

알고리즘

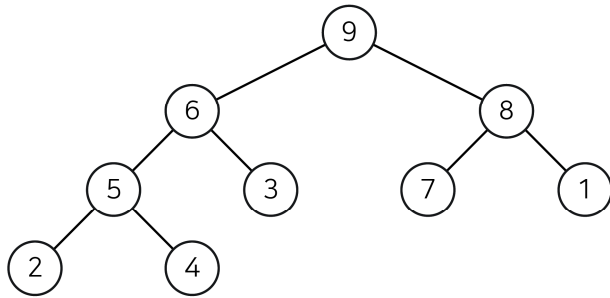
1. 알고리즘의 조건에 대한 설명으로 옳지 않은 것은?

- ① 입력: 외부에서 제공되는 자료가 0개 이상이어야 한다.
- ② 출력: 적어도 1개 이상의 결과를 만들어야 한다.
- ③ 명확성: 명령어들은 현재 실행 가능한 연산이어야 한다.
- ④ 유한성: 한정된 수의 단계 후에는 반드시 종료되어야 한다.

2. 점근 표기법에 대한 설명으로 옳은 것은?

- ① 알고리즘의 수행 시간을 대략적으로 나타내는 방법이다.
- ② $O(\text{Big-oh})$ 표기는 최소한의 알고리즘 수행 시간을 나타내기 위해 사용한다.
- ③ n 개 정수 배열에 대한 순차 탐색 알고리즘은 $O(1)$ 시간 복잡도를 가진다.
- ④ 증가 함수 $10n^2 + 3n + 2$ 는 $O(n)$ 으로 표기할 수 있다.

3. 다음 그림과 같은 최대 힙(max heap)을 이용하여 힙 정렬을 수행할 때, 최댓값 9가 배열의 마지막 요소인 4와 교환된 다음, 힙 구조 유지를 위한 연산이 완료된 상태에서의 배열 내용으로 옳은 것은?



[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
- 9 6 8 5 3 7 1 2 4

- ① - 8 4 6 7 3 5 2 1 9
- ② - 8 5 4 6 3 7 1 2 9
- ③ - 8 6 7 5 3 4 1 2 9
- ④ - 8 7 6 3 4 5 2 1 9

4. 다음은 초기상태 배열을 오름차순으로 정렬하는 단계들이다. 이와 같이 정렬을 수행하는 알고리즘으로 옳은 것은?

초기상태: [69, 10, 30, 3, 16, 8]

step 1: [3, 10, 30, 69, 16, 8]
step 2: [3, 8, 30, 69, 16, 10]
step 3: [3, 8, 10, 69, 16, 30]
step 4: [3, 8, 10, 16, 69, 30]
step 5: [3, 8, 10, 16, 30, 69]

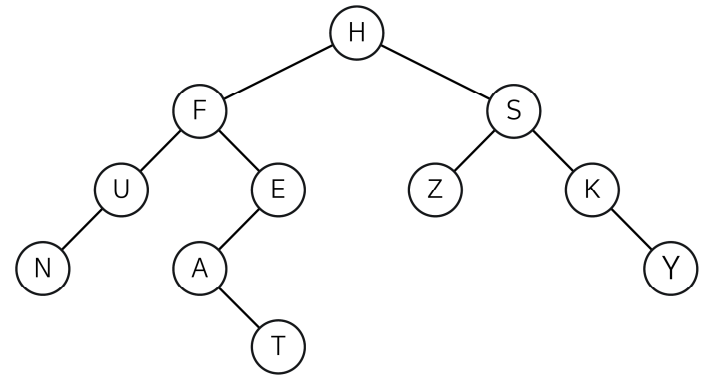
- ① 선택 정렬(selection sort) ② 버블 정렬(bubble sort)
- ③ 합병 정렬(merge sort) ④ 기수 정렬(radix sort)

5. 다음 중위 표기법 수식을 후위 표기법 수식으로 옳게 변환한 것은?

