CS 300 Midterm (2010-03-26)

Name : 정진근
Student Number : 20070764

75

1. (5 pts × 10 = 50 pts) Give T(True) or F(False) for each of the following statements. Justify your answers.

5

(1) Sorting 6 elements with a comparison sort requires at least 10 comparisons in the worst case.

True. 6개의 element 인 경우 가능한 입력순서는 $6!$ 가지이며 sorting은 이들을 모두 구분하여야한다. $6! = 720$ 인데, 9번의 비교로는 $2^9 = 512$ 가지 분류이다. (여기서 2는 ≤ or > 의 두가지 정보) 그러므로 최소 10번 ($2^{10} = 1024$)의 비교로 720 이상을 분류할수 있어야한다.

5

(2) Checking if there is a pair of non-equal elements in an array with n numbers requires $\Omega(n \log n)$ time.

False. 한쌍이라도 서로 다른값을 가진다면 True이므로, 결국 모두 같은 값의 array가 아니면 true이다. 모두같은 값인지를 보려면, A[0] (array의 첫번째 값)과 A[1]~A[n]을 하나씩 비교하며 모두같은지 보면 된다. C-like language로는

```
boolean chk (int *A, int n){
    int a = A[0];
    for (i=1; i<n; i++){
        if( a != A[i])
            return true;
    }
    return false
}
```

로 표현되며

→ $O(1)$
→ $O(n)$
→ $O(1)$

$O(n)$ 이므로 not $\Omega(n \log n)$

5

(3) Every array sorted in decreasing order is also a Max-Heap.

True. Max-Heap의 조건은 "parent가 ~~ch~~ 자신의 모든 child보다 크거나 같다." 이다. array 구현에서 parent의 index는 항상 child보다 작은데!. decreasing order로 sorting되어 있으면 모든 값 (A[i])는 $k(i-1)+j+1$ 으로 나타나는 k-ary heap의 자신의 child보다 크다. $(k>1), (j \geq 0)$

∴ Max-Heap

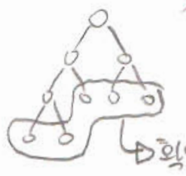
O

(4) Radix sort works correctly even if insertion sort is used as its auxiliary sort instead of counting sort.

True, RADIX sort는 각 자리를 낮은 자리부터 sorting하면 전체 sorting이 되는 것인데, auxiliary sort로 어떤 sort를 사용하였는가 하는것은 running time에만 영향을 줄뿐 Correctness에는 아무런 영향을 ~~줄수 없다.~~ insertion sort는 comparision sort로 어떤 범위등의 제한조건없이 sorting할 수 있어 범위제한의 counting sort 보다 광범위하여 counting sort로 가능한 모든 sort를 포함한다.

5

(5) The minimum element in a max-heap containing n elements can be found in $O(\lg n)$ time.



확인필요

False max-heap은 "parent는 그의 다른 어떤 child의 값보다 크거나 같다" 에 이는 child의 상한에만 제한을 줄뿐 하한에는 아무런 영향이 없다. 즉 parent와 관계 없이 child는 매우매우 작은값을 가질수 있다. 그러므로 minimum element를 찾기 위해서는 우리는 모든 leaf들(뿐만아니라 child가 없는 값모두) 적어도 한번 보아서 그값을 확인하여야 한다. leaf의 개수는 binary heap에서 $\lceil \frac{n}{2} \rceil$로 나타나는등 모든 heap에서 $\Omega(n)$이므로 $O(\lg n)$ time에는 불가능

(6) An adversary can present an input of n distinct numbers to RANDOMIZED-SELECT that will force it to run in $\Omega(n^2)$ time. False

5

나쁜놈이 Partition 결과를 편파적으로하게 하는 방법은 사실 모두 같은 값의 array를 제공하는것이다. 하지만 distinct number라는 조건이 있다.

