INDIAN STATISTICAL INSTITUTE

M. Tech (CS) - I Year, 2016-2017 (Semester - II)

Design and Analysis of Algorithms

Tutorial-1 Date: 13.01.2017

(Q1) Solve the following recurrences:

(i) $T(n) = 2 \cdot T(n/2 + 17) + n$.
Ans. $\Theta(n \log n)$.

(ii) $T(n) = 4 \cdot T(n/2) + n^2 \log n$.
Ans. $\Theta((n \log n)^2)$.

(iii) $T(n) = 2 \cdot T(n/2) + n/\log n$.
Ans. $\Theta(n \log \log n)$.

(iv) $T(n) = T(n/3) + T(2n/3) + n$.
Ans. $\Theta(n \log n)$.

(v) $T(n, r) = T(n-1, r) + T(n-1, r-1)$.
Ans. $\Theta(\binom{n}{r})$.

(vi) $T(m, n) = m + n + 2 \cdot T(m/2, n/2)$.
Ans. $\Theta((m+n)\log(m+n))$.

(Q2) Solve the following recurrence:

$$f(n) = \begin{cases} \sqrt{n} \cdot f(\sqrt{n}) + n & \text{if } n \geq 1; \\ 1 & \text{if } n = 1. \end{cases}$$

Ans.$\Theta(n \log \log n)$.

(Q3) Find an efficient algorithm to compute the second maximum of a set of n numbers.
Ans. Solve by using tournament method. The number of comparisons will be $n + \lceil \log n \rceil - 2$.

Also keep in mind the followiong. Any comparision based algorithm that finds second maximum, requires at least $n + \lceil \log n \rceil - 2$ many comparisons.

(Q4) Waiting for success bound: Let X be a binary random variable such that

$$X = \begin{cases} 1 & \text{if event E occurs;} \\ 0 & \text{otherwise.} \end{cases}$$

If $Pr(X = 1) = p$, then how many times we have to repeat the experiment to let E occur?
Ans. $1/p$.

Example: You are buying items. Each item comes with a coupon and there are n different types of coupons available. Your job is to collect n distinct coupons. How many items do you need to try on the expectation to collect n distinct coupons?

Ans. $nH_n = \Theta(n \log n)$, where $H_n = \sum_{i=1}^{n} \frac{1}{i}$.

(Q5) Let a, b be positive integers such that $b > a$. Find an algorithm to compute the gcd of b and a and deduce the time complexity.
Ans. Use the fact $GCD(a, b) = GCD(a, b \bmod a)$ and $b \bmod a \leq \frac{b}{2}$. The time complexity will be $O(n)$ divisions, where n is the number of bits required to represent b in binary form.

(Q6) Look at the most simple algorithm to find whether a natural number n is prime or not and analyse its time complexity.
Ans. Use the following fact. A number a is prime if and only if a has a divisor in between 2 and $\lfloor \sqrt{a} \rfloor$. So, the time complexity is $O(\sqrt{a})$. Let n be the input size i.e. number of bits required to represent a in binary form. Obseve that, $n = \log a$ and the time complexity is $O(2^{n/2})$. Hence, the time complexity is not polynomial in input size.