

알고리즘

1. 컴퓨터 알고리즘(algorithm)의 특성에 대한 설명으로 옳지 않은 것은?

- ① 알고리즘은 1개 이상의 결과(output)를 생성해야 한다.
- ② 알고리즘에는 반드시 입력(input)이 1개 이상 존재해야 한다.
- ③ 알고리즘은 한정된 횟수의 단계 실행 후 반드시 종료해야 한다.
- ④ 알고리즘의 각 명령은 모호하지 않고 명확해야 한다.

2. 다음 피보나치 수열의 n 번째 항을 계산하는 알고리즘에 대한 설명으로 옳지 않은 것은? (단, $n \geq 0$ 이다)

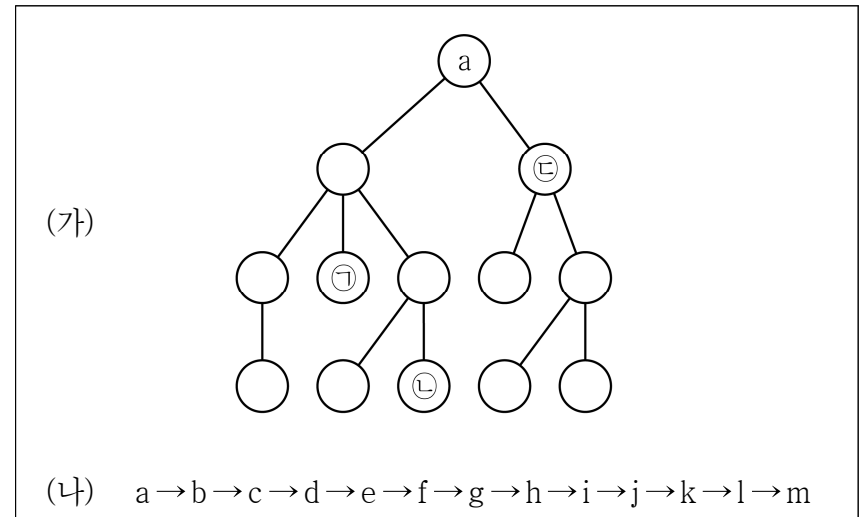
```
int fib(n) {
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    return fib(n - 1) + fib(n - 2);
}
```

- ① fib(7) 호출 시, fib(3)은 5번 호출된다.
- ② 알고리즘의 시간 복잡도는 $O(\log_2(n))$ 이다.
- ③ 알고리즘의 점화식은 $F_n = \begin{cases} 0, & n=0 \\ 1, & n=1 \\ F_{n-1} + F_{n-2}, & n > 1 \end{cases}$ 이다.
- ④ 반복문을 사용하여 알고리즘 성능을 개선할 수 있다.

3. 문제 해결을 위한 알고리즘을 기술하는 방법이 아닌 것은?

- ① 순서도(flowchart)를 사용하여 도식화한다.
- ② 의사 코드(pseudo code)를 사용하여 기술한다.
- ③ 추상 데이터 타입(abstract data type)만으로 기술한다.
- ④ 단계별로 명확한 문장(자연어)으로 기술한다.

4. 다음 (가) 트리(tree)를 깊이 우선 탐색(DFS, Depth First Search) 알고리즘으로 방문한 순서가 (나)와 같을 때, ㉠ ~ ㉣에 들어갈 노드는?



