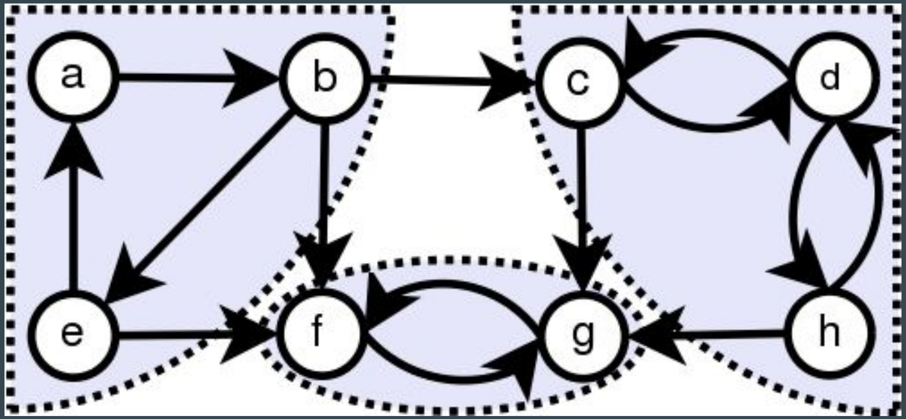
Kosaraju ('78) Sharir ('81) SCC



Lemma: Let v be the vertex to finish last in DFS. Then, v belongs to a source SCC.

Proof: Suppose not, so, let u -> w and w is in the same component as v. There are two cases (a) pre(u) < pre(v), (b) pre(u) > pre(v).

Kosaraju ('78) Sharir ('81) SCC



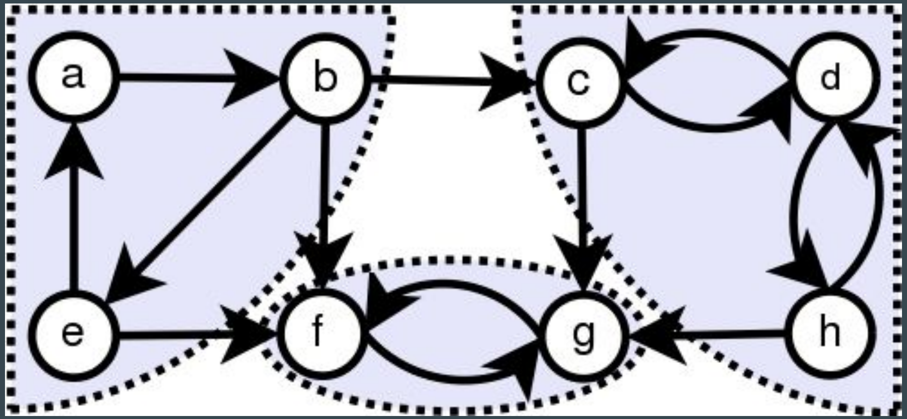
Lemma: If v belongs to a sink SCC, then SCC(v) = all vertices reachable from v.

Proof of 1st part: If u is in SCC(v), then by definition of SCC, u has a path to and from v.

Proof of 2nd part: (Proof by contradiction) Suppose v has a path to u and u is not in the SCC(v), so in a different SCC.

Consider that edges on the path from v to u and let e denote the edge that _first_ goes out of SCC(v), probably to SCC(u) or some other SCC. This edge indicates that there is an outgoing edge from SCC(v) and contradicts that fact that SCC(v) is a sink SCC.

Kosaraju ('78) Sharir ('81) SCC



Lemma: A sink SCC in G is a source SCC in rev(G).

Proof: Let C be a sink SCC in G. So, it has no edges going out from any vertex in C to a vertex in any other component. In rev(G), there would be no edges coming in from a vertex in any other component to any vertex in C. This is same as the condition for C to be a source SCC in rev(G).

Algorithm for finding all SCC

Lemma: Let v be the vertex to finish last in DFS. Then, v belongs to a source SCC.

Lemma: If v belongs to a sink SCC, then SCC(v) = vertices reachable from v.

Lemma: A sink SCC in G is a source SCC in rev(G).

Q: How can we find one source SCC?

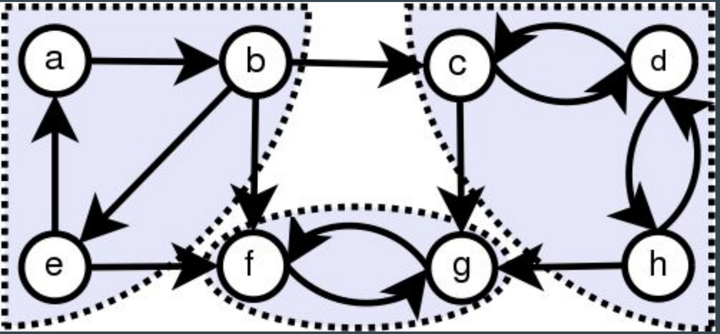
Q: How can we find one sink SCC?

Q: How can we find all SCCs?

Hint: Removing a source SCC or sink SCC does not change other SCCs.

Hint: Reverse graph has same SCCs.

Kosaraju ('78) Sharir ('81) SCC



2-DFS O(n+m) algorithm

Run DFS: L is ordered according to finish time

Reverse the edges of the graph: Grev

On Grev DFS(last finished yet-unmarked vertex in L)

    Output everything discovered as an SCC
    and mark all those that are output
    // This is source SCC of the current graph
    // Ignore/implicitly remove this SCC
    Goto: DFS(last finished... in L)