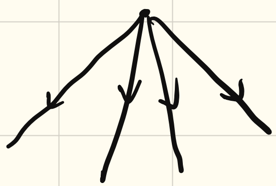
07 Apr 2025 - Algorithms - Week 14

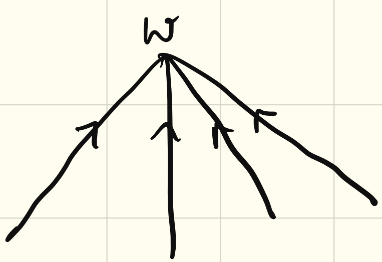
→ Recap

Source vertex : A vertex v : $\nexists\ (u, v) \in E$

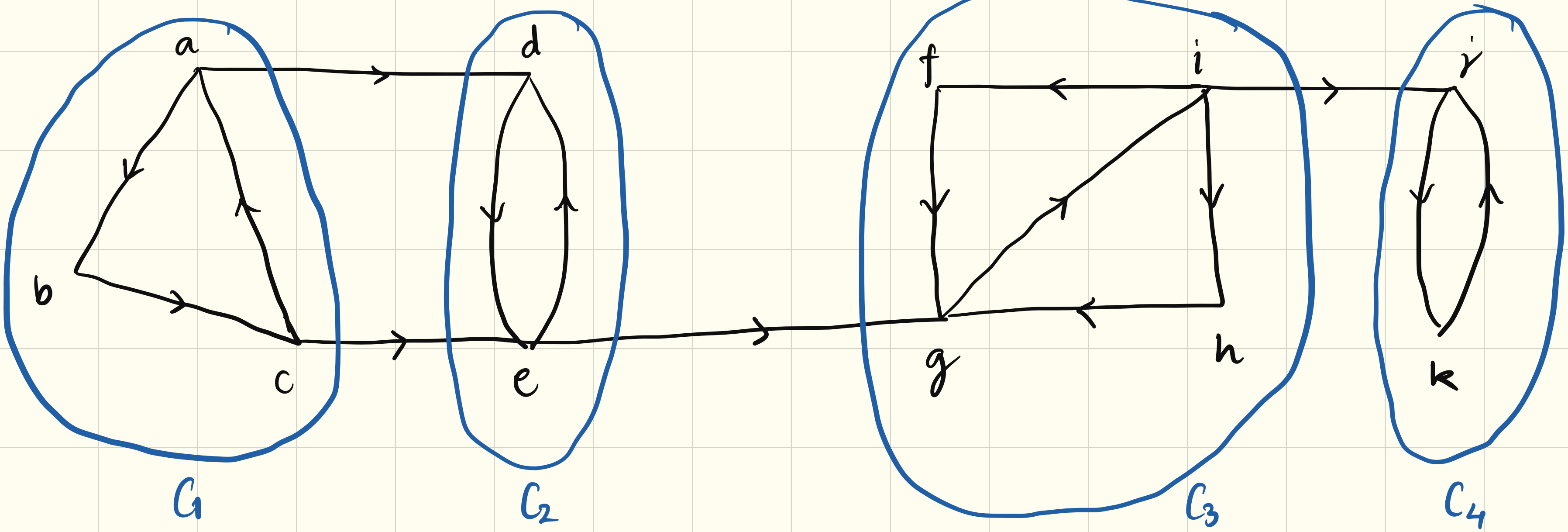
v — source vertex



Sink vertex :



