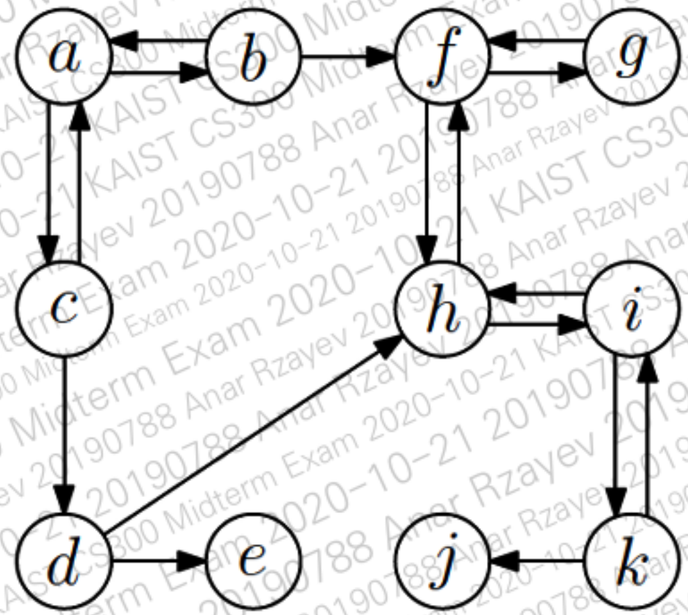
Consider the directed graph $G$ in the below figure. If we construct the meta-graph of $G$, how many edges does it have?




(A) 3

(B) 4

(C) 5

(D) 6

(E) 7

Consider the following pseudocode contains an error. Where is the error in the algorithm?

```
1:  function WorstCaseLinearTimeSelect(A[1..n], k)
2:      if n ≤ 64 then
3:          run quicksort
4:          return kth element
5:      m ← ⌈n/5⌉
6:      for i ← 0 to m − 1 do
7:          B[i] ← BruteForceMedian(A[5i + 1..5i + 5])
8:      mom ← WorstCaseLinearTimeSelect(B[1..m], ⌊m/2⌋)
9:      r ← Partition(A[1..n], mom)
10:     if k = r then
11:         return mom
12:     if k ≤ r then
13:         return WorstCaseLinearTimeSelect(A[1..r − 1], k)
14:     return WorstCaseLinearTimeSelect(A[r + 1..n], k − 1)
```

(A) Line 2
(B) Line 3
(C) Line 5
(D) Line 7
(E) Line 11
(F) Line 12
(G) Line 13
(H) Line 14

**Analyze the time complexity of the given algorithm.**

```
foo(a, n)
  if n == 0
    return 1
  if n is even
    x = foo(a, n/2)
    return x * x
  else
    x = foo(a, (n − 1)/2)
    return x * x * a
```

(A) $T(n) = T(n-1) + \Theta(1)$
(B) $T(n) = T(n-1) + \Theta(n)$
(C) $T(n) = T(n/2) + \Theta(1)$
(D) $T(n) = T(n/2) + \Theta(n)$
(E) $T(n) = 2T(n/2) + \Theta(n)$

$T(n)$ is a recurrence defined as $T(n) = 27T(n/9) + n^{(3/2)}$ for $n \geq 2$ and $T(1) = 1$. Which one of the following statement is TRUE?

A. $T(n) = \theta\left(n^{(3/2)}\right)$

B. $T(n) = \theta\left(n^{(3/2)} \log n\right)$

C. $T(n) = \theta\left(n^{\log_{27} 9}\right)$

D. $T(n) = \theta\left(n^{\log_9 27}\right)$

E. $T(n) = \theta\left(n^{\log_9 27} \log n^{(3/2)}\right)$

Given a set of $n$ distinct numbers, we need to find the $i$-th largest element. Choose an option that works in best worst-case running time.

(A) Sort the numbers with merge sort, and list the $i$-th largest element.
(B) Build a max-priority queue from the numbers, and extract maximum element $i$ times.
(C) Use an order-statistic algorithm that uses deterministic partitioning algorithm to find the $i$-th largest number.

Below are several sorting algorithms.
1) quicksort 2) mergesort 3) insertion sort 4) bubble sort
Which of these algorithms have lower bound of Ω(n lg n)?

