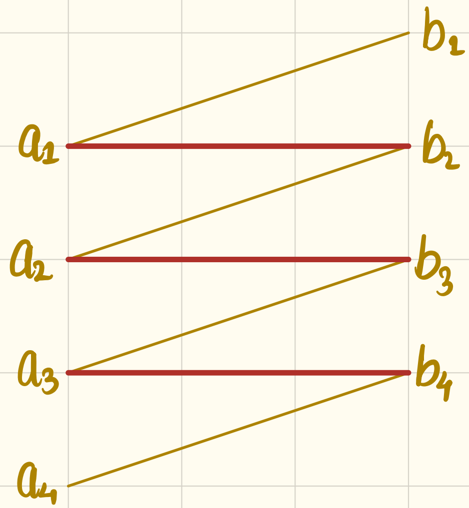
03 Apr 2025 – Algorithms – Week 13

Diameter.

Claim $\forall x \in V$,

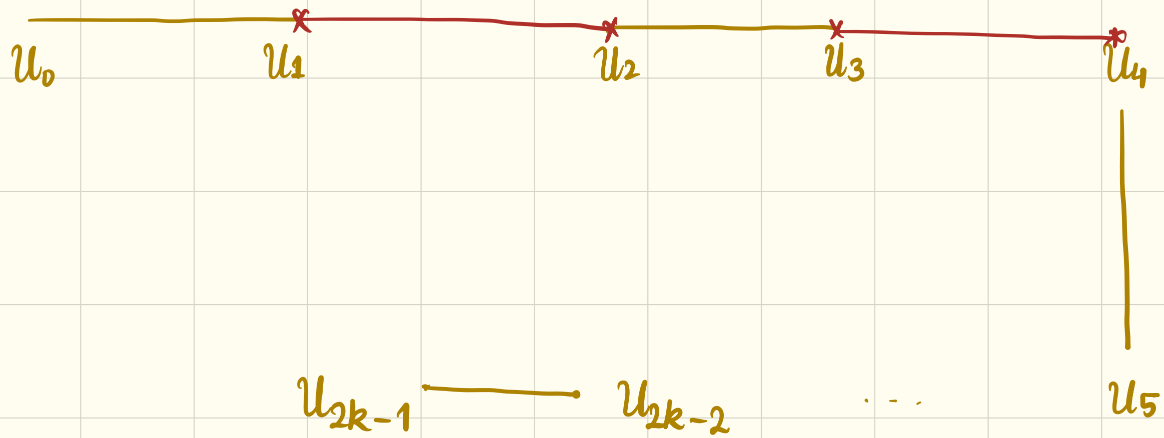
$$ecc(x) = \max \{ d(x,u), d(x,v) \}$$

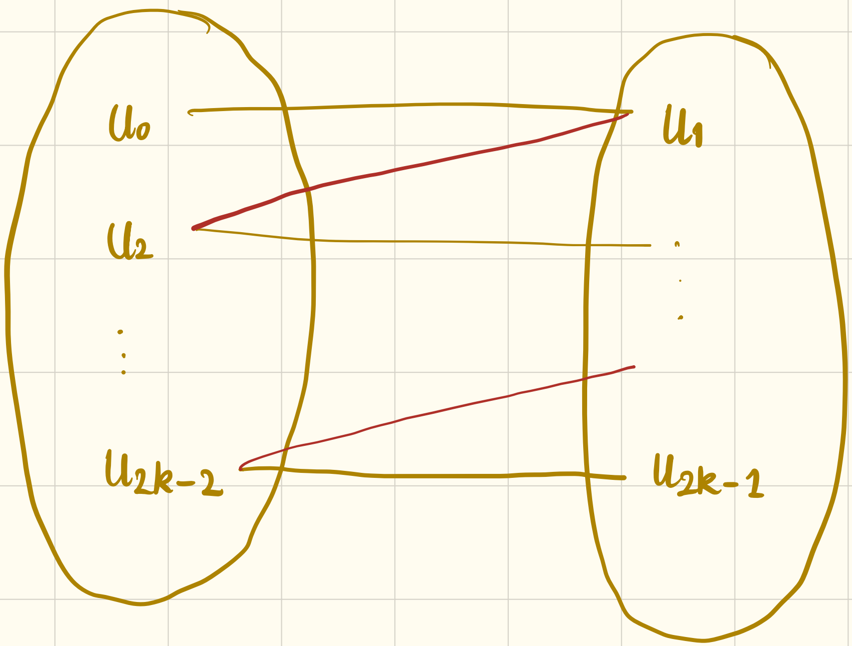
<u>Maximum matching problem</u> (in bipartite graphs)