$(A * B) - (C / D) - E$

- ① $A B * C D E - / -$
- ② $A B * C D / - E -$
- ③ $A B * C D - / E -$
- ④ $* A B - / C D - E$

6. 그림과 같이 연결되어 있는 11개의 여행지를 관광객들이 모두 방문할 수 있도록 안내하기 위해 C언어로 구현한 Course() 함수를 다음과 같이 제시하고 있다. 여행지 H를 파라미터로 하여 알고리즘을 호출했을 때, 출력결과 (가) ~ (라)에 들어갈 내용을 바르게 연결한 것은? (단, 상위 노드 기준으로 $n \rightarrow \text{left}$ 는 왼쪽으로 순회하고 $n \rightarrow \text{right}$ 는 오른쪽으로 순회한다)



```
typedef struct node {
    char name;
    struct node *left;
    struct node *right;
} node;

void Course(node *n) {
    if(n != null) {
        Course(n->left);
        Course(n->right);
        printf("%c->", n->name);
    }
}
```

○ 출력결과

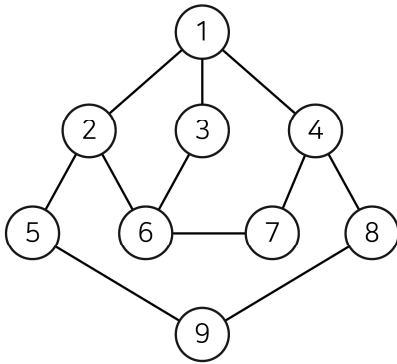
(가) -> [] -> (나) -> [] -> (다) -> [] ->
[] -> [] -> [] -> (라) -> [] ->

- | | (가) | (나) | (다) | (라) |
|---|-----|-----|-----|-----|
| ① | H | A | Z | K |
| ② | H | T | S | K |
| ③ | N | H | U | T |
| ④ | N | T | E | S |

7. 다음 데이터를 순서대로 삽입하여 불균형이 발생했을 때 회전 연산을 수행하여 균형 상태를 유지하는 AVL 트리를 구성하였다. 구성된 AVL 트리에서 단말 노드(leaf node)에 해당하는 값만을 모두 고르면?

50, 60, 70, 90, 80, 75, 73, 72, 78

- ① 72, 73, 90
 ② 72, 75, 90
 ③ 50, 73, 75, 90
 ④ 50, 72, 78, 90
8. 그림과 같은 그래프를 대상으로 다음 알고리즘을 적용하였을 때, 결과에 대한 설명으로 옳지 않은 것은? (단, 탐색의 시작 정점은 1이고, Q는 큐 객체이며, Q.enqueue()와 Q.dequeue()는 큐 Q에 대한 삽입과 삭제 연산이고, 각 정점의 방문을 표시하는 것은 visited[] 배열을 이용한다)



```

_FS(v) {
    visited[v] <- YES;    // 정점 v에 대한 방문 표시
    Q.enqueue(v)
    while(!Q.empty()) {  // Q가 비어있지 않은 동안
        w <- Q.dequeue();
        for all u ∈ (w에 인접한 정점) {
            if(visited[u] != YES) {
                visited[u] <- YES;
                Q.enqueue(u);
            }
        }
    }
}

```

- ① 알고리즘의 수행에 따른 노드의 방문 순서는 $1 \rightarrow 2 \rightarrow 5 \rightarrow 9 \rightarrow 8 \rightarrow 4 \rightarrow 7 \rightarrow 6 \rightarrow 3$ 이다.
 ② 너비 우선 탐색(BFS, Breadth First Search) 알고리즘이다.
 ③ 시작 정점으로부터 가까운 정점들을 먼저 방문하고 멀리 떨어져 있는 정점들을 나중에 방문하는 탐색방법이다.
 ④ 알고리즘은 그래프의 모든 정점을 방문한다.

9. 다음 C언어 함수 ABC()에 대한 설명으로 옳은 것은?

```

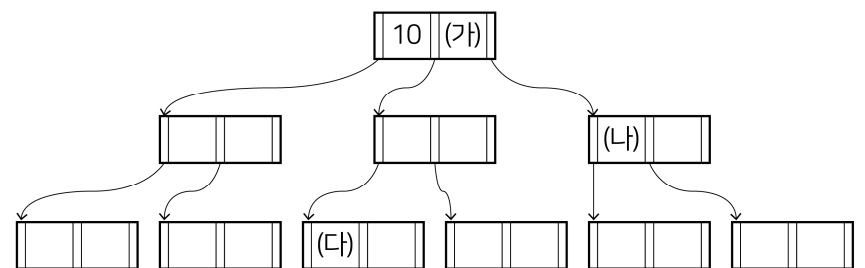
void ABC(int A[], int n) {
    int i, j, tmp;
    for(i = n - 1; i > 0; i--) {
        j = 0;
        while(j < i) {
            if(A[j] < A[j + 1]) {
                tmp = A[j];
                A[j] = A[j + 1];
                A[j + 1] = tmp;
            }
            j = j + 1;
        }
    }
}

```

- ① 코드 수행 완료 후, A[0]에는 가장 큰 원소가 저장되어 있다.
 ② 코드 수행 완료 후, 배열 A의 원소는 오름차순으로 정렬되어 있다.
 ③ 최선의 경우와 최악의 경우의 수행 시간 복잡도가 다르다.
 ④ 코드는 삽입 정렬(insertion sort)을 구현한 것이다.