A vertex w : $\nexists\ (w, v) \in E$



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

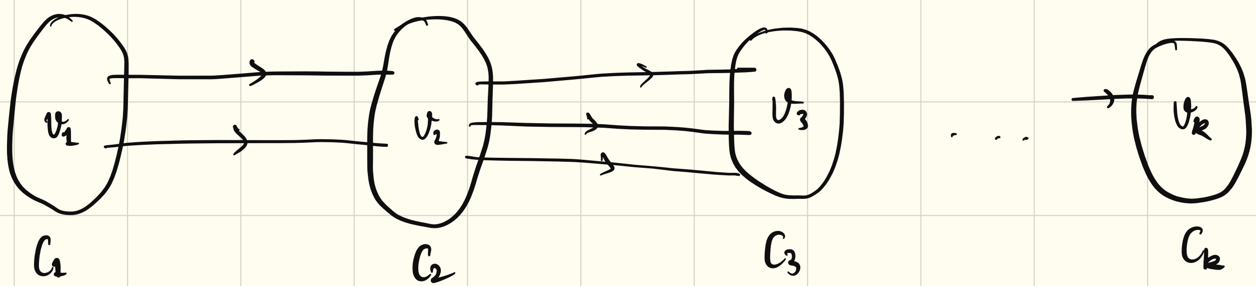
$DFS(G, i)$: $C_3 \cup C_4$

$DFS(G, d)$: C_2

$DFS(G, a)$: C_1

$$\begin{array}{ccccc} k, & g, & e, & a, & \ldots \\ \downarrow & \downarrow & \downarrow & \downarrow & \\ C_4 & C_3 & C_2 & C_1 & \end{array}$$

Want: sequence of components like k, g, e, a $\longrightarrow$ each vertex DFS $\longrightarrow$ only one component is discovered.

