알고리즘 중간고사

Open book, 1999. 10. 26.

1. (25점) 아래의 recurrence relation들에 대한 asymptotic running time을 구하라. (tight하게)
   A. $T(n) = 3T\left(\frac{n}{3}\right) + n \log n$
   B. $T(n) = T(\alpha n) + T(\beta n) + \theta(n),\ \alpha + \beta = 1$
   C. $T(n) = T(\alpha n) + T(\beta n) + \theta(n),\ \alpha + \beta < 1$

2. (5점) n 개의 element가 -5n 부터 5n 범위의 값을 갖는다. 이 n 개의 element들을 sorting하는데 필요한 asymptotic time은 어떻게 되는가?

3. (10점) mergesort를 다음과 같이 바꾼 경우의 asymptotic running time을 구하라. (tight하게)

```
mergesort( A, p, r )
   // p: starting index
   // r: ending index
   if ( p < r ) {
      q = p + (r-p)/3;
      mergesort( A, p, q );
      mergesort( A, q+1, r );
      merge ( A, p , q, r );
   }
merge( A, p, q, r )
      sort 되어 있는 두 list A[p], …, A[q]와 A[q+1], …, A[r]로부터 하나의 sort된 list를 만든다.
```

4. (15점) Linera-time selection algorithm의 골격은 아래와 같다. step c 에서 median 들의 median을 찾는 대신 median들 중 작은 순서로 3/5이 되는 M을 찾도록 수정하면 (예를 들어 median이 100개 있다면 이 중 60번째 것을 M으로 찾는다.) 여전히 linear-time selection이 가능한가? 당신의 대답을 justify하라. 단, 계산의 편의를 위해, step c를 수행한 후 median들은 항상 3:2로 정확하게 나누어진다고 가정해도 좋음. 같은 이유로 step a 에서도 각 group이 정확하게 5개씩 꽉 찬다고 가정해도 좋음.
   A. Divide the input into n/5 groups of 5 elements each.
   B. Find the median of each group.
   C. Find the median M of the medians of Step b by recursion.
   D. Use M as the pivot element and partition the whole elements as in Quicksort.
   E. Select the proper side of the partitions and recursively perform Steps a-e.

5. (15점) size 100인 hash table 상에서 open addressing을 위한 hash function이 다음과 같이 주어진다.

$$h'(x) = x \bmod 100$$
$$h_1(x) = (h'(x) + 2i + i^2) \bmod 100$$

   여기에 50개의 입력 데이터가 110, 200, …, 200, 210, 220, …, 500, 510, 520, …, 600의 순서로 들어온다. 이후에 이 데이터들을 search 하고자 할 때 성공적인 search의 probing 회수의 기대치는? (정확한 회수를 요구함)

6. (10점) n 개의 실수로 된 집합에서 가장 작은 k원소들 중 임의의 하나를 찾아내기 위해 최소 몇 번의 비교가 필요한가? 비교 회수를 가장 최소화하는 알고리즘을 제안하고 비교 회수를 밝히라. (말로 설명하는 것으로 충분함. asymptotic notation이 아닌 정확한 비교회수를 말함) 예를 들어 집합 {4, 5, 10, 2, 7, 9, 16}이 주어지고 k가 3이라면 알고리즘은 2, 4, 5 중 하나만 리턴하면 된다.

7. (20점) 첫 번째 숙제의 마지막 문제를 상기하자. 문제는 아래와 같고 이의 가능한 정답 하나를 제시한다.

[문제] 알파벳 $\Sigma = \{a, b, c\}$ 의 세 원소로 이루어지는 연산 table이 오른쪽과 같이 주어진다. 예를 들어 ab = b, ba = c, cc = c 와 같이 계산된다. a(b((cb)a)) = a이다. 괄호를 치는 방법에 따라 연산의 순서가 달라지고 결과도 달라진다. 테이블에서 알 수 있듯이 위의 연산은 commutative하지도 않고 associative하지도 않다. 임의의 스트링 $x_1x_2 \ldots x_n$ ($x_i \in \Sigma$ for i = 1, 2, ..., n)에 대하여 결과가 a가 되는 괄호치기 방법이 있는지 알아보는 (최대한 효율적인) 알고리즘을 작성하라. 예를 들어 bbbba를 입력으로 받았다면 결과가 a인 방법이 존재하므로 "yes"를 리턴해야 한다. [((b(b(b(ba)))) = a]

뒤

	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

앞

ca ba aa

[정답]

```
// V_ij : x_i ... x_j를 괄호쳐서 만들 수 있는 심볼들의 집합
for i = 1 to n
        V_ii = {x_i};
for d = 1 to n−1 {
        for i = 1 to n−d {
                j = i + d;
                for k = i to j−1
                        V_ij = {x | y ∈ V_ik, z ∈ V_k+1,j, x = yz};
        }
}
if V_1n ≠ ∅, then return "Yes";
        else return "No";
```

이제 문제를 약간 바꾸어 임의의 스트링 $x_1x_2 \ldots x_n$에 대하여 결과가 a인 괄호치기 방법의 총 수를 리턴하도록 수정한 아래의 알고리즘에서 '?' 부분을 채워 넣어라.

```
// C_ij^(a), C_ij^(b), C_ij^(c): x_i...x_j에서 a, b, c를 만들어 낼 수 있는 괄호치기 방법의 총 수
for i = 1 to n {
        C_ii^(x_i) = 1;
        C_ii^(y) = 0 for y ∈ Σ − {x_i};
}
for d = 1 to n−1 {
        for i = 1 to n−d {
                j = i + d;
                for k = i to j−1 {      < Cij(a) = Cik(c) x C
                        ?
                }
        }
}
return C_1n^(a);
```