ch1장

1. (3)

2. ADT Set
객체 정의: 집합은 원소(element)라 불리우는 데이터 요소들의 모임
연산 정의:
Create() := 집합을 생성하여 반환한다.
Insert(S, item) := 원소 item을 집합 S에 저장한다.
Remove(S, item) := 원소 item를 집합 S에서 삭제한다.
Is_In(S, item) := 집합 S에 item이 있는지를 검사한다.
Union(S1, S2) := S1과 S2의 합집합을 구한다.
Intersection(S1, S2) := S1과 S2의 교집합을 구한다.
Difference(S1, S2) := S1과 S2의 차집합을 구한다.

3. ADT Boolean
객체정의: 0과 1
연산정의:
And(b1, b2) := if b1=1 and b2=1 then return 1;
else return 0;
Or(b1, b2) := if b1=0 and b2=0 then return 0
else return 1;
Not(b) := if b=0 return 1;
else return 0;
Xor(b1, b2) := if (b1=1 and b2=1) or (b1=0 and b2=0) then return 0;
else return 1;

4. 시간 복잡도 함수 $n^2+10n+8$를 빅오 표기법으로 나나내면? (3) $O(n^2)$

5. (1)

6. (3)

7. $100^2 : 1 = x^2 : 100$에서 x는 1000

8. $O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(2^n) < O(n!)$

9. (1) test(int n)

```
{
        int i;
        int total=1;                                    1번의 대입연산
        for(i=2;i<n;i++){                       루프 제어 문자은 무시
                total *= n;                             n-2번의 곱셈과 대입연산
        }
        return n;
}
```

-> 1+n-2+n-2번의 연산 -> $O(n)$

(2)

```
float sum(float list[], int n)
{
        float tempsum;
        int i;
        tempsum = 0;                                    1번의 대입연산
        for(i=0;i<n;i++) {                      루프제어 연산 무시
                tempsum += list[i];             n번의 대입연산, 덧셈연산
        }
        tempsum += 100;                         1번의 대입연산, 덧셈연산
        tempsum += 200;                         1번의 대입연산, 덧셈연산
        return tempsum;
}
```

-> 1+n+n+2+2 -> $O(n)$

(3)

```
void sum(int n)
{
        int i,b;
        b=2;                                            1번의 대입연산
        i=1;                                            1번의 대입연산
        while(i <= n){                          루프 제어 연산 무시
                i = i*b;                        log2n번의 곱셈, 대입 연산
        }
}
```

-> 1+1+$\log_2 n$+$\log_2 n$ -> $O(\log_2 n)$

10. 알고리즘 A: $1000n^2 + 1000$

알고리즘 B: 2^n

$n \geq 19$이면 $1000n^2 + 1000 < 2^n$

11. $30n + 4$와 n^2

$n \geq 31$이면 $30n + 4 < n^2$

12. $a, b > 1$인 경우에 $O(\log_a n) = O(\log_b n)$ 임을 증명하라. (문제에 오타가 있었음)

log의 성질을 이용하면

$\log_a n = \frac{\log_b n}{\log_b a} = \frac{1}{\log_b a} log_b n = c \bullet \log_b n$

따라서 $O(\log_a n) = O(\log_b n)$

13. $O(nlog_2 n)$

14.

(1) $O(n)$

(2) $O(\log_2 n)$

(3) $O(\log_2 n)$

(4) $O(n)$

(5) $O(n^2)$

(6) $O(n^2)$

(7) $O(n^2)$

15. $O(nlog_2 n)$

16.

(1) $5n^2 + 3 = O(n^2)$

$n \geqq 1$인 경우 $5n^2 + 3 \leqq 1000n^2$이므로 $O(n^2)$

(2),(3),(4) 도 동일하게 증명 가능

17. $n > n_0$일때 $6n^2 + 3n < c \bullet n$을 만족하는 n_0와 c를 찾을 수가 없으므로 $6n^2 + 3n$은 $O(n)$이 될 수 없다.

