2023 알고리즘 기말고사

(1) 시험시간 : 6월 21일(수요일) 오전 8시 30분 - 10시 20분
(2) 풀이 중간과정을 충분히 기술하여 채점자에게 도움이 되도록 하세요.
풀이 과정이 충분하지 않거나 최종 답만 적으면 감점됩니다.
(3) 뒷면에 답을 쓰게 되면 작성자가 이 사실을 문제에 표시하기 바랍니다.

학번	이름	감독자 확인

문제 (배점)	점수	문제 (배점)	점수
1 (30)		4 (30)	
2 (40)		5 (30)	
3 (30)		6 (40)	
extra (10)		총합계 (210)	

[1] 1차원 배열 X[N]에 숫자가 정렬된 순서로 저장되어 있다. 우리는 어떤 숫자 k가 이 배열에 있는지를 ternary search(3분 탐색)로 검사하려고 한다. ternary search는 전체 탐색할 공간을 3개의 거의 같은 크기인 $n/3$으로 분할하여 각 경계의 자료가 k인지를 검사한다. 만일 아니라면 다시 답이 존재할 구간을 찾아서 탐색을 진행한다. 이 ternary search 알고리즘의 최악 시간 복잡도 $T(N)$를 $\Theta(\ )$ 기준으로 자세히 계산하시오.

[2] insertion sort는 X[N]의 앞쪽에서부터 정렬을 한다. 즉 X[1:i]까지는 sorting된 상황에서 X[i+1] 원소를 뒤에서부터 검사하여 적절한 곳에 삽입(insertion)하는 방식이다. 이와 달리 selection sort는 정렬되지 않은 구간에서 가장 작은 원소를 찾아(select)하여 이것을 X[0], X[1], X[2]의 원소와 바꾸어 나가는 방식이다.

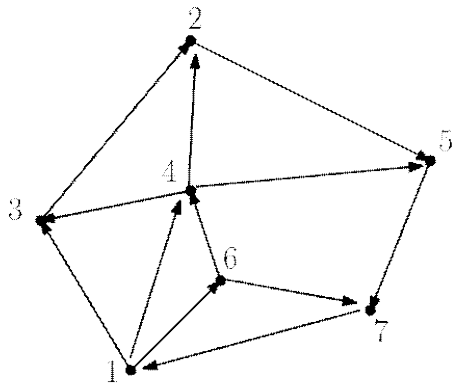
a) 두 원소의 비교(Key comparison)를 기본동작으로 할 때 insertion sort에서 worst case와 best case가 되는 X[9]={ }의 한 예를 제시하고 그때 수행된 비교 횟수를 각각 제시하시오.

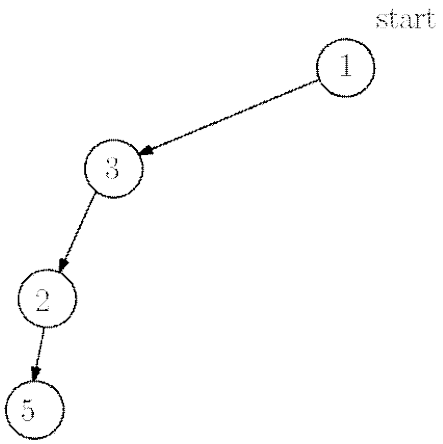
b) 두 원소의 비교(Key comparison)를 기본동작으로 할 때 selection sort에서 worst case와 best case가 되는 X[9]={ }의 한 예를 제시하고 그때 수행된 비교 횟수를 각각 제시하시오.

c) 이번에는 기본동작을 data movement, 즉 원소의 위치 이동을 기준으로 한다. 이 기준으로 insertion sort에서 worst case와 best case가 되는 X[9]={ }의 한 예를 제시하고 그때 수행된 이동 횟수를 각각 제시하시오.

d) 이동(Key movement) 기준으로 selection sort에서 worst case와 best case가 되는 X[9]={ }의 한 예를 제시하고 그때 수행된 이동 횟수를 각각 제시하시오.

[3] 다음 directed graph에서 node 1에서 출발하여 나머지 점을 모두 **edge 겹침없이** 방문하고 다시 1로 돌아오는 cycle을 찾으려고 한다. 이 작업을 backtracking으로 해결하고자 할 때, 구성되는 **backtracking tree를 제시하시오.** 단 방문순서는 unvisited node 중에서 번호가 빠른 순서를 먼저 방문한다. 단 만일 cut이 필요한 경우가 있으면 cut을 한 이유를 backtrack tree에 설명을 하면 된다.