Start with an arbitrary maximal matching.

* Start with an unmatched vertex u.





$k-1$ red edges

k matched edges

If $\exists$ a path $u_0 u_1 \dots u_{2k-2} u_{2k-1}$ such that $u_1 u_2$, $u_3 u_4$, $\dots$ $u_{2k-2} u_{2k-1}$ are in M (current matching).

$$M' = M \setminus \{ u_1 u_2, \; u_3 u_2, \; \dots \}$$
$$\cup \{ u_0 u_1, \; \dots, \; \}$$

Using a BFS from every unmatched vertex u_0, check if such an alternating path exists.

If such a path does not exist $\rightarrow$ $\underset{\text{current matching}}{M}$ is a maximum matching.

① DFS – recap: Topological Sorting

Application: Strongly - connected components.

② Shortest path algorithms weighted graphs.

→ single source

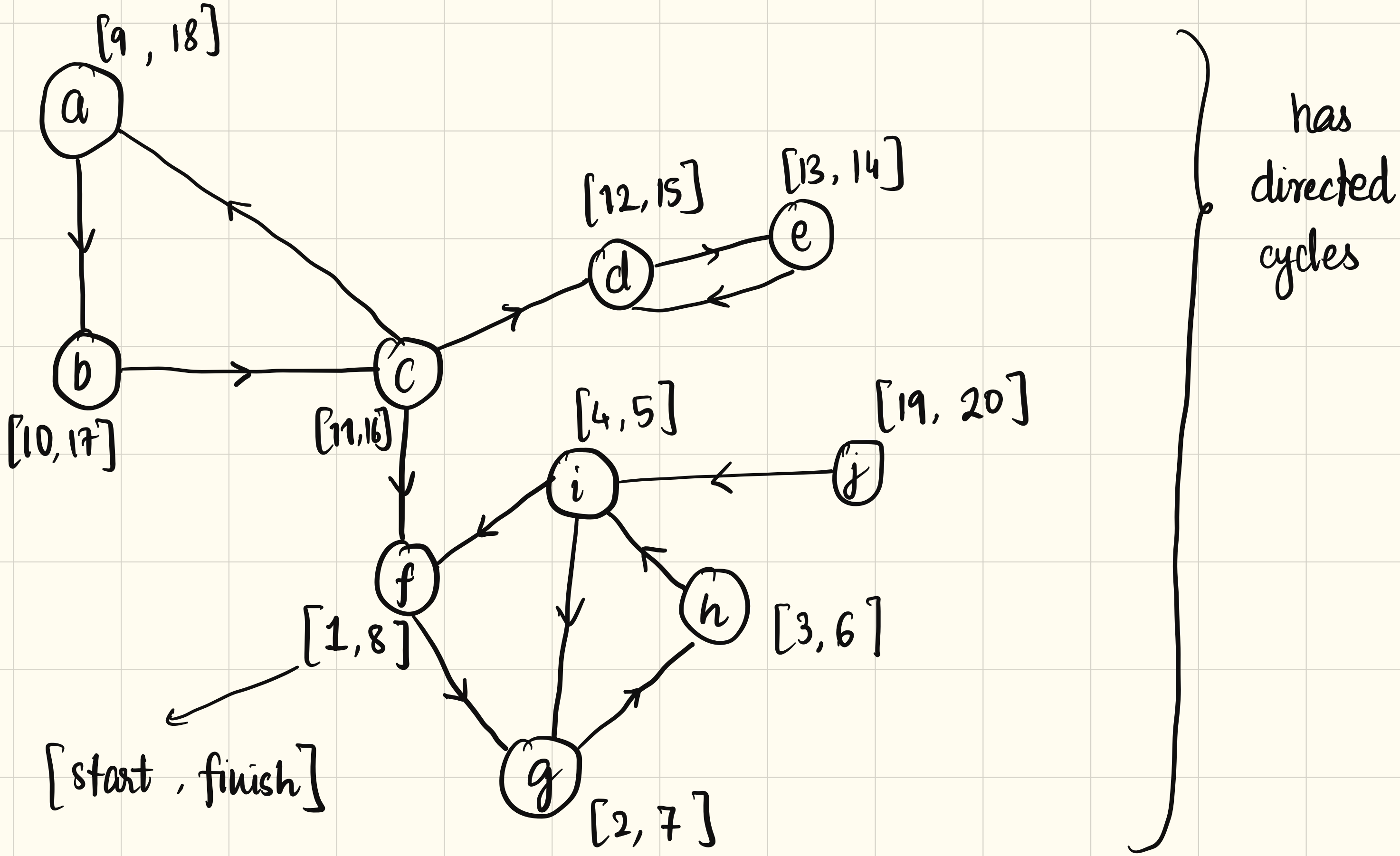
→ all pairs

③ Maximum flow

DFS(G)

For $v \in V$, DFS(G, V)

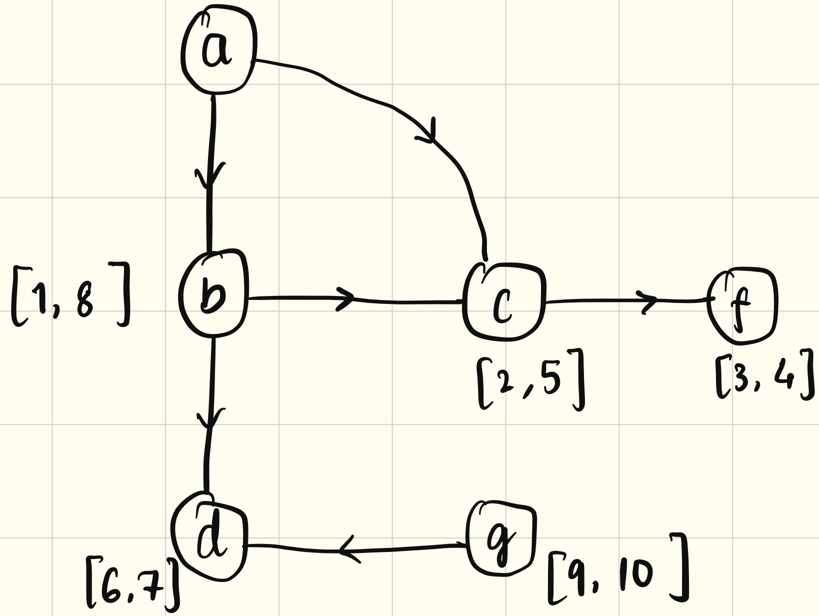
if v is not explored



Adjacency lists in alphabetical order.

$V = \{ f, a, b, c, g, i, j, h, e, d \}$

Directed acyclic graph
(DAG)

