자 료 구 조 론 (7급)

(과목코드 : 080)

2026년 군무원 채용시험　　응시번호 :　　성명 :

1. 점근적 표기법의 정의에 따라 다음 세 함수가 주어졌을 때, 이에 대한 설명으로 가장 적절한 것은?

$$f(n) = O(n^2),\ g(n) = O(n),\ h(n) = \Theta(n \log n)$$

① $f(n)$의 증가율은 $g(n)$의 증가율보다 크다.
② $g(n)$의 증가율은 $h(n)$의 증가율보다 크다.
③ $f(n)$의 증가율과 $h(n)$의 증가율은 같을 수 있다.
④ 세 함수의 증가율은 모두 같을 수 있다.

2. 재귀적 구현이 사용되는 알고리즘으로 가장 적절하지 않은 것은?
① 트리 레벨 순회
② 그래프 깊이 우선 탐색
③ 하노이탑
④ 퀵정렬

3. 다음과 같은 C언어 함수 xyz(12)를 수행했을 때, 결과 값으로 가장 적절한 것은?

```
int xyz(int n)
{
    if (n <= 1) return 1;
    else if (n % 2 == 1) return n + xyz(n/3);
    else return n + xyz(n-1);
}
```

① 15　　② 21
③ 27　　④ 35

4. C언어를 사용하여 다음과 같은 문자열 배열을 선언했을 때, 출력 값이 다른 하나는?

```
char *a[3] = { "abcde", "fghi", "mno" };
```

① printf("%c\n", a[0][2]);
② printf("%c\n", *a[2]);
③ printf("%c\n", *(*a+2));
④ printf("%c\n", **a+2);

5. 다음은 원형이 아닌 이중 연결 리스트에서 데이터를 저장하고 있는 어떤 노드 p를 삭제하는 함수의 코드이다. 이에 대한 설명으로 가장 적절한 것은? (단, 이 연결 리스트는 데이터를 저장하고 있지 않은 특별한 더미 노드가 맨 앞에 존재하지만, 맨 뒤에는 이러한 더미 노드가 존재하지 않는다)

```
1. e = p->data;
2. p->prev->next = p->next;
3. p->next->prev = p->prev;
4. free(p);
5. return e;
```

① 이 코드는 2번 문장에서만 실행 오류가 발생할 수 있다.
② 이 코드는 3번 문장에서만 실행 오류가 발생할 수 있다.
③ 이 코드는 2번과 3번 두 개 문장에서 모두 실행 오류가 발생할 수 있다.
④ 이 코드는 오류 없이 잘 동작한다.

6. 배열을 이용한 리스트 ADT 구현에 대한 설명으로 가장 적절하지 않은 것은? (단, n은 리스트에 저장된 원소의 개수이다)

① 선형 배열을 사용하는 경우, 삽입 연산의 최악의 경우는 리스트 맨 앞에 삽입하는 경우이다.

② 원형 배열을 사용하는 경우, 삭제할 위치의 배열 인덱스는 $O(1)$ 시간에 계산할 수 있다.

③ 원형 배열을 사용하는 경우, 리스트의 맨 앞의 원소가 리스트의 맨 끝의 원소보다 배열에서 뒤쪽에 위치할 수 있다.

④ 원형 배열이든 선형 배열이든 관계없이, 삭제 연산은 최악의 경우 최대 n-1개의 원소 이동이 필요하다.

7. 일반 리스트(General List) L = (e1, e2, e3, …, en)에 대해, 함수 head(L)은 첫 번째 원소인 e1을 반환하고, 함수 tail(L)은 리스트 (e2, e3, …, en)를 반환한다고 할 때, 다음과 같은 리스트 A에 대한 함수의 수행 결과로 가장 적절하지 않은 것은?