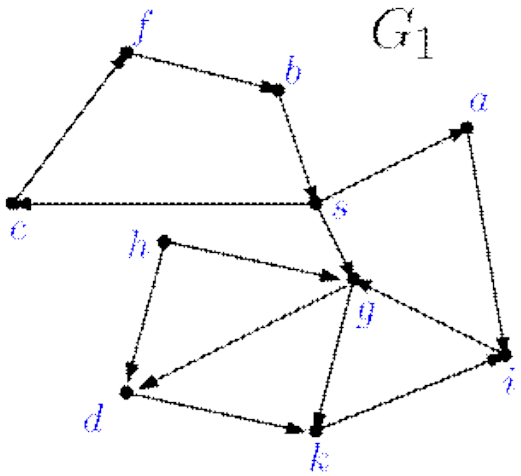




[4] 아래 directed graph G_1에서 아래 질문에 답을 하시오.

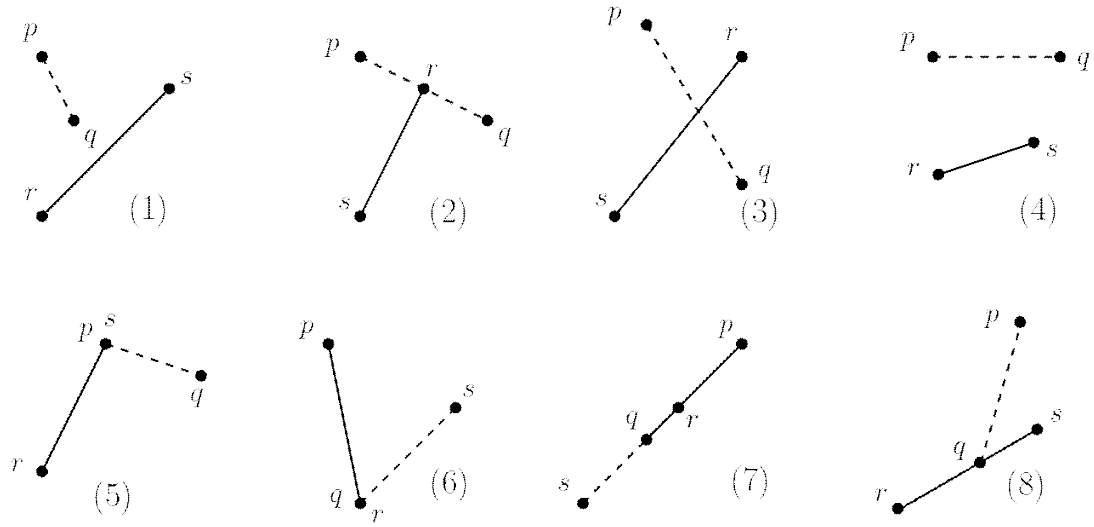
(a) Directed graph G(V,E)에서 strongly directed component를 정의하시오.

(b) 다음 그래프에서 strongly connected component를 찾는 과정을 보이시오. 단 자료구조는 adjacency list를 사용하며 한 정점에 연결된 vertex의 순서는 알파벳 순서로 가정한다. 즉 G1에서 s에 연결된 노드의 순서는 $s:\rightarrow a\rightarrow g;$ 이며 g의 경우라면 그와 나가는 방향으로 연결된 노드의 순서는 $g:\rightarrow d\rightarrow i\rightarrow k;$ 가 된다. 이것은 DFS 방문 순서를 결정하므로 중요하다. 강의 시간에 설명한 Kosaraju 알고리즘으로 진행하시오. 즉 DFS의 visit, finished number를 먼저 구해야 한다. DFS의 시작은 vertex a부터 시작한다. 이것이 끝나면 남아있는 unvisited node중에서 가장 빠른 알파벳 node에서 다시 시작한다. 먼저 아래 표를 채우시오. DFS는 dfs 방문번호, fin은 finished number를 의미한다.



Node	DFS	fin.
a		
b		
s		
f		
c		
h		
g		
d		
k		
i		

[5] **T-교차 선분** 선분(line segments)의 T-교차는 두 선분 (p,q)와 (r,s)이 영문 대문자 T자와 같이 구성된 것을 의미한다. 즉 한 선분의 끝 점이 다른 선분의 가운데에 있는 것을 의미한다. 아래 예에서 (2)과 (8) 우리가 원하는 T-교차이다.



