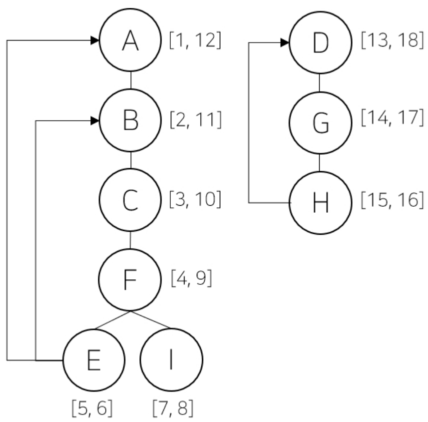
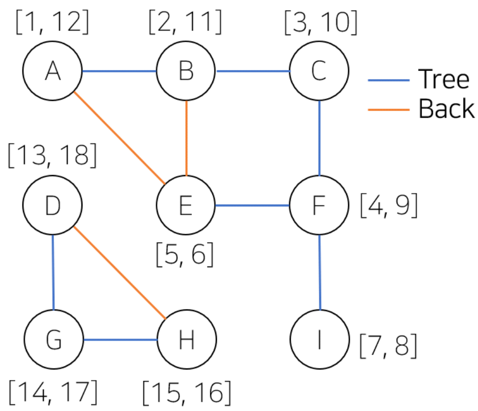
2020 알고리즘 Prof.조승범 Task (20.10.08)					
학과	소프트웨어	학번	2019038004	이름	조민우

1. 교과서 연습문제 3.1을 해결하시오.
3.1. Perform a depth-first search on the following graph; whenever there's a choice of vertices, pick the one that is alphabetically first. Classify each edge as a tree edge or back edge, and give the *pre* and *post* number of each vertex.





2. 교과서 연습문제 3.11 을 해결하시오 (저번 숙제와 마찬가지로 제시한 알고리즘이 왜 올바르게 동작하는지 설명(증명) 하고, 코드를 이용하여 설명하지 마시오). 참고로 교과서의 linear 의 의미는 'linear numer of vertices and edges' 이다.
3.11. Design a linear-time algorithm which, given an undirected graph G and a particular edge e in it, determines whether G has a cycle containing e.

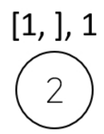
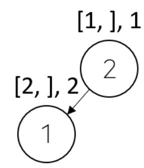
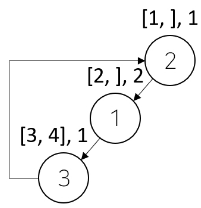
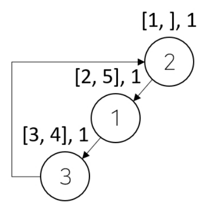
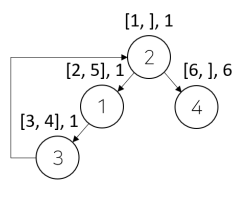
Edge e(u, v)이고 DFS Tree상 u가 v의 parent node라고 가정.
Cycle 존재 : v의 descendant 중 어느 하나는 u의 ancestor로의 Back Edge 존재.
Edge e를 포함한 Cycle을 찾아야 하므로, Graph G 중 Edge e를 포함한 Connected component G`만을 고려.

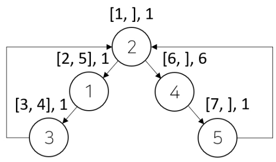
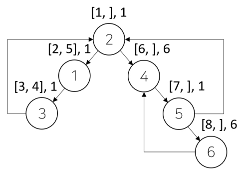
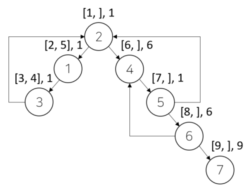
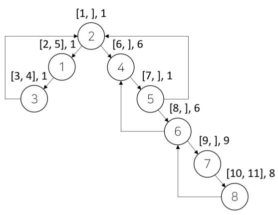
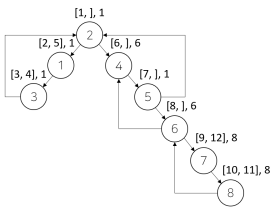
G`을 DFS하는 중, v를 explore할 때 부터, v의 explore가 종료될 때까지 pre number가 v의 pre number 이상인 어떤 vertex k($\therefore pre(k) \geq pre(v)$)에 대하여 pre number가 u의 pre number 이하인 어떤 vertex x ($\therefore pre(x) \leq pre(u)$)로의 Back Edge가 있는지 검사하면 된다.