10. 다음 자료를 차수가 3인 B-트리에 순서대로 삽입하여 그림과 같은 B-트리를 생성하였을 때, (가) ~ (다)에 들어갈 값을 바르게 연결한 것은?

75, 15, 10, 12, 5, 30, 1, 2, 25, 27, 34, 60



- | | (가) | (나) | (다) |
|---|-----|-----|-----|
| ① | 25 | 34 | 12 |
| ② | 25 | 60 | 15 |
| ③ | 30 | 34 | 15 |
| ④ | 30 | 60 | 12 |

11. 다음은 재귀 호출을 이용하여 피보나치(Fibonacci) 수열값을 찾는 파이썬 코드이다. 이 코드의 시간 복잡도는? (단, n 은 0 이상의 정수이다)

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)
```

- ① $O(n)$
 ② $O(n^2)$
 ③ $O(2^n)$
 ④ $O(\log n)$
12. 분할 정복(divide-and-conquer) 방식의 정렬 알고리즘에 대한 설명으로 옳지 않은 것은?
- ① 2-way 합병 정렬의 분할은 분할 대상을 반으로 나눈다.
 ② 퀵정렬(quick sort)의 분할은 분할 대상에 대하여 피벗(pivot)을 기준으로 나눈다.
 ③ 합병 정렬과 다르게 퀵정렬에서는 모든 분할이 완료되면 정렬도 완료된다.
 ④ 합병 정렬은 추가 데이터 저장공간이 필요 없는 제자리 정렬 알고리즘이다.

13. 배열로 운영하는 원형 큐에 대해 다음과 같이 연산 1에서 7까지의 연산을 차례로 수행했을 때, 수행이 완료된 후 큐의 상태로 옳은 것은? (단, 현재 상태에서 $front = 0$, $rear = 2$ 이다)

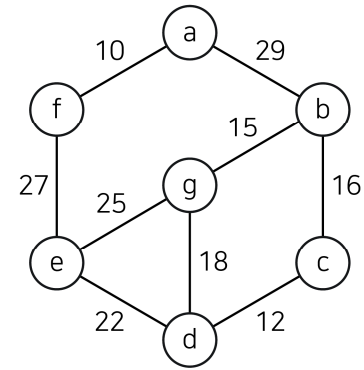
[0] [1] [2] [3] [4]

□ Z X □ □

연산 1. 원소 Y 삽입
 연산 2. 원소 3개 삭제
 연산 3. 원소 Q 삽입
 연산 4. 원소 M 삽입
 연산 5. 원소 A 삽입
 연산 6. 원소 K 삽입
 연산 7. 원소 2개 삭제

- ① [0] [1] [2] [3] [4]
 M □ □ □ Q
- ② [0] [1] [2] [3] [4]
 □ A K □ □
- ③ [0] [1] [2] [3] [4]
 □ □ □ Y Q
- ④ [0] [1] [2] [3] [4]
 M A □ □ □

14. 다음 그래프를 대상으로 크루스칼(Kruskal) 알고리즘을 적용하여 최소 신장 트리(MST, Minimum Spanning Tree)를 구성하고자 할 때, 생성되는 MST에 5번째 추가되는 간선과 비용을 바르게 연결한 것은?



간선	비용
① (b, c)	16
② (d, g)	18
③ (d, e)	22
④ (e, g)	25

15. 문자열 매칭을 위한 알고리즘으로 옳은 것은?

- ① 합병 정렬 알고리즘
 ② 라빈 카프(Rabin-Karp) 알고리즘
 ③ 백트래킹(backtracking) 알고리즘
 ④ 후위 순회(postorder traversal) 알고리즘

16. 다음은 단일 출발점 최단 경로를 찾기 위한 다익스트라(Dijkstra) 알고리즘을 의사(pseudo) 코드로 기술한 것이다. 이 코드에서 (가)에 들어갈 내용으로 옳은 것은?

