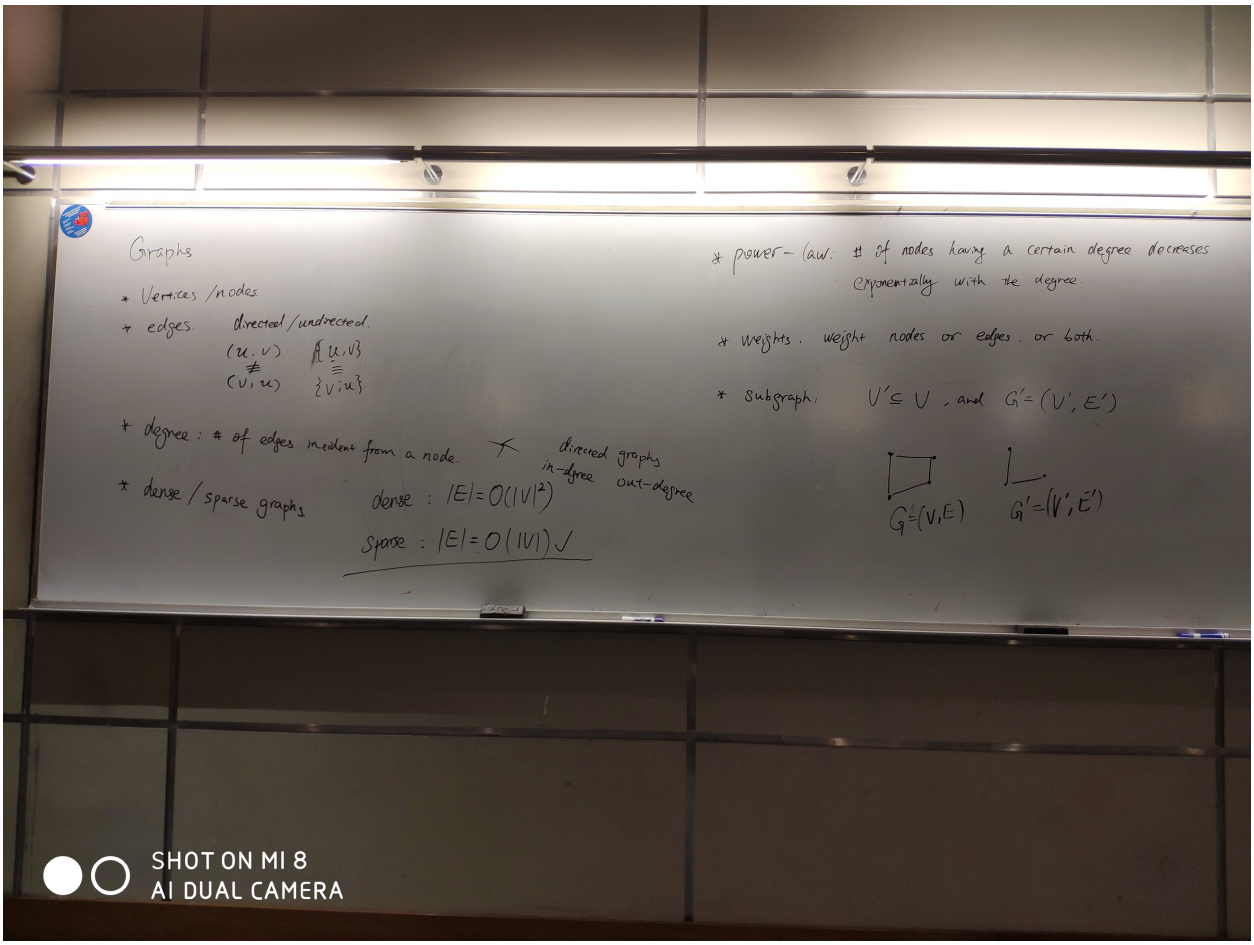
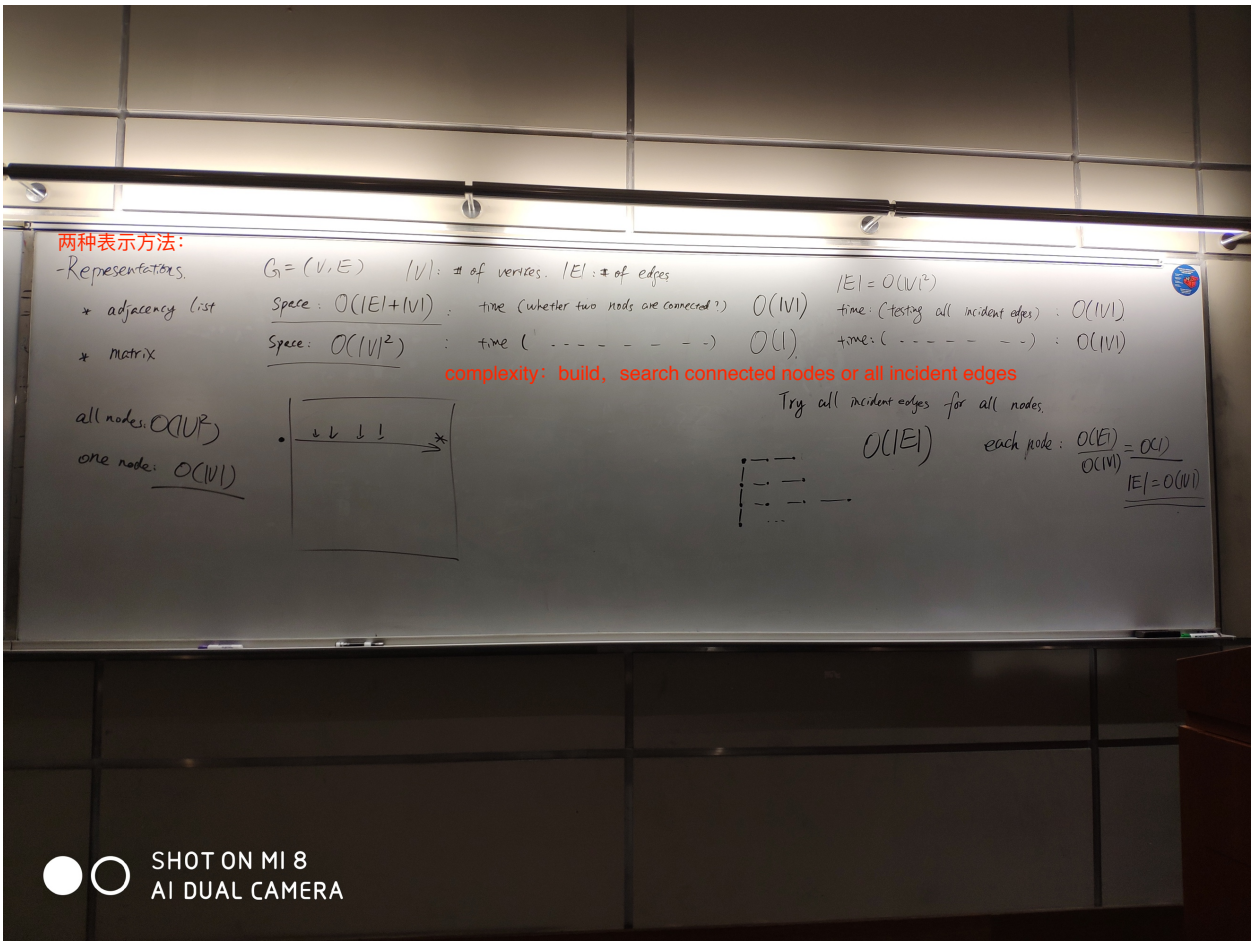
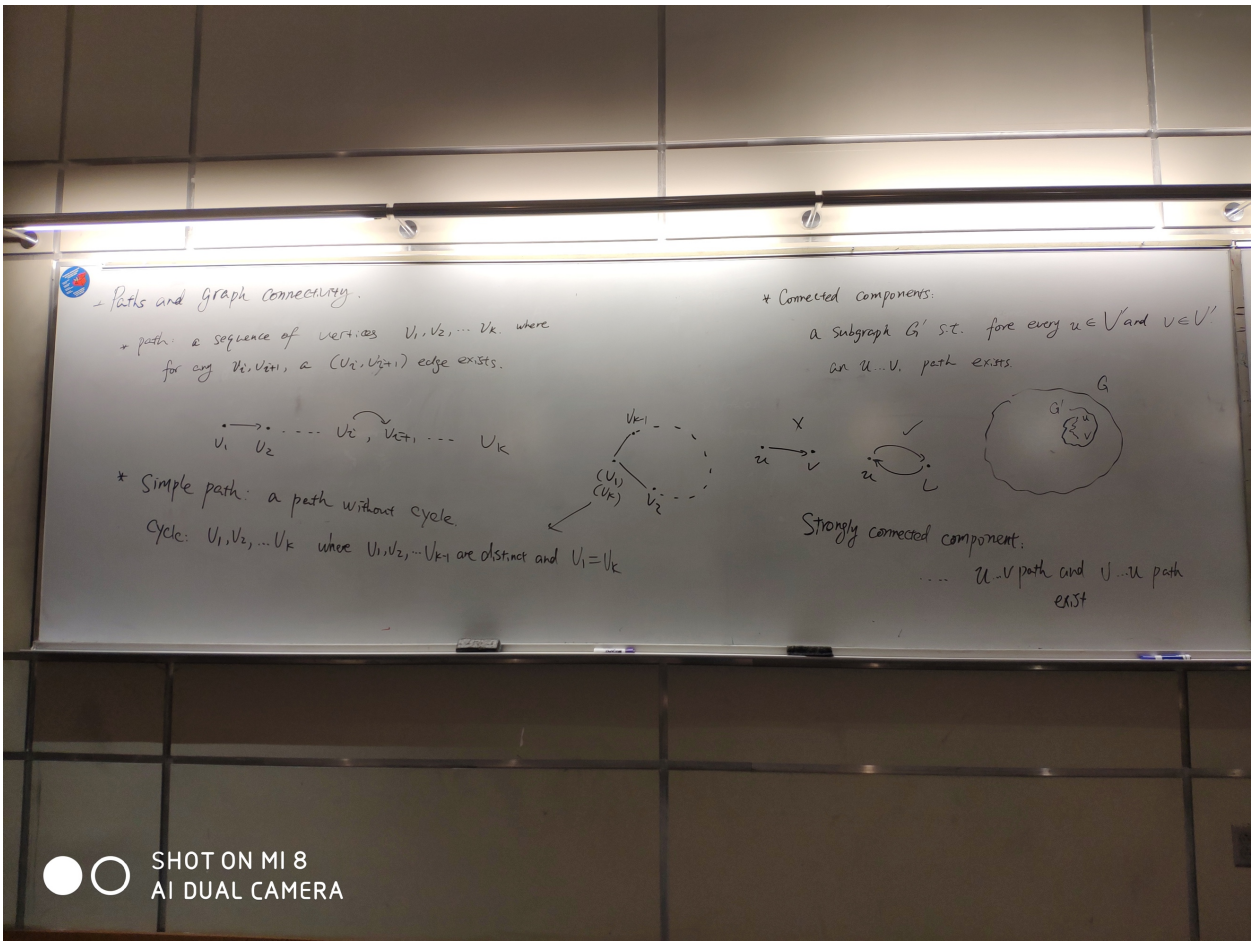
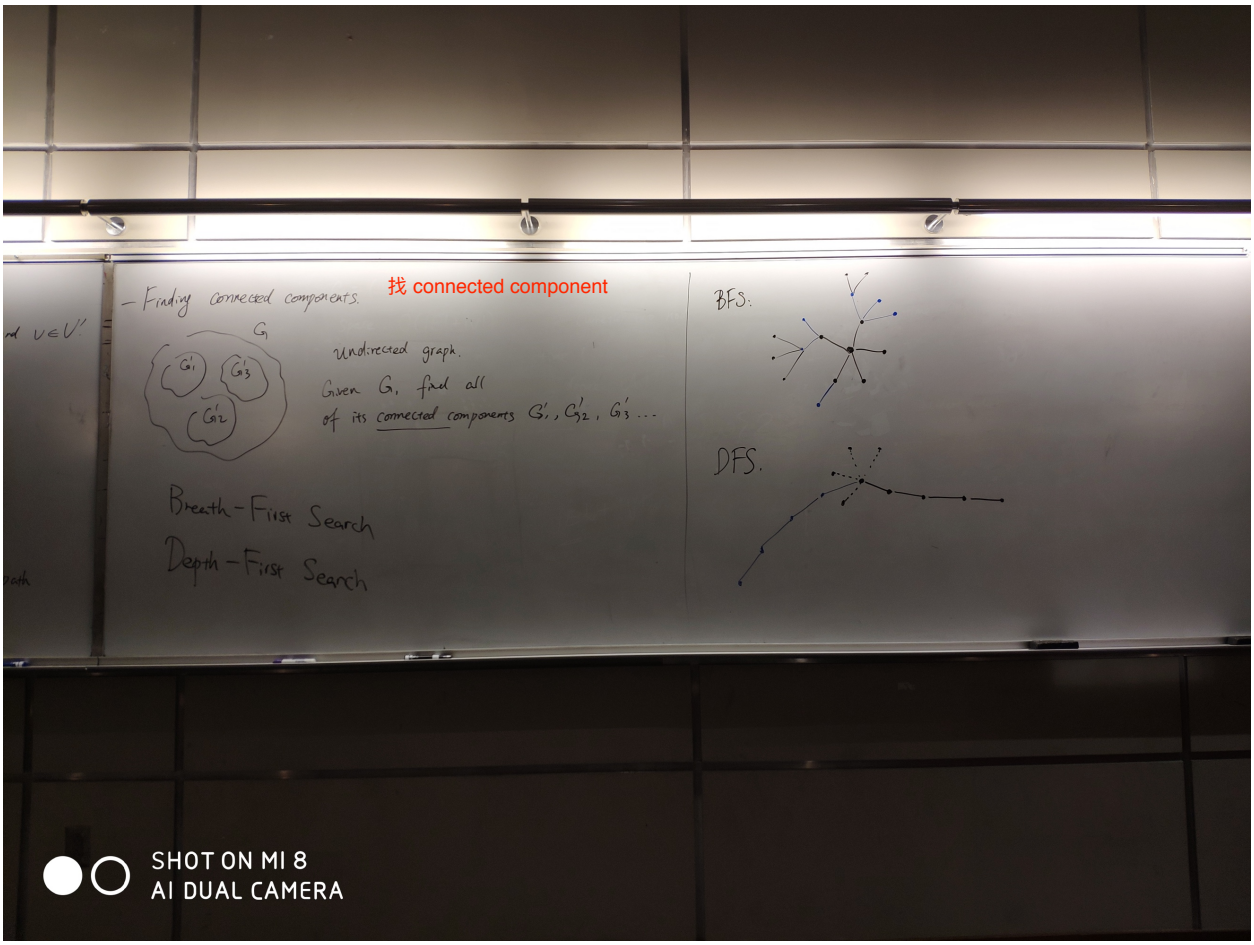
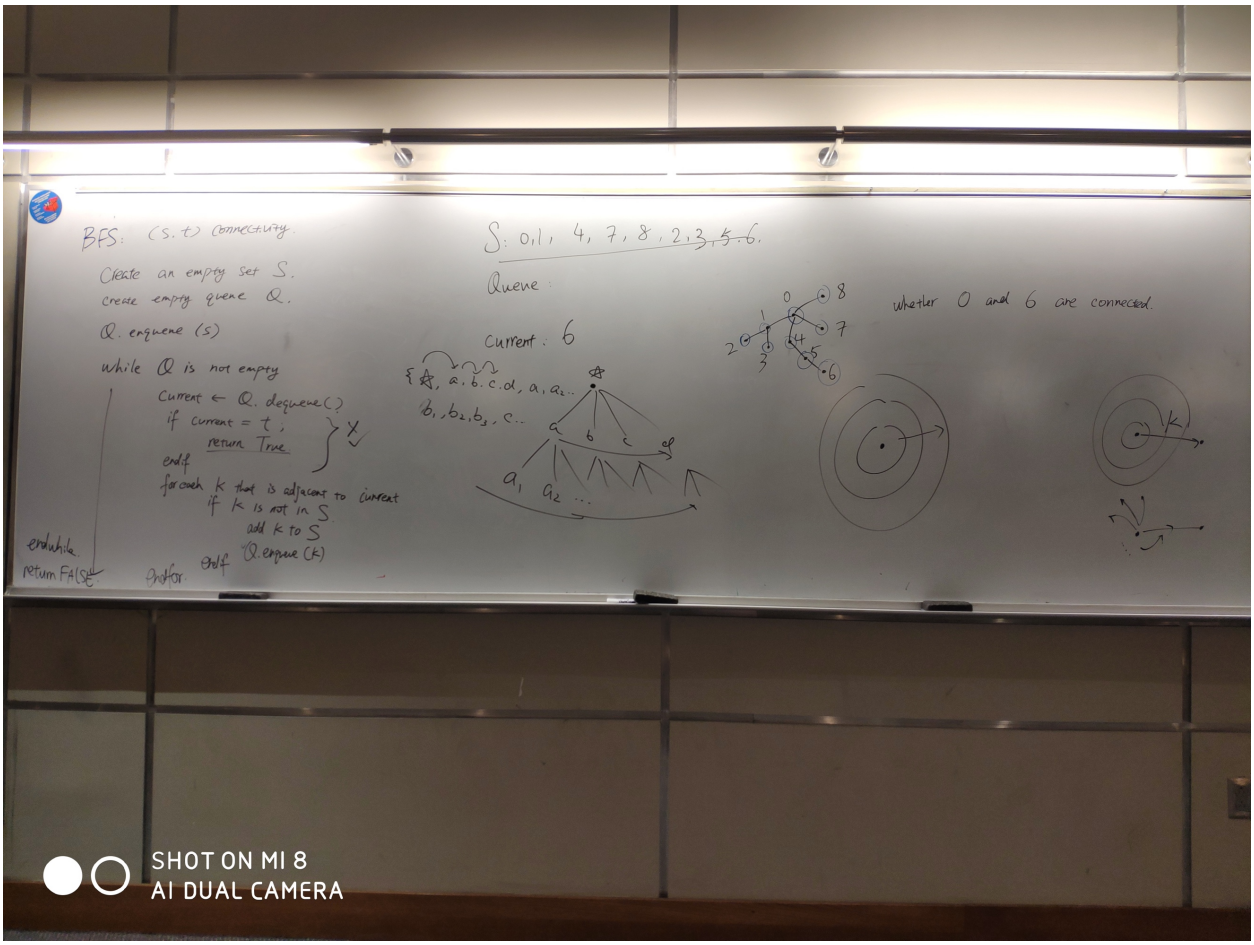

Graphs
* Vertices / nodes
* edges. directed / undirected.
(u, v) ≠ (v, u)
{u, v} ≡ {v, u}
* degree : # of edges incident from a node.
directed graphs
in-degree
