연습문제

1 다음 알고리즘들에 대해 입력의 크기와 기본 연산을 설명해 보라.

(1) 입력 리스트에서 가장 큰 항목을 찾는 알고리즘
(2) 입력 리스트의 모든 숫자를 더하는 알고리즘
(3) $n!$ 알고리즘
(4) 두 개의 $n \times n$ 행렬을 더하는 알고리즘
(5) 유클리드 알고리즘(알고리즘 1.7)
(6) 최대 공약수 알고리즘(알고리즘 1.5)
(7) 최대 공약수 알고리즘(알고리즘 1.6)

2 위 문제의 알고리즘들에 대해 입력의 구성에 따른 처리시간(최선, 평균, 최악) 특성을 각각 설명하라.

3 다음의 두 알고리즘이 입력의 크기에 따라 어떤 증가 속도를 갖는지를 설명하고, 입력의 구성에 따라 처리시간이 달라지는지를 설명하라.

(1) 입력 리스트에서 어떤 항목이 있는지를 검사하는 알고리즘
(2) 입력 리스트에서 어떤 항목이 몇 번 나타나는지를 구하는 알고리즘

4 다음 시간 복잡도 함수들의 증가 속도를 비교하고, =, $\rangle$, $\langle$ 기호로 표시하라.

(1) $n(n-100)(n-2000)$과 $50000n^3$
(2) $(\log_2 n)^2$과 $\log_2 n^2$
(3) $100000n$과 $0.00001n^2$
(4) 3^{n-3}과 3^n
(5) 1000^n과 $n!$

5 입력의 크기가 두 배로 늘어나면 다음 시간 복잡도 함수들의 처리시간은 몇 배가 늘어날까?

(1) $\log_2 n$ (2) n (3) n^2 (4) 2^n (5) $\sqrt{n}$

6 다음의 시간 복잡도 함수를 빅오 표기법으로 나타내라.

(1) $T(n) = n^2 + 10n + 8$
(2) $T(n) = n^3 + 10000n^2 + 50n$
(3) $T(n) = n^2 \log_2 n + n^3 + 3$
(4) $T(n) = 7(2^n) + 3^n$
(5) $T(n) = 3^n + n!$

7 다음의 빅오 표기법들을 실행 시간이 적게 걸리는 것부터 나열하라.

$$O(1)\ \ O(n)\ \ O(n^2)\ \ O(n^3)\ \ O(\log n)\ \ O(n \log n)\ \ O(n!)\ \ O(2^n)$$

8 다음의 시간 복잡도 함수들을 증가 속도의 오름차순으로 정렬하라.

$$2^{3n},\ (n-10)!,\ 0.0001n4+3n+1,\ 4^n,\ 3^n,\ \log_2 n,\ \ln^2 n$$

$0.0001n^4$

9* 정의 2.1~2.3을 이용해 다음을 증명하라.

(1) $0.1n^2 - 10000 \in O(n^2)$
(2) $2n(n^2+1) \in O(n^3)$
(3) $0.000001n^3 \notin O(n^2)$
(4) $5n^2 + 10000n \in \Omega(n^2)$
(5) $100000n + 8 \notin \Omega(n^2)$
(6) $2n^3 + 3n \in \Theta(n^3)$
(7) $2n^3 + 3n \notin \Theta(n^2)$
(8) $2n^3 + 3n \notin \Theta(n^4)$
(9) 최고 차수가 k인 모든 다항식 $\in O(n^k)$

10 똑같이 생긴 n개의 동전과 양팔 저울이 있다. 동전 중에서 하나는 가짜 동전이고 진짜 동전과 무게가 약간 다르다. 양팔 저울은 동전의 무게를 직접 측정할 수 없고, 양쪽의 무게가 같은지 또는 어느 쪽이 더 무거운지를 알 수 있을 뿐이다. 이 저울을 이용해 $O(1)$에 위조 동전이 진짜 동전보다 무거운지 가벼운지를 판단하는 알고리즘을 작성하라. 단, $n > 2$이다.

11 다음의 합을 구하라.

