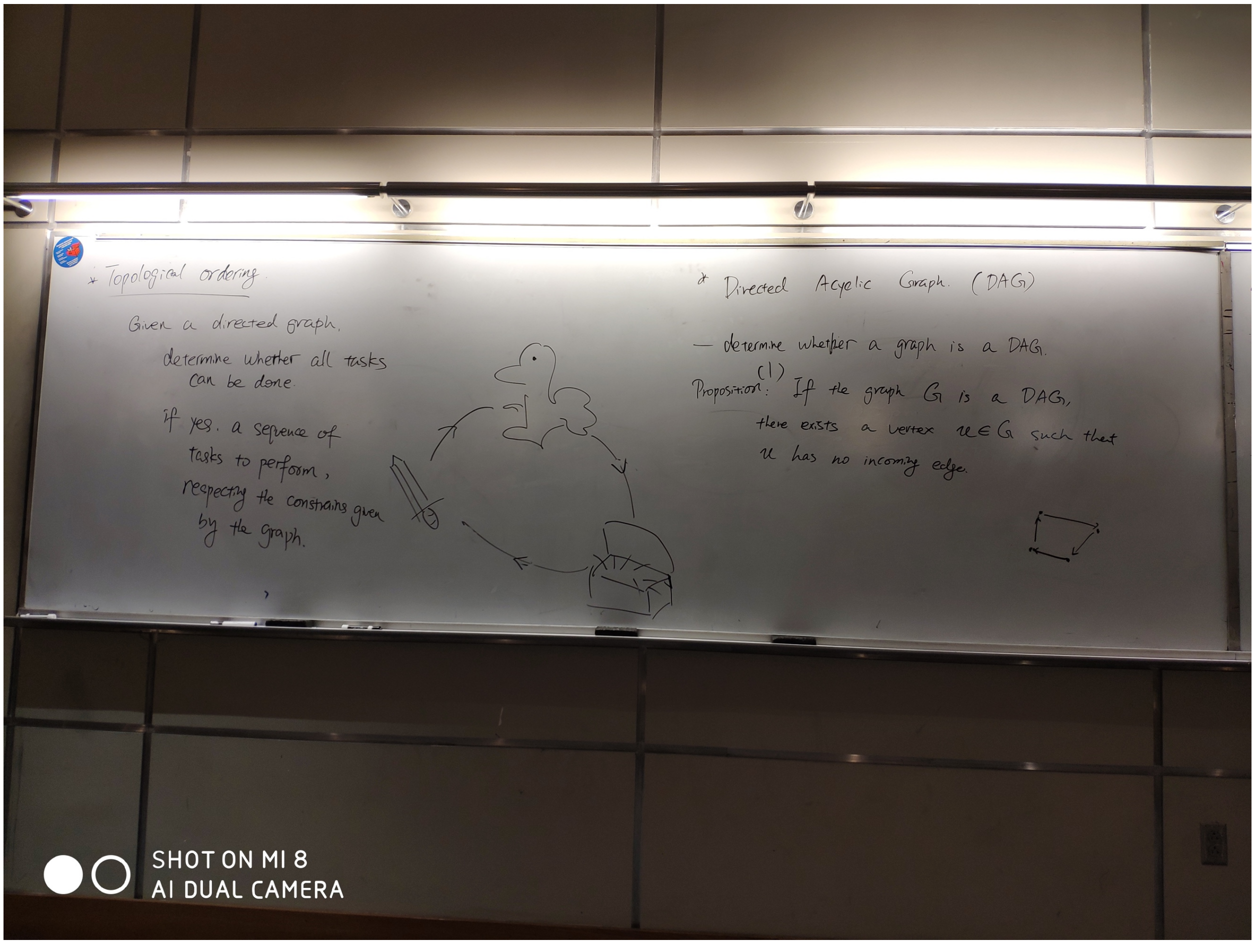

* Topological ordering
Given a directed graph,
determine whether all tasks can be done.
if yes. a sequence of tasks to perform, respecting the constrains given by the graph.
* Directed Acyclic Graph. (DAG)
— determine whether a graph is a DAG.
Proposition (1): If the graph G is a DAG, there exists a vertex u ∈ G such that u has no incoming edge.
SHOT ON MI 8
AI DUAL CAMERA

Proof: if the graph is DAG and all vertices have at least one incoming edge.

We can start from an arbitrary node and trace back using its incoming edge.

this process will never end.

We can do this for $n+1$ times, where $n = |V|$

by pigeonhole principle, we know that two node must be identical

$\Rightarrow$ cycle. $\neq$ DAG

* If G is a DAG, then there exists a topological ordering for G.

Proof: by induction.

(Base case): when G has only one node, then all tasks in G can be done.

(Induction): We assume that G with k nodes can be finished if G is a DAG.

If G is a DAG with $k+1$ nodes, by proposition (1) we know that there exists a node (v) with no incoming edge in G. We can perform the corresponding task first, and remove the node from $G \Rightarrow G' = G \setminus v$ G' becomes a graph with k nodes

by induction G' has a topological ordering.