out-degree
* dense / sparse graphs
dense : |E| = O(|V|²)
sparse : |E| = O(|V|) ✓
* power-law: # of nodes having a certain degree decreases exponentially with the degree
* weights. weight nodes or edges. or both.
* subgraph. V' ⊆ V, and G' = (V', E')
G = (V, E)
G' = (V', E')
SHOT ON MI 8
AI DUAL CAMERA


两种表示方法：
-Representations.
* adjacency list
* matrix
$G=(V,E)$ $|V|$: # of vertices. $|E|$: # of edges
Space: $O(|E|+|V|)$: time (whether two nods are connected?) $O(|V|)$
Space: $O(|V|^2)$: time (- - - - - - - - -) $O(1)$
$|E| = O(|V|^2)$
time: (testing all incident edges) : $O(|V|)$
time: (- - - - - - - - -) : $O(|V|)$
complexity: build, search connected nodes or all incident edges
all nodes: $O(|V|^2)$
one node: $O(|V|)$
Try all incident edges for all nodes
$O(|E|)$
each node: $\frac{O(|E|)}{O(|V|)} = O(1)$
$|E| = O(|V|)$
SHOT ON MI 8
AI DUAL CAMERA




– Finding connected components.
找 connected component
G
G'1
G'3
G'2
undirected graph.
Given G, find all
of its connected components G'1, G'2, G'3 ...
Breath-First Search
Depth-First Search
BFS:
DFS:
SHOT ON MI 8
AI DUAL CAMERA


BFS: (s, t) connectivity
Create an empty set S.
create empty queue Q.
Q. enqueue (s)
while Q is not empty
Current ← Q. dequeue()
if current = t;
return True
endif
for each k that is adjacent to current
if k is not in S
add k to S
Q. enqueue (k)
endif
endfor
endwhile
return FALSE
S: 0, 1, 4, 7, 8, 2, 3, 5, 6.
Queue:
current: 6
whether 0 and 6 are connected.
SHOT ON MI 8
AI DUAL CAMERA