1. 21개의 노드를 가진 그래프가 3개의 트리(tree)로 이루어진 숲(forest)의 형태일 때, 간선(edge)의 총 개수로 가장 옳은 것은?

① 18 ② 19
③ 20 ④ 21

2. 서로 다른 8개의 값에서 두 번째로 큰 값을 항상 찾기 위해 최악의 경우 필요한 최소 비교 횟수로 가장 옳은 것은?

① 8 ② 9
③ 13 ④ 24

3. 순환식 $T(n)=T(n/3)+T(2n/3)+n\ (n>1)$, $T(1)=1$로 표현되는 알고리즘의 점근적 시간 복잡도로 가장 옳은 것은?

① $\Theta(n)$
② $\Theta(n\log n)$
③ $\Theta(n^{1.5})$
④ $\Theta(n^2)$

4. 이진 트리(binary tree)의 전위 순회(preorder traversal)와 중위 순회(inorder traversal) 결과가 <보기>와 같을 때, 이 트리의 후위 순회(postorder traversal) 결과로 가장 옳은 것은?

<보기>
전위 순회: A − B − D − E − C − F
중위 순회: D − B − E − A − C − F

① D − B − E − C − F − A
② D − E − B − C − F − A
③ D − E − B − F − C − A
④ E − D − B − F − C − A

5. <보기>의 텍스트 문자열 T와 패턴 문자열 P에 대해 KMP(Knuth-Morris-Pratt) 알고리즘을 수행하여 인덱스 5에서 불일치(mismatch)가 발생하였다. 다음에 비교할 패턴 문자열 P의 인덱스로 가장 옳은 것은? (단, 패턴 문자열 P의 인덱스는 0부터 시작한다.)

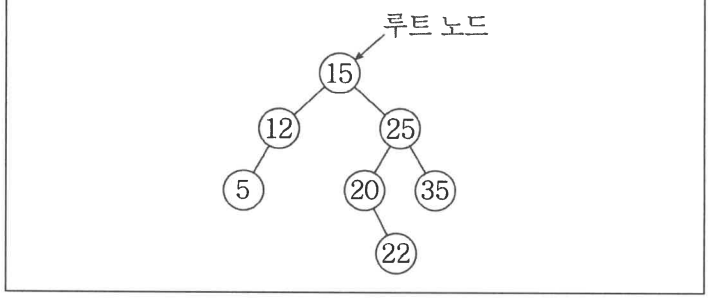
<보기>

	0	1	2	3	4	5	6	7	8	9	10
T	a	b	a	b	a	d	a	a	b	c	d
	=	=	=	=	=	≠					
P	a	b	a	b	a	c					

① 0 ② 1
③ 2 ④ 3

6. <보기>의 이진 검색 트리(binary search tree)에서 노드를 삭제하는 과정에 대한 설명으로 가장 옳지 않은 것은? (단, 노드 명칭과 노드 키 값은 동일하다.)

<보기>



① 루트 노드를 삭제하기 위해 오른쪽 서브트리의 최솟값을 갖는 노드로 대체할 수 있고, 이때의 전체 수행시간은 트리의 높이에 비례한다.
② 노드 12를 삭제하기 위해 노드 15에서 노드 12를 가리키는 포인터가 노드 5를 가리키도록 변경한다.
③ 노드 25를 삭제하기 위해 노드 22로 대체하고, 노드 15에서 노드 25를 가리키는 포인터를 NULL로 변경한다.
④ 노드 5를 삭제하기 위해 노드 12에서 노드 5를 가리키는 포인터를 NULL로 변경한다.

7. <보기 1>은 입력 초기 배열 (a)와 이를 오름차순으로 정렬하는 도중의 배열 (b)의 상태를 나타낸다. 이러한 중간 상태가 나타날 수 있는 정렬 알고리즘을 <보기 2>에서 모두 고른 것은? (단, 각 알고리즘은 배열의 앞쪽부터 처리하는 표준적인 방식으로 구현되었다. 중간 상태는 원소의 교환 또는 이동이 진행되는 중간이 아닌 하나의 원소에 대한 처리가 완료된 시점 또는 버블 정렬에서 한 번의 순환(cycle)이 완료된 시점의 상태이다.)