```
// 입력: 가중 방향 그래프 G, 인접 행렬 A, 시작 정점 s
// 출력: s로부터 다른 모든 정점까지의 최단 경로의 길이
Dijkstra(G, A, s) {
    U = {s};
    for(모든 정점 v ∈ G)
        D[v] = A[s][v];
    while(U != G) {
        D[w]가 최소인 정점 w ∈ (G - U)를 선택
        U = U ∪ {w};
        for(w에 인접한 모든 정점 v) {
            D[v] = (가);
        }
    }
}
```

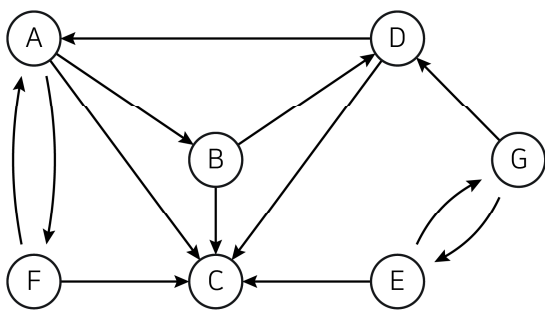
- ① $\min(D[v], D[v] + A[v][w])$
 ② $\min(D[v], D[w] + A[v][w])$
 ③ $\min(D[v], D[v] + A[w][v])$
 ④ $\min(D[v], D[w] + A[w][v])$

17. KMP(Knuth-Morris-Pratt) 알고리즘과 보이어-무어(Boyer-Moore) 알고리즘에 대한 설명으로 옳지 않은 것은?

- ① KMP 알고리즘은 패턴 문자열의 첫 문자부터 순차적으로 대상 문자열과 부분 문자열의 일치 여부를 판단한다.
- ② KMP 알고리즘은 패턴 문자열과 대상 문자열을 한 차례 비교한 후 얻을 수 있는 추가 정보로 비교 횟수를 감소시킨다.
- ③ 보이어-무어 알고리즘은 패턴 문자열의 마지막 문자부터 역순으로 대상 문자열과 부분 문자열의 일치 여부를 판단한다.
- ④ 보이어-무어 알고리즘은 전처리(preprocessing) 과정 없이 곧바로 일치 여부를 판단할 수 있다.

18. 다음 조건에 따라 주어진 그래프에 대해 깊이 우선 탐색(DFS, Depth First Search) 방법을 이용하여 트리를 만들 경우, 나올 수 있는 트리의 개수와 수행하는 백트래킹 횟수를 바르게 연결한 것은?

- DFS의 시작 정점은 A로 하고, 정점의 접근 순서는 알파벳 순서로 한다.
- DFS는 사이클을 형성하지 않으면서 백트래킹을 통해 시작 정점으로 돌아왔을 때, 시작 정점에서 방문하지 않은 접근 가능한 인접 정점이 없으면 종료한다.
- DFS의 종료시점에 그래프에 방문하지 않은 정점들이 존재하면, 방문하지 않은 정점들 중 알파벳 순서가 가장 앞선 정점을 시작 정점으로 하여 DFS를 다시 시작한다.
- 그래프 안에 방문하지 않은 정점이 존재하지 않을 때까지 DFS를 반복한다.



	트리의 개수	백트래킹 횟수
①	1	3
②	1	4
③	2	3
④	2	5

19. 다음은 n 개의 데이터에 대한 평균을 재귀적으로 계산하는 프로그램이다. (가) ~ (다)에 들어갈 내용을 바르게 연결한 것은? (단, $n \geq 1$ 이다)

```
float AVG(float (가), int n) {
    if(n == 1) return (나);
    else
        return (data[n-1] + (n-1) * AVG(data, n-1)) / (다);
}
```

	(가)	(나)	(다)
①	data[]	0	n
②	data[]	1	(n-1)
③	*data	data[0]	n
④	*data	data[n-1]	(n-1)

20. 작업 선택 문제를 다루는 다음 알고리즘은 각 작업의 시작 시간 s 와 완료 시간 f 가 정해진 n 개의 작업 $t[0..n-1]$ 가 주어질 때, 하나의 기계만 사용하여 충돌 없이 수행할 수 있는 최대 개수의 작업을 선택한다. 이 알고리즘의 (가), (나)에 들어갈 내용을 바르게 연결한 것은? (단, $n \geq 2$ 인 정수, count는 선택한 작업의 개수, $t[i].s$ 는 작업 $t[i]$ 의 시작 시간, $t[i].f$ 는 작업 $t[i]$ 의 종료 시간이고, jobs 배열은 선택된 작업을 저장한다)