- | | ㉠ | ㉡ | ㉣ |
|---|---|---|---|
| ① | e | g | c |
| ② | e | h | i |
| ③ | k | g | i |
| ④ | k | h | c |

5. 1부터 n 까지의 합을 계산하는 다음 알고리즘의 빈칸에 들어갈 코드는? (단, $n \geq 1$ 인 정수이다)

```
int sum(n) {
    if (n == 1)
        return 1;
    else
        ;
}
```

- ① return $n + \text{sum}(n - 1)$
- ② return $n - \text{sum}(n - 1)$
- ③ return $n + \text{sum}(n + 1)$
- ④ return $n - \text{sum}(n + 1)$

6. 힙(heap)에 대한 설명으로 옳지 않은 것은? (단, n 은 데이터의 개수이다)

- ① 힙을 구성하는 완전 이진 트리는 1차원 배열을 사용하여 구현할 수 있다.
- ② 최대 힙(max heap)의 모든 부모 노드 값은 그들의 자식 노드 값보다 크거나 같다.
- ③ 새로운 데이터 삽입 후 힙 성질을 만족하기 위한 연산의 시간 복잡도는 $O(n \log_2(n))$ 이다.
- ④ 루트 노드에서 데이터 삭제 후 힙 성질을 만족하기 위한 연산의 시간 복잡도는 $O(\log_2(n))$ 이다.

7. 병합(merge) 정렬과 퀵(quick) 정렬에 대한 설명으로 옳지 않은 것은?

- ① 병합 정렬과 퀵 정렬은 모두 재귀 함수(recursive function)로 구현할 수 있다.
- ② 병합 정렬과 퀵 정렬은 모두 분할 정복(divide-and-conquer) 알고리즘에 속한다.
- ③ 퀵 정렬은 병합 정렬과 달리 배열 내 요소 간 데이터를 교환 (swap)하지 않는다.
- ④ 퀵 정렬은 입력의 피벗(pivot) 값을, 병합 정렬은 중간 위치를 사용하여 배열을 분할한다.

8. 다음 버블(bubble) 정렬 알고리즘 (가)에 대한 입력 배열이 (나)일 때, 해당 알고리즘에서 데이터 교환(swap 함수 실행) 직후 (다) 배열 상태에서의 end와 cur의 값은?

(가) bubble_sort(arr[], arr_size) {
 // arr 배열의 인덱스는 0 ~ arr_size-1
 for end ← arr_size - 1 to 1
 for cur ← 0 to end - 1
 if (arr[cur] > arr[cur + 1])
 swap(arr[cur], arr[cur + 1]);
 }

(나) 6, 2, 3, 4, 0, 5, 1, 9, 8, 7

(다) 2, 3, 0, 4, 5, 1, 6, 8, 7, 9

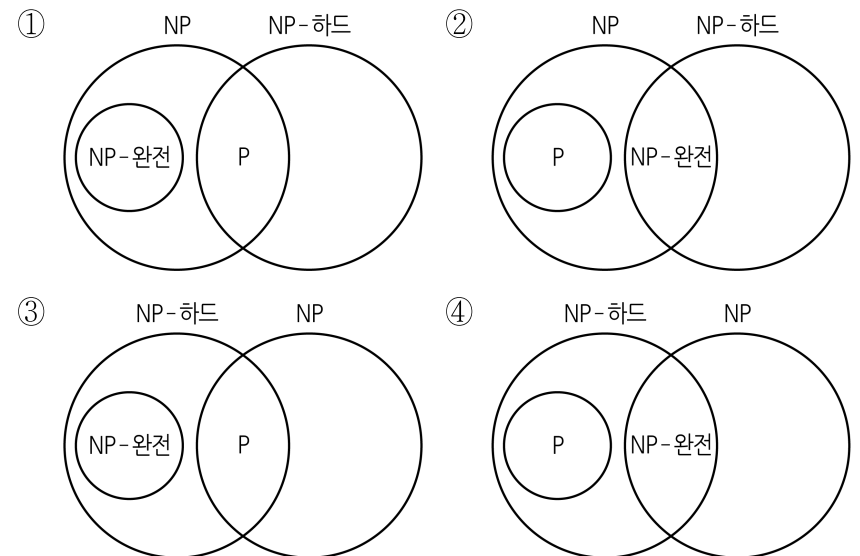
	end	cur
①	8	2
②	8	4
③	7	2
④	7	4

9. 다음 알고리즘에 해당하는 정렬은?

```
sort(arr[], n) { // arr 배열의 인덱스는 0 ~ n-1
  for i ← 1 to n - 1 {
    cur ← arr[i];
    j ← i - 1;
    while (j ≥ 0) and (arr[j] > cur) {
      arr[j + 1] ← arr[j];
      j ← j - 1;
    }
    arr[j + 1] ← cur;
  }
}
```

