자료구조론	국가 전산 7급	2019년 8월 17일

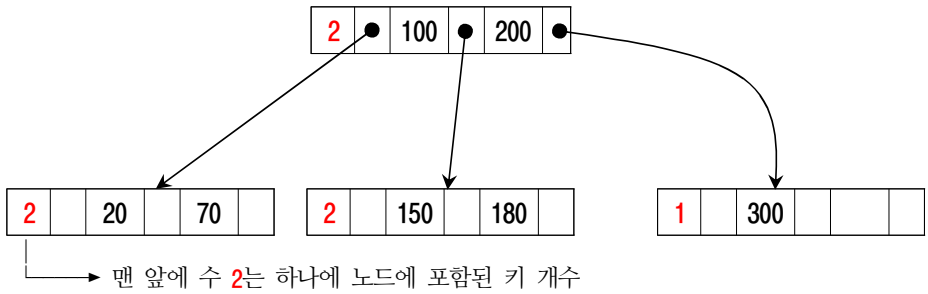
♤ 합격선/최종합격인원(74.16점/33명) - 선발예정인원 30명 ♤

1. m원 탐색트리(m-way search tree)에 대한 설명으로 옳지 않은 것은? [2019년 국가 7급]

① 공백(empty)이 아닌 m원 탐색트리의 루트(root) 노드는 최대 m개의 서브-트리(sub-tree)를 가질 수 있다.
② 공백이 아닌 m원 탐색트리의 루트노드에 저장된 키(key) 들은 정렬되어 있어야 한다.
③ 공백이 아닌 m원 탐색트리의 각 노드가 가지는 서브-트리 역시 m원 탐색트리이다.
④ 차수(degree)가 m이고 높이(height)가 h인 다원 탐색트리가 가질 수 있는 최대 노드 수는 (m^h-1)이다. (단, 루트만 있는 트리의 높이는 1로 한다)

♤ **m원 탐색트리(m-way search tree)**

- **이진탐색트리**를 **일반화**시킨 형태로 모든 노드들이 m 이하의 차수를 갖는 탐색트리이다.
- 각 노드는 최대 m-1개의 키를 가질 수 있다.
- 다음은 **3**원 탐색트리(m=3)이다.



- m=3이므로, 각 노드는 최대 m - 1 = 3 - 1 = 2(개)의 키를 가질 수 있다.
- 차수가 m이고 높이가 h이면, **최대 노드수**는 $(m^h-1)/(m-1)$이다.
- 차수가 m이고 높이가 h이면, **최대 원소수**는 (m^h-1)이다.
- m=3이고, h=2이면, **최대 노드수** = $(m^h-1)/(m-1)$ = $(3^2-1)/(3-1)$ = **4**

// m원 탐색트리의 노드 구조

n	A0	K1	A1	K2	···	Ai-1	Ki	Ai	···	An-1	Kn	An

n : 하나의 노드에 포함된 킷값의 수($1 \le n < m$)
Ai : 서브트리에 대한 포인터
Ki : 노드에 포함된 i번째 킷값($Ki < Ki+1$)

정답 : ④

2. 다음은 C 언어로 구현한 원형큐(circular queue) 삽입 알고리즘이다. ㉠, ㉡에 들어갈 내용으로 옳은 것은? [2019년 국가 7급]

```
-----------------------------------------
#define MAX_QUEUE_SIZE 10
int queue[MAX_QUEUE_SIZE];
int front = rear = -1;
void addq(int front, int *rear, int item) {
    _____㉠_____
  if (_____㉡_____) {
    printf("Queue is full!!\n");
    return -1;
  }
  queue[*rear] = item;
}
-----------------------------------------
```

	㉠	㉡
①	*rear=(*rear+1)%MAX_QUEUE_SIZE;	front==*rear
②	*rear=(front+1)%MAX_QUEUE_SIZE;	front==*rear+1
③	front=(front+1)% MAX_QUEUE_SIZE;	front==*rear
④	front=(*rear+1)%MAX_QUEUE_SIZE;	front==*rear+1

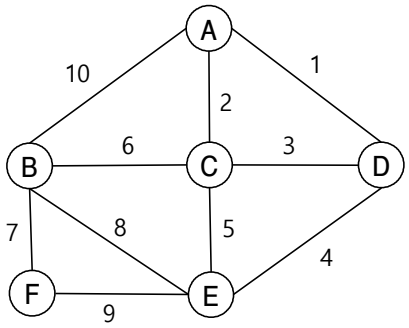
♤ **원형큐 삽입 알고리즘**

• **공백조건**의 원형큐 삽입 알고리즘이다. 분석해 보면

```
void addq(int front, int *rear, int item)   //*rear으로 선언된 것에 주의할 것!
{
  ㉠ *rear=(*rear+1)%MAX_QUEUE_SIZE; //*rear 값을 증가시키고
  if (㉡ front==*rear)               //Queue가 포화상태이면
  {
    printf("Queue is full!!\n");
    return -1;
  }
  queue[*rear] = item;              //자료 입력
}
```

정답 : ①

3. 다음 그래프를 Kruskal 알고리즘을 이용하여 최소비용 신장트리로 만들 때 선택되지 않는 간선은? (단, 알고리즘은 노드 A에서 시작한다) [2019년 국가 7급]



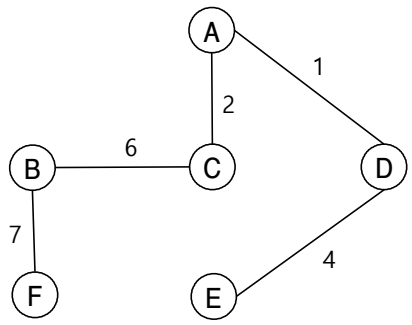
---

① (A, C)　② (B, C)
③ (B, F)　④ (C, D)

♤ Kruskal 알고리즘

- 그래프의 모든 간선을 비용 값을 기준으로 오름차순 정렬한 뒤
- 비용 값이 가장 작은 간선부터 선택하여
- 선택된 간선이 사이클을 이루지 않으면 신장트리에 포함시킨다.