A = (a, b, (c, d))

① head(A) = 'a'

② head(tail(A)) = 'b'

③ head(tail(tail(tail(A)))) = 'c'

④ head(tail(head(tail(tail(A))))) = 'd'

8. 다음은 단순 연결 리스트(Singliy Linked List)에서 마지막 노드를 삭제하는 C언어 함수이다. 빈 칸에 들어갈 내용으로 가장 적절한 것은? (단, 리스트에는 2개 이상의 노드가 있으며, p는 현재 노드를 가리키는 포인터이고, q는 p의 이전 노드를 가리키는 포인터이다)

```
typedef struct node {
    int key; struct node *link;
} ListNode;

void delete_last_node(ListNode *head)
{     ListNode *p, *q;
    q = head;
    p = head->link;
    while (p->link != NULL) {
        [ ㉠ ];
        [ ㉡ ];     }
    free(p);
    q->link = NULL;
}
```

① ㉠ : p = q
   ㉡ : q = q->link

② ㉠ : p = q
   ㉡ : p = p->link

③ ㉠ : q = p
   ㉡ : q = q->link

④ ㉠ : q = p
   ㉡ : p = p->link

9. 후위 수식으로 가장 적절하지 않은 것은? (단, 연산자는 모두 이항연산자이다.)

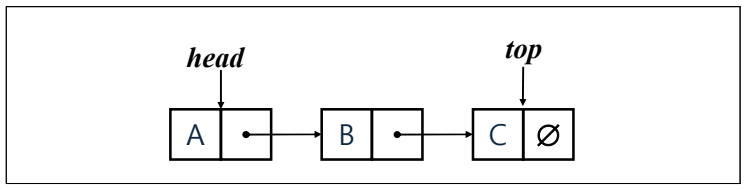
① ABCDEF+-/*+

② ABC++DE**F/

③ AB+CD-*+EF+

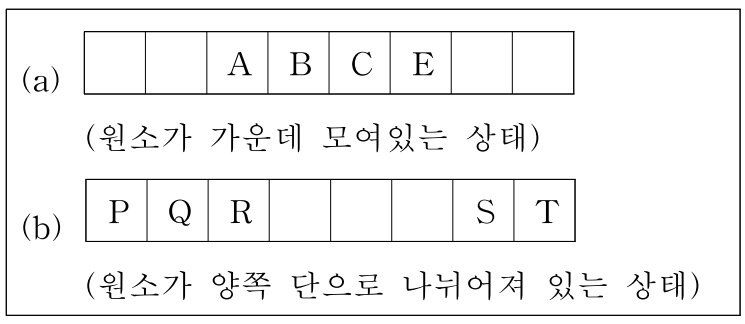
④ AB+C-D*EF/-

10. 스택을 아래와 같은 단일 연결 리스트로 구현하였을 때, 이에 대한 설명으로 가장 적절한 것은? (단, 스택에 가장 최근에 추가된 원소는 C이다)



① push와 pop 둘 다 $O(1)$ 시간에 수행할 수 있다.
② push는 $O(1)$ 시간에 수행할 수 있지만, pop은 $O(1)$ 시간에 수행할 수 없다.
③ push는 $O(1)$ 시간에 수행할 수 없지만, pop은 $O(1)$ 시간에 수행할 수 있다.
④ push도 $O(1)$ 시간에 수행할 수 없고, pop도 $O(1)$ 시간에 수행할 수 없다.

11. 원형 배열을 이용하여 큐(Queue) 또는 데크(Double - Ended Queue, Deque)를 구현하려고 한다. 다음 중 이에 대한 설명으로 가장 적절한 것은?



① 큐는 (a) 상태만 가능하지만, 데크는 (a)와 (b) 상태 모두 가능하다.
② 그림 (b)의 데크에서 T는 가장 최근에 삽입된 원소가 될 수 있다.
③ 데크에서 (a) 상태만 허용하고 (b) 상태를 허용하지 않는다면, 나머지 연산은 필요하지 않다.
④ 배열에 저장된 원소 개수를 저장하는 변수를 유지하지 않으면, 큐와 데크에 저장된 원소의 개수를 $O(1)$ 시간에 알 수 없다.