- ① 셸(shell) 정렬
- ② 기수(radix) 정렬
- ③ 선택(selection) 정렬
- ④ 삽입(insertion) 정렬

10. 문제 부류(class)간 포함관계로 옳은 것은? (단, $P \neq NP$ 일 경우이다)



11. 다음 설명에 해당하는 알고리즘은?

- 문제의 입력 사례 분할 후, 분할된 문제의 해답을 우선 구한다.
- 문제의 입력 사례를 분할할 때, 재귀 관계식을 사용할 수 있다.
- 분할된 문제의 해답을 저장하여 분할 전 문제의 해답을 구하는 상향식 방법이다.

- ① 브루트포스(bruteforce)
- ② 백 트래킹(backtracking)
- ③ 분기 한정법(branch-and-bound)
- ④ 동적 프로그래밍(dynamic programming)

12. 분할 정복(divide-and-conquer) 알고리즘에 대한 설명으로 옳은 것은?

- ① 이진 탐색(binary search)은 분할 정복 알고리즘에 속한다.
- ② 분할 정복에 기반을 둔 알고리즘의 시간 복잡도는 최악의 경우 $O(\log_2(n))$ 이다.
- ③ 반복문(iteration)만을 사용하여 큰 문제를 작은 문제들로 분할한다.
- ④ n 개의 정수들 중 k 번째로 큰 정수를 찾는 선택(selection) 문제에서 문제의 크기는 재귀적으로 절반씩 분할된다.

13. 다음 알고리즘에 대한 시간 복잡도는?

```
sum ← 0;
for j ← 1 to n
  for i ← 1 to j
    sum ← sum + 1;
for k ← 0 to n - 1
  sum ← sum + k;
```

- ① $O(n)$
- ② $O(n \log_2(n))$
- ③ $O(n^2)$
- ④ $O(2^n)$

14. a^x 을 계산하는 다음 알고리즘의 빈칸 ㉠ ~ ㉣에 해당하는 코드로 옳게 짝지어진 것은? (단, a 와 x 는 정수, $x \geq 0$, mod는 나머지 연산이다)

```
int power(a, x) {
  if (x == 1)
    return ㉠;
  else if (x == 0)
    return 1;
  if (x mod 2 == 0) {
    temp ← power(a, x / 2);
    return ㉡;
  } else {
    temp ← power(a, (x - 1) / 2);
    return ㉢;
  }
}
```

- | | ㉠ | ㉡ | ㉢ |
|---|---------|---------------|-------------------|
| ① | $a * a$ | $temp * a$ | $temp * temp * a$ |
| ② | $a * a$ | $temp * temp$ | $a * a * temp$ |
| ③ | a | $temp * a$ | $a * a * temp$ |
| ④ | a | $temp * temp$ | $temp * temp * a$ |

15. 정렬된 정수 배열을 입력으로 하는 이진 탐색 알고리즘에 대한 시간 복잡도는?

- ① $O(\log_2(n))$
- ② $O(n)$
- ③ $O(n \log_2(n))$
- ④ $O(n^2)$

16. 다음 설명에 해당하는 복잡도 함수에 대한 시간 복잡도는?