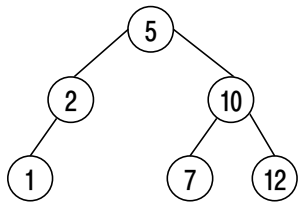
- 최소비용 신장트리는 다음과 같다.



- 간선(C, D)는 선택되지 않는다. 사이클이 형성되므로

정답 : ④

4. 다음은 이진트리(binary tree)이다. 매개변수 ptr로 함수 func를 호출하였을 때, 결과 값은? (단, ptr은 루트노드에 대한 포인터이다) [2019년 국가 7급]




```
struct node {
  int data;
  struct node *left;
  struct node *right;
};
int func(struct node *ptr) {
  if (ptr == NULL)                                          //완료조건
     return 0;
  else if (ptr->left == NULL && ptr->right == NULL)         //완료조건(단말노드인 경우)
     return 1;
  else
     return (ptr->data + func(ptr->left) + func(ptr->right)); //재귀호출
}
```

---

① 10　　② 17

③ 20　　④ 37

♤ 이진트리 재귀호출

• ptr->data + func(ptr->left) + func(ptr->right)

↓

= 5 + func(ptr->left) + func(ptr->right)

= 5 + 2 + func(ptr->left) + func(ptr->right)

= 5 + 2 + 1 + func(ptr->right)　　//여기서, 1은 완료조건의 return 1;

= 5 + 2 + 1 + 10 + func(ptr->left) + func(ptr->right)

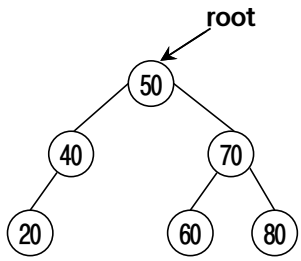
= 5 + 2 + 1 + 10 + 1 + func(ptr->right)　　//여기서, 1은 완료조건의 return 1;

= 5 + 2 + 1 + 10 + 1 + 1　　//여기서, 1은 완료조건의 return 1;

= 20

정답 : ③

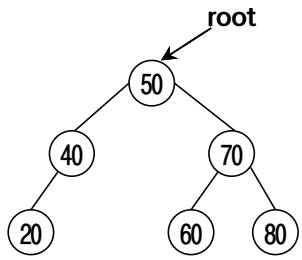
//앞의 이진트리 재귀호출 프로그램을 완성한 것이다.
//프로그램 구조와 데이터는 조금 변형하였다. 이해하기 쉽도록




```
#include <stdio.h>
#define  NULL  0
typedef struct NODE btree;
struct NODE{
    int     data;
    btree *left;
    btree *right;
} *ROOT = NULL;

btree *make_tree(btree *root, int key){                    //이진탐색트리 생성
    if(root==NULL){
        root = (btree *)malloc(sizeof(btree));
        root->data = key;
        root->left = root->right = NULL;
    }
    else if(key < root->data) root->left = make_tree(root->left, key);
    else root->right = make_tree(root->right, key);
    return root;
}

int func(btree *root) {
  if (root == NULL)                                        //완료조건
     return 0;
  else if (root->left == NULL && root->right == NULL)      //완료조건(단말노드인 경우)
     return 1;
  else
     return (root->data + func(root->left) + func(root->right)); //재귀호출
 }

void main(){
    int i, key[] = {50, 40, 70, 20, 60, 80};
    btree *root = NULL;
    ROOT = make_tree(root, key[0]); root = ROOT;
    for(i = 1; i < sizeof(key)/sizeof(int); i++) root = make_tree(root, key[i]);
    printf("\n");
    printf("%5d\n", func(root));              //출력 163 (50 + 40 + 1 + 70 + 1 + 1 = 163)
}
```

◈ 재귀함수 분석

```
int func(btree *root) {
  if (root == NULL)                                          //완료조건
     return 0; //공백트리이거나 자식이 없는 경우 0을 반환
  else if (root->left == NULL && root->right == NULL)        //완료조건(단말노드인 경우)
     return 1; //root가 단말노드를 가리키면 1을 반환
  else
     return (root->data + func(root->left) + func(root->right));   //재귀호출
 }
```



• root->data + func(root->left) + func(root->right)

↓

↓ root->data = 50

↓

= 50 + **func(root->left)** + func(root->right)

↓

↓ 재귀호출은 자신 모습 그대로 호출한다.

↓

= 50 + **root->data + func(root->left) + func(root->right)** + func(root->right)

↓ root->data = 40

= 50 + **40 + func(root->left) + func(root->right)** + func(root->right)

↓ 단말노드 20에 도착, 완료조건의 return 1;

= 50 + **40 + 1 + func(root->right)** + func(root->right)

↓ func(root->right)==NULL이므로 return 0;

= 50 + **40 + 1 + 0** + func(root->right)

↓ 재귀호출은 자신 모습 그대로 호출한다.

