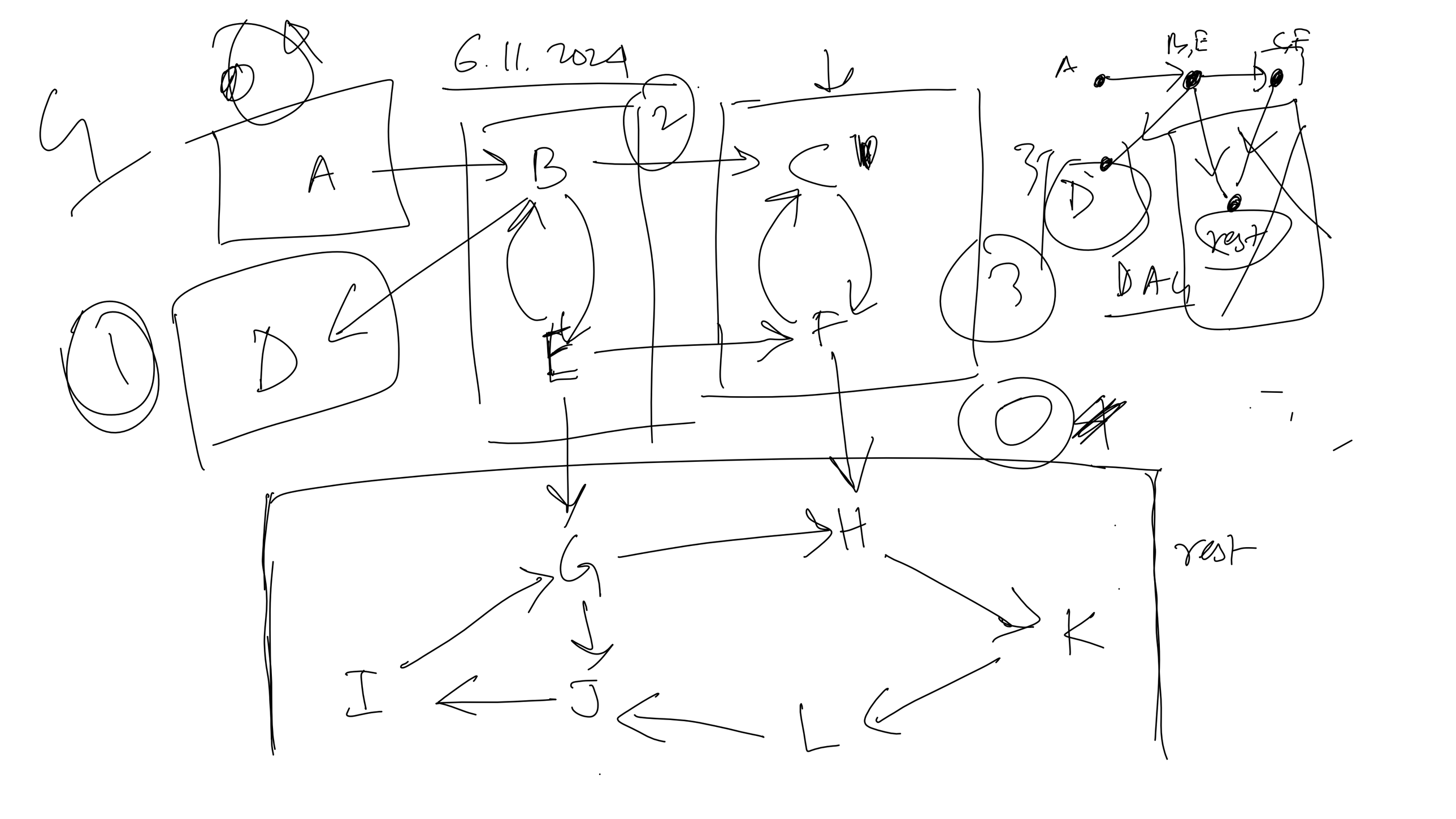