- $T(1) = 27$
- $T(n)$ 은 궁극적으로 감소하지 않는다.
- $T(n) = 9T(\frac{n}{3}) + 8n$ ($n > 1$ 이고, n 은 3의 거듭제곱이다)

- ① $\Theta(1)$
 ② $\Theta(\log_2(n))$
 ③ $\Theta(n)$
 ④ $\Theta(n^2)$

17. 다음 수식들에 대해 ㉠ ~ ㉣에 해당하는 빅오(O) 표기가 옳게 짝지어진 것은? (단, k 는 상수이다)

㉠	⌈	6,450,254	⌋	㉣
		$4\log(n) + 3$		
		$3n + 8,230$		
		$n\log(n) + 4n$		
		$2n^2 + 5n + 3$		
		$4n^3 + n^2 - 2n$		
		$3 \times 2^n + 8n^5$		

- | | | | |
|---|----------|--------------|----------|
| | ㉠ | ㉡ | ㉢ |
| ① | $O(2^n)$ | $O(1)$ | $O(n^k)$ |
| ② | $O(n^2)$ | $O(1)$ | $O(n^3)$ |
| ③ | $O(n^2)$ | $O(\log(n))$ | $O(n^k)$ |
| ④ | $O(n^2)$ | $O(\log(n))$ | $O(n^3)$ |

18. 다음 문제를 해결하기 위한 알고리즘 관련 설명으로 옳은 것은?

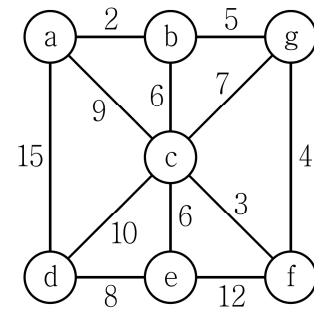
- n 개의 물건은 각기 고유의 무게와 가치를 갖는다.
- 물건은 나눌 수 없으며, 선택한 물건은 통째로 넣는다.
- 배낭 내 물건들의 총 무게는 배낭 용량을 초과할 수 없다.
- 최대의 가치를 갖도록 한정된 용량의 배낭에 물건을 넣는다.

- ① 탐욕 알고리즘을 사용할 경우 시간 복잡도는 $O(n)$ 이다.
 ② 탐욕 알고리즘은 문제에 대한 최적해를 계산할 수 없다.
 ③ 동적 프로그래밍 알고리즘은 가치 ÷ 무게(무게당 가치)를 사용한다.
 ④ 동적 프로그래밍 알고리즘은 힙(heap)을 구성하여 최적해를 탐색한다.

19. 각 정렬 알고리즘에 대한 사례별 시간 복잡도가 옳게 짝지어진 것은?

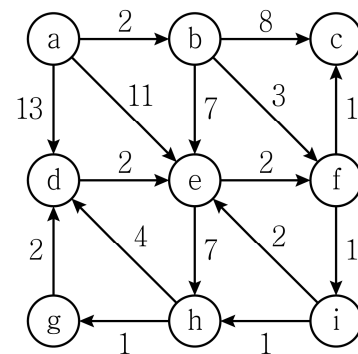
	최선의 경우	최악의 경우
① 삽입(insertion) 정렬 :	$\Omega(n)$	$O(n^2)$
② 선택(selection) 정렬 :	$\Omega(n)$	$O(n^2)$
③ 힙(heap) 정렬 :	$\Omega(n \log_2(n))$	$O(n^2)$
④ 퀵(quick) 정렬 :	$\Omega(n \log_2(n))$	$O(n \log_2(n))$

20. 다음 그래프에 대해 크루스칼(Kruskal) 알고리즘에 의한 최소 신장 트리(MST, Minimum Spanning Tree)를 계산할 때, 5번째로 MST에 포함되는 간선은? (단, 초기의 MST는 $\emptyset$ 이다)



- ① (b, c)
 ② (b, g)
 ③ (c, e)
 ④ (d, e)

21. 다음 방향 그래프의 a 노드에서 각 노드별 최단 경로를 다익스트라(Dijkstra) 알고리즘으로 계산할 때, 각 노드별 최단 거리 갱신 횟수로 옳은 것은? (단, a 노드를 제외한 모든 노드의 최단 경로 초기값은 ∞ 이다)