= 50 + 40 + 1 + 0 + **root->data + func(root->left) + func(root->right)**

↓ root->data = 70

= 50 + 40 + 1 + 0 + **70 + func(root->left) + func(root->right)**

↓ 단말노드 60에 도착, 완료조건의 return 1;

= 50 + 40 + 1 + 0 + **70 + 1 + func(root->right)**

↓ 단말노드 80에 도착, 완료조건의 return 1;

= 50 + 40 + 1 + 0 + **70 + 1 + 1**

= **163**

5. n×n 하삼각 행렬은 총 n(n+1)/2개의 원소를 갖는다. 원소 a_{ij}(1≤i≤j≤n)가 최소 공간을 사용하도록 1차원 배열 b[k]에 행우선순서(row-major order)로 저장했을 때, 하삼각 행렬의 원소 a_{ij}가 저장될 배열 b[k]의 색인 k를 계산하는 식으로 옳은 것은? (단, i는 행 색인 값, j는 열 색인 값으로 하고, 1차원 배열 b에는 하삼각 행렬의 0 값은 저장되지 않으며, b의 색인 k의 값은 0부터 시작한다) [2019년 국가 7급]

정방행렬(square matrix) 중에서 대각선보다 위의 모든 원소가 0인 경우를 특별히 하삼각 행렬(lower triangular matrix)이라 하며, 다음은 4 × 4 하삼각 행렬의 예이다. (단, a_{ij}는 0이 아닌 어떤 실수 값을 의미하고, 0은 반드시 0이어야 함을 의미한다)

$$\begin{pmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

---

① (i − 1) × n + (j − 1)
② j × (j − 1) / 2 + (i − 1)
③ i × (i − 1) / 2 + (j − 1)
④ (j − 1) × n + (i − 1)

♤ n×n 하삼각 행렬

• 이런 문제는 **공식**을 **유도**하지 않고, 다음처럼 대입하여 풀어도 된다.**(객관식이므로)**

$$\begin{pmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

↓1차원 배열 b[k]에 저장된 모습

k	0	1	2	3	4	5	6	7	8	9	…
b[k]	a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}	a_{44}	…

↓ a_{ij} = a_{21}인 경우

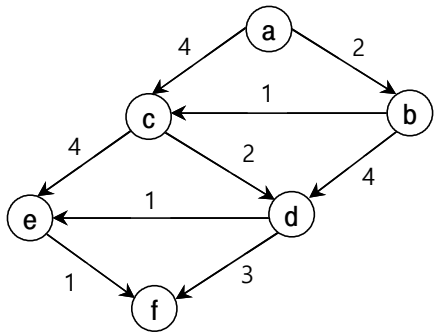
• k = i × (i − 1) / 2 + (j − 1) = 2 × (2 − 1) / 2 + (1 − 1) = **1**

↓ a_{ij} = a_{42}인 경우

• k = i × (i − 1) / 2 + (j − 1) = 4 × (4 − 1) / 2 + (2 − 1) = **7**

정답 : ③

6. 다음과 같은 방향그래프에 대하여 Dijkstra의 알고리즘을 적용하여 최단경로를 구하고자 한다. 노드 a에서 시작하여 노드 f로 가는 최단경로를 찾아 가는 과정에서 노드 a에서부터 다른 노드까지 최단경로를 차례로 알게 된다. 이 과정에서 알게 되는 최단경로의 노드 순서로 옳은 것은?



---

① a-b-c-d-e-f
② a-b-d-f
③ a-c-d-f
④ a-c-d-e-f

♤ **Dijkstra 최단경로 알고리즘 – 최단경로가 발견된 정점들의 순서**

• 간선에 가중값이 부여된 방향그래프에서 두 정점 사이에는 하나 이상의 경로가 존재할 수 있다.
• 이때 경로상의 가중값의 합이 최소인 경로를 두 정점 사이의 최단경로라 한다.

• 방문하게 되는 정점 순서는 비용의 합이 적은 순서부터이다.

// 정점 a에서 다른 모든 정점까지의 최단경로를 구할 때 방문하게 되는 정점 순서

• a→b : 2
• a→b→c : 2 + 1 = 3
• a→b→c→d : 2 + 1 + 2 = 5
• a→b→c→d→e : 2 + 1 + 2 + 1 = 6
• a→b→c→d→e→f : 2 + 1 + 2 + 1 + 1 = 7

• 방문 순서 : **a-b-c-d-e-f**

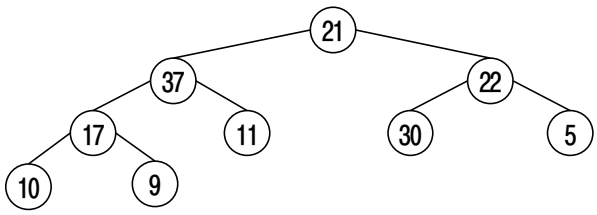
정답 : ①

7. 다음과 같이 배열에 저장된 데이터에 대하여 내림차순으로 히프정렬(heap sort)을 수행할 때, 첫 번째 데이터를 출력하고 히프를 재구성 한 후에 배열의 여섯 번째에 있는 값은? (단, 배열의 첫 번째 색인 값은 1이다) [2019년 국가 7급]

1	2	3	4	5	6	7	8	9
21	37	22	17	11	30	5	10	9

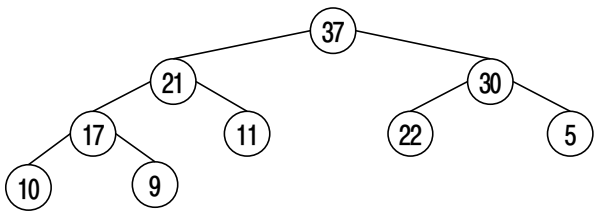
① 5 ② 9 ③ 10 ④ 11

♤ **내림차순 히프정렬 – 최대힙 이용**

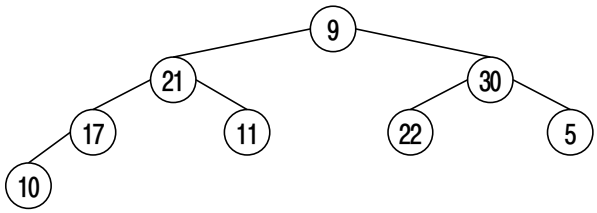


• 완전이진트리

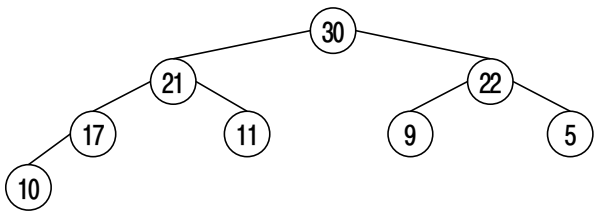
↓초기 힙구조로 고치면(근노드를 가장 큰 값으로)



