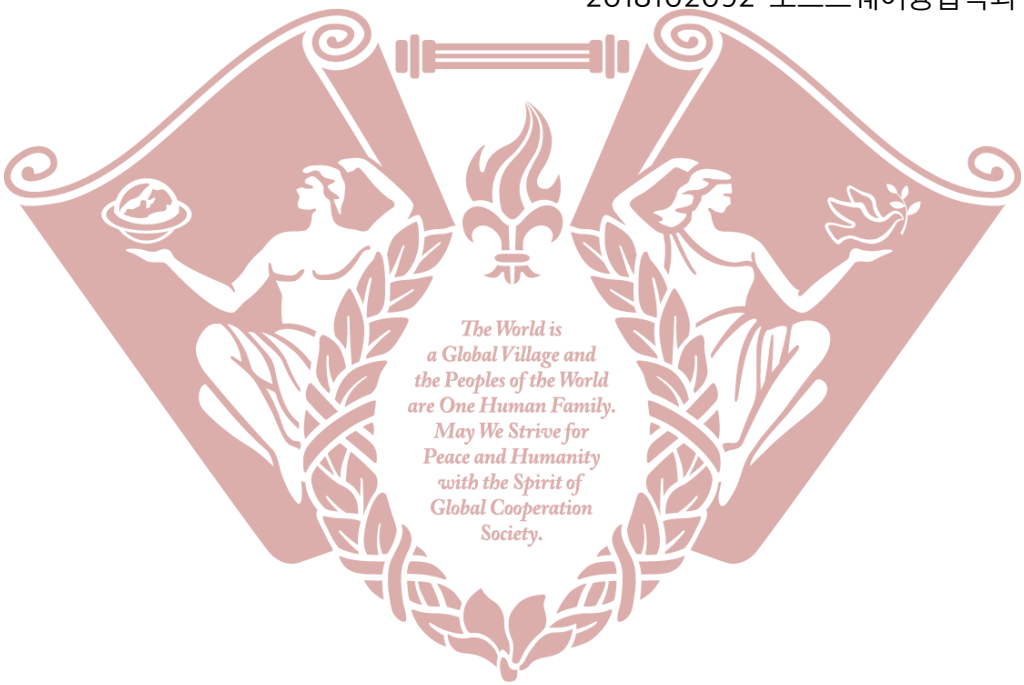
2020학년도 2학기 알고리즘분석 01 분반 한치근 교수님

알고리즘 분석

과제 1

2018102092 소프트웨어융합학과 권민지



1) 다음 n에 대해 alg1, alg2 수행시간을 측정하여 테이블을 완성하고, 수행시간 차이의 원인을 설명하라.

	수행시간 [단위: 초]	
n	alg1	alg2
10,000	10.970644235610962	-
20,000	44.250595569610596	-
30,000	95.7547357082367	-

[표 1] 수행시간 테이블

1-1) 천의 자리 n에 대한 alg1, alg2 수행시간 테이블

	수행시간 [단위: 초]	
n	alg1	alg2
1,000	0.0838162899017334	0.001954793930053711
2,000	0.38596582412719727	0.002993345260620117
3,000	0.8566873073577881	0.00498652458190918
4,000	1.7872817516326904	0.004921913146972656
5,000	3.036912441253662	0.0069925785064697266
6,000	3.9095396995544434	0.008976459503173828

[표 2] 천 단위 n에 대한 수행시간 테이블

```
7000 1

12255250.0
alg 1 time :  5.363646745681763

Traceback (most recent call last):
  File "C:/Users/Gwon/Downloads/알분/과제/assignment1.py", line 45, in recursive
    result1 = list[i] + i * (recursive(list, i-1, n ,total))
  File "C:/Users/Gwon/Downloads/알분/과제/assignment1.py", line 45, in recursive
    result1 = list[i] + i * (recursive(list, i-1, n ,total))
  File "C:/Users/Gwon/Downloads/알분/과제/assignment1.py", line 45, in recursive
    result1 = list[i] + i * (recursive(list, i-1, n ,total))
  [Previous line repeated 996 more times]
  File "C:/Users/Gwon/Downloads/알분/과제/assignment1.py", line 36, in recursive
    if(i == 0):
MemoryError: Stack overflow
```