<보기 1>
(a) 입력 초기 배열: 3, 2, 1, 6, 5, 4
(b) 정렬 중간 배열: 1, 2, 3, 6, 5, 4

<보기 2>
ㄱ. 삽입 정렬(insertion sort)
ㄴ. 선택 정렬(selection sort)
ㄷ. 버블 정렬(bubble sort)

① ㄱ, ㄴ ② ㄱ, ㄷ
③ ㄴ, ㄷ ④ ㄱ, ㄴ, ㄷ

8. 이진 힙(binary heap)과 이를 이용한 힙 정렬(heap sort)에 대한 설명으로 가장 옳은 것은? (단, n은 힙에 저장된 원소의 개수이다.)

① 최소 힙에서 최댓값은 $O(\log n)$ 시간에 찾을 수 있다.
② 삽입식 힙 생성 알고리즘은 입력 순서와 무관하게 $\Theta(n\log n)$ 시간이 필요하다.
③ 힙 정렬은 별도의 배열 없이 제자리(in-place)에서 수행 가능하지만, 안정(stable) 정렬은 아니다.
④ 힙 정렬은 $O(n)$ 시간에 수행 가능하다.

9. <보기>와 같은 간선(edge)을 갖는 방향 그래프에 대하여 가능한 위상 정렬(topological sort) 결과의 개수로 가장 옳은 것은? (단, 정점은 A, B, C, D, E 5개이다.)

<보기>

A→B　A→C　B→E　C→E　D→E

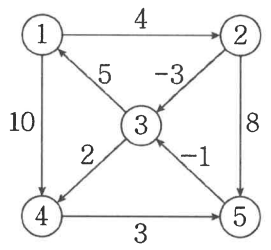
① 4　② 6
③ 8　④ 12

10. 그래프에서 최단 경로를 구하는 벨만-포드(Bellman-Ford) 알고리즘에 대한 설명으로 가장 옳지 않은 것은? (단, N은 정점(vertex)의 개수, M은 간선(edge)의 개수이다. 각 정점은 1, 2, …, N으로 번호가 매겨져 있고, 1번이 시작점이다.)

① 1번 정점에서 도달 가능한 음의 가중치를 갖는 사이클이 그래프에 존재하는지 여부를 판단하는 데 사용할 수 있다.
② 총 N-1번의 라운드로 이루어지며, 라운드 K에서는 정점 1부터 정점 K까지의 최단 경로를 구한다.
③ 시간 복잡도는 O(NM)이다.
④ 사이클이 없는 그래프에서 알고리즘이 수행되는 동안, 모든 정점에 대한 거리 추정치는 실제 최단 경로의 거리와 같거나 이보다 크다.

11. <보기 1>의 가중치 방향 그래프 G에 대해 모든 정점 쌍의 최단 경로를 구하는 플로이드-워셜(Floyd-Warshall) 알고리즘을 수행하는 과정에서 <보기 2>의 행렬 D_2를 얻었다. 다음 단계에서 얻을 행렬 D_3의 2행으로 가장 옳은 것은? (단, $D_k[i, j]$는 정점 i에서 정점 j로 가는 경로 중 중간 정점이 모두 1번부터 k번 사이의 정점인 최단 경로의 길이이고, 그러한 경로가 없으면 ∞이다.)