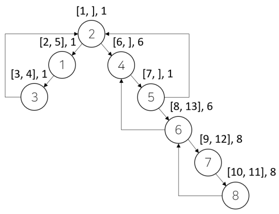
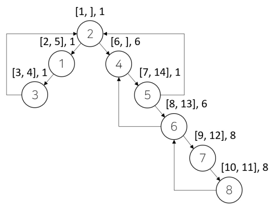
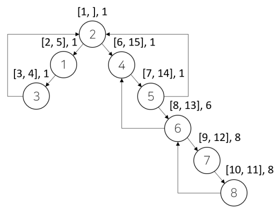
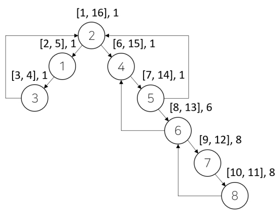
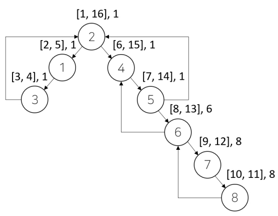
이는, DFS 탐색 중 해결할 수 있으므로 Time Complexity는 DFS와 동일한 O(n+m) (worst-case : u가 Start Point고, G가 Connected Graph일때, DFS를 전체 진행해야 함)이다.

증명) vertex k -> vertex x로의 Back Edge가 있지만 Cycle이 성립하지 않는다고 가정
$pre(k) \geq pre(v)$이므로, vertex k는 vertex u의 descendant
$pre(x) \leq pre(u)$이므로, vertex x는 vertex v의 ancestor
∴x -> ... -> u -> v -> ... -> k 로 connected함을 알 수 있다. 이때, vertex k -> vertex x로의 Back Edge가 있지만 Cycle이 성립하지 않으려면 이것이 disconnected 해야하므로 가정에 모순이 됨으로 이를 증명할 수 있다.

3. Chap 3 강의자료 34 페이지부터 나오는 예제를 처음부터 끝까지 완성하시오(각 step 마다 dfs tree와 stack 상태 표현 등, 예시와 같은 방법으로 나타내시오). 단, 강의자료와는 다르게 각 vertex의 low 값이 explore를 호출하고 종료할 때마다 어떻게 변하는지 또한 같이 나타내시오(강의자료에는 각 vertex의 최종 low 값이 나와 있음). 필요한 경우 첨부파일에 있는 양식 2_3.pptx 를 이용해도 무방. 단 최종저장본은 반드시 pdf 파일이어야 함.

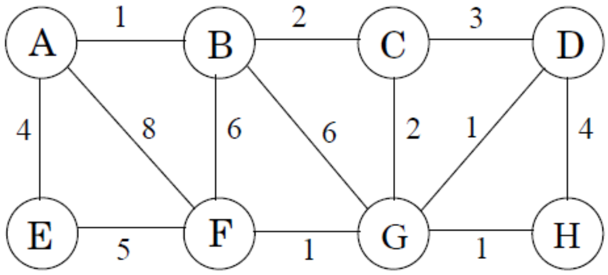
(1) explore(2) 호출	(2, 1)	{}	
(2) explore(2) explore(1) 호출	(1, 3) (2, 1)	{}	
(3) explore(2) explore(1) explore(3) 호출 및 종료	(3, 2) (1, 3) (2, 1)	{}	
(4) explore(2) explore(1) 종료 $low(1) \geqq pre(2)$		{{(3, 2),(1, 3),(2, 1)}}	
(5) explore(2) explore(4) 호출	(4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}}	