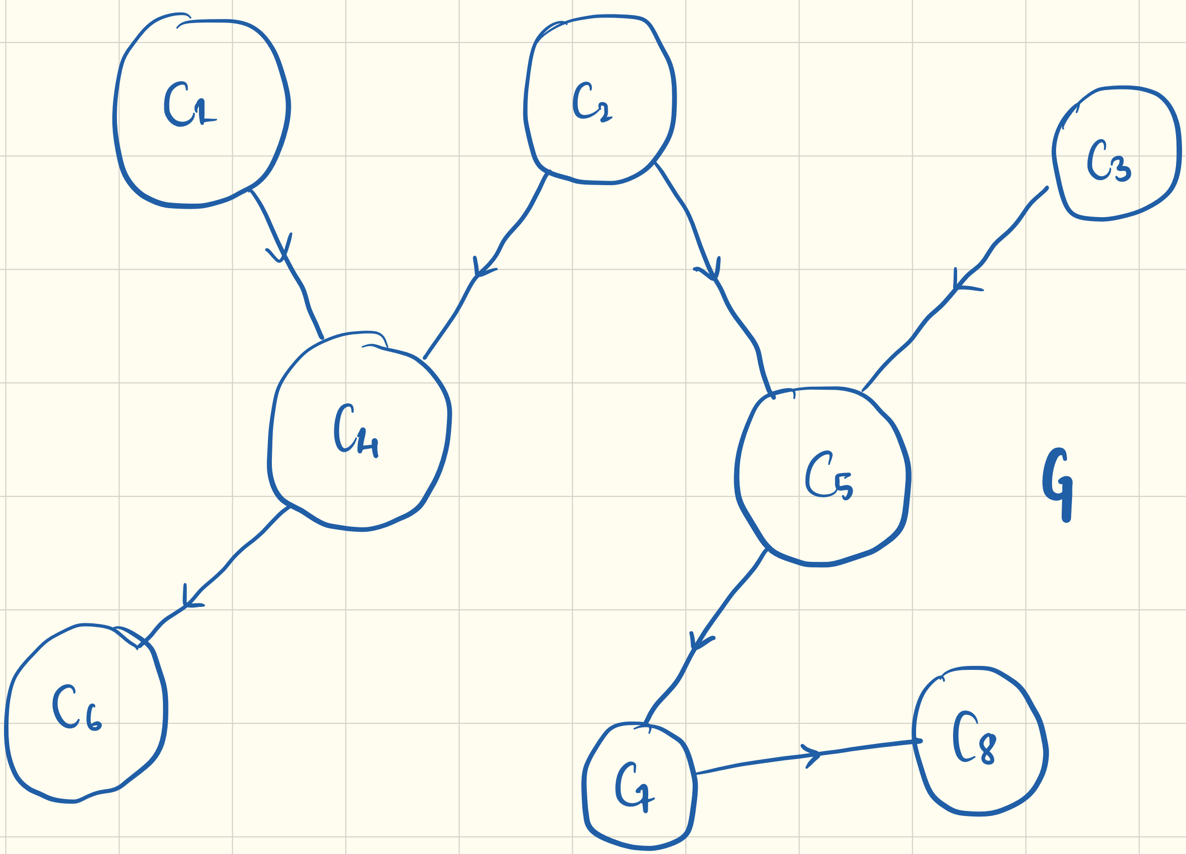


$$\begin{array}{ccccc} v_k, & v_{k-1}, & \ldots, & v_1 \\ \downarrow & \downarrow & & \\ C_k & C_{k-1} & & \end{array}$$

Goal: Find a vertex ordering : $v_1, \dots, v_n$:

for 1 to n

DFS (G, v_i) $\longrightarrow$ discover exactly one SCC per BFS.