18.

```
int i, k;
for(i=0; i<(n-2); i++){                    루프 제어 문장 무시
        for(k=0; k<30; k++){               루프 제어 문장 무시
                buffer[i][k] = 0;          30*(n-2)번 수행
        }
}
```

(1) $T(n) = 30*(n-2)$ 따라서 $O(n)$

(2) $O(n)$이므로 수행속도는 입력에 정비례한다. 따라서 수행시간은 변함이 없다.

19.

```
answer  = 1.0;                             1번
temp = a;                                  1번
```

```
k = n;                                          1번
while( k > 0 ) {                                루프제어문장 무시
        if( (k % 2) != 0 ) answer *= temp;      logn번의 비교연산,곱셈,대입연산(최대)
        k = (int) k/2;                          1번
        if( k != 0 ) temp *= temp;              logn번의 비교연산,곱셈, 대입연산(최대)
}
```

T(n)= 6logn+4(최대)
따라서 O(logn)

20.
(1) 배열의 n번째 숫자를 화면에 출력한다.-> 최악 O(1) 최선 O(1)
(2) 배열안의 숫자 중에서 최소값을 찾는다.-> 최악 O(n) 최선 O(n)
(3) 배열의 모든 숫자를 더한다.-> 최악 O(n) 최선 O(n)

21. 1시간에 10개의 입력을 처리할 수 있는 프로그램이 있다. 만일 속도가 100배 빠른 컴퓨터를 구입하여 동일한 작업을 한다면 프로그램의 시간 복잡도가 다음과 같은 경우, 1시간에 처리할 수 있는 입력의 개수는 얼마가 되겠는가?
->
(1) $T(n)=n$ ->
$T(n)=n$이므로 10개의 입력이라면 대략 10개의 연산이 수행되었다. 컴퓨터의 속도가 100배 빨라졌다면 단위시간당 수행하는 연산의 수가 100배가 된 것이므로 1시간에 10*100 연산을 수행할 수 있고 따라서 입력도 10*100개를 처리할 수 있다.

(2) $T(n)=n\log_{10}n$ ->
$T(n)=n\log_{10}n$이므로 10개의 입력이라면 1시간당 $10\log_{10}10$=10개의 연산이 수행되었다. 컴퓨터가 100배 빨라졌으므로 1시간에 처리할 수 있는 연산의 수가 100배 증가했다고 보면 1시간에 수행될 수 있는 연산의 수는 $100\times10\log_{10}10$=1000개가 된다. $T(n)=n\log_{10}n=1000$을 만족하는 n을 구하면 된다.

(3) $T(n)=n^2$
역시 같은 식으로 $T(n)=n^2$이므로 10개의 입력이라면 1시간당 10^2개의 연산이 수행되었다. 컴퓨터가 100배 빨라졌으므로 1시간에 처리할 수 있는 연산의 수가 100배 증가했다고 보면 1시간에 수행될 수 있는 연산의 수는 100×10^2개가 된다. $T(n)=n^2=100\times10^2$을 만족하는 n을 구하면 된다.

(4) $T(n)=n^3$
같은 방법으로 해결한다.

(5) $T(n) = 10^n$
같은 방법으로 해결한다.

22.

(1)
```
void insert_array(int loc, int value){
        int i;
        for(i=items-1; i>=loc;i--)
                array[i+1] = array[i];    // items-loc개의 대입연산
        array[loc] = value;               // 1개의 대입연산
        items++;                          // 1개의 산술연산
}
```

(2) 최선의 시간복잡도: 배열의 마지막에 삽입 2=$O(1)$
최악의 시간복잡도: 배열의 처음에 삽입 $O(items+2) = O(n)$
평균적인 시간복잡도: (2+3+4+...+(items+2))/items=O(items)=O(n)

23.

```
typedef struct {
        float real;
        float imaginary;
} complex;
```