(6) explore(2) explore(4) explore(5) 호출	(5, 6) (5, 2) (4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}}	
(7) explore(2) explore(4) explore(5) explore(6) 호출	(6, 7) (6, 4) (5, 6) (5, 2) (4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}}	
(8) explore(2) explore(4) explore(5) explore(6) explore(7) 호출	(7, 8) (6, 7) (6, 4) (5, 6) (5, 2) (4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}}	
(9) explore(2) explore(4) explore(5) explore(6) explore(7) explore(8) 호출 및 종료	(8, 6) (7, 8) (6, 7) (6, 4) (5, 6) (5, 2) (4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}}	
(10) explore(2) explore(4) explore(5) explore(6) explore(7) 종료 $low(7) \geqq pre(6)$	(6, 4) (5, 6) (5, 2) (4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}, {(8 ,6), (7, 8), (6, 7)}}	

(11) explore(2) explore(4) explore(5) explore(6) 종료	(6, 4) (5, 6) (5, 2) (4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}, {(8 ,6), (7, 8), (6, 7)}}	
(12) explore(2) explore(4) explore(5) 종료	(6, 4) (5, 6) (5, 2) (4, 5) (2, 4)	{{(3, 2),(1, 3),(2, 1)}, {(8 ,6), (7, 8), (6, 7)}}	
(13) explore(2) explore(4) 종료 $low(4) \geqq pre(2)$		{{(3, 2),(1, 3),(2, 1)}, {(8 ,6), (7, 8), (6, 7)} {(6, 4), (5, 6), (5, 2), (4, 5), (2, 4)}}	
(14) explore(2) 종료		{{(3, 2),(1, 3),(2, 1)}, {(8 ,6), (7, 8), (6, 7)} {(6, 4), (5, 6), (5, 2), (4, 5), (2, 4)}}	
최종 결과		{{(3, 2),(1, 3),(2, 1)}, {(8 ,6), (7, 8), (6, 7)} {(6, 4), (5, 6), (5, 2), (4, 5), (2, 4)}}	

4. 교과서 5.2 의 (a) 와 (b)를 해결하시오. (a) 같은 경우 cost 뿐만 아니라 각 step 마다 pre와 H의 상태 또한 모두 나타내시오(chap 4 강의자료 41 페이지와 똑같은 방법으로 하면 됨). (b) 같은 경우 chap 4 slide 25 페이지 부터를 참고하되, path compression을 써야함에 유의하시오. 필요한 경우 첨부파일에 있는 양식 2_4a.pptx와 2_4b.pptx를 이용해도 무방. 단 최종저장본은 반드시 pdf 파일이어야 함.

5.2. Suppose we want to find the minimum spanning tree of the following graph.



(a) Run Prim's algorithm; whenever there is a choice of nodes, always use alphabetic ordering (e.g., start from node A). Draw a table showing the intermediate values of the *cost* array.
(b) Run Kruskal's algorithm on the same graph. Show how the disjoint-sets data structure looks at every intermediate stage (including the structure of the directed trees), assuming path compression is used.

Answer은 6페이지(a)와 7페이지(b)에 풀어져 있음.

(a)

1

	A	B	C	D	E	F	G	H
cost(v)	0	∞	∞	∞	∞	∞	∞	∞
pre(v)	nil	nil	nil	nil	nil	nil	nil	nil
H	A	B	C	D	E	F	G	H

2

	A	B	C	D	E	F	G	H
cost(v)	0	1	∞	∞	4	8	∞	∞
pre(v)	nil	A	nil	nil	A	A	nil	nil
H		B	C	D	E	F	G	H

3

	A	B	C	D	E	F	G	H
cost(v)	0	1	2	∞	4	6	6	∞
pre(v)	nil	A	B	nil	A	B	B	nil
H			C	D	E	F	G	H

