알고리즘 H/W #1

Due. 10/7(금) 5:00pm
제출: 302-313-2(최적화연구실)

1. (각 5점) Asymptotic running time을 구하고 과정을 밝히시오. Master Thm을 이용할 수 있는 곳에는 이용해도 좋음. Master Thm을 이용하지 않는 경우에는 O( )와 $\Omega()$ 모두 구하시오.

   1.1 $T(n)=4T(n/4)+b$ b: a constant
   1.2 $T(n)=3T(n/3)+n\log n$
   1.3 $T(n)=5T(n/5)+3n$
   1.4 $T(n)=T(n/4)+T(3n/4)+\Theta(n)$
   1.5 $T(n)=3T(n/3+9)+n$

2. (20점) 아래 알고리즘의 수행시간을 구하라.

```
sample(A[ ], p, r)
{
        if (r-p≤3) return 1;
        sum = 0;
        for i = p to r
                sum = sum + A[i];
        q = ⌊(r - p + 1)/4⌋;
        tmp = sum + sample(A, p, p+q-1) + sample(A, p+2q, r);
        return tmp;
}
```

3. (25점) 아래 알고리즘 test(n)의 수행 시간의 upper bound를 asymptotic notation으로 나타내어라. n은 양의 정수다. 문제를 풀면서 절대 $\frac{n}{3}+5$과 $\frac{2n}{3}+7$를 $\frac{n}{3}$으로 근사 취급해서는 안된다.

```
int test(n)
{
        if (n ≤ 50) then return n ;
        else return (test(n/3+5) + test(2n/3+7)) ;
}
```

4. (20점) Selection Sort 알고리즘이 제대로 sorting 한다는 것을 수학적 귀납법으로 증명하라.

5. Quicksort의 worst-case running time은 $\Theta(n^2)$이다. Worst-case에도 $\Theta(n\log n)$이 되도록 quicksort 알고리즘을 수정하고, Worst-case running time이 $\Theta(n\log n)$임을 증명하라.

6. (20점) Mergesort에서 둘로 나누는 대신 16 개로 나누어도 sorting은 된다. 즉, 최상위 레벨에서 mergesort를 16 번 부른 다음 merge를 한다. 이 경우의 알고리즘을 기술하고 (상

식적인 선에서 기술. 너무 자세할 필요 없음.) complexity를 분석하라. Heap을 이용해서 merge 부분을 효율적으로 하는 방법도 기술에 포함시킬 것.

7. 아래는 Mergesort 알고리즘이다. Array A[0...n-1]을 mergesort 하는 과정에서
   7.1 temporary array tmp[...]는 총 몇 회 생성되는가? Asymptotic 횟수를 말함.
   7.2 생성되는 temporary array tmp[...]의 사이즈의 총합은 어떻게 되는가? Asymptotic 사이즈를 말함.

```
mergeSort(A[], p, r):    ◀ A[p...r]을 정렬한다.
    if (p < r)
        q ← ⌊(p+r)/2⌋          ◀ p, r의 중간 지점 계산
        mergeSort(A, p, q)      ◀ 전반부 정렬
        mergeSort(A, q+1, r)    ◀ 후반부 정렬
        merge(A, p, q, r)       ◀ 병합
merge(A[], p, q, r):
◀ local array tmp[]는 여기서 정확히 필요한 크기 만큼만 생성함
    i ← p; j ← q+1; t ← 0
    while (i ≤ q and j ≤ r)
        if (A[i] ≤ A[j]) tmp[t++] ← A[i++]   ◀ 의미: tmp[t] ← A[i]; t++; i++;
        else tmp[t++] ← A[j++]               ◀ 의미: tmp[t] ← A[j]; t++; j++;
    while (i ≤ q)          ◀ 왼쪽 부분 배열이 남은 경우
        tmp[t++] ← A[i++]
    while (j ≤ r)          ◀ 오른쪽 부분 배열이 남은 경우
        tmp[t++] ← A[j++]
    i ← p; t ← 0
    while (i ≤ r)          ◀ 결과를 A[p...r]에 저장
        A[i++] ← tmp[t++]
```