↓첫 번째 데이터 출력(즉, 근노드 값37을 출력하고)
↓마지막 노드 9를 근드로 이동(완전이진트리 유지)



↓히프를 재구성



• 6번째 데이터는 **9**

정답 : ②

8. 6개 외부노드의 가중치가 각각 q1=3, q2=4, q3=6, q4=8, q5 =10, q6=15인 허프만 트리의 설명으로 옳은 것은? (단, 허프만 트리는 $\sum_{1 \le i \le n} q_i d_i$가 최소가 되는 트리로서, q_i는 외부노드의 가중치이고 d_i는 루트노드에서부터 외부노드까지의 거리이며, n은 외부노드의 수이다) [2019년 국가 7급]

① 외부노드에서 루트까지의 거리와 외부노드의 가중치 곱의 총합($\sum_{1 \le i \le n} q_i d_i$)은 112이다.

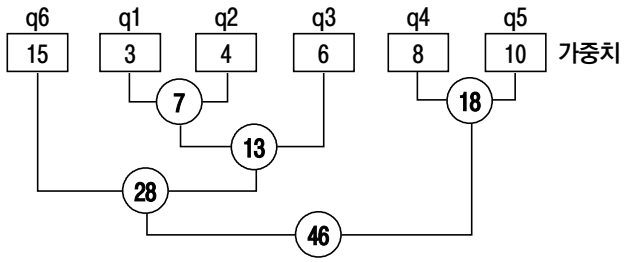
② 루트까지의 거리가 가장 긴 외부노드에서 루트까지 거리는 5이다.

③ 루트까지의 거리가 가장 긴 외부노드는 q1, q2, q3이다.

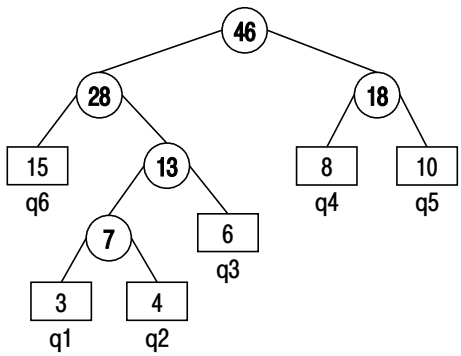
④ 루트까지의 거리가 가장 짧은 외부노드는 q6이고 그 다음으로 짧은 외부노드는 q4와 q5이다.

♤ **허프만 트리**

• 먼저, 허프만 트리를 그리면



↓거꾸로 뒤집으면



• 외부노드의 가중치 곱의 총합 = 15 • 2 + 3 • 4 + 4 • 4 + 6 • 3 + 8 • 2 + 10 • 2 = **112**

정답 : ①

9. 다음과 같이 배열에 저장된 입력자료들을 퀵정렬(quick sort)로 오름차순 정렬하고자 한다. 정렬과정에서 단계별 정렬 순서로 나타날 수 없는 것은? (단, 피봇(pivot)은 마지막 레코드로 선택한다) [2019년 국가 7급]

---
15 3 1 35 20 4 18 26
---

① 15, 3, 1, 18, 20, 4, 26, 35
② 15, 3, 1, 4, 20, 18, 26, 35
③ 1, 3, 4, 18, 20, 15, 26, 35
④ 1, 3, 4, 15, 20, 18, 26, 35

♤ 퀵정렬 – 피봇키는 마지막 원소

	큰값(a++) →					← 작은값(b--)				
	0	1	2	3	4	5	6	7	left	right
초기배열	15	3	1	35	20	4	18	**26**	0	7
				↑ a			↑ b		35과 18을 교환	
	15	3	1	18	20	4	35	**26**	인덱스 교차	
						↑ b	↑ a		35와 26을 교환	
1 pass	15	3	1	18	20	**4**	[26]	35		
	↑ a		↑ b						15와 1을 교환	
	1	3	15	18	20	**4**	[26]	35	인덱스 교차	
		↑ b	↑ a						4와 15를 교환	
2 pass	1	**3**	[4]	18	20	15	[26]	35	부분 정렬 완료	
	↑ a, b									
3 pass	1	3	[4]	18	20	**15**	[26]	35	15보다 작은 값 없음	
				↑ a, b					18과 15를 교환	
4 pass	1	3	[4]	[15]	20	**18**	[26]	35	18보다 작은 값 없음	
					↑ a, b				20과 18을 교환	
5 pass	1	3	[4]	[15]	[18]	20	[26]	35	정렬 완료	

// 마지막 원소를 제어키로 사용하면,

- 인덱스 a, b가 교차되면, 피벗키를 a가 가리키는 키와 교환한다.
- 리스트는 피벗키를 기준으로 양분된다.