```
ActSelect(n, t[]) {
    (가)의 오름차순으로 작업 t[]를 정렬;
    count = 0, last = 0, jobs[];
    jobs[count] = 0;
    for(i = 0; i < n; i++)
        if((나)) {
            count++;
            last = i;
            jobs[count] = last;
        }
    return jobs;
}
```

	(가)	(나)
①	시작 시간	$t[i].s \geq t[last].f$
②	시작 시간	$t[i].f \geq t[last].s$
③	완료 시간	$t[i].s \geq t[last].f$
④	완료 시간	$t[i].f \geq t[last].s$

21. 동적 프로그래밍(dynamic programming) 방법으로 최소의 원소단위 곱셈 횟수를 갖도록 연쇄 행렬곱셈(matrix-chain multiplication)의 순서를 정하려 한다. $c_{i,j}$ 를 행렬 M_i 부터 M_j 까지의 연쇄 행렬곱셈의 최소 비용이라고 할 때, $c_{i,j}$ 에 대한 관계식으로 옳은 것은? (단, $i < j$ 이고, 각 행렬의 차원이 p_0, p_1, \dots, p_n 일 때, 행렬 M_i 의 크기는 $p_{i-1} \times p_i$ 이다)

- ① $c_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k \leq j-1} \{c_{i,k} + c_{k+1,j} + p_{i-1}p_kp_j\} & \text{if } i < j \end{cases}$
- ② $c_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k \leq j-1} \{c_{i,k} + c_{k,j} + p_{i-1}p_{k-1}p_j\} & \text{if } i < j \end{cases}$
- ③ $c_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k \leq j-1} \{c_{i,k-1} + c_{k,j} + p_{i-1}p_{k-1}p_j\} & \text{if } i < j \end{cases}$
- ④ $c_{i,j} = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k \leq j-1} \{c_{i,k} + c_{k,j} + p_{i-1}p_kp_j\} & \text{if } i < j \end{cases}$

22. 동적 프로그래밍은 한 번 계산된 값을 메모리에 저장한 후, 저장된 값을 필요할 때마다 가져다 쓰는 방법이다. 동적 프로그래밍을 적용하여 구현한 다음 Java 프로그램에서 피보나치 수열 fibo(6)을 구하기 위해 저장되어 있는 fibo(3)의 결과를 가져다 쓰는 횟수는?

```
public class Fibonacci {
    static int[] dp = new int[100];

    public static void main(String[] args) {
        System.out.println(fibo(6));
    }

    public static int fibo(int num) {
        if(num == 0) {
            return 0;
        }
        if(num == 1) {
            return 1;
        }
        if(dp[num] != 0) {
            return dp[num];
        }
        return dp[num] = fibo(num - 1) + fibo(num - 2);
    }
}
```

- ① 0
② 1
③ 2
④ 3

23. 배낭 문제(knapsack problem)에 대한 설명으로 옳은 것은?

- ① 0-1 배낭 문제와 분할 가능 배낭 문제는 다항식 시간(polynomial-time) 문제이다.
- ② 분할 가능 배낭 문제는 최적 부분 구조를 갖지만, 0-1 배낭 문제는 그렇지 않다.
- ③ 0-1 배낭 문제는 물건을 나눌 수 없어서 그리디(greedy) 알고리즘을 적용하여 해결할 수 있으며, 분할 가능 배낭 문제는 물건을 나눌 수 있어 동적 프로그래밍으로 해결해야 한다.
- ④ 분할 가능 배낭 문제는 '단위 무게당 가치가 가장 높은 물건을 선택'하는 방식이 항상 최적해를 보장하지만, 0-1 배낭 문제는 그렇지 않다.

24. 알고리즘들에 대한 설명으로 옳지 않은 것은?

- ① 크루스칼의 최소 신장 트리 알고리즘은 가중치가 가장 작으면서 사이클을 만들지 않는 간선을 추가하여 트리를 구성한다.
- ② 프림(Prim)의 최소 신장 트리 알고리즘은 현재까지 만들어진 트리에 최소 가중치로 연결되는 정점을 트리에 추가한다.
- ③ 허프만(Huffman) 압축에서 파일에 빈번히 나타나는 문자는 긴 이진코드를 할당하고, 드물게 나타나는 문자는 짧은 이진코드를 할당한다.
- ④ 벨만-포드(Bellman-Ford)의 최단 경로 알고리즘은 음의 가중치를 갖는 간선이 존재하는 상황에서도 단일 출발점 최단 경로 문제를 해결하는 알고리즘이다.

25. 다음 설명에 해당하는 알고리즘은?

- 주어진 문제에 대해 근사해나 일정 확률 이상의 최적해를 출력함
- 근사해나 간혹 틀린 해를 허용할 수 있는 경우에 사용 가능함
- 효율적인 결정론적 알고리즘이 밝혀져 있지 않은 문제에 효과적임

- ① 유클리드(Euclid) 알고리즘
② 스트라센(Strassen) 알고리즘
③ 몬테카를로(Monte Carlo) 알고리즘
④ 플로이드-워셜(Floyd-Warshall) 알고리즘