2020 알고리즘 Prof.조승범 Task (20.09.16)

학과	소프트웨어	학번	2019088004	이름	조민우

1. 챕터 1 강의자료 24 페이지에 나와있는 알고리즘 2 (Figure 1.1) 의 Pseudocode 가 올바르게 동작한다는 것을 mathematical induction on y 를 이용해 증명하시오.

(Base Case) $y=0$ 일때 성립한다고 가정

(Induction Step) $y=y'$ 일때 성립

$\Rightarrow$ x**y'이 정상적으로 return

$\rightarrow$ $y=y'+1$ 일때

출력 return값 $(x*(y'+1)) = \underline{x**y'+x}$ ①

i) $y'\%2==0$

$\rightarrow$ $y'+1$은 홀수

$\therefore z = x*\frac{y'}{2}$

이때 return $= x+2z = x+2*x*\frac{y'}{2}$

$= \underline{x**y'+x}$ ②

ii) $y'\%2==1$

$\rightarrow$ $y'+1$은 짝수

$\therefore z = x*\frac{y'+1}{2}$

이때 return $= 2z = 2*x*\frac{y'+1}{2}$

$= x*(y'+1)$

$= \underline{x**y'+x}$ ③

① = ② = ③

$\therefore$ Correct.

2. 교과서 0.1 의 b, j, k, 번을 해결하시오 (정확한 정의에 의해 증명할 필요는 없지만 이유는 간단히 언급해야 함).

0.1. In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \theta(g)$).

	f(n)	g(n)
(b)	$n^{1/2}$	$n^{2/3}$
(j)	$(\log n)^{\log n}$	$n/\log n$
(k)	$\sqrt{n}$	$(\log n)^3$

(b) n에 t^6을 대입 $\rightsquigarrow$ t^3 t^4

g(n)이 f(n)을 dominate.

$\therefore f(n) = O(g(n))$

(j) n에 2^t을 대입 $\rightsquigarrow$ t^t $\frac{1}{t}2^t$

f(n)이 g(n)을 dominate

$\therefore f(n) = \Omega(g(n))$

(k) n에 2^{2t}을 대입 $\rightsquigarrow$ 2^t $8t^3$

f(n)이 g(n)을 dominate

$\therefore f(n) = \Omega(g(n))$

3. 교과서 2.5 의 b, d, g 번을 해결하시오 (풀이과정도 같이 언급하시오).

2.5. Solve the following recurrence relations and give a bound for each of them.

(b) $T(n) = 5T(n/4) + n$
(d) $T(n) = 9T(n/3) + n^2$
(g) $T(n) = T(n-1) + 2$

(b) master-theorem

$a=5 \quad b=4 \quad d=1$

$\log_4 5 > 1$

$\therefore O(n^{\log_4 5})$

(d) master-theorem

$a=9 \quad b=3 \quad d=2$

$\log_3 9 = 2$

$\therefore O(n^2 \log n)$

(g) $T(n) = T(n-1) + 2$

$= T(n-2) + 2 + 2$

$\vdots$

$= T(1) + 2 + \cdots + 2$ ($n-1$)

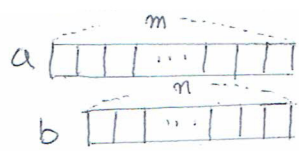
Base-case ↵

$\therefore O(1) + (n-1) * O(1)$

$= O(n)$

4. 교과서 2.22 (코드를 사용하지 말고 설명하고, correctness 및 time complexity 를 증명하시오).

2.22. You are given two sorted lists of size m and n. Give an O(logm + log n) time algorithm for computing the kth smallest element in the union of the two lists.



time-complexity 증명.

worst-case : a→0 b→1일때까지 반복

$\frac{m}{2^{x+1}} = 1 \quad 2^{x+1} = m$

$x-1 = \log m$

$x = \log m + 1$

$\frac{n}{2^y} = 1 \quad 2^y = n$

$y = \log n$

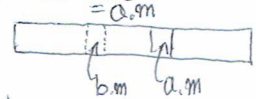
반복문관리하는시간 $= O(1)$

$\therefore O(\log m + \log n + 1)$

$= O(\log m + \log n)$

if) a의 median > b의 median

$= a.m$ $= b.m$



└b의절반이상 + a의반 a의반┘

if) $K < \frac{m+n}{2}$

$[1 \sim \frac{1}{2}m][1 \sim n]$ K

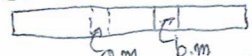
→상한선 (Upper bound 제한

else).

$[1 \sim m][\frac{1}{2}n \sim n]$ $K - \frac{n}{2}$

→하한선 lower bound 제한.

else)



└ a의절반이상 + b의반 b의반┘

if) $K < \frac{m+n}{2}$

$[1 \sim m][1 \sim \frac{1}{2}n]$ K

else)

$[\frac{1}{2}m \sim m][1 \sim n]$ $K - \frac{m}{2}$

→한 리스트가 0이될때까지 반복후 남은리스트의 K(고려할 수 있음) 번째 index반환

Upper-bound 증명은 method 1

lower-bound 증명은 method 2

Correctness 증명.

(Base-case) K=1,

Array a와 b를 method 1로 각각 최소값만 남을때까지 진행하고, 각각 최소값일때 서로 median이 되어 비교후, 큰값은 divided by 2에 의해 소거되므로 최소값만 남아 K=1일때 성립함을 증명.

(Induction-case) K=t이하일때 성립한다 가정.

K=t+1 일때, method 2를 이용해 lowerbound를 설정해 탈락하면 $K=t+1-\frac{m}{2}$ (혹은 $-\frac{n}{2}$)으로 귀납법으로 가설에의해 성립함.

method 2를 쓰지않을경우 (method 1 만사용한 경우) 는, K는 최소값을 탐색할때와 동치이므로, K=1일때 성립함을 증명했으므로 성립함.

5. Selection using median of medains 알고리즘의 첫번째 과정에서 각 subarray 의 크기를 3 또는 9 로 해도 worst-case O(n) 시간 안에 동작하는가 ? (나머지 과정은 모두 동일) 두 경우에 대해 동작하는 (혹은 동작하지 않는) 이유를 설명하시오.

① 크기 3

↳ Median O(1) subarray $\frac{n}{3}$

array M(size : $\frac{n}{3}$)

if) M.size ≤ 25 sort

else) recursion.

subproblem의 최대개수 = $n - (\frac{2}{3}n \times \frac{1}{2})$

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

$\therefore T(n) = T(n/3) + T(2n/3) + O(n)$

$\fallingdotseq T(n)$

Master Theorem : $a=1$ $b=1$ $d=1$

$\therefore O(n \log n)$

② 크기 9

↳ Median O(1) subarray $\frac{n}{9}$

array M(size: $\frac{n}{9}$)

if) M.size ≤ 25 sort

else) recursion

subproblem의 최대개수 = $n - (\frac{5}{9} \times \frac{1}{2})$

$= \frac{13}{18}n$

$\therefore T(n) = T(n/9) + T(13n/18) + O(n)$

$\fallingdotseq T(5n/6)$

Master Theorem : $a=1$ $b=\frac{6}{5}$ $d=1$

$\log_{\frac{6}{5}} 1 < 1$

$\therefore O(n)$