여러분은 이 작업은 signarea(a,b,c)만 이용해서 구해야 한다. signarea(a,b,c)는 서로 다른 3점의 순서로 구성되는 삼각형의 부호면적의 부호만을 return한다. 즉 이 함수의 결과 값은 {+1, 0, -1}중 하나이다. 이를 위한 함수 Tinter(p,q,r,s)를 구성하시오. 필요한 경우 int i,j 등과 같은 보조 변수는 얼마든지 사용할 수 있다. 그 reutrn 값은 boolean으로 true(T-교차가 맞음) or false(아님)이다. (비슷하게 python으로 작성해도 좋다()

코드 작성 전에 자신의 방법을 먼저 글로 설명하는 것이 좋다.

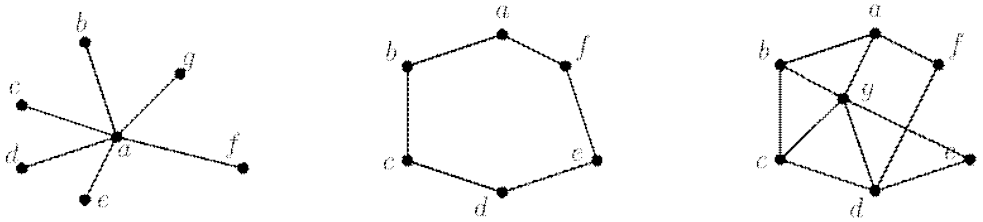
```
bool signarea( ) ;
bool Tinter( point p, point q, point r, point s ) {
  bool bt ;
  bt = signarea(p,q,r) ; // p→q→r 삼각형의 부호면적




}
```

[6] 그래프 G에서 한 정점(vertex) x를 선택할 때 이와 연결된 모든 edge는 x에 대하여 커버(cover)된다고 말을 한다. 아래 그래프 G_1에서 a를 선택하면 edges { (a,b), (a,g), (a,c), (a,d), (a,e), (a,f)}가 covered-되었다고 말을 한다.

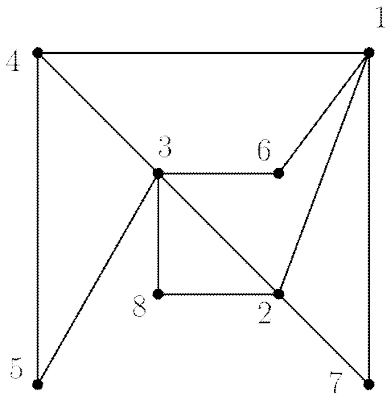
이에 관련된 문제로는 가장 적은 수의 vertex를 선택해서 모든 edge를 cover하는 것이 있다. 이렇게 선택된 vertex 집합은 minimum vertex cover(**MVC**,최소정점 커버)라고 말을 한다. 아래 그림에서 G_1의 MVC의 크기는 1이며 {a}가 그 답이다. G_2의 경우 MVC는 {a,c,e}혹은 {b,d,f}이며 그 크기는 3이다.



G_1 G_2 G_3

이 문제는 잘 알려진 NP-hard 문제이라 사람들은 적절한 Greedy 알고리즘으로 대신하고 있다. 가장 간단한 Greedy 알고리즘은 아직 cover돠지 못한 edge에 대하여 가장 많은 수의 edge를 cover할 수 있는 vertex를 선택하는 것이다. 만일 cover할 수 있는 edge수가 같을 경우에는 정점의 번호가 빠른 것을 먼저 선택한다.(tie breaking rule). 예를 들어 시작 단계에서 G_1이라면 a, G_3라면 {g}를 선택할 것이다.

(a) 다음 그래프에서 설명한 Greedy algorithm으로 MVC를 구하는 과정을 보이시오.



(b) 이러한 차수-기반의 Greedy 알고리즘을 사용했을 때 최적해(optimal solution)를 구해주는 경우와 아닌 경우의 그래프를 하나씩 제시하시오. 단 Greedy 전략으로 선택되는 정점의 순서는 표시해야 한다. **그리고 제시한 그래프 정점의 수는 최소 7 이상 되어야 한다.**

Greedy 전략으로 찾을 때 MVC를 제대로 구하는 경우의 그래프 (5점)
(Greedy MVC 크기와 Optimal MVC의 크기가 같음을 보여야 함.)

Greedy 전략으로 찾을 때 MVC를 제대로 찾지 못하는 경우의 그래프 (15점)
(Greedy MVC 크기와 Optimal MVC의 크기가 다름을 보여야 함.)