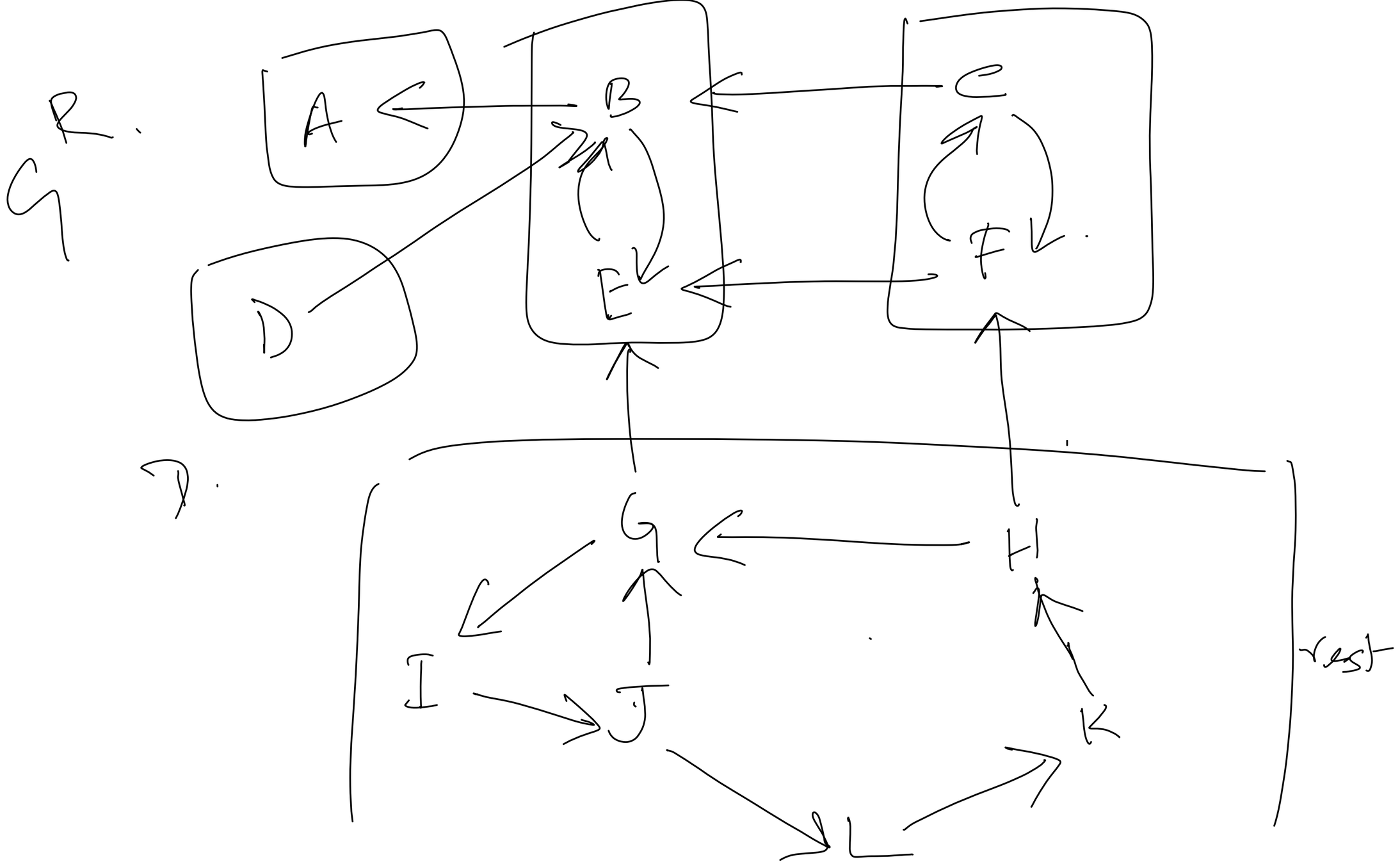