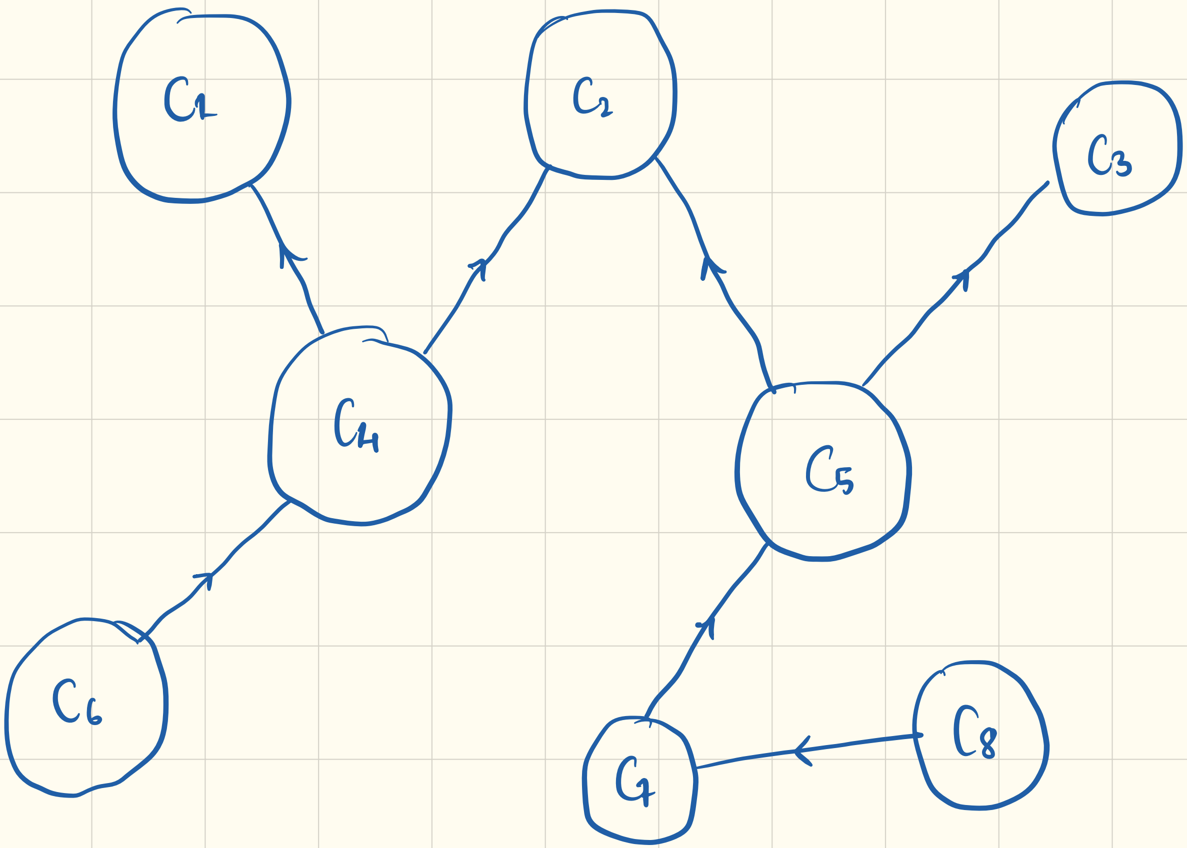
Sink component

First vertex should belong to C_6 or C_8

$$
\begin{array}{cccccccc}
v_6, & v_8, & v_4, & v_7, & v_5, & v_2, & v_1, & v_3 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
C_6 & C_8 & C_4 & C_7 & C_5 & C_2 & C_1 & C_3
\end{array}
$$

Want: a sequence of vertices in which the first vertex is from a sink component.

Observation: In a DFS, the vertex to finish last is from a source component.



G^T

1. Find $G^T(V, E^T)$ $\therefore$ $E^T = \{(v, u) : (u, v) \in E\}$
2. Do DFS(G^T) and sort vertices in ↓ order of finish times.
3. DFS(G) in order (step 2)

$\Big\}\; O(|V| + |E|)$

The Single Source Shortest Path Problem

→ Slides

→ Shortest path need not have the fewest no. of edges.

→ Google Maps

→ Negative edges

→ Negative cycles need to be handled.