<보기 1>



<보기 2>

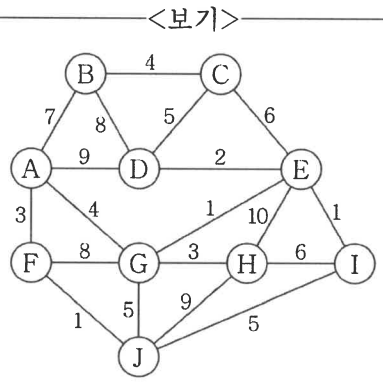
	1	2	3	4	5
1	0	4	1	10	12
2	∞	0	-3	∞	8
3	5	9	0	2	17
4	∞	∞	∞	0	3
5	∞	∞	-1	∞	0

D_2

① (∞, 0, -3, ∞, 8)
② (∞, 0, -3, ∞, 2)
③ (2, 0, -3, -1, 8)
④ (2, 0, -3, -1, 2)

12. <보기>의 그래프에 대해 다익스트라(Dijkstra) 알고리즘을 이용하여 정점 A에서 다른 모든 정점까지의 최단 경로를 찾으려고 한다. 마지막으로 최단 경로를 찾게 되는 정점으로 가장 옳은 것은? (단, 방문할 수 있는 정점이 둘 이상인 경우, 알파벳 순으로 방문한다.)

<보기>

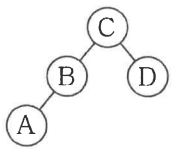


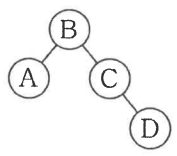
① C　② D
③ H　④ I

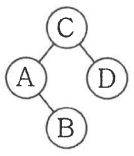
13. <보기>와 같이 입력키와 각각의 검색확률이 주어질 때, 최적의 이진 검색 트리(optimal binary search tree)로 가장 옳은 것은? (단, 최상단 노드가 루트 노드이다.)

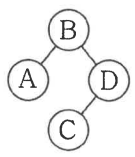
<보기>

입력키	A	B	C	D
검색확률	0.3	0.4	0.1	0.2

① 

② 

③ 

④ 

14. 로봇 청소기가 청소할 지역들을 표현한 $n \times n$ 크기의 행렬 M이 있다. <보기>는 4×4 크기의 행렬 M에 대한 예시로, 행렬의 각 원소 $M_{i,j}$는 지역 (i, j)을 청소했을 때 얻을 수 있는 양의 정수인 청소 점수이다. 한편, 로봇 청소기는 오른쪽이나 아래쪽으로만 이동할 수 있으며, 왼쪽, 위쪽, 대각선 이동은 허용하지 않는다. 로봇 청소기가 지역 (1, 1)에서 청소를 시작하여 지역 (n, n)에 도착할 때까지, 가장 많은 청소 점수를 얻고자 한다. 동적 프로그래밍(dynamic programming)을 이용하여 이 문제를 해결하기 위한 점화식으로 가장 옳은 것은? (단, 테이블 D는 동적 프로그래밍을 위한 $(n+1) \times (n+1)$ 크기의 테이블이고, $D_{i,j}$는 지역 (1, 1)에서 지역 (i, j)까지 이동하면서 얻을 수 있는 최대 청소 점수의 합이다.)

<보기>

행렬 M (예: $M_{3,1}=4$)

	1	2	3	4
1	5	3	2	1
2	1	9	1	2
3	4	2	8	1
4	1	1	3	7

① $D_{i,j}=\begin{cases}0 \\ M_{i,j}+\min(M_{i,j-1},\ M_{i-1,j})\end{cases}$ if $i=0$ or $j=0$, *otherwise*.

② $D_{i,j}=\begin{cases}0 \\ M_{i,j}+\max(M_{i,j-1},\ M_{i-1,j})\end{cases}$ if $i=0$ or $j=0$, *otherwise*.

③ $D_{i,j}=\begin{cases}0 \\ M_{i,j}+\min(D_{i,j-1},\ D_{i-1,j})\end{cases}$ if $i=0$ or $j=0$, *otherwise*.

④ $D_{i,j}=\begin{cases}0 \\ M_{i,j}+\max(D_{i,j-1},\ D_{i-1,j})\end{cases}$ if $i=0$ or $j=0$, *otherwise*.