[그림 1] 오류 화면

[그림 1]과 같이 재귀함수의 경우 호출 횟수가 많아 메모리 오류(stack overflow)가 뜬다. 램 12기가의 노트북에서도, 16기가의 데스크탑에서도 호출 횟수가 6000을 넘어가자 [그림 1]의 오류가 발생했다. 따라서 [표 1]의 alg2는 측정이 불가능했고, [표 2]와 같이 1000에서부터 6000까지의 수행시간을 측정하였다.

1-2) 소스코드

```
import time # measure the time

### set the limit on the number of call recursions
import sys
sys.setrecursionlimit(100000)

### create list
list1 = []
for i in range(7000):
    list1.append(i+1)

print(len(list1), list1[0], '\n')


### alg 1
def alg1(list):
    n = len(list)
    total = 0
    for i in range(n):
        sum = 0
        for j in range(i+1):
            if(i == 0):
                sum += list[i]
            else:
                sum += list[j]
        sum /= i + 1
        total += sum
    return total


startTime1 = time.time()
print(alg1(list1))
endTime1 = time.time()


print("alg 1 time : ", endTime1-startTime1, '\n')


### alg 2 : recursive function
def alg2(list, i, n, total):
    if(i == 0):
        total[0] += list[i]
        return list[i]

    if(i == n-1):
        result = ((alg2(list, i-1, n, total))*i + list[i]) / n
        total[0] += result
        return result

    result1 = list[i] + i * (alg2(list, i-1, n ,total))
    result2 = result1 / (i+1)
    total[0] += result2
    return result2
```

```
total = [0]
n = len(list1)
alg2(list1, n-1, n, total)

startTime2 = time.time()
print(total[0])
endTime2 = time.time()


print("alg 2 time : ", endTime2-startTime2)
```

[표 3] 코드 전체

1-3) 수행시간 차이의 원인

alg1의 경우, i-루프 안에 j-루프가 들어있는 형태로 정의하였다. 따라서, j-루프가 수행될 때마다 덧셈연산(j번째 요소까지의 총합을 더해 sum 변수에 넣는다)이 한 번씩 수행된다. 덧셈 연산의 총 수행 횟수는 [표 4]와 같다. 여기서 n은 리스트 요소의 개수(입력 데이터의 크기, len(list))를 의미한다. 따라서, $T(n) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ 이다. 궁극적으로는 n^2이 지배한다고 할 수 있다.

i	j-루프 수행 횟수
1	1
2	2
3	3
…	…
n	n

[표 4] j-루프 내의 덧셈연산의 총 수행 횟수

alg2의 경우, alg2(list, i, n, total)라는 재귀함수로 정의하였다. call by reference와 같은 기능을 하기 위해 각 평균 값을 받아올 total을 리스트로 정의하였고, n은 alg1과 마찬가지로 입력하는 리스트의 크기를 의미한다. alg2(n)을 계산하기 위해 alg2 함수를 호출하는 횟수를 T(n)이라 하자. 이때, [표 3]의 코드에서는 i가 리스트에 접근하기 위한 인덱스이므로 아래의 수식과는 달리 n 크기의 리스트에 대해 0부터 n-1까지 들어가게 되는 점을 참고하길 바란다. 따라서, alg2는 n이 지배한다고 할 수 있다.

$$T(1) = 1$$

$$T(2) = 2$$

…

$$T(n) = n$$

[표 2]의 수행시간 측정 결과를 보면, n = 4000인 경우부터 아예 두 알고리즘의 수행시간이 1초 이상의 차이가 나는 것을 확인할 수 있다. 이는 alg1은 n^2이, alg2는 n이 지배하기 때문이다. 따라서 n의 값이 커짐에 따라 알고리즘의 수행시간, 시간복잡도가 중요함을 확인할 수 있다.