정답 : ②

10. 다음과 같이 중위표기법(infix notation)으로 되어 있는 수식을 후위표기법(postfix notation)으로 변환하기 위해 스택을 이용한다. 변환 과정에서 스택에 토큰이 가장 많이 쌓여 있을 때의 스택 내 연산자 토큰 수는? [2019년 국가 7급]

```
----------------------
a * (b - c * d) / e
----------------------
```

① 3개 ② 4개
③ 5개 ④ 6개

♤ **중위표기법을 후위표기법으로 변환**

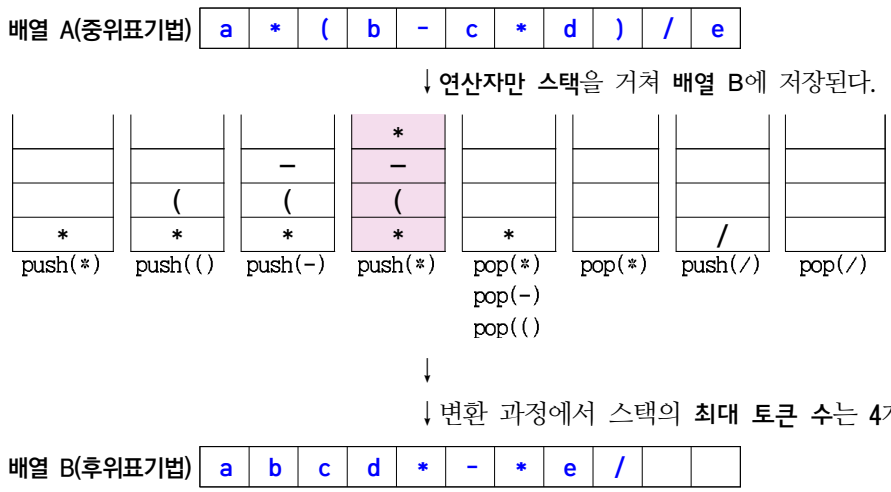
1. 먼저, 중위표기법을 저장하는 배열 A와 후위표기법을 저장하는 배열 B를 준비한다.
2. 배열 A의 피연산자와 연산자를 차례로 읽어서, 다음 규칙에 따라 배열 B로 옮긴다.

① **피연산자** : 피연산자는 **스택을 거치지 않고** 바로 배열 B에 저장된다.

② **연산자** : 연산자는 스택에 임시 저장되는데 다음 **규칙**에 의한다.

규칙	• **읽은 연산자**가 스택의 top이 가리키는 연산자보다 연산우선순위가 **빠르면, 그냥 스택에 저장** • **우선순위가 같거나 늦으면**, 스택의 top이 가리키고 있는 **연산자를 꺼내 후위식에 기록**한다. • 그리고, 방금 읽은 자신(연산자)은 스택에 저장된다. • 배열 A를 읽는 도중에 여는 괄호 '('는 무조건 스택에 저장한다. [괄호도 연산자로 취급] • 여는 괄호 **'('는 연산우선순위가 가장 늦은 것으로 취급**한다. • 닫는 괄호 **')'를 만나면 '('를 만날 때까지 스택에서 꺼내 후위식에 기록**하는데, '('는 버린다.

3. 배열 A의 피연산자를 배열 B로 모두 옮겼으면, 스택의 연산자도 모두 배열 B로 옮긴다.



정답 : ②

11. 그래프에 대한 설명으로 옳은 것만을 모두 고르면? [2019년 국가 7급]

ㄱ. 무방향그래프(undirected graph)를 인접행렬(adjacency matrix)로 표현하면 주대각선을 중심으로 상위행렬과 하위행렬이 대칭이 된다.
ㄴ. 사이클이 없고 n - 1개의 간선(edge)을 갖는 연결그래프(connected graph)를 트리(tree)라고 한다.
ㄷ. n개의 정점(vertex)을 가진 방향그래프의 최대 간선 수는 n(n+1)/2개이다.
ㄹ. 경로를 구성하는 간선의 수를 길이라고 할 때, 방향그래프에서 사이클(cycle)의 길이는 항상 3 이상이다.

---

① ㄱ, ㄴ ② ㄱ, ㄷ ③ ㄴ, ㄷ ④ ㄷ, ㄹ

♤ 그래프

ㄷ. n개의 정점(vertex)을 가진 방향그래프의 최대 간선 수는 **n(n+1)/2**개이다.**(×)**
→ n개의 정점(vertex)을 가진 방향그래프의 최대 간선 수는 **n(n−1)**이다.
ㄹ. 경로를 구성하는 간선의 수를 길이라고 할 때, 방향그래프에서 사이클(cycle)의 길이는 항상 **3** 이상이다.**(×)** → 방향그래프에서 사이클(cycle)의 길이는 **2**가 될 수도 있다. 〈a, b〉와 〈b, a〉

정답 : ①

12. 일반리스트(general list) L = (e1, e2, ... , 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 결과 중 문자 'c'를 반환하는 것은? [2019년 국가 7급]

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

① head(head(A)) ② head(tail(A))
③ head(tail(head(A))) ④ head(tail(tail(A)))

♤ 일반리스트

---

① head(head(A)) = head(a, b) = a
② head(tail(A)) = head(c, d) = **c**
③ head(tail(head(A))) = head(tail(a, b)) = head(b) = b
④ head(tail(tail(A))) = head(tail(c, d)) = head(d) = d

정답 : ②

13. 다음은 단순연결리스트(singly linked list)에서 특정 값을 가진 노드의 개수를 반환하는 C 함수이다. ㉠, ㉡에 들어갈 내용으로 옳은 것은? [2019년 국가 7급]

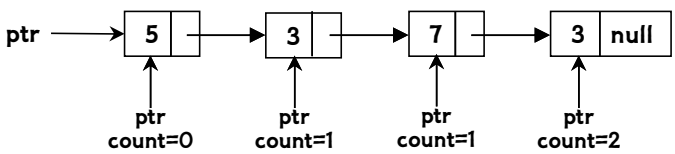
```
-----------------------------------------------
struct node {
  int data;
  struct node *link;
};
int countNode(struct node *ptr, int value) {
  int count = 0;
  while (_____㉠_____) {
    if ( ptr->data == value ) count++;
       _____㉡_____
  }
  return count;
}
-----------------------------------------------
```