A) 1 and 2
B) 2 and 4
C) All of the above
D) None of the above

Given $T_1(n) = \Theta(n^2)$, $T_2(n) = O(n \log n)$, $T_1(1) = O(1)$ and $T_2(1) = O(1)$, which of the following statement is FALSE?

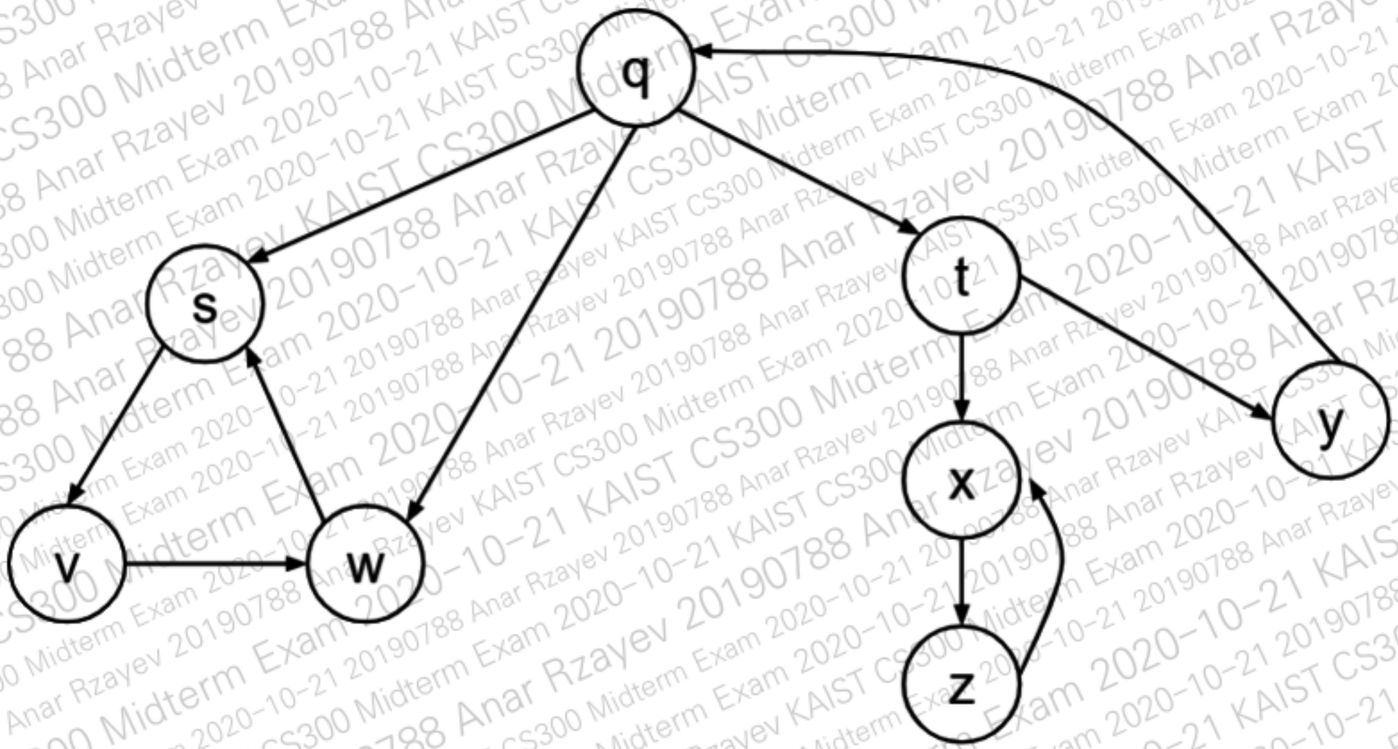
(A) $T_1(n) = O(n^3)$

(B) $T_2(n) = O(n \log_b n)$ for all $b > 1$

(C) $T_1(n) + T_2(n) = \Omega(n \log n)$

(D) $T_1(n) + T_2(n) = O(n \log n)$

(E) There exist constants $c$ and $n_0$ such that $T_2(n) \leq cn^2$ for all $n > n_0$.




Traverse the above graph by depth-first search (in alphabetical order). Start the traversal at vertex q.
Classify each edge. What is not tree edge?