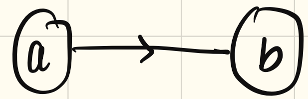


A topological ordering of V is a sequence $V_1, \ldots, V_n$ such that $(v_i, v_j) \in E$

$$\Rightarrow \quad i < j$$

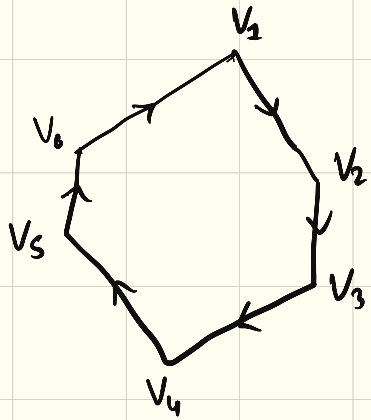
Topological sorting:

think of vertices as tasks / processes



b has a as its prerequisite

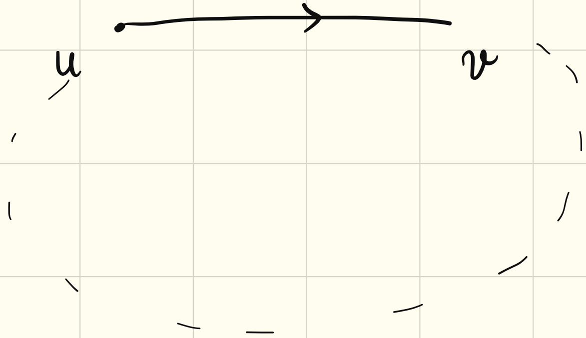
Directed cycle $\Rightarrow$ no such sequence



$$\boxed{V_1 < V_2 < V_3 < V_4 < V_5 < V_1}$$

Reverse order of finish times $\longrightarrow$ topological ordering

$$a \longrightarrow g \longrightarrow b \longrightarrow d \longrightarrow c \longrightarrow f$$



If $(u, v) \in E$, then

$$f[u] > f[v]$$

Start exploring u first $\Longrightarrow$ v finishes,
traceback to finish you
finish all children tasks first.

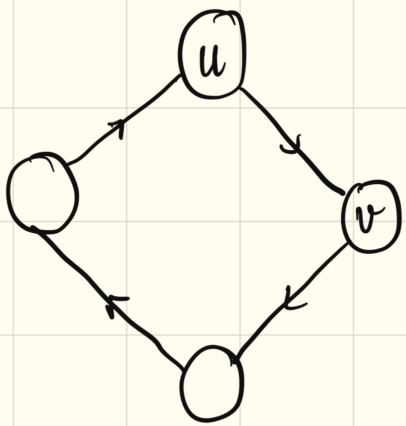
Start exploring v first $\Longrightarrow$ finish v
Explore u later