	㉠	㉡
①	ptr == NULL	ptr->link = ptr;
②	ptr == NULL	ptr = ptr->link;
③	ptr != NULL	ptr->link = ptr;
④	ptr != NULL	ptr = ptr->link;

♤ 단순연결리스트

```
while (㉠ ptr != NULL) {          //공백리스트 또는 마지막 노드의 NULL이 아닌 경우
  if (ptr->data == value) count++;  //특정 값을 가진 노드 카운터
    ㉡ ptr = ptr->link;             //리스트 추적
}
```

// 특정 값(3)을 가진 노드 카운터



- 그림은 특정 값 3을 가진 노드를 카운터하는 것을 보여준다.
- ptr = ptr->link; //리스트 추적
- ptr == NULL이 되면 프로그램은 종료된다.

정답 : ④

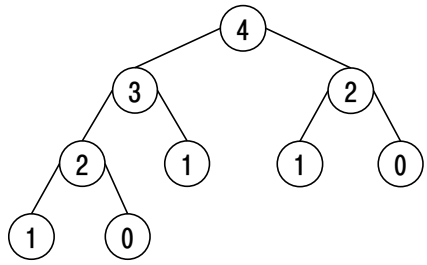
14. 다음은 Fibonacci 수열을 계산하는 C 함수이다. 함수 fibonacci(4)를 호출하였을 때 화면에 출력되는 숫자의 순서로 옳은 것은? [2019년 국가 7급]

```
-----------------------------------------
int fibonacci(int n) {
  printf("%3d", n);
  if(n==0) return 0;
  else if(n==1) return 1;
  else return fibonacci(n-1) + fibonacci(n-2);
}
-----------------------------------------
```

① 4 3 2 2 1 1 0 1 0　　② 4 3 2 1 0 2 1 1 0
③ 4 3 2 1 0 1 2 1 0　　④ 4 3 2 1 0 2 1 0 1

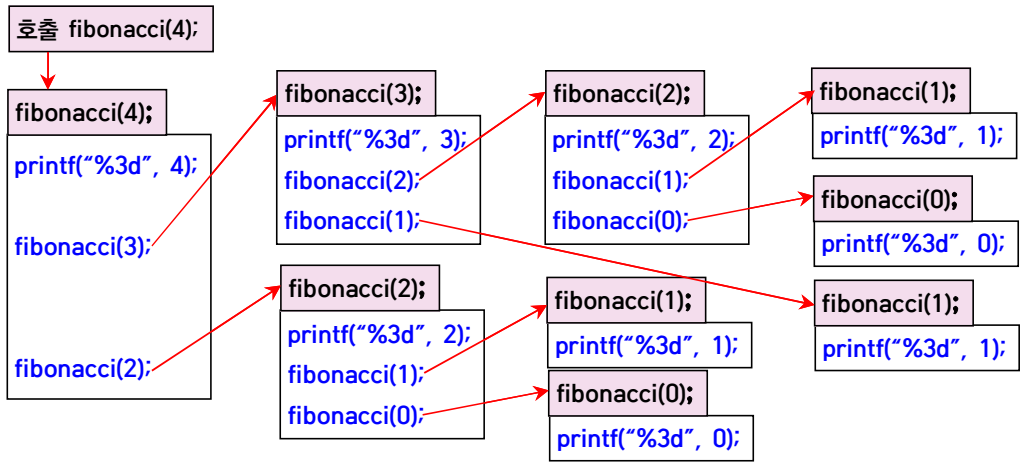
♤ Fibonacci 수열

• 먼저, 출력되는 숫자는 다음 재귀트리를 전위운행하면 된다.(출력문이 먼저 있는 경우)



전위운행(중,좌,우) : 4 3 2 1 0 1 2 1 0

// 재귀호출 수행 과정을 그림으로 나타내면 다음과 같다.



정답 : ③

15. 다음은 이진탐색을 수행하는 C 함수이다. 이 함수를 사용하여 0부터 20까지의 정수가 차례대로 저장되어 있는 배열 b에 대한 이진탐색을 수행할 때, key 값과 a[middle] 값을 비교하는 횟수가 다른 것은? (단, 배열의 색인은 0부터 시작한다) [2019년 국가 7급]

```
int binarySearch(int *a, int key, int low, int high) {
  int middle;
  while (low <= high) {
    middle = (low + high) / 2;
    if (key == a[middle])
        return middle;
    else if (key > a[middle])
        low = middle + 1;
    else
        high = middle - 1;
  }
  return -1;
}
```

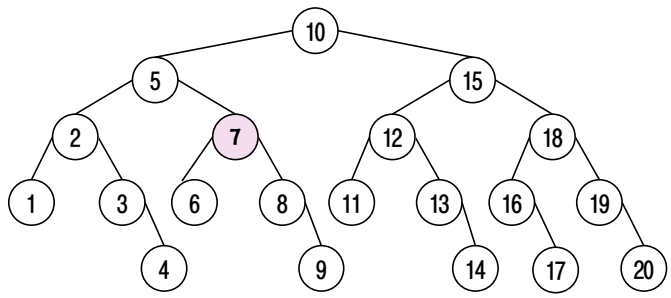
① binarySearch(b, 4, 1, 20)　② binarySearch(b, 7, 1, 20)
③ binarySearch(b, 9, 1, 20)　④ binarySearch(b, 14, 1, 20)

♤ 이진탐색

// 먼저, 문제에서 배열 b에는 다음처럼 저장되어 있다고 하였다.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

↓그런데, 항목에서는 binarySearch(b, 7, 1, 20)처럼
↓인덱스 low=1 high=20으로 설정
↓즉, **0은 탐색 대상에서 제외**된 것으로
↓문제에 맞게 트리를 그리면



7만 레벨 3에 있다.