(A). (q, s)

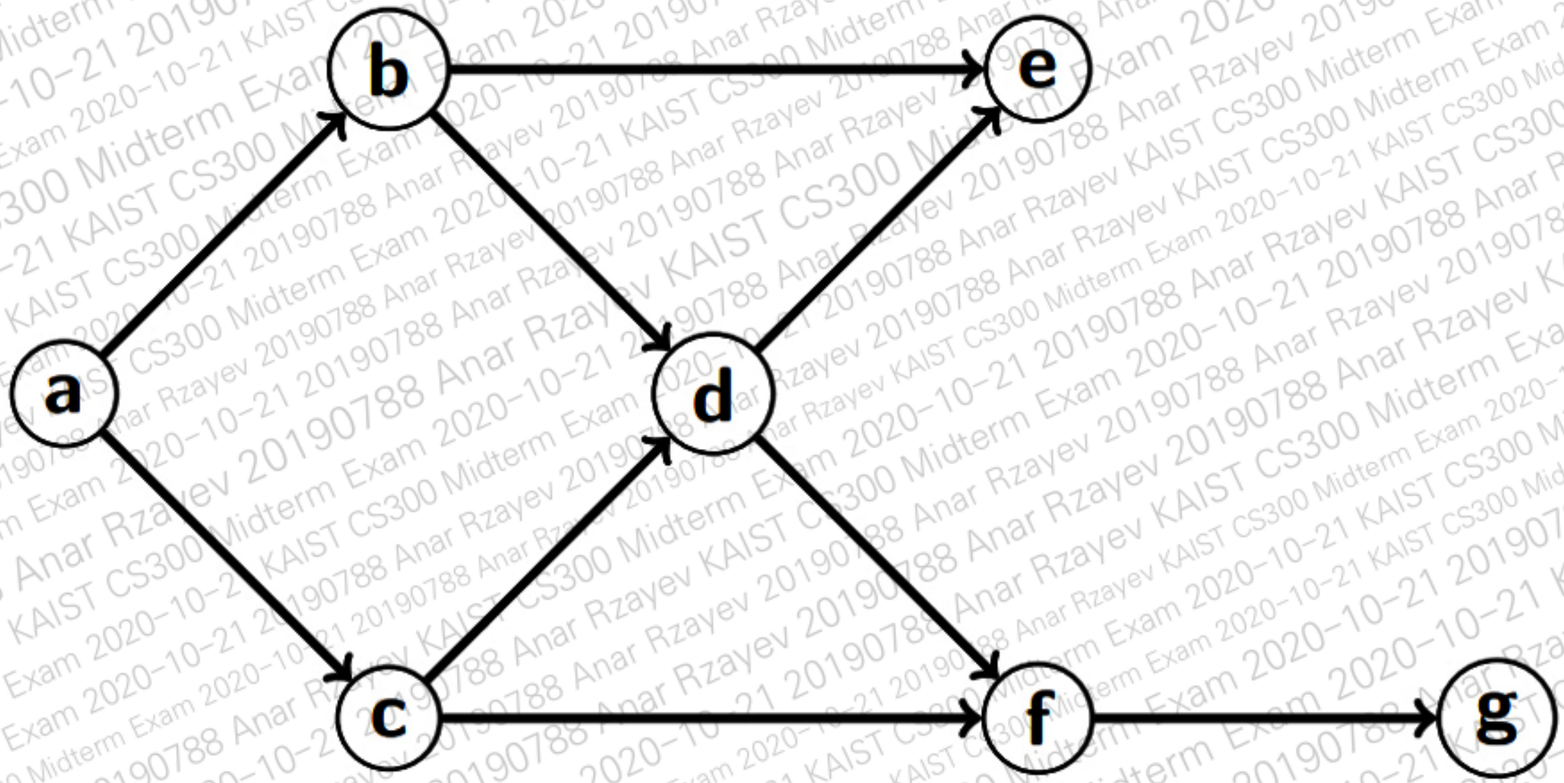
(B). (q, w)

(C). (t, y)

(D). (s, v)

(E). (v, w)

Which ordering is NOT a topological sort of the DAG shown below?




