Purdue University | S. Hambrusch - C. Guerra
CS 381 | Fall 1999

Solution Sketches to Assignment 1

1) (Graded by Tian Luan.)
Find a closed form for the expression $1\star 2+2\star 3+3\star 4+\ldots+n\star(n+1)$ and prove it correct by induction.

(i) Let $S_n = 1\star 2+2\star 3+3\star 4+\ldots+n\star(n+1)$. Using the sums given on page 21 of the text we get $S_n = n(n+1)(n+2)/3$.

(ii) Prove by induction on n.
Base case: for $n=1$ we have $S_1 = 1*2 = 2$ and $1*(1+1)*(1+2)/3 = 2$. Hence, the claim holds for the base case.
Induction hypothesis: Assume that $S_k = k(k+1)(k+2)/3$ for $k = 2, 3\ldots, n-1$.
Consider now S_n:

$S_n = S_{n-1} + n(n+1)$, by definition
$= (n-1)n(n+1)/3 + 3n(n+1)/3$, by induction hypothesis
$= [n(n+1)(n-1+3)]/3$
$= n(n+1)(n+2)/3$ $\square$

2) (Graded by Tian Luan.) Partition the following functions representing running times into equivalence classes so that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. Rank the classes from smallest to largest (in terms of growth rate with respect to n).

The ordering of the functions is:
$4\log n, 4\log(n^3) < \sqrt{4n} + 4\log n < 3^{\log_3 n} < \sqrt{n} + 4n\log n < \frac{n}{6}(\log n)^2 < 3^{\log n} < 3n^2 + 4n\log n < n^{\log 6} < n^{10} - n^6 + 15n^3 < 3^{n/3} < 2^n < 3^{n-3}, 3^n < n^{n/2}$

Recall that to prove $f(n) = \Theta(g(n))$, we need to show the existence of constants c_1, c_2, and n_0 such that $c_2 g(n) \leq f(n) \leq c_1 g(n)$ for every $n > n_0$. One method of determining whether a function $f(n)$ is $\Theta(g(n))$ is to use the formal definitions and find the constants satisfying the inequalities. It helps to first simplify the functions. Here are some examples:

$$4\log(n^3) = 12\log(n)$$

It is clear that this function belongs to $\Theta(\log(n))$.

$$3^{\log_3 n} = n$$

and we know that it belongs to $\Theta(n)$.

$$3^{\log n} = n^{\log_2 3}$$

and we can conclude that the function is polynomial in n: $\Theta(n^{\log_2 3}) = \Theta(n^{1.59\ldots})$.
Another method to compare two functions is to determine the limit of $\lim_{n\to+\infty} \frac{f(n)}{g(n)}$.
The rule of thumb is that, if this limit is

a) a constant, then $f(n) = \Theta(g(n))$

b) equal to zero, then $f(n) = O(g(n))$

c) approaches ∞, then $f(n) = \Omega(g(n))$

For example:

$$\lim_{n\to\infty} \frac{3^n}{3^{n-3}} = \lim_{n\to\infty} \frac{3^n}{3^n \star 3^{-3}} = 9$$

Thus they two belong to same equivalence class $\Theta(3^n)$.

$$\lim_{n\to\infty} \frac{3^n}{3^{\frac{n}{3}}} = \lim_{n\to\infty} 3^{\frac{2n}{3}} = \infty$$

Thus 3^n grows faster than $3^{\frac{n}{3}}$. They are not in the same complexity class.

$$\lim_{n\to\infty} \frac{2^n}{3^{\frac{n}{3}}} = \lim_{n\to\infty} (\frac{2}{\sqrt[3]{3}})^n = \infty$$

Thus 2^n grows faster than $3^{\frac{n}{3}}$. They are not in the same complexity class.

3) (Graded by Biana Babinsky.)
The health department needs to test n water samples for pesticide X. For this purpose water is collected and the samples are labeled. There exists an expensive test to determine whether water contains pesticide X. One may mix portions of water samples to conduct tests on several water samples simultaneously. (A positive outcome implies that at least one original sample contains the pesticide.)

i) Describe an efficient method to determine p, the number of water samples which contain pesticide X. The amount of water available for each sample is not a constraint. Your method should be efficient when p is considerably smaller than n. State the number of tests necessary in terms of n and p.

Assume (without loss of generality) that n is a power of 2.
General idea of the algorithm: Take a drop of each sample, combine into a mixture (call this mixture S1) and test it for the pesticide. If S1 does not contain the pesticide, we are done. If S1 contains the pesticide, divide the samples into two groups, each of size $n/2$. Solve the problem recursively for each group (a recursive call returns an integer count and these counts are added up.) Recursion ends when group corresponds to one sample or a group contains no pesticide.

Most solutions handed in gave an iterative description. The description for the first test is then as above. The next step in the iterative description is to take a drop of each sample in the first group, generating sample S2 and to take a drop of each sample in the second group, generating sample S3. Test S2 and S3. If a sample does not contain pesticide, discard it. If it does, divide its group into halves and continue testing. The testing will continue until all the groups are discarded or until a group corresponds to a single sample. When our testing brings us to a single sample, we know that that this sample contains the pesticide.

This process can be represented by a binary tree having p leaves. The tree has the following characteristics: a node is either a leaf or an interior node with 2 children. The tree has height at most $\log n$.

Next, turn to the analysis of this method. We don't know p and even when we knew p, the exact number of tests depends on where the p contaminated samples are when groups are formed. The worst case situation for a given p corresponds to a binary tree with p leaves in which the p nodes on a level are generated "as fast as possible". Such a tree has height $\log p$ and we perform $O(p)$ tests. Once a level contains p nodes, testing continues along p paths leading to p leaves. To follow a single path down, we need to do $O(\log n - \log p)) = O(\log(n/p))$ tests (more precisely, we need two tests at every level). In total, we need $O(p + p * \log(n/p)) = O(p * \log(n/p))$ tests. (Note that for $p = 0$ we need one test and this is covered in the big-O notation used.)

ii) Your solution for (i) only needs to determine the number of water samples containing pesticide X. Can your algorithm also be used to identify the samples with pesticide X? If yes, state how. If no, describe a solution which can.
The algorithm described above will also show which samples are contaminated, since it is going to trace a path down to each contaminated samples, and, thus, find them.