• 0을 탐색 대상에서 제외한 것이 출제자의 의도인지? 실수인지? 알 수가 없다.

정답 : ②

16. 시간복잡도 함수에 대한 점근표기법으로 옳은 것만을 모두 고르면? [2019년 국가 7급]

ㄱ. $n^{2n} + 6 \cdot 2^n = O(n^{2n})$
ㄴ. $n^2/\log n = \Theta(n^2)$
ㄷ. $n^3 2^n + 6n^2 3^n = O(n^2 2^n)$
ㄹ. $6n^3 + \log n = O(n^3)$

---

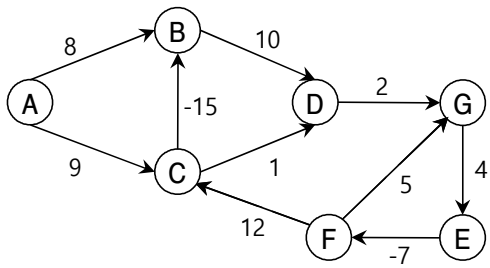
① ㄱ, ㄷ ② ㄱ, ㄹ ③ ㄴ, ㄷ ④ ㄴ, ㄹ

♤ **시간복잡도**

ㄴ. $n^2/\log n = \Theta(n^2/\log n)$
ㄷ. $n^3 2^n + 6n^2 3^n = O(n^2 3^n)$

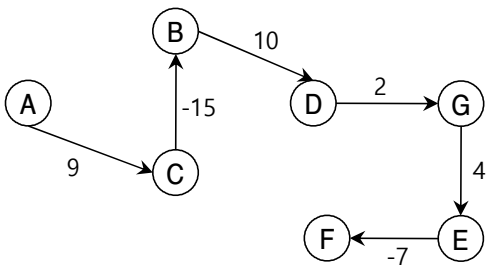
정답 : ②

17. 다음 그래프에는 음의 가중치가 존재한다. 이 그래프에서, 노드 A에서 노드 F로의 최단경로를 구하고자 할 때, 최소 비용은? [2019년 국가 7급]



① 3 ② 5 ③ 7 ④ 9

♤ **벨만-포드(Bellman-Ford) 알고리즘 최단경로(음의 가중치가 존재)**



최소 비용 = 9 − 15 + 10 + 2 + 4 − 7 = **3**

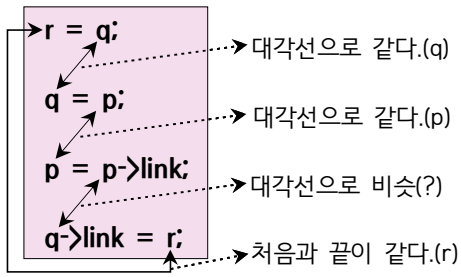
정답 : ①

18. 다음은 단순연결리스트(singly linked list) L에서 리스트의 원소를 역순으로 바꾸는 함수이다. 함수에서 ㉠ 부분에 들어갈 프로그램 코드(code)로 옳은 것은? [2019년 국가 7급]

```
------------------------------
typedef struct node{
  struct node *link;
  char data;
} listNode;
typedef struct{
  listNode *head;
} linkedList;
void reverse(linkedList *L)
{
  listNode *p, *q, *r;
  p = L->head;
  q = r = NULL;
  while ( p != NULL)  {
    r = q;
    q = p;
    p = p->link;
      ㉠
  }
  L->head = q;
}
------------------------------
```

① q = q->link;　② q->link = r;
③ p->link = q;　④ r->link = q;

♤ 단순연결리스트에 대한 역리스트 문제 푸는 방법



- 좌측에, 역리스트 문제 푸는 방법을 제시하였다.
- 무슨 말인지? 이해할 수 있을 것이다.
- 만약, 좌측에 제시한 방법 이외의 방법으로 풀면 1분 안에 풀기가 어려울 것이다.
- 최종적으로 공무원 시험은 시간 싸움이다.

정답 : ②

19. 주어진 정수 배열 {26, 13, 77, 61, 35, 11, 8, 48, 15, 19}에 대한 합병정렬(merge sort)의 순환 알고리즘(recursive algorithm)을 다음과 같이 구현하였다. 이 프로그램의 수행 과정에서 형성되는 배열의 순서로 옳지 않은 것은? [2019년 국가 7급]

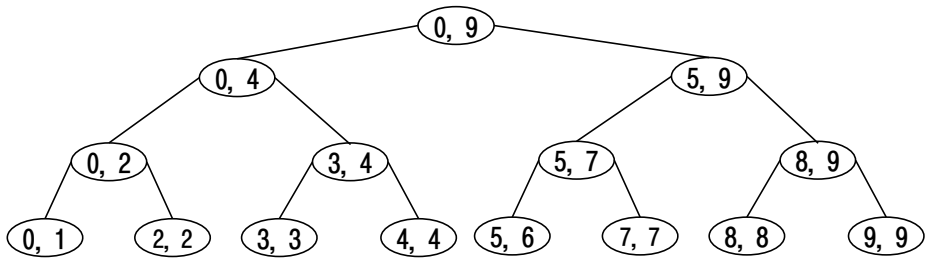
```
#include <stdio.h>
#define MAX 10
void merge(int list[], int temp[], int left, int middle, int right) {
  int i=left, j=middle+1, k=left, t;
  for( ; i<=middle && j<=right;  ) {
        if(list[i] <= list[j]) temp[k] = list[i++];
        else temp[k] = list[j++];
        k += 1;
  }
  if(i > middle) { for(t=j; t<=right; ) temp[k++] = list[t++]; }
  else { for(t=i; t<=middle; ) temp[k++] = list[t++]; }
  for(t=left; t<=right; t++ ) list[t] = temp[t];
}
void mergeSort(int list[], int temp[], int left, int right) {
  int middle = 0;
  if( left < right ) {
     middle = (left+right)/2;
     mergeSort(list, temp, left, middle);
     mergeSort(list, temp, middle+1, right);
     merge(list, temp, left, middle, right);
   }
}
int main() {
   int source[10] = {26, 13, 77, 61, 35, 11, 8, 48, 15, 19};
   int temp[10] = {0x0, };
   mergeSort(source, temp, 0, 9);
}
```

