Institut for Matematik og Datalogi
Syddansk Universitet

November 3, 2010
JFB

Introduction to Computer Science
E10 – Cubic Root

The following is an algorithm for computing an integer cubic root, m, of N, such that $m^3 \leq N < (m+1)^3$. This uses binary search, but one is searching a range of integers (initially from 1 to N), rather than an interval in an array (when searching an interval of an array, one is also searching a range of integers, the indices for that interval of the array).

As long as the input is a positive integer, N, the algorithm cannot fail, so there is no check for the size of the interval being less than one.

```
Cubic(N)
{
if (N <= 0)
    then return(Error)
    else SearchCubic(1,N,N)
}
function SearchCubic(low,high,N)
{
 m <- (low + high)/2     # integer division, so it rounds down
 mc <- m*m*m
 m1c <- (m+1)*(m+1)*(m+1)
 if (mc <= N < m1c)
    then return(m)
    else if (mc > N)
            then SearchCubic(low,m-1,N)
            else SearchCubic(m+1,high,N)
}
```

With very large numbers, such as those used for cryptographic purposes, we cannot assume that a number fits into one register, so the multiplications do not, in general take constant time, using most standard algorithms. Using

the most standard algorithms, generalizing the multiplication base 10 which we learned as school children to base 2, multiplying a number with s bits by one with t bits takes $\Theta(s \cdot t)$ time. (If s and t are constants, this is constant, but otherwise it is not.) Suppose our number N has b bits, then m has $r \leq b$ bits. Computing $m * m$ takes $\Theta(r^2)$ time and the result has $2r$ or $2r - 1$ bits. Thus, computing $m * m * m$ takes time $\Theta(2r * r)$ time after $m * m$ is computed and $\Theta(r^2)$ time in all (since constants are absorbed into the Θ). Similarly, computing $(m + 1)^3$ will take $\Theta(r^2)$ time.

As with the binary search algorithm in the textbook, each time the function is called, it is searching in a range which has at most half of the previous size, so the number of calls is $\Theta(\log(N))$, which is $\Theta(b)$, since b is $\Theta(\log(N))$. Since two numbers are cubed in each call to the function (and we are using this as the fundamental operation), the total running time is $\Theta(b^3)$ or $\Theta(\log^3(N))$.

Note that if an array is created with the integers from 1 to N, where N has b bits, just creating the array takes time $\Theta(b \cdot 2^b)$ since there are that many bits to write. Thus, such an array should not be created.

In general, we express the running time of an algorithm as a function of the size of the input. In this case, the input is N, which has $b = \Theta(\log(N))$ bits, so we should express the running time as a function of b.