2) n=10,000,000에 대해 alg1 수행시간을 추정하라. 큰 시간 단위(년, 월, 일 등)를 이용하여 표시하라.

n	alg1 수행시간 [단위: 초]
$1{,}000 = 1 \times 10^3$	$0.1 = 1^2 \times 10^{-1}$
$2{,}000 = 2 \times 10^3$	$0.4 = 2^2 \times 10^{-1}$
$3{,}000 = 3 \times 10^3$	$0.9 = 3^2 \times 10^{-1}$
…	…
$10{,}000 = 10 \times 10^3$	$11.0 \sim 10^2 \times 10^{-1}$
$k \times 10^3$	$k^2 \times 10^{-1}$

[표 5] 소수점 둘째 자리에서 수행시간을 반올림한 alg1의 수행시간 테이블

[표 1]과 [표 2]의 수행시간을 [표 5]와 같이 소수점 둘째 자리에서 반올림해 정리할 수 있다. 이때, $10{,}000{,}000 = 10000 \times 10^3$이므로 수행시간은 아래와 같이 추정할 수 있다.

$$10000^2 \times 10^{-1} = 10^8 \times 10^{-1} = 10^7 = 10{,}000{,}000(sec)$$

$$10000000 \div 60 \div 60 \div 24 = 115.740740741\ldots(day)$$

따라서 n이 10,000,000일 경우, alg1의 수행시간은 위의 과정에 따라 대략 116일 정도가 걸린다는 것을 추정할 수 있다.

3) alg1, alg2를 1시간 동안 수행할 때 해결할 수 있는 문제 크기 n을 각각 추정하라.

3-1) alg1을 1시간 수행할 경우

$$3600 = k^2 \times 10^{-1}$$

$$k^2 = 3600 \times 10$$

$$k \sim 189.73665961$$

$$k \times 10^3 \sim 189736.65961$$

[표 5]에 따라 alg1을 1시간(=60분=3600초) 동안 수행할 경우, 위의 수식에 따라 해결 가능한 문제 크기 n은 189,737이다.

3-2) alg2를 1시간 수행할 경우

n	alg2 수행시간 [단위: 초]
$1{,}000 = 1 \times 10^3$	$0.002 = 1 \times 2 \times 10^{-3}$
$2{,}000 = 2 \times 10^3$	$0.003 = 2 \times 1.5 \times 10^{-3}$
$3{,}000 = 3 \times 10^3$	$0.005 \sim 3 \times 1.7 \times 10^{-3}$
…	…
$6{,}000 = 6 \times 10^3$	$0.009 = 6 \times 1.5 \times 10^{-3}$
$k \times 10^3$	$k \times 1.4 \times 10^{-3}$

[표 6] 소수점 넷째 자리에서 수행시간을 반올림한 alg2의 수행시간 테이블

[표 6]은 [표 2]의 alg2의 수행시간을 소수점 넷째 자리에서 반올림하여 정리한 테이블이다. k는 음이 아닌 정수라 했을 때 $n = k \times 10^3$인 경우, alg2의 수행시간은 $k \times 1.4 \times 10^{-3}$초에 근사한다. 이때, 수식의 1.4는 [표 6]

의 alg2 수행시간 열의 2, 1.5, 1.7, …, 1.5의 평균이다. 따라서 1시간 동안 alg2를 수행하였을 때, 해결 가능한 문제 크기 n은 대략 2,571,428,570이다.

$$3600 = k \times 1.4 \times 10^{-3}$$

$$k = 3600 \div 1.4 \times 10^{3} \sim 2571428.57$$

$$k \times 10^{3} \sim 2571428570$$