6.11.2024
A
B
C
D
E
F
1
2
3
A
B,E
C,F
D
rest
DAG
G
H
I
J
K
L
rest


R.
A
B
C
D
E
F
G
H
I
J
K
L
rest

Procedure SCC

- You perform DFS from any node to the $G^R$. & get a ordering based on decreasing finish time.
- Process the vertices in the previous sorted order & perform DFS on G.

(each restart of DFS increments another counter which is the tag of the connected component)

time complexity: $O(|V| \log |V|) \rightarrow O(|V|)$

**Fact:** Let $C_1$ & $C_2$ be two strongly connected component such that $\exists$ a path from $C_1$ to $C_2$.

$u_1 \rightsquigarrow u_2$

Then,

the highest finish time in $C_1$

the highest finish time in $C_2$.

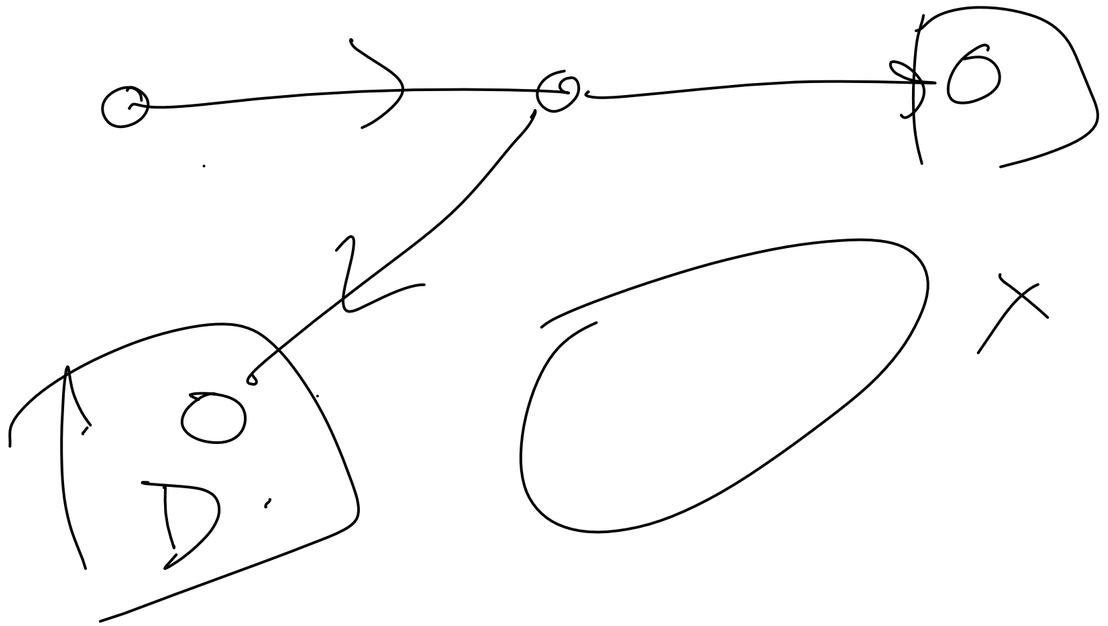
(no matter where you start DFS).