이는항상 0:n-1의 partition을 제공할 것이다

RANDOM이므로 더이상 나쁜놈이 할수있는일은 없고 ~~expectation 값이~~ 확률분포에 따라 나오게 된다

|0|0|1|1|1|1|…| -> 2^n

최대최소 교대로 선택 or 최소+1, 최대-1, …

distinct에서는 모든 확률분포가 골고루 존재하며 $\Omega(n^2)$으로 만드는 입력은 $\frac{O(c^n)}{n!}$으로 주어지며 나쁜놈이 인위적으로 이확률을 바꿀수 없다.

5

(7) The decision-tree model lower bound on comparison sorting can be used to prove that the number of comparisons needed to build a heap of n elements is $\Omega(n \lg n)$ in the worst case.

False,

Build heap은
$$\sum_{h=0}^{\lfloor \lg n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h) = O\left(n \cdot \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) = O(n)$$

$$\left(\because \sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{\frac{1}{2}}{(1-\frac{1}{2})^2} = 2\right)$$

으로 $\Omega(n \lg n)$이 아니다

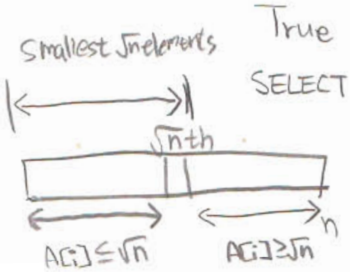
5 (8) One can sort n integers between 1 and m in time $O(n \log_n m)$. true

RADIX sort의 running time은 $\Theta(d(n+k))$로 d는 최대 자리수, k는 구성 수의개수 (k진수)이다. integer를 모두 n진수로 바꾸자. 그러면 $k=n$이된다. 그리고 최댓값은 n^d이 된다. $n^d = O(m)$ 이므로 $d = O(\log_n m)$ 이고, 모든 integer를 n진수로 바꾸는 작업은 각 integer당 $O(\log_n m)$ (-1 우시하자) 이므로 총 $O(n \cdot \log_n m)$, 이후 RADIX sort도 $O(d(n+k)) = O(2n \log_n m) = O(n \log_n m)$

총 running time도 $O(n \cdot \log_n m)$ 이된다.

5 (9) The sum of the smallest $\sqrt{n}$ elements in an unsorted array of n distinct numbers can be found in $O(n)$ time.

True



SELECT $\sqrt{n}$ th smallest value, PARTITION with pivot that value. 하면 이 된다. 그 후 $A[0]$부터 $A[\sqrt{n}-1]$까지 더하면된다. 이과정은

$$\left.\begin{array}{l} \text{SELECT} \to O(n) \\ \text{PARTITION} \to O(n) \\ \text{sum } A[0:\sqrt{n}-1] \to O(\sqrt{n}) \end{array}\right\}$$

$O(n) + O(n) + O(\sqrt{n}) = O(n)$ 이다.

(SELECT는 5개 median으로 worst-case $O(n)$)

5 (10) The collection $\mathcal{H} = \{ h_1, h_2, h_3 \}$ of hash functions is universal, where the three hash functions map the universe $\{A, B, C, D\}$ of keys into the range $\{0, 1, 2\}$ according to the following table :

x	$h_1(x)$	$h_2(x)$	$h_3(x)$
A	**1**	0	2
B	0	1	2
C	0	0	0
D	**1**	1	0

True

X_{ij} : $h_1(i) = h_2(j)$라하면.

$X_{AB} = \frac{\#\{h_3\}}{\#\{h_1, h_2, h_3\}} = \frac{1}{3}$ $X_{BC} = \frac{\#\{h_1\}}{\#\{h_1, h_2, h_3\}} = \frac{1}{3}$

$X_{AC} = \frac{\#\{h_2\}}{\#\{h_1, h_2, h_3\}} = \frac{1}{3}$ $X_{BD} = \frac{\#\{h_2\}}{\#\{h_1, h_2, h_3\}} = \frac{1}{3}$

$X_{AD} = \frac{\#\{h_1\}}{\#\{h_1, h_2, h_3\}} = \frac{1}{3}$ $X_{CD} = \frac{\#\{h_3\}}{\#\{h_1, h_2, h_3\}} = \frac{1}{3}$

즉, for all $x, y \in \{A, B, C, D\}$, $x \neq y$, $\left| \{ h(x) = h(y) \} \right|_x = 1 = \frac{|\mathcal{H}|}{m}$

이므로 universal

6

2. (7 pts) Give an algorithm to find the second smallest of n elements. Show how many comparisons are necessary in the worst case. An algorithm with less number of comparisons will get more credit.

