# 알고리즘 과제 #1

201723274 소프트웨어학과
이서현

12. Write a Θ(n) algorithm that sorts n distinct integers, ranging in size between 1 and kn inclusive, where k is a constant positive integer. (Hint: Use a kn element array.)

Answer:

```
Sort( keytype A[]){
 Create array C[kn+1]
 for(i=0 ; i<n ; i++)
   initialize array C[i]
 for(i=0 ; i<n-1 ; i++)
   C[A[i]]=1;
 for(i=0 ; i<kn ; i++)
   if(C[i]==1){
     A[k]=i;
     k++;
   }
}
```

13. Algorithm A performs 10n^2 basic operations, and algorithm B performs 300ln(n) basic operations. For what value of n does algorithm B start to show its better performance?

Answer: $10n^2>=300lnn$ 이므로 $n^2>=30lnn$

n=7 이므로 49 <= 1.95 * 30 ( 58.5 ) 이므로 A algorithm이 better performance이다.

n=8 이므로 64 >= 2.08 * 30 ( 62.4 ) 이므로 B algorithm이 better performance이다.

따라서 n>=8 부터 B algorithm이 better performance가 된다.

15. Show directly that f ( n ) = n^2 + 3n^3 ∈ Θ( n^3 ). That is, use the definitions of O and Ω to show that f ( n ) is in both O( n^3 ) and Ω( n^3 ).

Answer: $f(n)=n^2+3n^3$ 이므로

$f(n)=O(g(n))$ 는 $0<=f(n)<=c*g(n)$ 이다. (c, k 는 양의 상수이고 모든 n>=k)

$0<=f(n)<=4n^3$ 이므로 c=4이고 $g(n)=n^3$, (n>0)

따라서 $f(n)=O(n^3)$

$f(n)= \Omega(g(n))$ 는 $0<=c*g(n)<=f(n)$ 이다. (c, k 는 양의 상수이고 모든 n>=k)

$0<=3n^3<=f(n)$ 이므로c=3이고 $f(n)=n^3$, (n>0)

따라서 $f(n)= \Omega(n^3)$

19. The function f(x)=3n^2 + 10nlogn + 1000n + 4logn + 9999 belongs in which of the following

complexity categories:
(a) θ (lg n ) (b) θ ( n ^2 log n ) (c) θ ( n ) (d) θ ( n lg n ) (e) θ ( n ^2 ) (f) None of these
Answer: (e)

20. The function f(x)=(logn)^2 + 2n + 4n + logn + 50 belongs in which of the following complexity categories:
(a) θ (lg n ) (b) θ ( n ^2 log n ) (c) θ ( n ) (d) θ ( n lg n ) (e) θ ( n ^2 ) (f) None of these
Answer: (b)

21. The function f(x)=n + n^2 + 2^n + n^4 belongs in which of the following complexity categories:
(a) θ (lg n ) (b) θ ( n ^2 log n ) (c) θ ( n ) (d) θ ( n lg n ) (e) θ ( n ^2 ) (f) None of these
Answer: (f)

28. Presently we can solve problem instances of size 30 in 1 minute using algorithm A, which is a Θ(n^2) algorithm. On the other hand, we will soon have to solve problem instances twice this large in 1 minute. Do you think it would help to buy a faster ( and more expensive) computer?
Answer: T(n)= 2^n이므로
현재 computer: $T(30)=C_1 2^{30}$= 1분 이므로 $C_1=1/2^{30}$ (min) 이다.
문제 해결된 computer: $T(60)=C_2 2^{60}$=1분 이므로 $C_2=1/2^{60}$(min) 이다.
$C_1/C_2=(1/2^{30})/(1/2^{60})=2^{30}$ 이므로
10억(1G)배 빠른 컴퓨터를 사야 한다.

34. What is the time complexity T(n) of the nested loop below? For simplicity, you may assume that n is a power of 2. That is, n=2^k for some positive integer k.

```
i=n;
while (i>=1) {
j=i;
while (j<=n) {
<body of the while loop> // Needs Θ(1).
j=2*j;
}
i=[i/2];
}
```

Answer: 바깥 while문은 i=1에서 i=n-1까지를 세어야 하고 내부 while문은 j=1에서 $\log_2(i)$까지를 세어야 한다. 따라서 $T(n)=\sum_{i=1}^{n-1}\sum_{j=1}^{|log2(i)|} c$ = c $\sum_{i=1}^{n-1} |log2(i)|$
T(n)=c[(nlog2(n-1)-2log(n-1)+1] 이므로 T(n)= Θ(n(logn)) 이다.