$v_1, \; v_2, \; \dots, \; v_n$

$$f[v_1] > f[v_2] > \dots > f[v_n]$$

suppose $\quad v_i \longleftarrow v_j \qquad f[v_j] > f[v_i]$

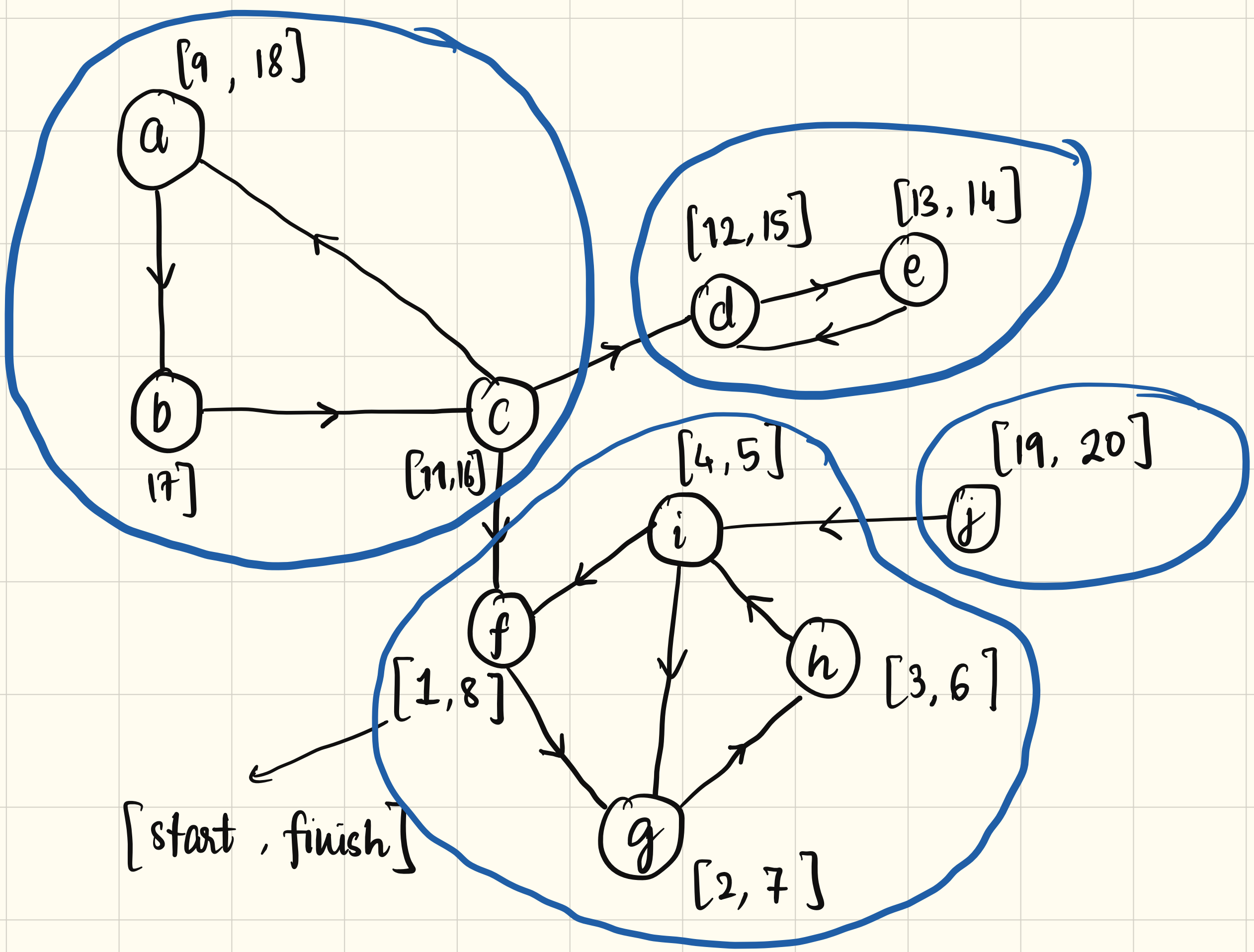
$\Rightarrow\Leftarrow$

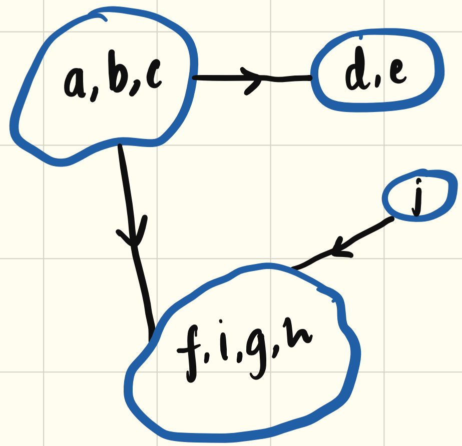


Schedule

$u, \; v, \; z, \; w$

in parallel





DAG

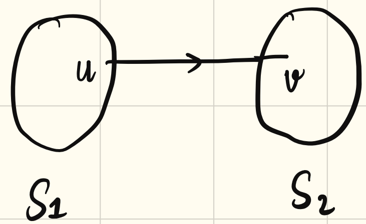
Strongly - connected component

A set $S \subseteq V$:

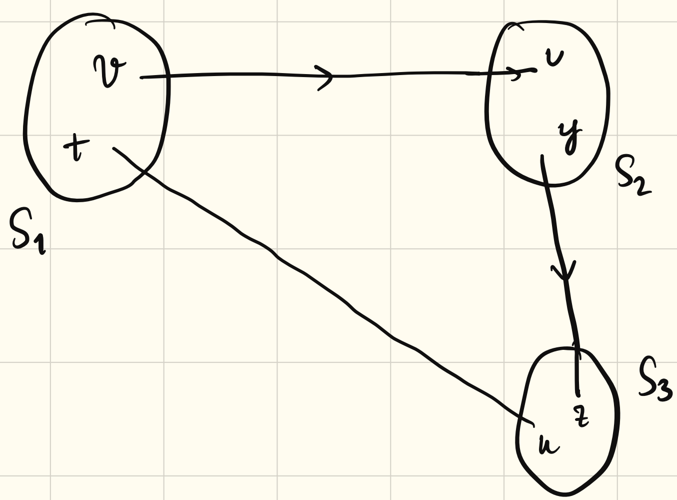
(i) $\forall\ u, v \in S, \ \exists\ u \to v$ path in S

(ii) S is maximal (adding any vertex should break (i))

Block Graph : SCCs of G

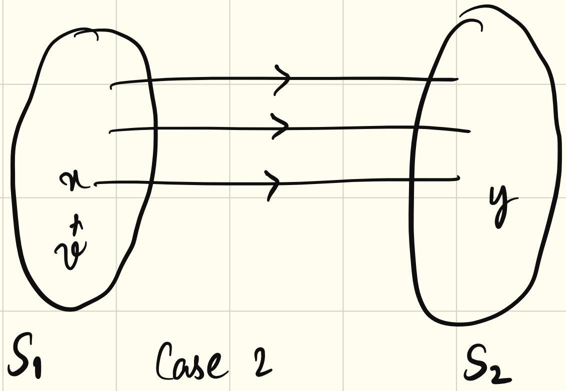


Block graph must b DAG.



$S_1 \cup S_2 \cup S_3$ is a bigger SCC.
(every pair of (u, v) $\in S_1 \cup S_2 \cup S_3$, path exists from u to v)

$\Rightarrow\Leftarrow$



The vertex which finishes last is in S_1

Case 1: Start the DFS in S_2

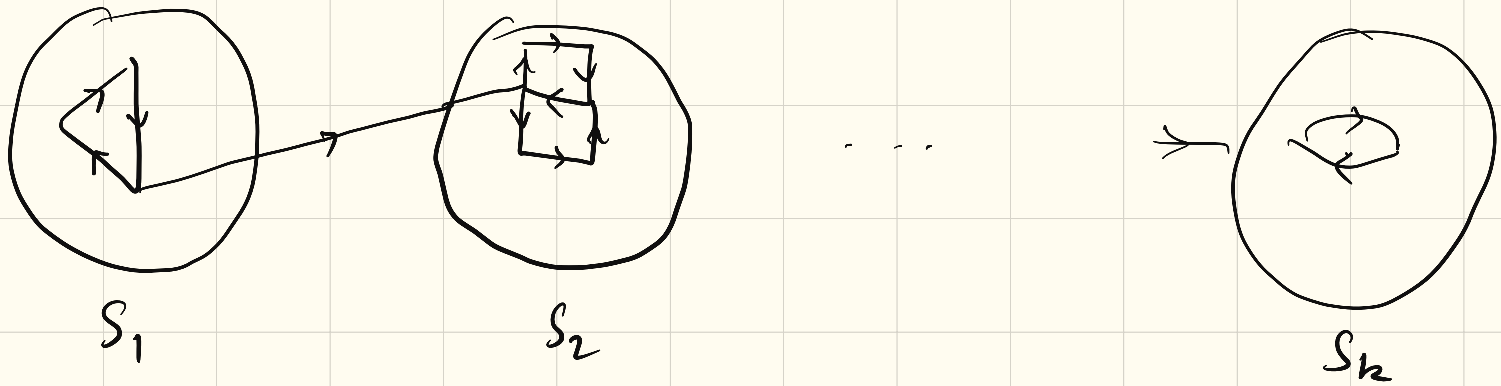
S_2 is explored completely.

S_1 is explored later.

Case 2: S_1 is explored first.

Block graph

G :



Vertex that finishes the last will be a vertex in S_1.