4

	A	B	C	D	E	F	G	H
cost(v)	0	1	2	3	4	6	2	∞
pre(v)	nil	A	B	C	A	B	C	nil
H				D	E	F	G	H

5

	A	B	C	D	E	F	G	H
cost(v)	0	1	2	1	4	1	2	1
pre(v)	nil	A	B	G	A	G	C	G
H				D	E	F		H

6

	A	B	C	D	E	F	G	H
cost(v)	0	1	2	1	4	1	2	1
pre(v)	nil	A	B	G	A	G	C	G
H					E	F		H

7

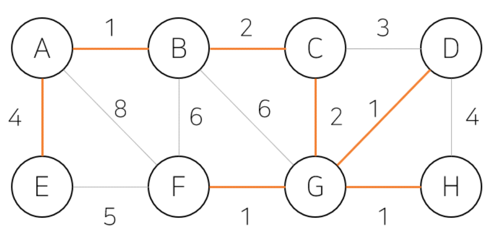
	A	B	C	D	E	F	G	H
cost(v)	0	1	2	1	4	1	2	1
pre(v)	nil	A	B	G	A	G	C	G
H					E			H

8

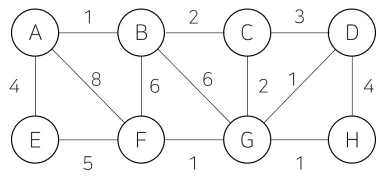
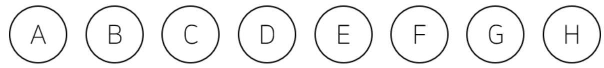
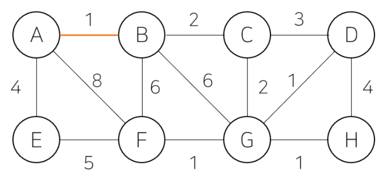
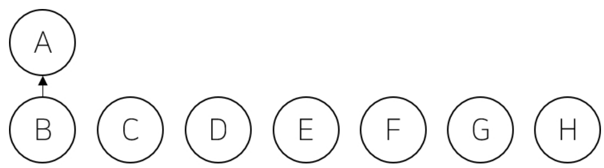
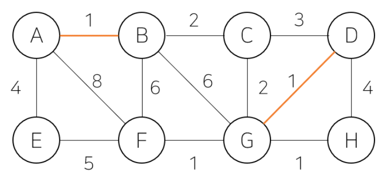
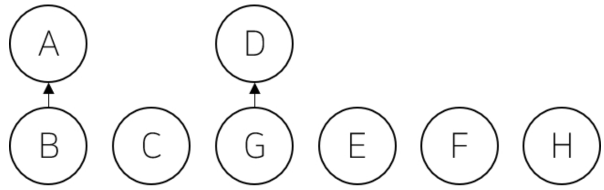
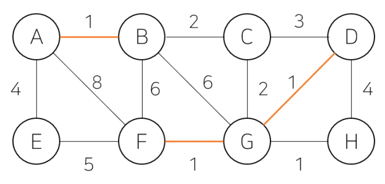
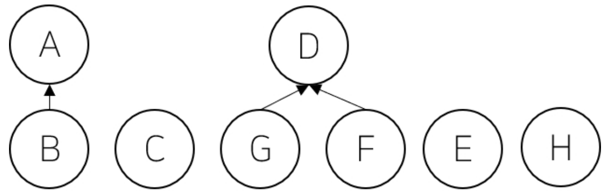
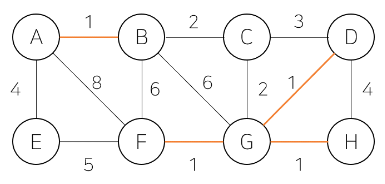
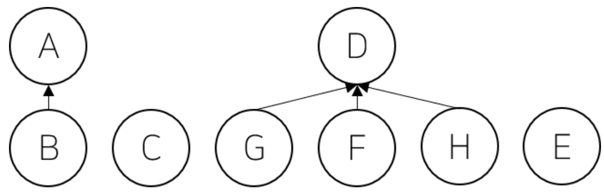
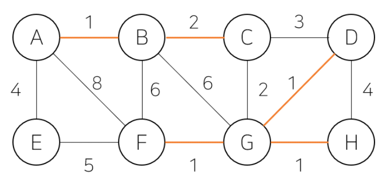
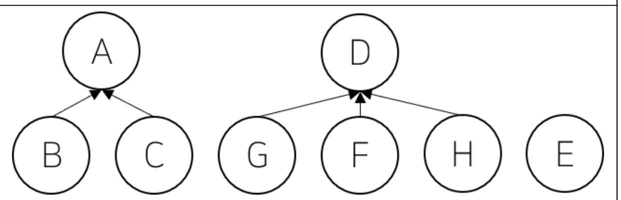
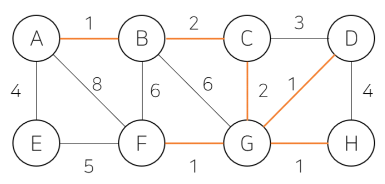
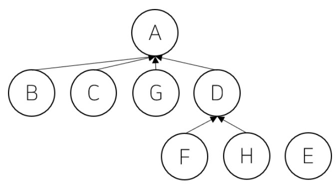
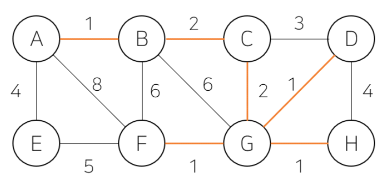
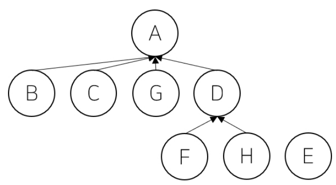
	A	B	C	D	E	F	G	H
cost(v)	0	1	2	1	4	1	2	1
pre(v)	nil	A	B	G	A	G	C	G
H					E			