Dijkstra's Algorithm

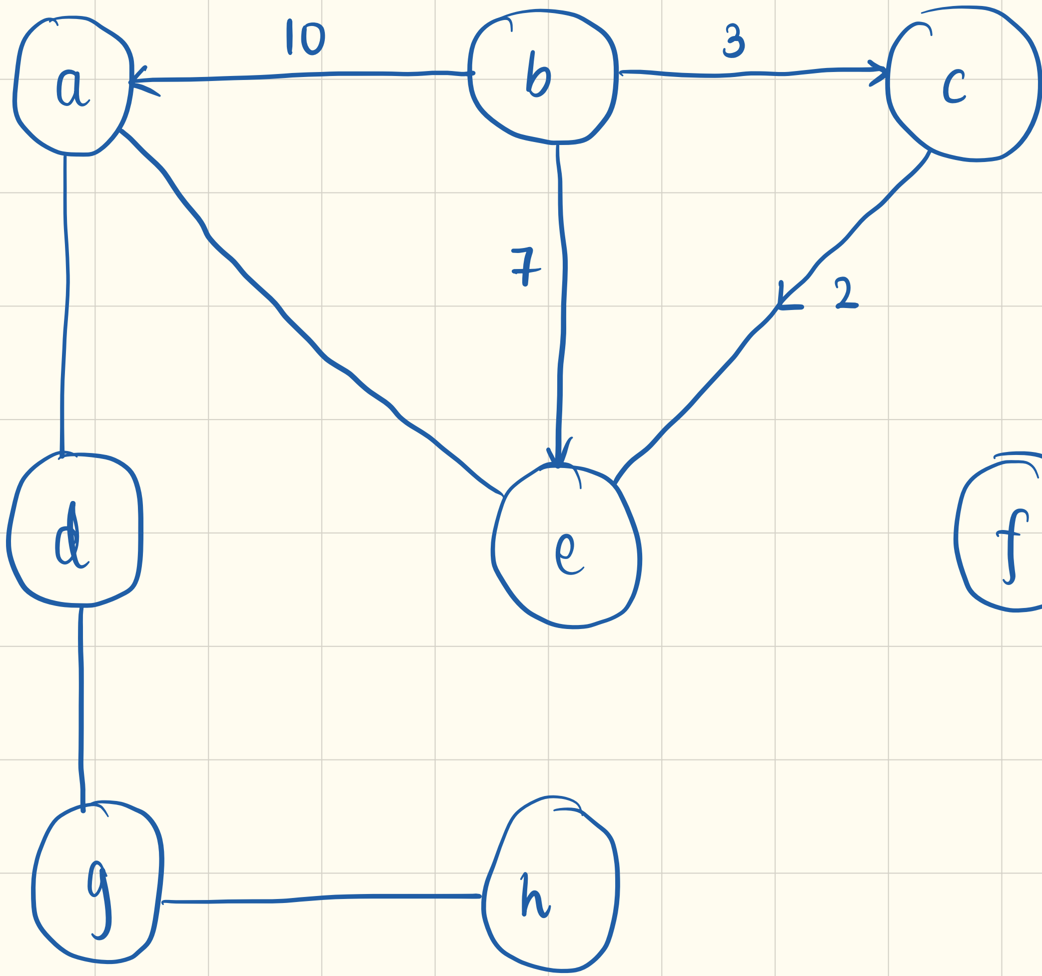
$d[s] = 0$

$d[v] = \infty \quad \forall \; v \neq s$

optimal cost

Maintain a set S such that $\forall \; v \in S, \; d[v] = d[s, v]$

Add one vertex to S at each step



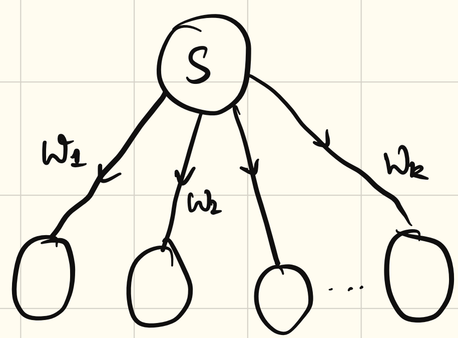
Initial $S = \{b\}$

Step 1:

$S = \{b, c\}$

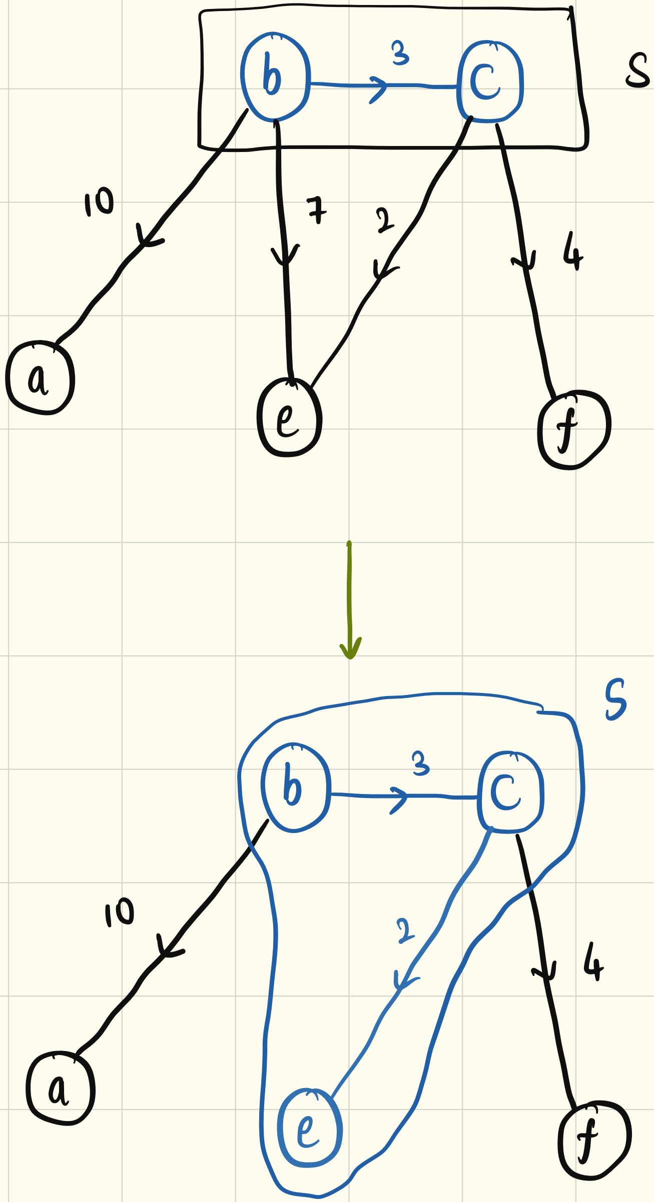
Step 2:

$S = \{b, c, e\}$



$w_1 \leq \dots \leq w_k$

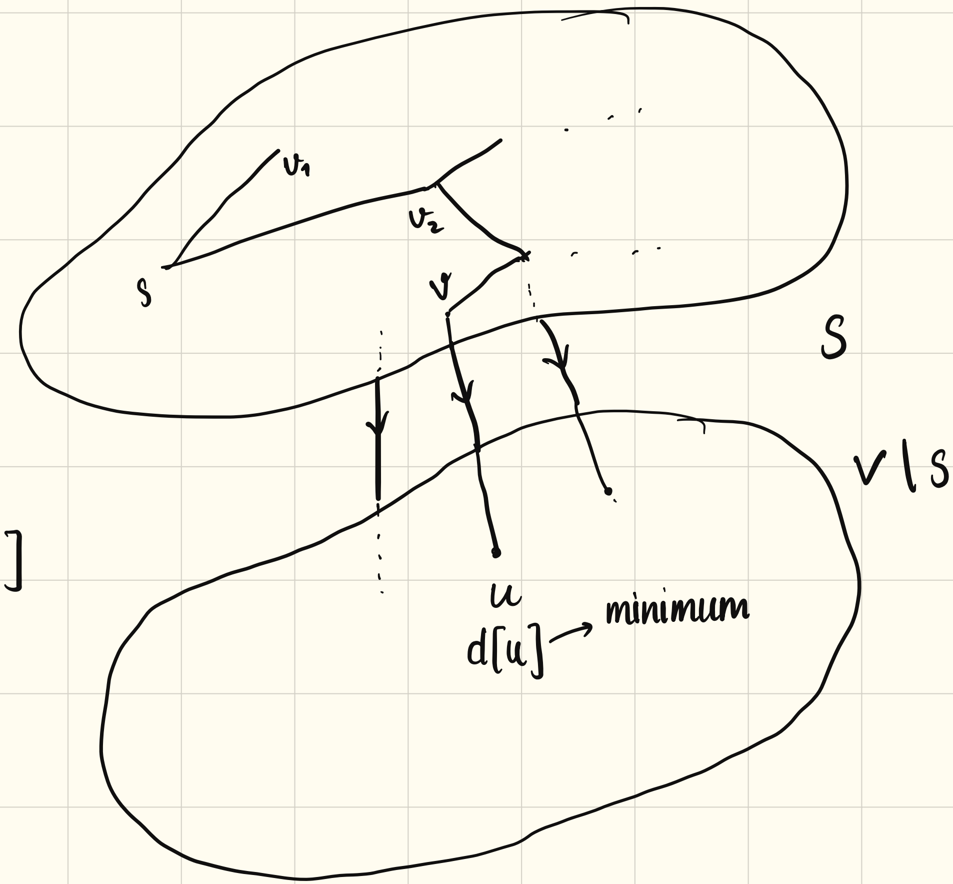
~ Greedy algorithm



Maintain a $E' \subseteq E[S]$

which induces a tree T:

Path from s to v in T



RELAX (u, v)

If $d[u] + w(u, v) < d[v]$

$d[v] = d[u] + w(u, v)$