**pf:** two cases

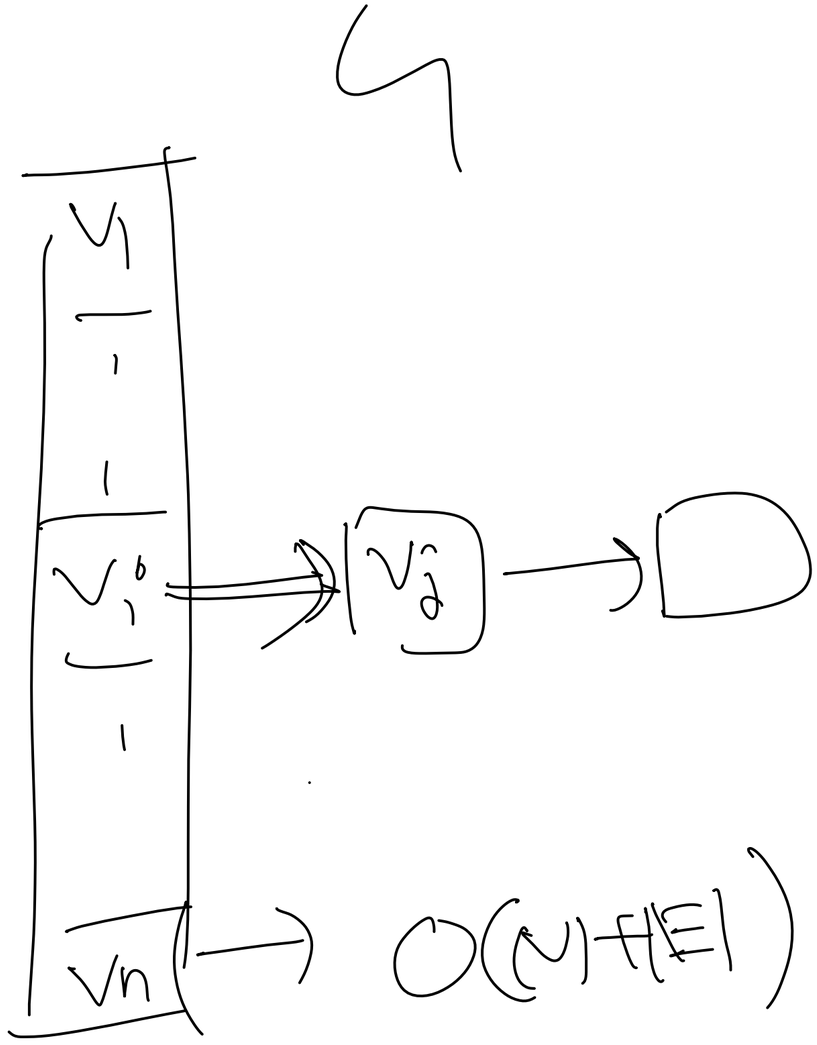
- process $C_1$ earlier.
  - ---
- process $C_2$ earlier.
  - ---

Once we have the ~~to~~ previous fact,

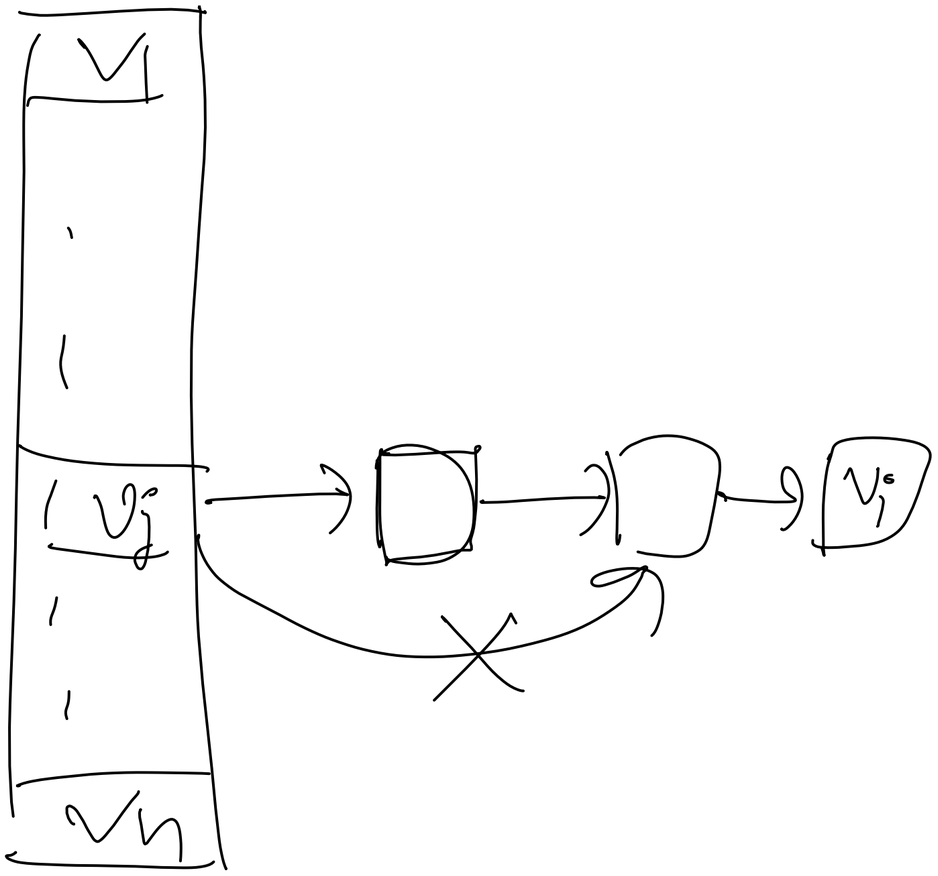
correctness of the algorithm is obvious!