(A) a b c d e f g
(B) a b c d f e g
(C) a c b d f e g
(D) a c d b e f g
(E) a c b d f e g

**Find the right answer for $f(n)$, $g(n)$**

* $lg$ means logarithm with base 2

Base Case: $T(1) = \theta(1)$

Divide: $\theta(n^2)$

Conquer: $8T(n/5) + \theta(lg^6 n)$

Combine: $\theta(n)$

=> Recurrence relation: $T(n) = aT(n/b) + \theta(f(n)) = \theta(g(n))$

(A) $f(n) = n^3, g(n) = n^3$

(B) $f(n) = n^3, g(n) = n^2$

(C) $f(n) = lg^6 n, g(n) = n^3$

(D) $f(n) = lg^6 n, g(n) = lg^6 n$

(E) $f(n) = n^2, g(n) = lg^6 n$

(F) $f(n) = n^2, g(n) = n^2$

Consider a Quicksort algorithm where the partition algorithm always chooses the first element as a pivot. Which one of the following is **not** true?

(A) The best-case running time of Quicksort is $\Theta(nlogn)$.

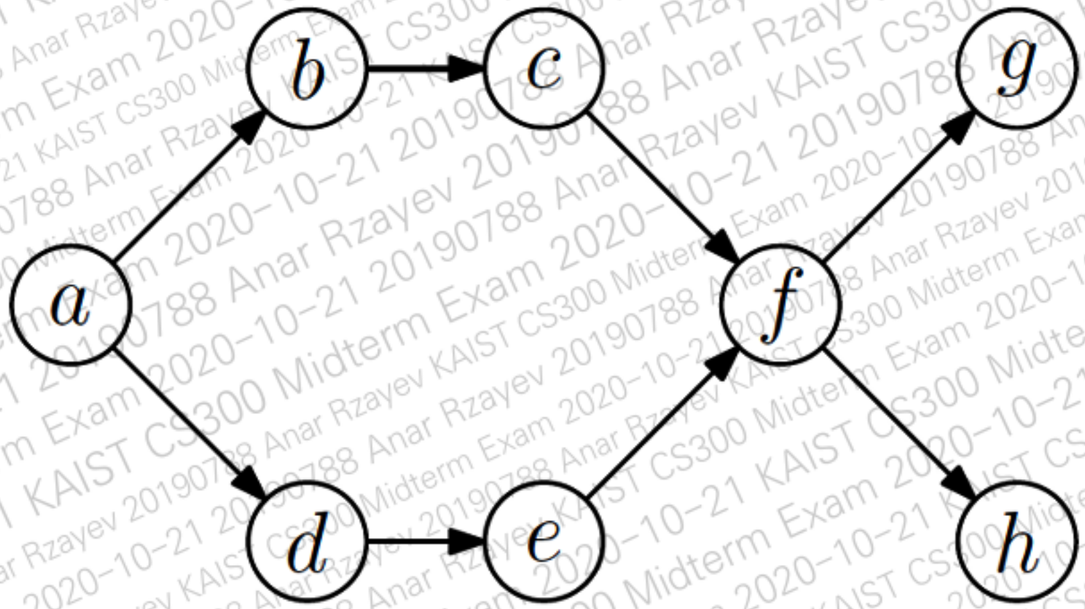
(B) If we pick the pivot randomly instead of the first element, then the expected running time of Quicksort is $\Theta(nlogn)$.

(C) The running time of Quicksort for the input array $[1, 2, 3, \ldots, n]$ is $\Theta(nlogn)$.

(D) The behavior of Quicksort depends on the relative ordering of the array elements given as the input, not on the particular values in the array.

(E) If we perform the first partitioning of the array $[8, 2, 16, 4, 20, 6, 10]$, the left group will consist of $\{2, 4, 6\}$ and the right group will consist of $\{16, 20, 10\}$.

Consider the directed acyclic graph $G$ in the below figure. How many topological orderings (possible linearizations) does it have?



Consider a Quicksort algorithm where the PARTITION algorithm always chooses the first element as a pivot. Given an input array $L = \{13, 19, 9, 5, 12, 8, 7, 4, 21\}$, find the result array $L'$ constructed by the following procedure:

1. PARTITION $L$ into left subarray $A$, right subarray $B$, and the pivot $x$ selected.

2. Concatenate 3 subarrays $A$, $x$, $B$ in this order to form a new array $L'$

Write your answer in "$a_1, a_2, ..., a_n$" format (Write only comma between numbers, no alphabets and no special characters are allowed except for comma.)