(if G is a DAG, if we remove some node from it, G remains a DAG)

(Conclusion)

...

If G is a DAG, can we identify the sequence of tasks to perform?

For a graph G.

identify a node v that has no incoming edge.

Do the task and remove the node from G.

update G.

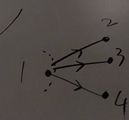
iterate.

$n = |V|$

$O(n)$

$O(|V| + |E|)$ ✓

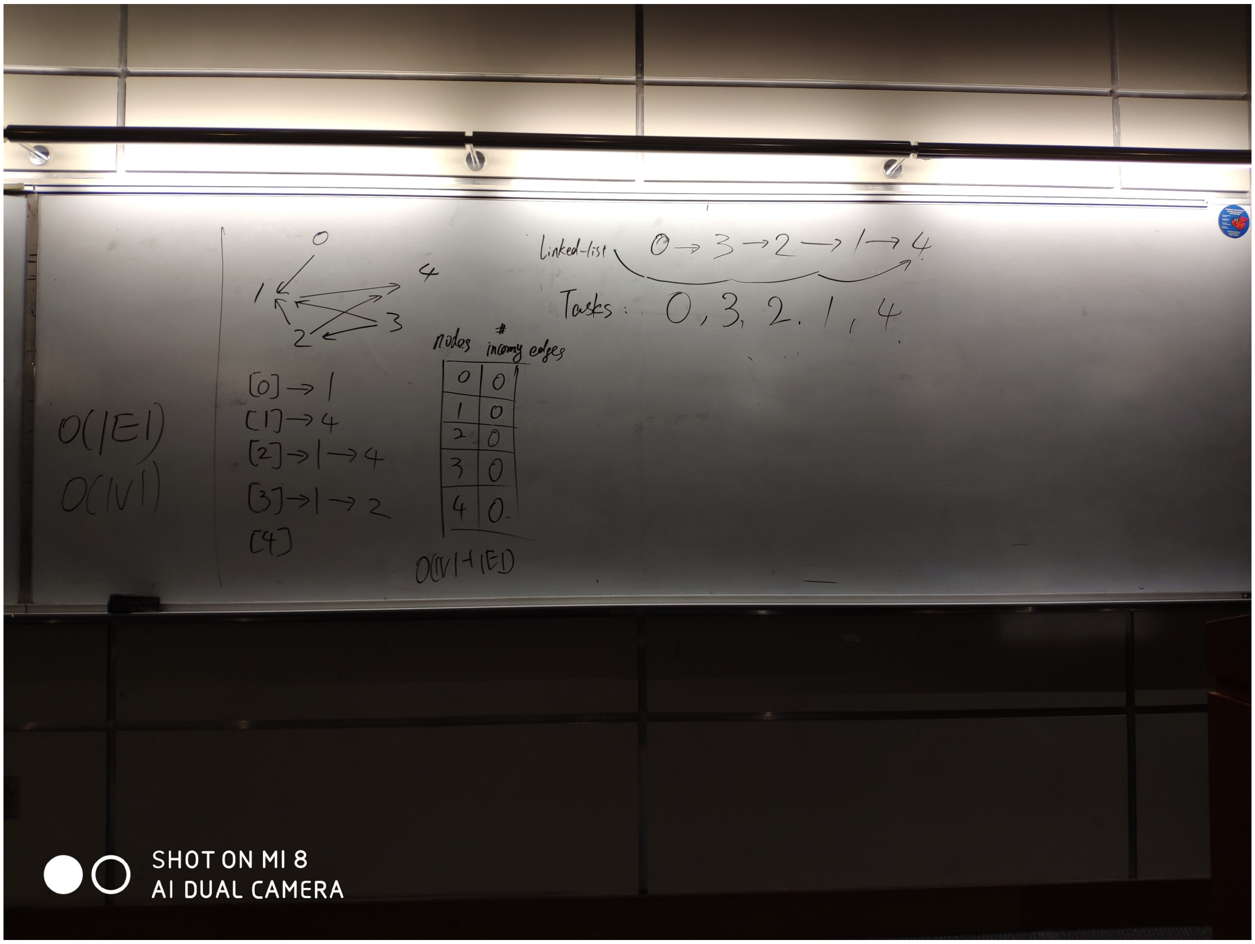
$O((|V| + |E|)^2)$



$\otimes \to 0 \to 1 \to 2 \to 3$

3

node	# of incoming edges
1	0
2	0
3	0
4	0


0
1
4
2
3
[0] → 1
[1] → 4
[2] → 1 → 4
[3] → 1 → 2
[4]
O(|E|)
O(|V|)
nodes
incoming edges
0 0
1 0
2 0
3 0
4 0
O(|V|+|E|)
Linked-list 0 → 3 → 2 → 1 → 4
Tasks: 0, 3, 2, 1, 4
SHOT ON MI 8
AI DUAL CAMERA