CS 303, Paul Cao, Lecture 16

Exercise class for exam 1

Topics

1. Efficiency analysis
   - definition of O, Ω, Θ
   - Properties of asymptotic notations
   - efficiency analysis methods (substitution, recursion tree, master theorem)
   - Divide and conquer algorithms (binary search, powering of a number, strassen's algorithm, max-subarray)
2. Sorting algorithms
   - insertion, mergesort, quicksort, heapsort, counting sort, radix sort
   - their efficiency (best worst average and when does it happen)
   - how each algorithm works
3. Hashing
   - idea of hashing
   - open hashing
     - idea
     - hashing function (division and multiplication)
   - closed hashing
     - idea
     - hashing function(linear, quadratic, double hashing)

1. prove that if $f(n) \in O(g(n))$, then $g(n) \in \Omega(f(n))$.

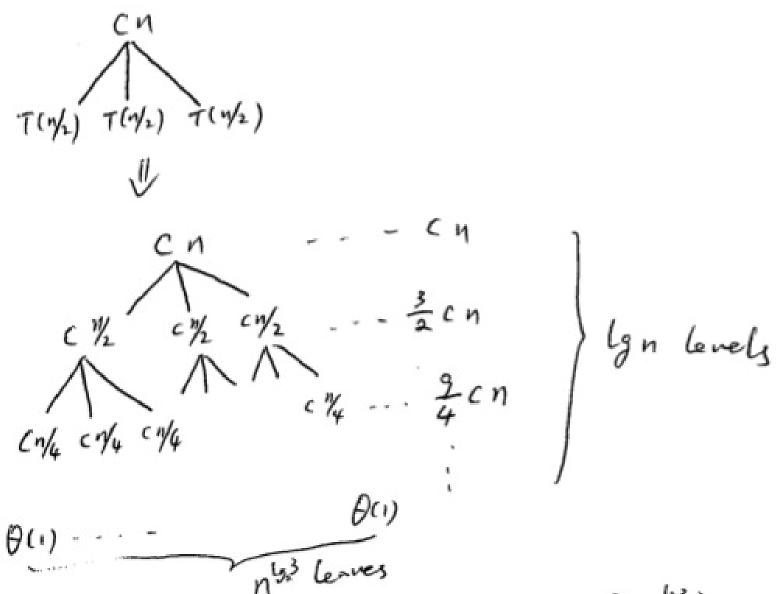
if $f(n) \in O(g(n))$, then $\exists\ c > 0, n_0$ s.t.

$f(n) \le c\, g(n)$ for all $n > n_0$

thus. $g(n) \ge \frac{1}{c} f(n)$. this is the def of Ω. Thus proved.

2. Use recursion tree method to find the efficiency of $T(n)$ if

a. $T(n) = 3T(n/2) + \underset{\underset{cn}{\parallel}}{\Theta(n)}$



Thus $T(n) = \underbrace{\left(cn + \frac{3}{2}cn + \frac{9}{4}cn + \cdots\right)}_{\lg n \text{ terms}} + \Theta(n^{\lg 3})$

$= cn\left(1 + \frac{3}{2} + \frac{9}{4} + \cdots\right) + \Theta(n^{\lg 3})$

$= cn \cdot \frac{\left(\frac{3}{2}\right)^{\lg n} - 1}{\frac{3}{2} - 1} + \Theta(n^{\lg 3})$

$= 2cn\left(\frac{3^{\lg n}}{2^{\lg n}} - 1\right) + \Theta(n^{\lg 3}) = 2c\, n^{\lg 3} - 2cn + \Theta(n^{\lg 3})$

$\in \Theta(n^{\lg 3})$

3. prove $T(n) = 4T(n/2) + \Theta(n)$ is $O(n^2)$

suppose $T(k) \le C_1 k^2 - C_2 k$ for all $k \le n$, $C_1 > 0$, $C_2 > 0$

Then

$$T(n) = 4T(n/2) + C'n$$

$$= 4\left(C_1\left(\tfrac{n}{2}\right)^2 - C_2\left(\tfrac{n}{2}\right)\right) + C'n$$

$$= C_1 n^2 - C_2 n - (C_2 - C')n$$

if we pick $C_2 > C'$, then the above is

$\le C_1 n^2 - C_2 n$

thus proved.

4. use heapsort to sort the following array

-7 11 -8 5 10 -2 100

① heapify: 100 11 -2 5 10 -7 -8

② root swap / heapify:

11 5 10 -8 -2 -7 | 100

10 5 -7 -8 -2 | 100 11

5 -2 -7 -8 | 100 11 10

-2 -8 -7 | 100 11 10 5

-7 -8 | 100 11 10 5 -2

-8 | 100 11 10 5 -2 -7

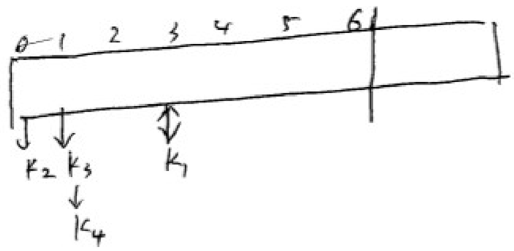
100 11 10 5 -2 -7 -8

done

5. use the following hash function to perform hashing into the hash table.

a. open hashing w/ $h(k) = k \bmod 7$

$k_1 = 10$, $k_2 = 21$, $k_3 = 8$, $k_4 = 22$



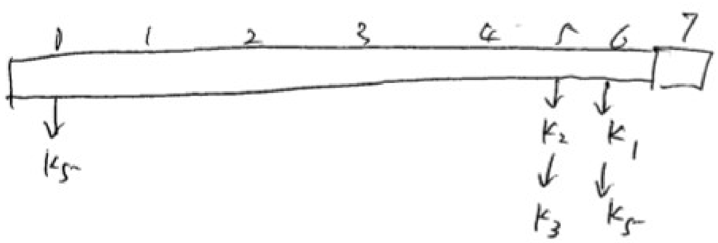
b. open hashing w/ $h(k) = (k * A \bmod 2^w) \gg (w-r)$

where $r=3$, $A = 1011011$, $w=7$, $m = 2^3 = 8$

$k_1 A = 1101101$ $\quad$ $k_4 A = 11001$

$k_2 A = 1011101$ $\quad$ $k_5 A = 0000000$

$k_3 A = 1010000$



c. closed hashing w/ $m = 13$, $\overset{h(k,i)}{\cancel{h(k)}} = (k \bmod 13 + i^2 + 5i) \bmod 13$

$k_1 = 5$ $\quad$ $k_3 = 6$

$k_2 = 18$ $\quad$ $k_4 = 70$