15. 4×4 체스판에 서로 공격할 수 없도록 퀸 4개를 놓는 N-Queens 문제를 백트래킹(backtracking)으로 해결하고자 한다. 다음 <보기>의 조건에 따라 탐색을 진행할 때, 가장 먼저 찾게 되는 해로 가장 옳은 것은?

<보기>

(조건1) 좌표 체계는 체스판의 가장 왼쪽 위 칸을 (1행, 1열)로 정의한다. 행 번호는 위쪽에서 아래쪽으로 1부터 4까지, 열 번호는 왼쪽에서 오른쪽으로 1부터 4까지 순서대로 부여된다.
(조건2) 탐색 순서는 1행부터 4행까지 행 번호가 증가하는 순서대로 퀸을 배치한다.
(조건3) 후보 시도는 각 행 내에서는 1열부터 4열까지 열 번호가 작은 칸부터 순서대로 퀸 배치를 시도한다.
(조건4) 퀸은 가로, 세로, 대각선 8방향에 대하여 자기 자신이 위치한 칸을 제외한 모든 칸을 공격할 수 있다. 예를 들어, (1행, 1열)에 위치한 퀸은 (1행, 1열)을 제외하고 1행 전체, 1열 전체, (2행, 2열), (3행, 3열), (4행, 4열)을 공격할 수 있다.
(조건5) 백트래킹 규칙은 어떤 칸에 퀸을 놓았을 때 이미 배치된 다른 퀸에게 공격받는 위치라면 같은 행의 다음 열을 시도한다. 만약 특정 행에서 4개의 열을 모두 시도했음에도 퀸을 놓을 수 없다면, 직전 행으로 되돌아가(backtrack) 해당 행에 배치된 퀸의 위치를 다음 열부터 다시 탐색한다.
(조건6) 해는 1행, 2행, 3행, 4행에 놓인 퀸의 열 번호를 순서대로 나열하여 표기한다.

① (1, 2, 3, 4)　② (2, 4, 1, 3)
③ (2, 4, 3, 1)　④ (3, 1, 4, 2)

16. <보기>는 배열 arr에서 k번째로 작은 원소를 찾는 알고리즘이다. (가), (나)에 들어갈 내용으로 가장 옳게 짝지은 것은? (단, partition(arr, p, r) 함수는 탐색 구역의 시작 인덱스 p부터 끝 인덱스 r 사이에서 피벗(pivot)을 기준으로 피벗보다 작은 원소는 왼쪽으로, 큰 원소는 오른쪽으로 재배치하는 분할을 수행한 후, 피벗의 최종 위치 인덱스를 반환한다.)

<보기>

```
select(arr, p, r, k)
    if p == r
        return arr[p]
    q←partition(arr, p, r)
    i←q - p + 1
    if k == i
        return arr[q]
    else if k < i
        return select(arr, p, q-1, (가))
    else
        return select(arr, q+1, r, (나))
```

	(가)	(나)		(가)	(나)
①	k	k − i	②	k	k
③	k − i	k − i	④	k − i	k

17. 연결된 무방향 가중 그래프(G)의 모든 간선의 가중치에 동일한 양수 c를 더하여 새로운 그래프(G')를 만들었다. G와 G'를 비교하였을 때, 최소 신장 트리(minimum spanning tree)의 변화 여부와 모든 두 정점 간 최단 경로의 변화 여부에 대한 설명으로 가장 옳은 것은?

① 최소 신장 트리와 최단 경로가 모두 그대로 유지된다.
② 최소 신장 트리는 그대로 유지되지만, 최단 경로는 달라질 수 있다.
③ 최소 신장 트리는 달라질 수 있지만, 최단 경로는 그대로 유지된다.
④ 최소 신장 트리와 최단 경로가 모두 달라질 수 있다.