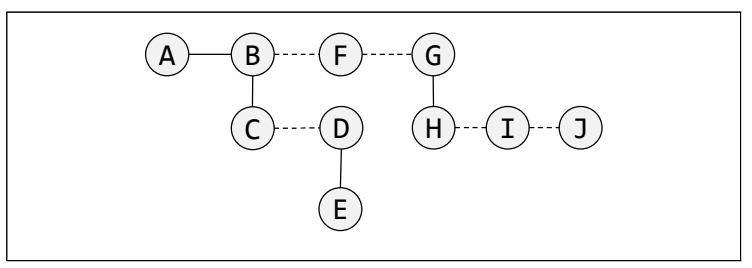
12. 다음과 같은 빈도수를 가지는 5개의 문자가 있다고 하자.

문자	a	b	c	d	e
빈도수	5	3	4	8	11

위에 있는 문자를 사용하여 허프만 코드(Huffman Code)를 생성하였을 때 각 문자에 대한 허프만 코드의 길이로 가장 적절한 것은?

① a : 2비트　　② b : 2비트
③ c : 2비트　　④ d : 3비트

13. 다음은 어떤 트리를 첫째 자식 - 다음 형제 표현으로 나타낸 그림이다. 이 트리에 대한 설명으로 가장 적절한 것은? (단, 실선은 첫째 자식 링크를, 점선은 다음 형제 링크를 나타낸다)



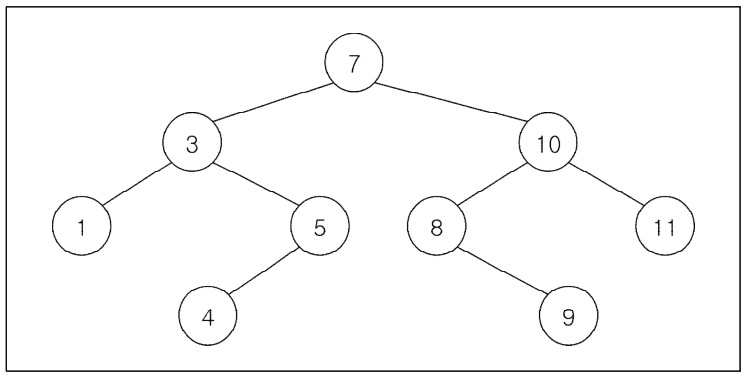
① 이 트리의 레벨 순회 결과는 A - B - C - F - D - E - G - H - I - J이다.
② 이 트리의 차수는 4이다.
③ 이 트리의 높이는 3이다. (단, 루트만 있는 트리의 높이는 1로 가정한다)
④ 이 트리에서 단말 노드(Leaf)의 개수는 6개이다.

14. 다음 이진 트리에 대한 의사코드가 반환하는 값에 대한 설명으로 가장 적절한 것은? (단, v는 루트가 아니고, 모든 내부 노드는 두 개의 자식을 가진다고 가정한다)

```
p ← parent(v)
if (rightChild(p) == v)
        return p
u ← rightChild(p)
while( !isExternal(u) ) // 외부 노드가 아닌 동안
        u ← leftChild(u)
return u
```

① v의 전위순회 계승자
② v의 전위순회 선행자
③ v의 후위순회 계승자
④ v의 후위순회 선행자

15. 다음과 같은 이진 탐색 트리(Binary Search Tree)에서 키 값 7을 가진 노드를 삭제했을 경우에 대한 설명으로 가장 적절한 것은?