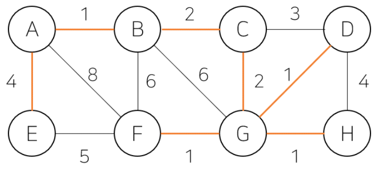
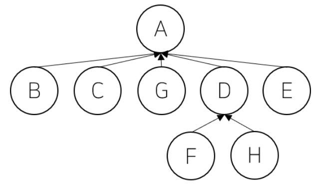
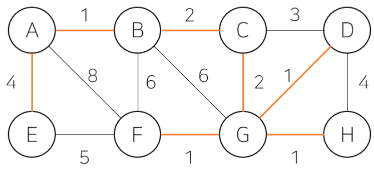
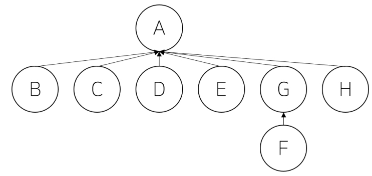
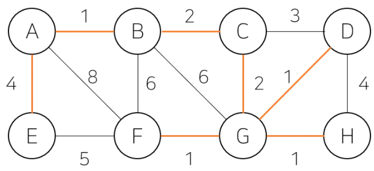
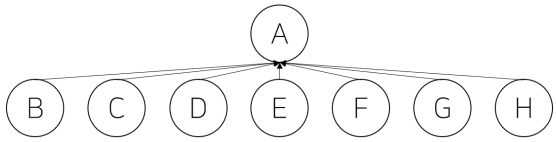
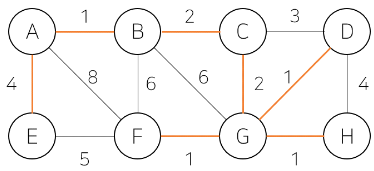
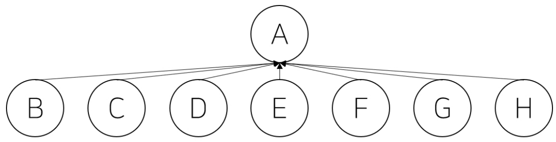
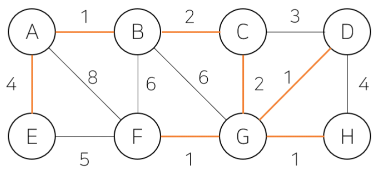
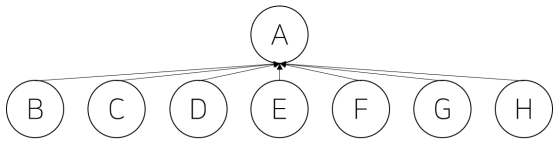
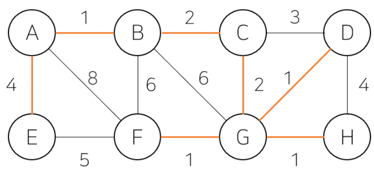
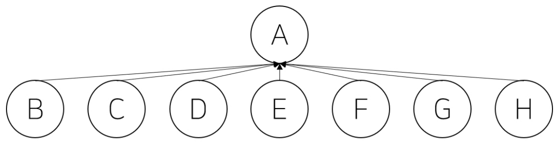
9