(1) $11 + 12 + 13 + \cdots + 19$
(2) $2 + 4 + 6 + 8 + \cdots + 4096$
(3) $\sum_{i=1}^{n} 1$
(4) $\sum_{i=1}^{n}\sum_{j=1}^{n} 1$
(5) $\sum_{i=1}^{1000}\sum_{j=1}^{n} 1$
(6) $\sum_{i=1}^{n} i$
(7) $\sum_{i=1}^{n} i^2$
(8) $\sum_{i=1}^{n}\sum_{j=1}^{n} ij$

12 문제 11의 (3)~(8)에 대한 시간 복잡도를 빅오 표기법으로 나타내라.

13 다음 알고리즘에 대해 물음에 답하라.

```
def mystery1(n) :                # n은 음이 아닌 정수
    sum = 0
    for i in range(1, n+1) :
        sum = sum + i
    return sum
```

(1) 이 알고리즘의 용도를 설명하라.
(2) 이 알고리즘에서 기본 연산을 찾아라.
(3) 기본 연산이 몇 번 수행되어야 하는가?
(4) 이 알고리즘의 복잡도를 설명하라.
(5) 더 좋은 알고리즘이 있다면 제시하고, 복잡도를 설명하라.

14 다음 알고리즘에 대해 문제 13의 (1)~(5)에 답하라.

```
def mystery2(A) :                # A는 실수의 리스트
    min = A[0]
    max = A[0]
    for i in range(1, len(A)) :
        if max < A[i] :
            max = A[i]
        if min > A[i] :
            min = A[i]
    return max-min
```

15 다음 알고리즘의 시간 복잡도를 빅오 표기법으로 나타내라.

(1)

```
sum = 0
for i in range (n):
  for j in range (n):
    sum = sum + j
```

(2)

```
sum = 0
k = 1
while k <= n :
  sum = sum + k
  k = k * 2
```

(3)

```
sum = 0
for i in range (n):
  for j in range (i+1, n, 2):
    sum = sum + j
```

(4)

```
def algorithm(n) :
  k = 0
  while n > 1 :
    n = n / 2
    k++
  return k
```

16 다음의 순환 알고리즘에서 잘못된 점을 찾아라.

(1)

```
def recursive1(n) :
    if n==1 : return 0
    return n * recursive1(n)
```

(2)

```
def recursive2(n) :
    print('현재 n = ', n )
    return n * recursive2(n-1)
```

17 다음의 순환 알고리즘을 sub(3)과 같이 호출할 때 sub()가 호출되는 전체 횟수를 구하라.

```
def sub(n) :
    if n <= 1 : return n
    return sub(n-1) + sub(n-2)
```

18 다음 함수를 sum(5)로 호출하였을 때, 화면에 출력되는 내용과 함수의 반환 값을 구하라.

```
def sum(n) :
    print(n)
    if n<1 : return 0
    else : return n + sum(n-1)
```

19 다음 함수에서 asterisk(5)와 같이 호출할 때 출력되는 *의 개수는?

```
def asterisk(i) :
    if i > 1 :
        asterisk(i/2)
        asterisk(i/2)
    print("*", end='')
```

20 다음의 순환 관계식을 풀어라.

(1) $T(n) = T(n-1)+3 \quad \text{for } n>1, \quad T(1)=0$

(2) $T(n) = 2T(n-1) \quad \text{for } n>1, \quad T(1)=4$

(3) $T(n) = T(n-1)+n \quad \text{for } n>0, \quad T(0)=0$

(4) $T(n) = T(n/2)+1 \quad \text{for } n>1, \quad T(1)=1 \ (n=2^k \text{ 라고 가정})$

(4) $T(n) = T(n/3)+n \quad \text{for } n>1, \quad T(1)=1 \ (n=3^k \text{ 라고 가정})$

21* 다음과 같은 순환 알고리즘이 주어졌다. 물음에 답하라.

```
def mystery3(n) :                # n은 양의 정수
   if n == 1 : return 1
   else : return mystery3(n-1) + 2 * n - 1
```

(1) 이 알고리즘의 반환값에 대한 순환 관계식을 만들고 풀어라. 이것은 무슨 알고리즘인가?

(2) 기본 연산으로 곱셈을 사용했을 때의 복잡도를 순환 관계식으로 만들고 풀어라. 시간 복잡도를 설명하라.

(3) 기본 연산으로 덧셈과 뺄셈을 사용했을 때의 복잡도를 순환 관계식으로 만들고 풀어라. 시간 복잡도를 설명하라.