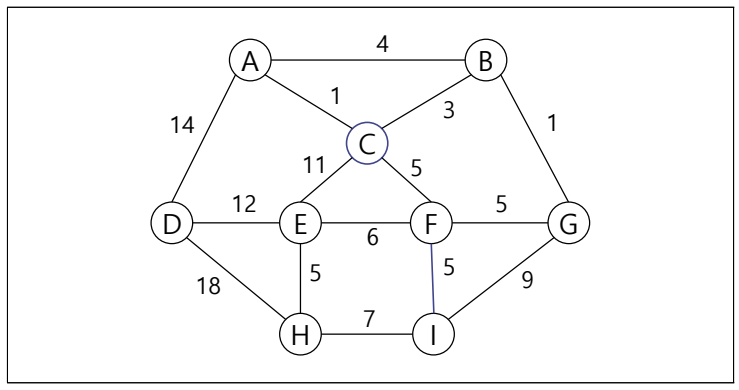


① 키 값 1 또는 키 값 11을 가진 노드가 후계자가 된다.
② 키 값 3 또는 키 값 10을 가진 노드가 후계자가 된다.
③ 키 값 4 또는 키 값 9를 가진 노드가 후계자가 된다.
④ 키 값 5 또는 키 값 8을 가진 노드가 후계자가 된다.

16. 그래프의 깊이 우선 탐색(DFS)과 너비 우선 탐색(BFS)에 대한 설명으로 가장 적절하지 않은 것은?

① 두 탐색 모두 무방향 그래프의 연결성을 검사하기 위해 활용될 수 있다.
② 깊이 우선 탐색은 FIFO 방식이고 너비 우선 탐색은 LIFO 방식의 전략을 사용한다.
③ 두 탐색 방식 모두 방향 그래프 내 사이클 존재 여부를 판단할 수 있다.
④ 그래프를 인접 행렬로 표현했을 때, 두 탐색 알고리즘의 시간 복잡도는 $O(|V|^2)$으로 동일하다.

17. 다음 그래프에 대하여 Kruskal 알고리즘을 수행하는 과정에서 나타날 수 있는 정점의 연결 상태로 가장 적절하지 않은 것은?



① {A, C} {B, G} {D} {E} {F} {H} {I}
② {A, C} {B, G} {D} {E} {F, I} {H}
③ {A, B, C, F, G} {D} {E, H} {I}
④ {A, B, C, F, G, I} {D} {E} {H}

18. Bellman-Ford 알고리즘에서 해가 존재하지 않는지 (즉, 최단 경로 및 거리가 정의될 수 없는지) 판단하는 코드로 가장 적절한 것은? (단, 해 존재 여부 판단은 그래프의 모든 간선을 확인하며 거리 갱신 작업을 (|V|-1)번 반복한 후에 수행한다고 가정하라. 또한, d[v]는 정점 v의 거리로 계산된 값, w(u,v)는 간선 (u,v)의 가중치를 의미한다)

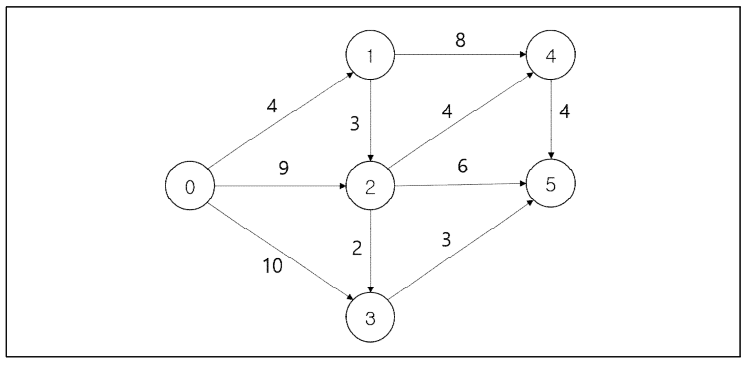
① for each u ∈ V
    if(d[u] > 0) then output "해 없음"

② for each u ∈ V
    if(d[u] < 0) then output "해 없음"

③ for each (u,v) ∈ E
    if(d[u] + w(u,v) > d[v])
        then output "해 없음"

④ for each (u,v) ∈ E
    if(d[u] + w(u,v) < d[v])
        then output "해 없음"

19. 다음과 같은 방향 그래프(Directed Graph)에 대해 Dijkstra 알고리즘을 이용하여 시작 정점 0에서 다른 모든 정점까지 최단 경로를 구해 배열에 저장한다고 할 때, 중간 단계로 나타나는 배열의 모습으로 가장 적절하지 않은 것은?



