REPORT

학과명 : 컴퓨터공학전공
교과명 : 데이터구조
교수명 : 박웅규
학 번 : 202311442
이 름 : 주동현
제출일 : 2024/03/13

서원대학교
SEOWON UNIVERSITY

연습문제

1 다음 중 선형 자료구조로 볼 수 없는 것은?

① 스택 ② 큐
③ 트리 ④ 연결 리스트

2 동일한 문제를 해결하는 알고리즘 A, B, C, D의 시간 복잡도가 다음과 같이 계산되었다고 하자. n이 충분히 크다고 할 때, 실행 시간이 적은 것부터 순서대로 나열해보라.

A: O(n)	B: $O(n^2)$	C: O(nlogn)	D: $O(2^n)$

3 시간 복잡도 함수 $n^2+10n+8$를 빅오 표기법으로 나타내면?

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

4 3개의 숫자 중에서 가장 큰 수를 찾는 알고리즘을 흐름도로 작성해보자.

5 다음의 빅오표기법들을 실행 시간이 적게 걸리는 것부터 나열하라.

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

6 다음 코드의 시간 복잡도는 어떻게 되는가?

```
int i, j;
for(i=0; i<n; i++) {
    for(j=0; j<n; j++) {
        printf("%d %d \n", i, j);
    }
}
```

7 다음 코드의 시간 복잡도는 어떻게 되는가?

```
int i;
for (i = 1; i <= n; i *= 2) {
    printf("%d \n", i);
}
```

2^k = n O(log n)

8 다음 코드의 시간 복잡도는 어떻게 되는가?

```
int i;
for (i = 1; i <= 10; i++) {
    printf("%d \n", i);
}
```

O(1) 상수 복잡도

9 $30n+4$가 $O(n)$임을 증명해보자. $O(n)$의 정의를 이용한다.

f(n) = 30n+4, g(n) = n 30n+4 ≤ c·n 30n+4 ≤ n, c=34, n0=1, n≥1, 34×n ≥ 30n+4 ∴ 30n+4 = O(n)

10 집합을 나타내는 Set 추상 데이터 타입을 정의해보자. 어떤 연산자들을 생각할 수 있는가? 예를 들어서 다음과 같은 연산자들을 우리말로 정의해보자.

(1) union(S,T): S와 T의 합집합

(2) intersection(S,T): S와 T의 교집합

(3) difference(S,T): S와 T의 차

(4) subset(S,T): S와 T의 부분집합

(5) is_empty(S): S가 비어있는가

(6) is_element_of(x, S): x가 S의 원소인가

(7) size(S): S의 크기

11 다음 알고리즘의 시간 복잡도를 n에 대한 함수로 나타내고, 빅오 표기법으로도 나타내어라.

```
int algorithm(int n) {
    int k = 0;
    while (n > 1) {
        n = n/2;
        k++;
    }
    return k;
}
```

n+3 O(n)

12 배열에 정수가 들어 있다고 가정하고 다음 작업의 최악의 시간 복잡도를 빅오 표기법으로 말하라.

(1) 배열의 n번째 숫자를 화면에 출력한다. O(1)

(2) 배열안의 숫자 중에서 최소값을 찾는다. O(n)

(3) 배열의 모든 숫자를 더한다. O(n)

프로그래밍 프로젝트

1 1부터 n까지의 합을 구하는 방법은 다음과 같이 3가지가 있다.

알고리즘 A: $sum = n(n+1)/2$ 공식 사용
알고리즘 B: $sum = 1+2+\cdots+n$
알고리즘 C: $sum = 0+(1)+(1+1)+(1+1+1)+\cdots+(1+1+\cdots+1)$

