Lec 2) DIJKSTRA'S algorithm, DAGs, geographical networks.

Large group tutorial.

1) What does the following algorithm compute?

$+\infty$ / $-\infty$ > what weight?

MAXNODEWEIGHT ($\bar{G}$, w) {$\bar{G} = (V, E)$, w - edge weight}

maxweight $\leftarrow -\infty$

for each node $v \in V$ do // n

nodeweight $\leftarrow 0$

for each node u adjacent to node v do // n. times

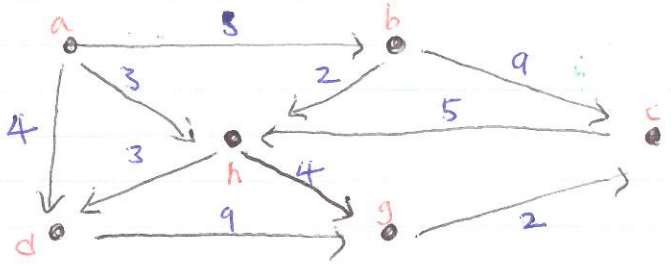
nodeweight $\leftarrow$ nodeweight + w(u, v)

If nodeweight > maxweight then maxweight $\leftarrow$ nodeweight

return maxweight.

Computation PART

example input



1) Adjacent Matrix (Representation 1)

n = # nodes
m = # edges
* # = number of

• $V = |n| = 6$

Is it adjacent? weight

	a	b	c	d	g	h
a		1/3		1		1
b			1			1
c						1
d					1	
g			1			
h			1			

Space needed = $\Theta(n^2)$

time needed = $\Theta(n^2)$

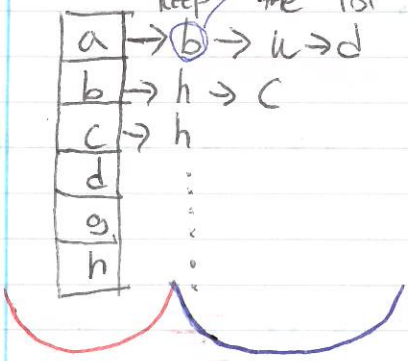
2) Adjacent List (representation 2)

- Make a adjacent List.

constant of memory needed to store this element.

Keep the list of edges (out going edges) = Size as number of adjacent no

// 순서 상관 없음 원하는 alphabet 순서대로.



$\Theta(n) + \Theta(m) = \Theta(n+m)$

n = # nodes

m = # edges

Space needed = $\Theta(n+m)$

// Node 하고 Edge 각각 저장.

$\Theta(n) + \Theta(m) = \Theta(n+m)$ < $\Theta(n^2)$

Adjacent List — Adjacent Matrix.

$0 \le n \le n(n-1) \sim n^2$

그래프에 아무것도 없어서 edges의 수는 0

최대 가질 수 있는 Edge의 수

So Linear is much less than Quadratic

↳ need much time to set output.

$\Theta(n+m)$ time

$\Theta(n+m)$ Space

2) what is the running time of this algorithm, if the input graph is given as

a) an array of adjacency lists;
b) an adjacency matrix ;

How would you implement this algorithm, if the input graph was given as a list of edges, in arbitary order?

.) Let's assume

{ G=(V,E), w - edge weights }

(a,b,3) , (b,c,9) , (h,g,h) ...

m

$\Theta(m)$

space needed.