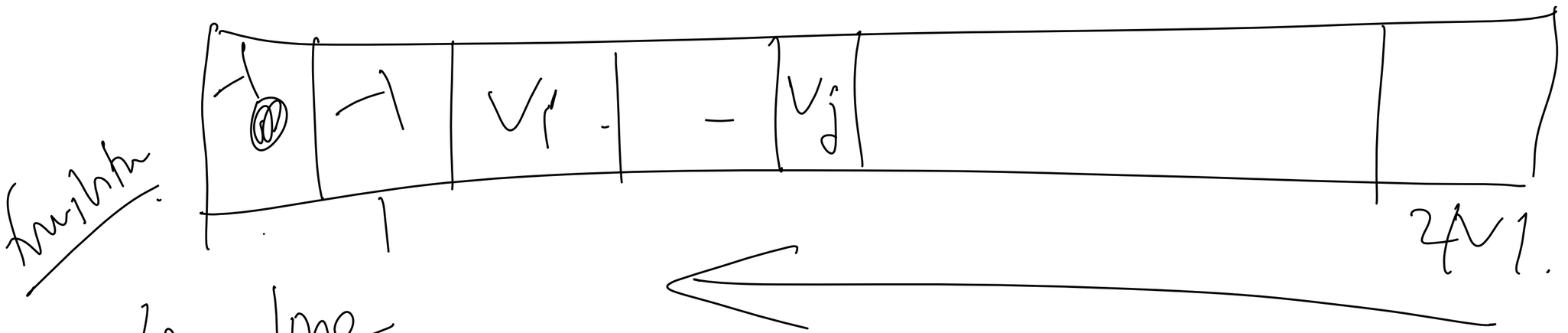


How to reverse adjacency list




$G^R$.






timeline

array indexed by

$O(|V|)$ time.

$O(|V| + |E|)$ total time!

Assignment: Read Dasgupta DFS chapter & Solve exercises.