3개의 list를 만든다.

A_0 : 비교후 한번도 다른 다른원소보다 크지않은 원소　　　A: 원래 list (비교한적 없음)

A_1 : 비교후 딱 한번 다른 원소보다 큰 원소

A_2 : 나머지.

☆ 비교한 두 값이 같을경우 하나는 더 작고 하나는 더 큰것으로 간주한다.

i) $|A_0| = 0$, $|A_1| = 0$ (처음; 이후로 $|A_0|=0$은 없다.)

→ A에서 두개를 골라 비교하고 작은 것은 A_0, 큰것은 A_1에 넣는다

ii) $|A_1| \geq 2$

→ ~~A_1 내부에서 비교하며 큰 녀석을 A_2에 넣는다.~~

$|A_1|=1$ $|A_0|=2$ $|A| \geq 2$

iii) $|A_1| = 1$, $|A_0| > 1$

→ A_0 내부에서 비교하며 큰녀석을 A_1에 넣는다

$|A_1| = 2$ $|A_0| = 2$

$|A| \geq 2$

iv) $|A_1| = 1$, $|A_0| = 1$, $|A| \geq 2$

→ A에서 두개를 골라 비교하고 작은것은 A_0, 큰것을 A_1에 넣는다.

v) $|A_1| = 1$, $|A_0| = 1$, $|A| = 1$

→ A의 마지막원소와 A_1의 원소를 비교하며 A가 크면 A를 A_2에 넣고 A가 작으면 A_0와 비교하며 큰것을 A_1에 작은것을 A_0에 넣는다

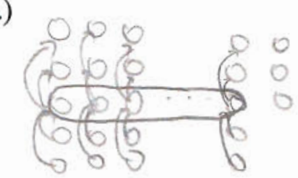
vi) $|A_1| = 1$, $|A_0| = 1$, $|A| = 0$

→ A_1의 원소가 second smallest.

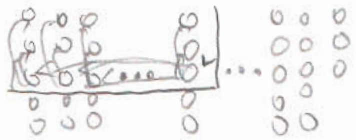
횟수분석을 위해서 순서를 조금바꿔 A를 먼저 비워보자. $n = 2k$라면 k회의 비교로 작은것은 모두 A_0에, 큰것은 A_1에 넣는다. 이후 A_0내부에서 $k-1$회 비교하여 큰것을 A_1으로 옮긴다. 이제 A_1의 크기는 $2k-1$개, 여기서 $2k-2$회실시하며 큰것을 A_2에 넣는다. $|A_0| = |A_1| = 1$, $|A| = 0$ 으로 종료, 총회수는 $k + (k-1) + (2k-2) = 4k-3 = 2n-3$회

$n = 2k+1$이라면 위의 상황후에 최대 2회비교추가이므로 $4k-1 = 2n-3$회로 같다.

3. (7 pts) Modify Quicksort to run in O(n lg n) time in the ***worst*** case, assuming that all elements are distinct. (You may use any procedure that you have learned in class.)



5개씩묶고 각 묶음에서 median을 찾는다. 이 $\lfloor \frac{n}{5} \rfloor$개의 median 중에서 또다시 median을 찾고, 이값을 pivot으로 사용한다.



→ pivot보다 작거나 같은 값은 적어도 $3 \cdot \lfloor \frac{n}{5} \rfloor$개
〃 큰 값도 적어도 $3 \cdot \lfloor \frac{n}{5} \rfloor$개 이다.

이값은 n이 충분히클때 $\frac{1}{4}$보다 크다. 이 말은 다른한쪽 partition (큰쪽)은 커봐야 $\frac{3}{4}$ 보다작음을 의미한다.

worst case: $T(n) \leq O(n) + T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right)$ →

∵ $T(n) = \omega(n)$ 이므로 $T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) \geq T\left(\frac{n}{4}+\alpha\right) + T\left(\frac{3}{4}n-\alpha\right)$ $(\alpha > 0)$

SELECT 5-median pivot 은 $O(n)$이고 partition도 $O(n)$이다.

