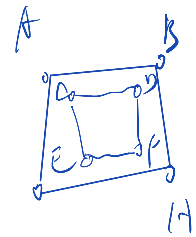
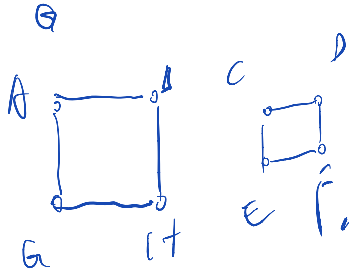
connected component.

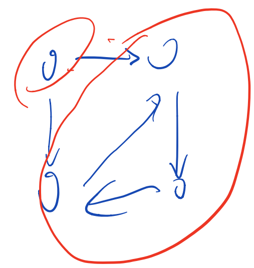




strongly connected component.

↓

directed graph.



CONNECTED COMPONENTS USING DFS.

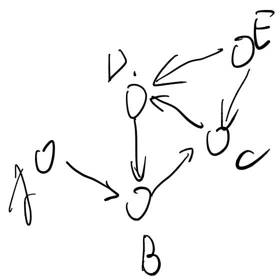
```
def cc
    c = 0
    for each v in V
        if not visited[v]
            c++
            explore(v, c)
```

STRONGLY CONNECTED COMPONENTS USING DFS.

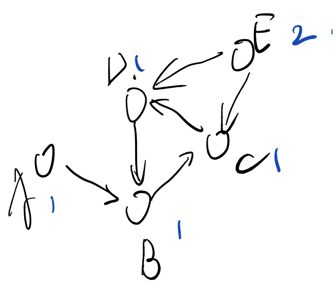
G^R = G with all edges reversed.

L = output of linearization algorithm on G^R.

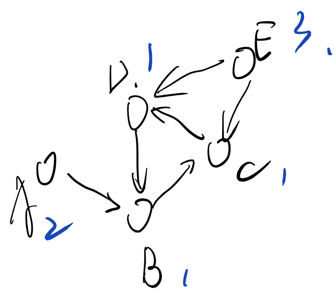
Run CC algorithm on G, enumerating vertices as in L



starting from A.

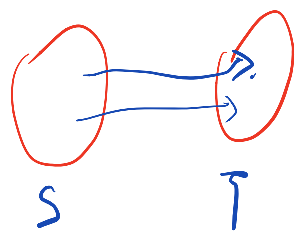


starting from B



G: directed graph.

L list of vertices in reverse order of termination of explore (v)



Let S, T be strongly cc of G with ≥1 edges from S to T.

Then first vertex of S in L comes before the first vertex of G.

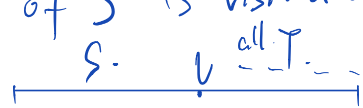
First vertex to be discovered in $S \cup T$ is $v \in S$.

Then all of S, all of T are discovered inside of explore(v)

explore (v) terminates after all of T.

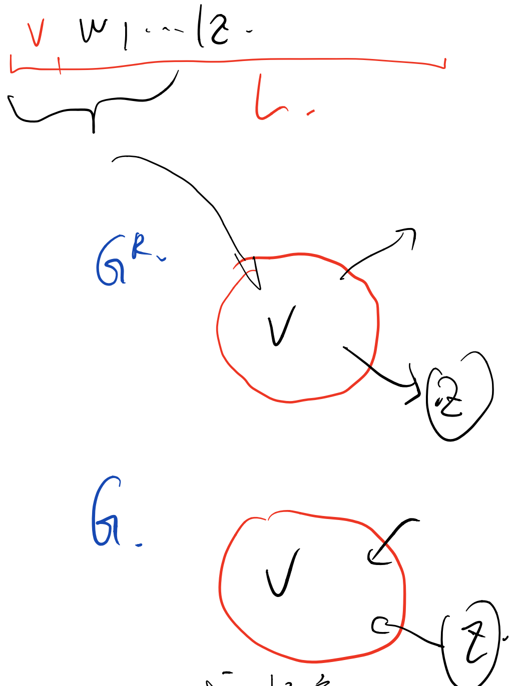
. . . - - $\forall \in T$.

explore(v) terminates discovering all of T but at time in which no vertex of S is visited.



v comes before all of T in L.   so all of S before all of T

> Let S, T be strongly cc of G with ≥ 1 edges from S to T.
> Then first vertex of S in L comes before the first vertex of G.

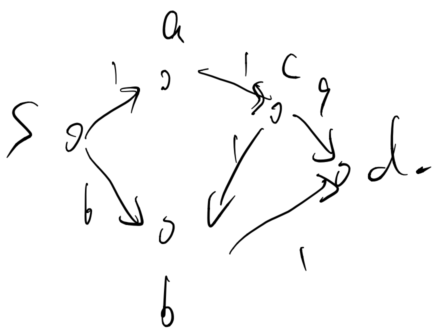


(no edge comes in)

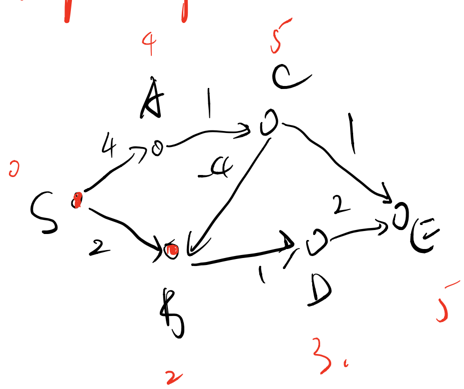
1. 反向 DFS 找顺序.

2. 根据顺序正向 DFS.

Shortest path.



Negative weight edges.



using dij ?

S
B.
D.
A.
E
C

Bellman-Ford.

dist = array indexed by V initialized to ∞

prec = array indexed by V initialized to $\perp$

for $i = 1$ to $|V|-1$

for each v in $V - \{s\}$

for each edge (u, v):

$O(E)$

if $dist[u] + l(u,v) < dist[v]$

$dist[v] = dist[u] + l[u,v]$

$prec[v] = u$.

$O(|V||E|)$.

correctness: At step l

for every v, $dist[v] \le$ length of shortest path

from s to v that

uses $\le l$ edges.