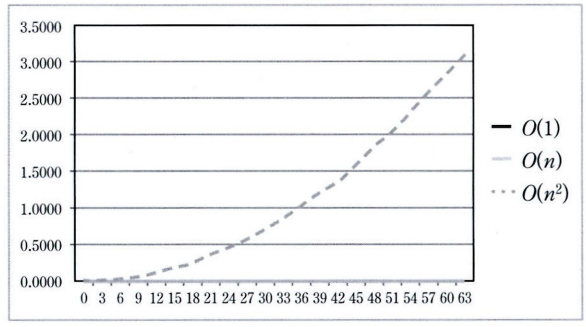
(1) 각 알고리즘을 함수로 구현하라. n을 매개 변수로 전달받고 결과를 반환한다.

```
예) int sumAlgorithmA ( int n );          // 알고리즘 A 구현 함수
```

(2) 비교적 작은 n에 대해 각 함수를 호출하여 세 알고리즘의 계산 결과가 동일함을 확인하라.

(3) 세 알고리즘의 시간 복잡도를 이론적으로 분석해 보고 빅오 표기법으로 나타내라.

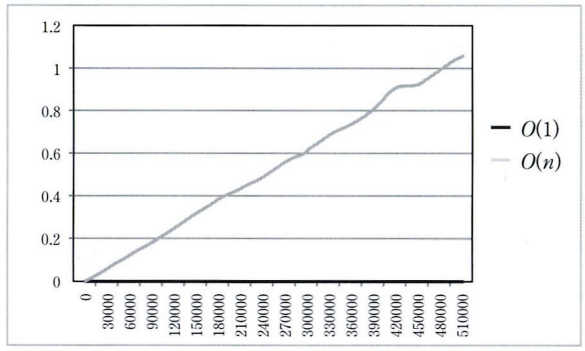
(4) 각 알고리즘의 실제 실행시간을 측정하여 이론적인 분석과 같게 나오는지를 조사해 보라. 실행 시간 측정을 위해 한 번의 루프에서 n은 2000 정도씩 증가시키고, 알고리즘 C의 실행시간이 1초가 이하일 때 까지 실행시킨 결과를 표로 만들고, 다음과 같이 선 그래프로 그려라(마이크로소프트 엑셀 등을 사용함).



| 그림 1.16 세 가지 알고리즘에 대한 실행시간 측정 결과의 예. 가로축은 n이고, 세로축은 실행 시간을 나타냄. A와 B는 변화가 없고, C는 n의 제곱에 비례해서 증가하는 것을 확인할 수 있음.