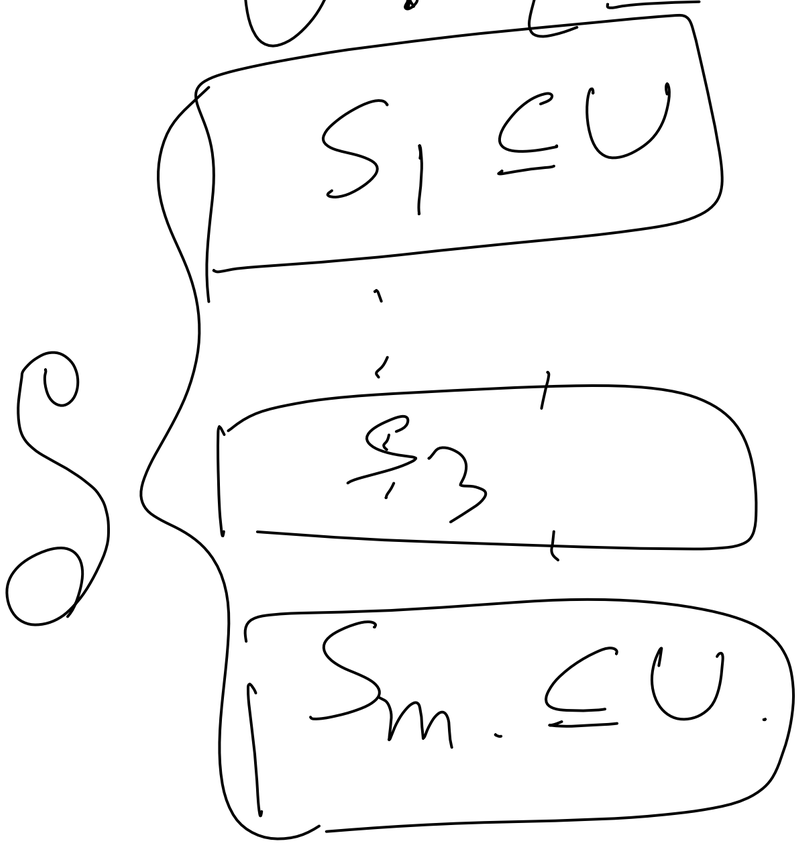
---

Greedy ~~DFS~~ as a "Heuristic"

- no proof of Correctness
- based on some intuition "always ~~correct~~"
- work well in practice.

# Set cover problem :

$U : \{x_1, \ldots, x_n\}$




**Goal:**

Pick the smallest number of subsets that covers $U$.

$$\underset{T \subseteq S}{\arg\min} \; |T|$$

$$\bigcup_{S_j \in T} S_j = U$$

$U$: $\{1, 2, 3, 4, 5, 6, 7, 8\}$

$S_1$: $\{1, 3, 5, 7\}$

$S_2$: $\{2, 4, 7\}$

$S_3$: $\{8, 5\}$

$S_4$: $\{3, 8\}$

$S_5$: $\{2, 4, 6, 8\}$.

- $S_1 \cup S_5 = U$
- $\nexists$ any single set that covers $U$
- answer is 2.

Try out all subsets $2^m$ time.

— no algorithm that can do in poly $(n, m)$ $nm^2, n^3m^9$

$\in$ NP-hard

easy to verify ⇒ $\subseteq$ EXP

but difficult to solve.

## Greedy algorithm

- Untill U is covered
  - Pick $S_j$ that that covers most # of additional elements

Claim: The greedy rule returns a solution that is at most $\ln n$ worse than the optimal solution.

proof: Lets assume the best solution has size $k$.

→ first set must cover at least $n/k$ elements.

$n_t \leftarrow$ # leftover elements to cover after $t$ iterations.

$n_0 \leftarrow n$.

Consider after $(t-1)$ iterations.

$n_{t-1}$ elements are remaining.

greedy chosen set covers $\geq \frac{n_{t-1}}{k}$ new elements.

$$n_t \leq n_{t-1}\left(1 - \frac{1}{k}\right)$$

Suppose we take $m$ steps of the greedy algo. By repeated applicat of

$$n_m \leq n_0\left(1 - \frac{1}{k}\right)^m.$$

$e^x \geq 1+x$ $\forall x \in \mathbb{R}$

$$= n\left(1 - \frac{1}{k}\right)^m$$

$$\leq n e^{-m/k} \leq 1$$

When $n_m < 1$ we are done.

What is good choice of $m$?

$$n e^{-m/k} \leq 1$$

$$e^{m/k} \geq n$$

$$m \geq k \ln n$$

We'll set $m = \lceil k \ln n \rceil$ □

Competitive ratio / approximation ratio : $\ln n$

$V = \{1, 2, \ldots, 8\}$.

$S_1 = \{5, 6, 7, 8\}$

$S_2 = \{4, 5, 6, 7\}$

$S_3 \{3, 4, 5, 6\}$.

$S_1 = \{1, \ldots, 50\}$

$S_2 =$

$S_3$

$S_1 = \{1, 3, 5, 7\}$

$S_2 = \{2, 4, 6, 8\}$.

U: {1, 2, 3, 4, 5, 6}.

$S_1$: {1, 2, 3}

$S_2$: {4, 5, 6}.

1 2 3

4 5 6

~~$S_3$: {1, 4}~~

$S_4$: {2, 3, 5, 6}.

{$S_3$, $S_4$} Greedy is OK.

Ref greedy : $\{S_4, S_1, S_2\}$.

best : $\{S_1, S_2\}$

Question: This ratio $\ln n$ is best or can we improve the greedy analysis?

Ans: $\exists$ cases where greedy will be $c \ln n$ factor worse D