Given $n = 2^m$, $m \geq 1$ and $m \in \mathbb{N}$, define an array $A[1..n]$ as follow.

$$A[i] = \begin{cases} 1 & \text{if } 1 \leq i \leq m \\ 0 & \text{otherwise} \end{cases}$$

When we sort $A[1..n]$ in non-decreasing order with the insertion sort, the asymptotic running time is $\Theta(f(n))$.
Find a $f(n)$. No proof is necessary.

Consider a special type of sorting case where the input array A[1 . . . n] satisfies: A[j] ∈ {1, 2, · · · , k}, for all j, where j and k are integers and $n >> 2^k$. Does the lower bound for the problem change? If so, does it increase or decrease?

**Specify if the below statement is TRUE or FALSE.**

***In the worst-case linear-time algorithm for selection, the input elements are divided into groups of 5. Will the algorithm work in linear time if they are divided into groups of 3?***

Given adjacency list and below, perform Depth First Search starting with 'B'.

```
adj = dict()
adj['A'] = ['E']
adj['B'] = ['A', 'E', 'C']
adj['C'] = []
adj['D'] = ['A']
adj['E'] = ['D','F']
adj['F'] = ['B']
```

Fill in the three blanks where $n$ is the number of elements.

- Worst-case linear-time selection algorithm runs in $\Theta($ $f(n)$ $)$ time.
- Finding the *second largest* element of a given max-heap requires $\Theta($ $g(n)$ $)$ time in the worst case.
- The modified quicksort below runs in $\Theta($ $h(n)$ $)$ time in the worst case. In the pseudocode below, RANDOMIZEDDIVIDEANDCONQUERSELECT recursively picks a random pivot and partitions the data into two parts.

```
1: function MODIFIEDQUICKSORT(A, p, r)
2:     if p < r then
3:         m ← RANDOMIZEDDIVIDEANDCONQUERSELECT(A, p, r, ⌊(r − p + 1)/2⌋)
4:         q ← MODIFIEDPARTITION(A, p, r, pivot=m)
5:         MODIFIEDQUICKSORT(A, p, q − 1)
6:         MODIFIEDQUICKSORT(A, q + 1, r)
```

- $f(n) =$ ?
- $g(n) =$ ?
- $h(n) =$ ?

Suppose we multiply two $n$ digit quinary (base-5) numbers X and Y using divide-and-conquer algorithms. We divide X into {A and B} and Y into {C and D}, where A, B, C, D are $n/2$ digit quinary (base-5) numbers. Consider two implementations.

$$X = p^{n/2}A + B$$

$$Y = p^{n/2}C + D$$

Following algorithm is the **first implementation**, and the recurrence for the running time of the algorithm is $T(n) = aT(n/b) + \theta(n^c)$.

$$XY = p^{n}AC + p^{n/2}BC + p^{n/2}AD + BD$$

And following is the **second implementation** where the recurrence for the running time is $T(n) = dT(n/e) + \theta(n^f)$.

$$XY = (p^{n} - p^{n/2})AC + p^{n/2}(A+B)(C+D) + (1 - p^{n/2})BD$$

Find the constants $a$, $b$, $c$, $d$, $e$, $f$ and $p$ in the recurrence for the running time. Write your answer in "a, b, c, d, e, f, p" format. (Write only commas between numbers, no alphabets and no special characters are allowed except for commas.)

We define the Fibonacci numbers and we can obtain Fibonacci numbers by using a recursive squaring formula.