- ① b: 1회, c: 1회
 ② f: 2회, g: 1회
 ③ d: 2회, i: 2회
 ④ e: 3회, h: 1회

22. 하노이탑 문제를 해결하기 위한 다음 알고리즘의 빈칸 ㉠, ㉡에 들어갈 코드가 옳게 짝지어진 것은? (단, n 은 원판의 개수이다)

```

hanoi(n, from, temp, to){
    if (n == 1)
        move(from, to); // from에서 to로 이동
    if (1 < n){
        hanoi(n - 1, ㉠);
        move(from, to); // from에서 to로 이동
        hanoi(n - 1, ㉡);
    }
}

```

㉠

㉡

- | | |
|------------------|----------------|
| ① from, to, temp | from, temp, to |
| ② from, to, temp | temp, from, to |
| ③ temp, from, to | from, to, temp |
| ④ temp, from, to | from, temp, to |

23. 라빈-카프(Rabin-Karp) 알고리즘에 대한 설명으로 옳지 않은 것은? (단, 패턴 문자열의 길이는 m , 대상 문자열의 길이는 n , $m \leq n$ 이다)

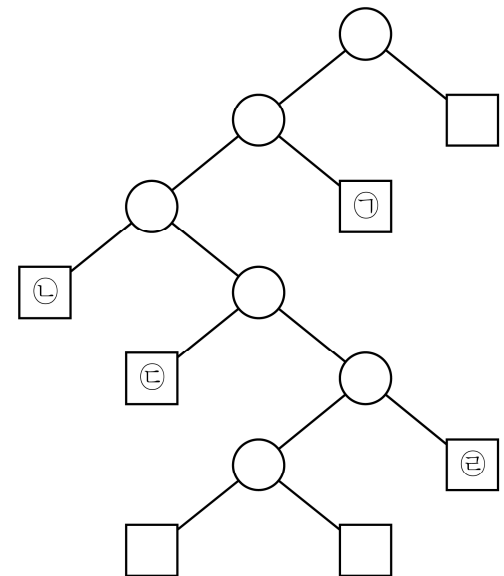
- ① 패턴 문자열에 대한 해시 값 1개, 대상 문자열에 대한 해시 값이 최대 $n - m + 1$ 개 생성된다.
- ② 문자열의 해시 값을 계산하는 과정에서 호너의 방법(Horner's rule)을 사용할 수 있다.
- ③ 해시 값이 컴퓨터 레지스터의 용량을 초과할 경우, 나머지 연산(mod)을 사용한다.
- ④ 패턴 문자열의 해시 값과 대상 문자열의 일부 문자열에 대한 해시 값이 일치하면 알고리즘을 종료한다.

24. KMP(Knuth-Morris-Pratt) 알고리즘에 대한 설명으로 옳지 않은 것은? (단, 패턴 문자열의 길이는 m , 대상 문자열의 길이는 n , $m \leq n$ 이다)

- ① 알고리즘의 전처리(preprocessing) 과정을 포함한 시간 복잡도는 $O(mn)$ 이다.
- ② 대상 문자열과 패턴 문자열의 문자들을 각각 비교하며 문자열 매칭 여부를 판단한다.
- ③ 매칭이 실패할 경우 미리 생성된 위치 정보를 이용하여 비교를 다시 시작한다.
- ④ 대상 문자열과 패턴 문자열에서 일치하는 부분 문자열을 이용하여 비교 횟수를 감소시킨다.

25. 다음 문자와 빈도수에 대해 허프만(Huffman) 알고리즘을 이용하여 트리를 구성했을 때, 빈칸 ㉠ ~ ㉣에 들어갈 문자는? (단, ○는 빈도수 노드, □는 문자 노드이다)

문자	a	b	d	e	g	i	k
빈도수	50	4	2	8	1	25	6



㉠

㉡

㉢

㉣

- | | | | |
|-----|---|---|---|
| ① a | i | b | k |
| ② a | i | k | b |
| ③ i | e | d | g |
| ④ i | e | k | b |