① 13, 26, 35, 61, 77, 11, 8, 48, 15, 19
② 13, 26, 77, 61, 35, 11, 8, 48, 15, 19
③ 13, 26, 35, 61, 77, 8, 11, 15, 19, 48
④ 13, 26, 35, 61, 77, 8, 11, 15, 48, 19

♤ **합병정렬의 순환 알고리즘**

// 먼저, 이 문제는 리스트가 분할되는 원리를 모르면 짧은 시간에 풀 수가 없다.

- 즉, 순환 합병정렬 원리는 리스트가 분할되는 것을 알고 있어야 풀 수 있다.
- 해서, 리스트가 분할되는 것을 기준으로 해설하기로 한다.
- 리스트가 분할되는 인덱스를 **재귀트리**로 그리면 다음과 같다.



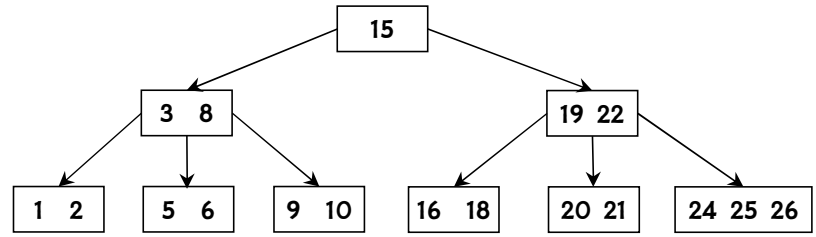
// 순환 합병정렬에서 리스트 분할

- 26 13 77 61 35 11 8 48 15 19
- 13 26 77 61 35 11 8 48 15 19
- 13 26 77 61 35 11 8 48 15 19
- 13 26 77 35 61 11 8 48 15 19
- 13 26 35 61 77 11 8 48 15 19
- 13 26 35 61 77 8 11 48 15 19
- 13 26 35 61 77 8 11 48 15 19
- 13 26 35 61 77 8 11 48 15 19
- 13 26 35 61 77 8 11 15 19 48
- 8 11 13 15 19 26 35 48 61 77 – 정렬 완료

- 순환 합병정렬의 분할, 합병되는 구조는 '반복 합병정렬'과는 다르다.
- 이 문제는 과거에는 종종 출제되었다.
- 그런데, 10년 이상 출제되지 않다가 2019년에 출제되었다.
- 아무튼, 리스트가 분할되는 원리를 모르면 짧은 시간에 풀 수 없다.

정답 : ④

20. 다음은 노드의 차수가 5인 B-트리의 현재 상태를 표현한 것이다. 현 상태에서 값 9를 삭제하였을 때, 결과 트리에 존재하는 노드 수를 종류별로 옳게 나타낸 것은? (단, n-노드란 (n-1)개의 키를 저장한 노드를 의미한다) [2019년 국가 7급]

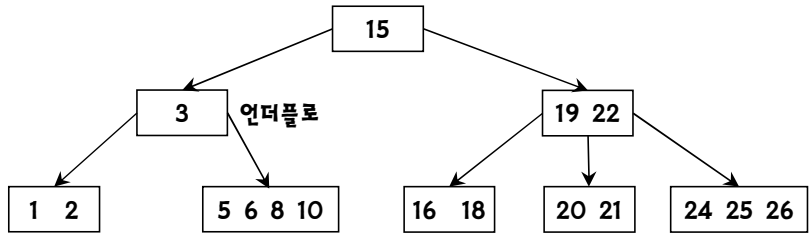


	2-노드	3-노드	4-노드	5-노드
①	2	6	1	0
②	2	4	1	1
③	0	2	3	1
④	0	3	1	2

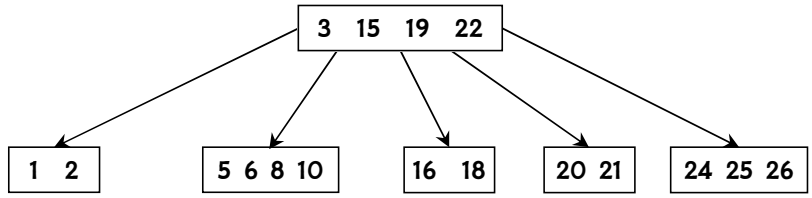
♤ 차수 5인 B-트리

- **근노드와 실패노드를 제외**한 각 노드는 적어도 ⌈m/2⌉ 개의 **자식**을 갖는다.(⌈5/2⌉ = ⌈2.5⌉ = 3)
- **근노드와 실패노드를 제외**한 각 노드는 적어도 ⌈m/2⌉ - 1의 **키**를 갖는다.(3 - 1 = 2)

- **값 9를 삭제** : 9를 삭제하고, 8을 9 위치로 이동하고, 인접노드와 통합



- **언더플로 발생** : 값 3 노드에서 언더플로 발생, 3개의 노드를 통합한다.



정답 : ④