$T(n) = O(n \lg n)$이라 하면 Substitution method?

$T(n) \leq c \cdot n \lg n$

(증식) $T(n) \leq O(n) + cn \cdot \frac{1}{4} \log \frac{n}{4} + cn \cdot \frac{3}{4} \log \frac{3}{4} n$

$= cn \log n - \frac{1}{2} cn - \frac{3}{4} \cdot \log \frac{4}{3} cn + c_2 n$

$= cn \log n - \left[\left(c_2 - \frac{1}{2}c - \frac{3}{4} \log \frac{4}{3} c\right) n\right]$

c를 충분히 크게 잡으면.

$\leq cn \log n$ ∴ worst-case $O(n \log n)$

$n \rightarrow m$

4. (7 pts) You use a hash function h to hash n distinct keys into an array T of length m. Assuming ***simple uniform hashing***, what is the expected number of collisions? Justify your answer.

서로다른 두키 x, y가 겹칠확률은 $\frac{1}{m}$이다.

$\left(x\text{의 키가 } a_i\text{라 할때 } y\text{의 키가 } a_i\text{일확률, 이므로} \quad P\left(h(y)=a_i \mid h(x)=a_i \right) = \frac{P(h(y)=a_i \cap h(x)=a_i)}{P(h(x)=a_i)} = \frac{\frac{1}{m}\cdot\frac{1}{m}}{\frac{1}{m}} \right)$

X_{ij} 를 서로다른 두키 x_i의 x_j가 충돌한사건이라면 $\quad X = \sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}$

$$E[X] = E\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}\right] = \underbrace{\sum_{i=1}^{n}\sum_{j=i+1}^{n} E[X_{ij}]}_{\text{linearity \& independent}} = \sum_{i=1}^{n}\sum_{j=i+1}^{n} \frac{1}{m}$$

$$= \frac{1}{m}\cdot\sum_{i=1}^{n}\sum_{j=i+1}^{n} 1 = \frac{1}{m}\cdot {}_nC_2 = \frac{1}{m}\cdot\frac{n(n-1)}{2}$$

7

5. (7 pts) Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A. Suppose that the elements of A form a uniform random permutation of <1, 2, …, n>. Use indicator random variable to compute the expected number of inversions.

A[i]가 A[1~i] 중에서 i-k+1 번째 작은 값이라면 A[1~i-1] 중 A[i]와 inversion은 i-k개이다.

↳ $1 \le k \le i$

A[i]가 A[1~i]중 i-k+1 번째일 확률은 $\frac{1}{i}$ 이다. ($\because$ random) → Y_{ik}라면

X_i를 A[i]와 A[1~i-1]에 있는 값들에 의한 inversion 이라면

$$E[X] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \sum_{k=1}^{i} E[Y_{ik}] = \sum_{i=1}^{n} \sum_{k=1}^{i} \left(\frac{1}{i} \cdot (i-k)\right)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{i} \left(1 - \frac{k}{i}\right) = \sum_{i=1}^{n} \left(i - \frac{1}{\cancel{i}} \cdot \frac{\cancel{i}(i+1)}{2}\right) = \sum_{i=1}^{n} \left(\frac{i}{2} - \frac{1}{2}\right) = \frac{n(n+1)}{4} - \frac{1}{2}n$$

$$= \frac{n(n-1)}{4}$$ //

3

$$\frac{n^2}{4} + \frac{n}{4} - \frac{n}{2}$$

6. (7 pts) Use a ***substitution method*** to solve the following recurrence.
$T(n) = 4\,T(n/3) + n$

$T(n) = O(n^{\log_3 4})$ 라고 가정하자.

