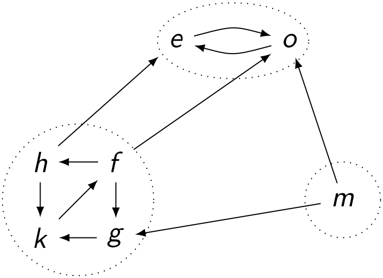
CSCB63 – Design and Analysis of Data Structures

Anya Tafliovich[1]

[1] based on notes by Anna Bretscher and Albert Lai

strongly connected component (SCC)

Strongly connected component (SCC): maximal subset of vertices reachable from each other.

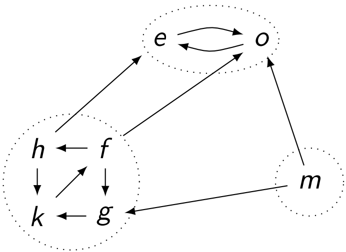


Three strongly connected components: $\{e, o\}$, $\{f, g, h, k\}$, $\{m\}$.

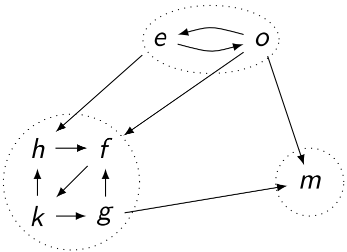
transposed graph

Transpose of G (G^{T}) means a graph with the same vertices as G and the edges are the reverse of G's.
Why is it called "transpose"? matrix representation is transpose

G:



G^{T}:



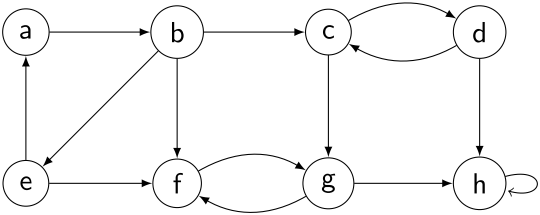
G^{T} has the same strongly connected components as G's.
How much time to compute adjacency lists of G^{T}: $O(|V| + |E|)$

computing SCCs: idea

1. DFS on G
   - visit all vertices
   - store all finish times
   - accumulate vertices in reverse finish-time order
2. Compute adjacency lists of G^{T}
3. DFS on G^{T}
   - use the above order to pick start/restart vertices
4. Each tree found has the vertices of one strongly connected component.

Total time: $O(|V| + |E|)$

computing SCCs: example



computing SCCs: DFS(G)

```
mark all vertices white
time := 0
R := []
for each vertex v:
  if v is white:
     DFS-visit(v)

DFS-visit(u):
  mark u gray
  for each v in adjacency list of u:
     if v is white:
       DFS-visit(v)
  mark u black
  finish-time(u) := ++time
  insert u at the front of R
```

computing SCCs: DFS(G^{T})

```
mark all vertices white
for each vertex v in R's order:
  if v is white:
     SCC := []
     DFS-visit2(v)
     output/record SCC

DFS-visit2(u):
  add u to SCC
  mark u gray
  for each v in u's adjacency list in G^T:
    if v is white:
      DFS-visit2(v)
  mark u black
```

computing SCC: proof

Prove: each depth-first tree found in DFS(G^T) is a SCC.

Let C and C' be distinct SCC's of G.
Define $max\_finish(C) = \max\{finish\_time(u) \mid u \in C\}$.

Proof steps:

- If some vertex $u \in C$ has an edge in G to some $v \in C'$, then $max\_finish(C) > max\_finish(C')$.
  - C discovered earlier: will go from C into C' (via (u, v) or otherwise), then finish C', then back to finish C.
  - C' discovered earlier: C' finished without visiting C because no path from C' to C: Why no such path?
- In G^{T}, if some vertex $v \in C'$ has an edge to some $u \in C$, then $max\_finish(C) > max\_finish(C')$.

computing SCC: proof

Proof steps (continued):

- In G^{T}, if some vertex $v \in C'$ has an edge to some $u \in C$, then $max\_finish(C) > max\_finish(C')$.
- If $max\_finish(C) > max\_finish(C')$, then in G^{T} no edge from C to C'.
- DFS(G^T):
    - start vertex $s \in C$ with largest $max\_finish(C)$ of all SCCs
    - visit all vertices reachable from s
    - G^T has no edge from C to another SCC C'
    - never visit another SCC C'
    - then select another start vertex $s_2 \in C_2$ with largest $max\_finish(C)$ of all SCCs except for C
    - ...

Complete proof: textbook / exercise: by induction on the number of depth-first trees found.