18. 곱셈 가능한 4개의 연쇄행렬 $A_1A_2A_3A_4$가 주어졌을 때, 동적 프로그래밍(dynamic programming) 방법으로 최소 곱셈 횟수를 갖도록 연쇄행렬 곱셈(matrix-chain multiplication)의 순서를 찾으려 한다. $1 \le i \le j \le 4$를 만족하는 i, j에 대하여, <보기 1>과 같이 $s[i, j]$에 $A_i \cdots A_j$가 최소 곱셈 횟수를 갖도록 $A_i \cdots A_k$와 $A_{k+1} \cdots A_j$로 분할하는 위치 k를 저장할 때, <보기 2>의 알고리즘으로 출력되는 연쇄행렬 곱셈의 순서로 가장 옳은 것은? (예를 들어, $s[1, 2]=1$)

<보기 1>

	1	2	3	4
1		1	1	1
2			2	2
3				3
4				

테이블 s

<보기 2>

```
PRINT(s, i, j)
   if i == j
        print "A"_i
   else print "("
        PRINT(s, i, s[i, j])
        PRINT(s, s[i, j]+1, j)
        print ")"
```

① $(((A_1A_2)A_3)A_4)$ ② $((A_1(A_2A_3))A_4)$
③ $(A_1((A_2A_3)A_4))$ ④ $(A_1(A_2(A_3A_4)))$

19. <보기>의 최대 구간합 문제를 재귀 전략을 활용하여 해결하는 방법 중, 계산량을 최소화하여 올바른 값을 구하는 방법으로 가장 옳은 것은?

<보기>

[최대 구간합 문제]
Input: 크기가 n인 정수 배열 A (시작 인덱스는 0)
Output: 배열 A의 부분 배열 구간합 중 최댓값
예시) 크기가 3인 정수 배열 [-1 | 2 | -3]의 Output는 2이다.
* 부분 배열 A[i..j]($0 \le i \le j < n$) 구간합: 연속한 부분 배열에 포함된 모든 원소의 합, 즉 A[i]+A[i+1]+…+A[j]를 의미한다.
* 재귀 전략: 배열 A를 전반부 A[0..m−1]과 후반부 A[m..n−1]로 나누어 해결한다. ($m = \lfloor n/2 \rfloor$) (단, n=1이면, A[0]을 반환한다.)

① 전반부의 최대 구간합 a와 후반부의 최대 구간합 b를 재귀적으로 구하고, max(a, b)를 취한다.
② 전반부의 최대 구간합 a와 후반부의 최대 구간합 b를 재귀적으로 구하고, max(a, b, a+b)를 취한다.
③ 전반부의 최대 구간합 a와 후반부의 최대 구간합 b를 재귀적으로 구하고, 시작 위치가 전반부에 있고 끝 위치가 후반부에 있는 모든 부분 배열의 구간합을 각각 계산하여 그중 최댓값 c를 구한 후, max(a, b, c)를 취한다.
④ 전반부의 최대 구간합 a와 후반부의 최대 구간합 b를 재귀적으로 구하고, 전반부에서 A[m-1]을 마지막 원소로 하는 부분 배열의 최대 구간합 c와 후반부에서 A[m]을 첫 원소로 하는 부분 배열의 최대 구간합 d를 각각 구한 후, max(a, b, c+d)를 취한다.

20. NP-완비(complete) 이론에 대한 설명으로 가장 옳지 않은 것은?

① 문제 A가 어떤 알려진 NP-하드(hard) 문제 B로 다항 시간 내 환원된다면 문제 A는 NP-하드이다.
② N개의 노드, M개의 간선이 있는 그래프의 최소 신장 트리를 찾는 문제는 NP에 속한다.
③ NP-complete에 속하는 문제 중 하나를 푸는 다항식 시간 알고리즘이 발견된다면 P=NP가 된다.
④ 무방향 해밀토니안 사이클(Hamiltonian-cycle) 문제는 NP에 속한다.