$$F_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2; \end{cases}$$

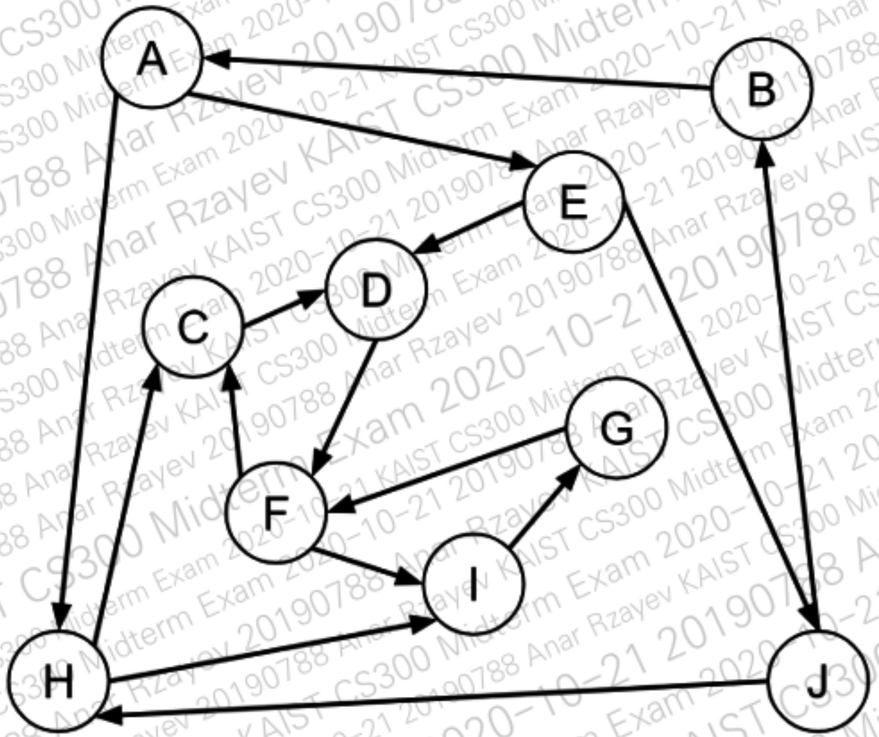
$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1}$$

Now we define a Fibonacci-like sequence and we can obtain Fibonacci-like numbers by using a new recursive squaring formula.

$$G_n = \begin{cases} 3 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ G_{n-1} + G_{n-2} & \text{if } n \geq 2; \end{cases}$$

$$\begin{bmatrix} G_{n+1} & G_n \\ G_n & G_{n-1} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix}^{n-1}$$

Find the constants $a$, $b$, $c$, $d$, $e$, $f$, $g$ and $h$ in the recursive squaring formula. Write your answer in "a, b, c, d, e, f, g, h" format. (Write only commas between numbers, no alphabets and no special characters are allowed except for commas.)




Find all the strongly connected components from the above graph. (Example of answer: (A,B),(C,D,E),…)

What is the correct order of the following three running times? This means, if $T_\alpha(n) = n$ and $T_\beta(n) = n^2$, the answer should be $T_\alpha < T_\beta$. Note that you can assume $T(1) = 1$ for any of these three running times.

- $T_1(n) = 3T(n/3) + n^2$
- $T_2(n) = 8T(n/2) + 1$
- $T_3(n) = n^2 \log^2 n^2 + 1$

Give an asymptotic tight bound on the solution to the following recurrence. You **should justify** your answer. Otherwise, you will get 0 points.

$$T(n) = \begin{cases} 16T(n/2) + 8k & \text{if } n^2 > k \\ k & \text{if } n^2 \leq k \end{cases} \quad (k \text{ is a constant}) \tag{1}$$

Specify whether the following statements are true or false and explain why your answers are correct.

(A) For any graph $G$, the meta-graph of $G$ has no directed cycle.

(B) Let $G$ be a graph with $n$ vertices where $n$ is an even number. For each vertex $v \in G$, if $deg_G(v) \geq n/2$, then $G$ is connected. ($deg_G(v)$ means the number of vertices which are adjacent to $v$).

For any two distinct people **a** and **b**, **a** may or may not know **b**. In the context of this problem, the "knows" relation is not necessarily symmetric: **a** may know **b**, but **b** may not know **a**.

At a party, everyone knows someone else except a celebrity. Now the celebrity joins the party - everyone knows him (the celebrity), but he knows no one.

Let $F$ be an $n \times n$ binary 2D array representing the "knows" relation for n people in the party: $F[a][b]$ = 1 if and only if **a** knows **b**

(A) Design an algorithm to find the celebrity in $O(n^2)$ time. (You can use pseudo code or explain your algorithm only with plain English or Korean.)

(B) Design an algorithm to find the celebrity in $O(n)$ time. (You can use pseudo code or explain your algorithm only with plain English or Korean.)

- You do not have to proof the time complexity of your algorithm. Describing proper algorithm for each problem is enough.