$T(n) \leq c_1 n^{\log_3 4} + c_2 n$

$c_2 = 1 + \frac{c_2}{3}$

$3c_2 = 3 + c_2$

$2c_2 = 3$

$c_2 = \frac{3}{2}$

(증식) $T(n) = 4 \cdot T(\frac{n}{3}) + n$

$\leq 4c_1 \left(\frac{n}{3}\right)^{\log_3 4} + 4c_2\left(\frac{n}{3}\right) + n$

$= c_1 n^{\log_3 4} + (1 + \frac{4}{3}c_2)n$

$= c_1 n^{\log_3 4} + c_2 n + (1 + \frac{1}{3}c_2)n$

$\leq c_1 n^{\log_3 4} + c_2 n$ $\quad$ ($c_2 \leq -3$ 이면)

3

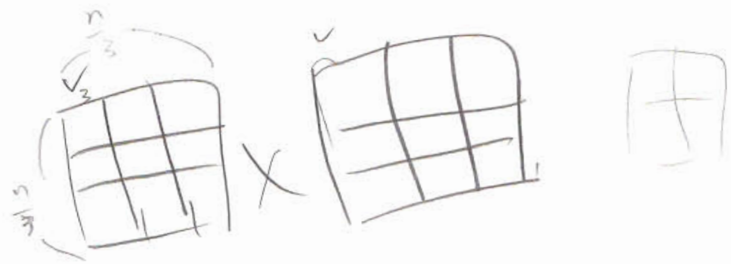
$\therefore T(n) = O(n^{\log_3 4})$

여기서 $c_2 = -3$으로하여

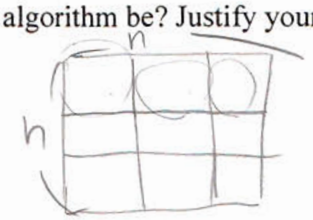
$T(n) = c_1 n^{\log_3 4} - 3n$

(증식) $c_1 n^{\log_3 4} - 3n = 4c_1 \cdot \left(\frac{n}{3}\right)^{\log_3 4} - 4n + n$

$= c_1 n^{\log_3 4} - 3n$



7. (7 pts) What is the largest k such that if you can multiply 3×3 matrices using k multiplications (not assuming commutativity of multiplication), then you can multiply $n \times n$ matrices in time $o(n^{\lg 7})$? What would the running time of this algorithm be? Justify your answers.



$$T(n) = 9 \cdot T\left(\frac{n}{3}\right) + k$$

if $T(n) < c \cdot n^{\lg 7}$

$$T(n) < 9 \cdot c \cdot \left(\frac{n}{3}\right)^{\log 7} + k$$

$$= 9 \cdot c \cdot n^{\log 7} - 9 \cdot c \cdot 3^{-\lg 7} + k$$

$k < 9 \cdot c \cdot 3^{-\lg 7}$ 이면 true. ($o(n^{\lg 7})$)

initial case, $T(3) = c \cdot 3^{\lg 7} < k$

$$c < \frac{k}{3^{\lg 7}} \text{ 이므로}$$

$T(n) = \quad \frac{n}{3}$

8. (8 pts) Suppose that you are given a sequence of n elements to sort. The input sequence consists of n/k subsequences, each containing k elements. The elements in a given subsequence are all smaller than the elements in the succeeding subsequence and larger than the elements in the preceding subsequence. Thus, all that is needed to sort the whole sequence of length n is to sort the k elements in each of the n/k subsequences. Show an $\Omega(n \lg k)$ lower bound on the number of comparisons needed to solve this variant of the sorting problem. (Hint : It is not rigorous to simply combine the lower bounds for the individual subsequences.)

우리는 각 sublist를 각각 sorting만하면 된다. ~~그 중에 가장~~

그들은 각각 $\Theta(k\lg k)$ time에 sorting된다.

i 번째 subsequence의 실행시간을 $c_i k\lg k$라 하자.

총시간은 $\sum_{i=1}^{\frac{n}{k}} c_i k\lg k$, 여기서 c_i중 최소값을 c라하면

$$\sum_{i=1}^{\frac{n}{k}} c_i k\lg k \geq \sum_{i=1}^{\frac{n}{k}} c\, k\lg k = \frac{n}{k}\cdot c \lg k = c \cdot n\lg k = \Omega(n\lg k)$$

즉 총시간의 lower bound는 $\Omega(n\lg k)$이다.