①

[0]	[1]	[2]	[3]	[4]	[5]
0	4	7	9	11	12

②

[0]	[1]	[2]	[3]	[4]	[5]
0	4	7	9	11	13

③

[0]	[1]	[2]	[3]	[4]	[5]
0	4	7	9	11	15

④

[0]	[1]	[2]	[3]	[4]	[5]
0	4	7	10	12	∞

20. 다음 입력 전체에 대해 퀵 정렬의 분할 함수를 한 번 수행한 후 가능한 전체 원소 나열 상태로 가장 적절한 것은? (단, 분할 기준으로 가장 마지막 원소를 사용하고, 최종적으로 기준 원소를 제자리에 위치시킨다.)

2 - 3 - 6 - 8 - 7 - 4 - 1 - 5

① 2 - 3 - 1 - 4 - 5 - 8 - 6 - 7
② 2 - 3 - 1 - 5 - 4 - 6 - 7 - 8
③ 1 - 4 - 2 - 3 - 6 - 5 - 7 - 8
④ 1 - 3 - 6 - 4 - 5 - 8 - 2 - 7

21. 다음 의사코드가 나타내는 정렬 알고리즘으로 가장 적절한 것은? (단, 배열 A의 인덱스는 0부터 n - 1까지이다)

```
for pass ← 1 to n - 1
    save ← A[pass],  j ← pass - 1
    while ( (j ≥ 0) & (A[j] > save) )
        A[j + 1] ← A[j]
        j ← j - 1
    A[j + 1] ← save
```

① 선택정렬
② 삽입정렬
③ 쉘정렬
④ 기수정렬

22. 탐색을 위한 자료구조 및 알고리즘에 대한 설명으로 가장 적절한 것은?

① 해싱은 키들의 대소 관계를 기반으로 만들어진 탐색 자료구조가 아니다.
② 이진 탐색 트리의 키 삽입 연산은 루트에서 삽입할 위치의 단말노드까지 따라 내려가는 작업이 필요하므로 $O(\log n)$ 시간에 수행된다.
③ 루트 노드의 왼쪽 서브 트리와 오른쪽 서브 트리의 높이 차이가 상수 배로 보장되면 균형 이진 탐색 트리가 될 수 있다.
④ 순서 배열에서의 키 삭제 연산은 이진 탐색을 통해 키가 삭제될 위치를 찾으면 되므로 $O(\log n)$ 시간에 수행된다.

23. 다음 원소들을 빈 이진 탐색 트리에 순서대로 삽입할 때, 최종 트리의 루트 노드와 높이는? (단, 루트만 있는 트리의 높이는 1이라고 가정한다)

48, 15, 33, 62, 39, 7, 40, 26, 54

① 39, 4
② 39, 5
③ 48, 4
④ 48, 5

24. 공백 트리에 키 값 2, 1, 8, 9, 7, 3, 6을 차례대로 삽입하여 2-3-4 트리를 생성하면 1개의 4-노드가 만들어진다. 4-노드에 있는 원소로 가장 적절한 것은?
① 2
② 6
③ 8
④ 9

25. 테이블 크기가 $m = 7$인 해시 테이블에 다음 키들이 순서대로 삽입될 때, 최종 해시 테이블의 상태로 가장 적절한 것은? (단, 충돌은 2차 조사법 $h(k, i) = (k + i^2) \bmod m$을 사용하여 해결한다)

15, 12, 22, 29, 9

①

[0]	[1]	[2]	[3]	[4]	[5]	[6]
-	15	22	9	29	12	-

②

[0]	[1]	[2]	[3]	[4]	[5]	[6]
-	15	22	29	-	9	12

③

[0]	[1]	[2]	[3]	[4]	[5]	[6]
-	15	22	29	9	12	-

④

[0]	[1]	[2]	[3]	[4]	[5]	[6]
-	15	22	29	-	12	9