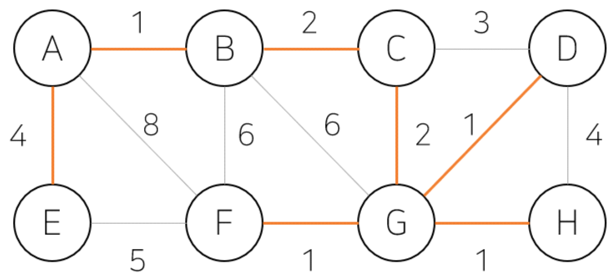
	A	B	C	D	E	F	G	H
cost(v)	0	1	2	1	4	1	2	1
pre(v)	nil	A	B	G	A	G	C	G
H								



(b)

1			
2	E(A, B)		
3	E(D, G)		
4	E(F, G)		
5	E(G, H)		
6	E(B, C)		
7	E(C, G)		
8	E(C, D)		

9	E(A, E)		
10	E(D, H)		
11	E(E, F)		
12	E(B, F)		
13	E(B, G)		
14	E(A, F)		



5. 교과서 연습문제 5.6을 해결하시오(귀류법을 써서 증명 가능).
5.6. Let G = (V;E) be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

In Kruskal's Algorithm, Kruskal의 return set을 S, Optimal Solution set을 S`로 두고, $S \neq S'$라고 가정한다면, 어떤 Edge e(u, v)가 S` - S에 속해야한다.
Kruskal 알고리즘 구현 정의 상에서 S에서 u->v path들을 이루는 Edge cost는 모두 Edge e의 cost보다 작다. (클 경우, 정의에 모순된다.)
∴S` - { e }∪{ e` }(e`은 S상에서 u->v까지의 path들 중 아무 Edge)의 total cost는 S`의 total cost보다 작고, 해당 Edge들은 여전히 Spanning Tree를 이룬다.
-> 가정에 모순됨

In Prim's Algorithm, Prim의 return set을 S, Optimal Solution set을 S`로 두고, $S \neq S'$라고 가정한다.
Cut property에 의해 어떤 임의의 vertex set S⊂V(G)에 대해, MST는 S와 V(G)-S를 이어주는 Edge들 중 cost가 가장 작은 Edge를 포함해야한다.
S와 S`의 start point는 동일하니 그곳에서부터 판단할 때, 어떤 임의의 포인트에서 Prim과 Optimal이 다른 Edge를 선택해야 $S \neq S'$가 성립한다. 이는, 가장 작은 cost를 선택해야 하는 판단에서 다른 것을 골라야함을 의미하므로, 최소 2개의 선택지가 존재해야 한다. 이때 가정에서 weight(cost)들이 distinct하다고 했으므로 가정에 모순된다.