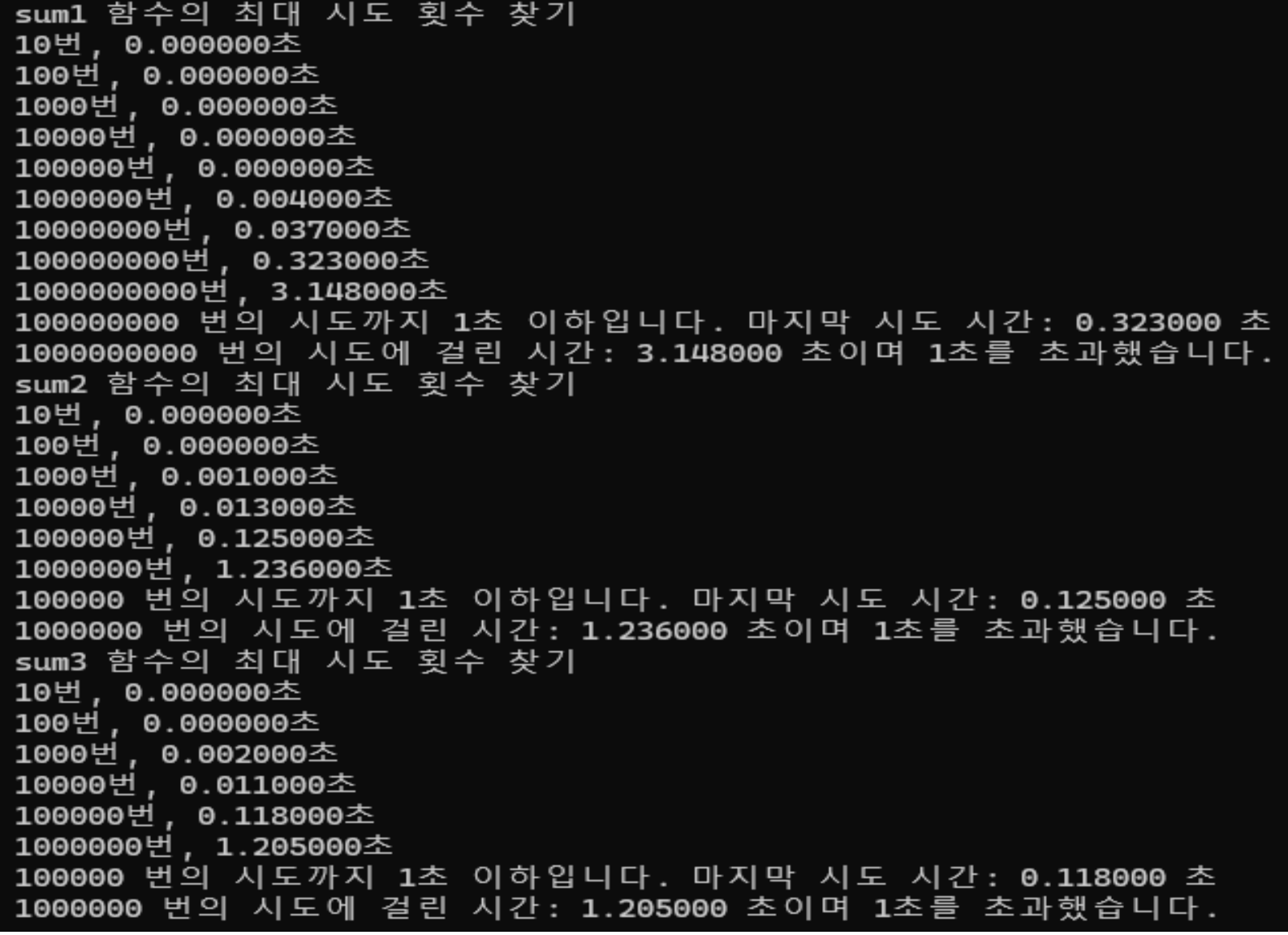
| Hint | 실행시간 측정에는 프로그램 1.2의 코드를 참조하면 된다. 그런데 우리가 사용하는 컴퓨터가 너무 빠르기 때문에 n이 커도 각 함수들을 한번 실행하는데 걸리는 시간이 너무 짧아 clock()함수가 정확한 값을 내지 못할 것이다. 따라서 안정적인 시간 측정을 위해서는 같은 함수를 1000번 실행한 시간을 측정하고, 이를 1000으로 나누는 것과 같은 방법을 사용하는 것이 좋다. 정확한 시간보다는 n에 따른 실행시간 그래프의 형태에 주목하라.

(5) 앞에서 알고리즘 A와 B의 시간 차이가 거의 나타나지 않았을 것이다. 그 이유를 생각해 보라. 이제 시간이 많이 걸리는 C알고리즘을 제외하고 이 두 알고리즘만을 비교해 보자. 한 번의 루프에서 n을 보다 더 크게 증가시켜 알고리즘 B가 1초 이하일 때까지 실행시킨 결과를 표로 만들고, 그래프로 그려라. 결과가 이론적인 분석과 같은지 조사해보라.



| 그림 1.17 알고리즘 A와 B만을 실행한 결과의 예. A는 O(1)이고 B는 O(n)으로 증가하는 것을 알 수 있음.

책에서처럼 a와 b가 실행 시간이 시도 횟수가 많아질수록 비슷하긴 하지만, 일정 시도횟수를 넘기면 b와 c의 실행 시간이 비슷해 진다. (rtx 3050, i5 11400H, 16gb ram 기준)



위 사진은 코드로 실행 시간을 측정한 것 이다. 첫 번째 사진과 같이 일정 시도횟수를 넘어가면 b와 c의 실행 시간이 비슷해진다. 그리고 a의 시도횟수, 1억번과 10억번 사이의 시간 차이가 크게 나온다.