

CSE 101 Spring 2002, Homework 1 Solutions

CSE 101

April 19, 2002

Problem 1

Consider the case where we have two polynomials $ax + b$ and $cx + d$. The product of these two polynomials is $(ac)x^2 + (ad + bc)x + bd$. So the multiplication problem is reduced to that of finding the coefficients of x^2 , x^1 , and x^0 ; these are ac , $(ad + bc)$, and bd respectively. Let the multiplications be as follows:

$$m_1 = (a + b)(c + d) = ac + ad + bc + bd$$

$$m_2 = ac$$

$$m_3 = bd$$

m_2 and m_3 are two of the coefficients that we are looking for. The final coefficient, that of x , is achieved by subtracting m_1 and m_2 from m_3 . As the example shows we have reduced the number of required multiplications from 4 to 3. We can generalize this as follows.

Let $P(n, x)$ be a polynomial of x with degree n . Write $P(n, x)$ as $P(n, x) = a_n x^n + \dots + a_0$. Then, $P(n, x)$ can be written as the sum of two polynomials: $P(n, x) = x(a_{2k-1}x^{2k-2} + \dots + a_3x^2 + a_1) + (a_{2k}x^{2k} + \dots + a_2x^2 + a_0)$ where we assume that $n = 2k$ (i.e. is even, if n is odd, the last coefficient of the highest degree be zero). Now we write $P(n, x)$ as $P(n, x) = xP_{\text{odd}}(n, x) + P_{\text{even}}(n, x)$, where P_{odd} and P_{even} are polynomials of y where $y = x^2$. Now each of P_{odd} and P_{even} only contain half of the terms of $P(n, x)$.

For the algorithm assume that each polynomial has degree n , if no pad it with zeroes as the coefficients. Divide each polynomial up as described above, and recursively multiply the smaller polynomials in the manner described by part a. This gives us a running time of $T(n) = 3T(n/2) + n$, which by the Master method is $\Theta(n \lg^3)$.

Problem 2

The problem can be solved recursively as follows. First we consider the base case, $n = 1$, where we have to move a single disk from A to B . Since direct moves are disallowed this requires 2 moves, and hence $T(1) = 2$.

We define the task of moving n disks from peg A to peg B recursively as follows.

Move the top $n - 1$ disks from A to B	\longrightarrow	$T(n - 1)$
Move the largest disk from A to the middle	\longrightarrow	1
Move the $n - 1$ disks from B to A	\longrightarrow	$T(n - 1)$
Move the largest disk from the middle to B	\longrightarrow	1
Move the $n - 1$ disks from A to B	\longrightarrow	$T(n - 1)$

After performing these moves all the n disks will be in order on peg B . Thus we can see that the total time required to transfer the n disks is $T(n) = 3T(n - 1) + 2$. Given the previously derived base case the solution to this recurrence is $T(n) = 3^n - 1$.

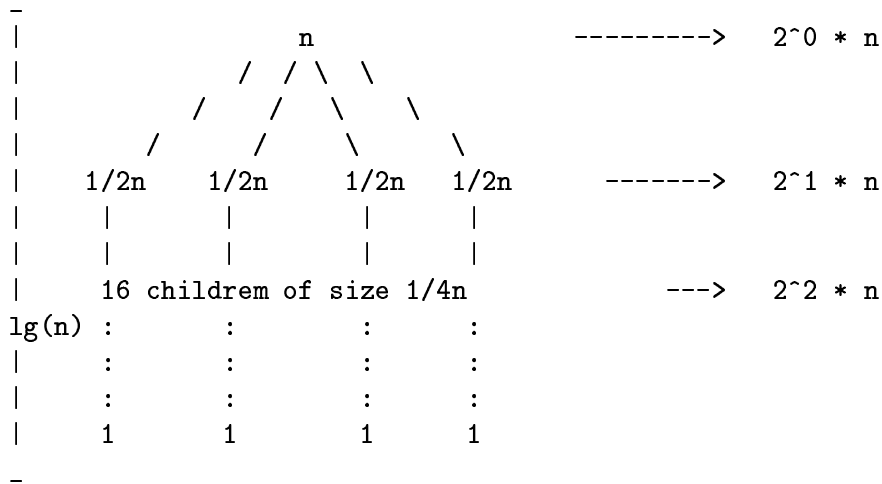
Problem 3

Consider the most general case, where planes inserted are not parallel, and no set of more than 2 planes intersect in the same line. For the n -th plane inserted, all the $n - 1$ previously inserted planes will intersect the n -th plane and create $n - 1$ cuts on that plane. As it was shown in class these $n - 1$ cuts will divide the n -th plane into at most $1 + \frac{n(n-1)}{2}$ regions. The most number of new pieces that can be introduced by the n -th cut is exactly this number.

Thus the recurrence relation that describes the maximum number of pieces attainable using n cuts is $T(n) = T(n - 1) + 1 + \frac{n(n-1)}{2}$. Given that the base case is $T(0) = 1$. The solution to this recurrence is $T(n) = \frac{(n-1)n(n+1)}{6} + n + 1$, which can be verified by substituting this solution back into the recurrence.

Problem 4

The following picture illustrates the recursion tree for the given recurrence relation.



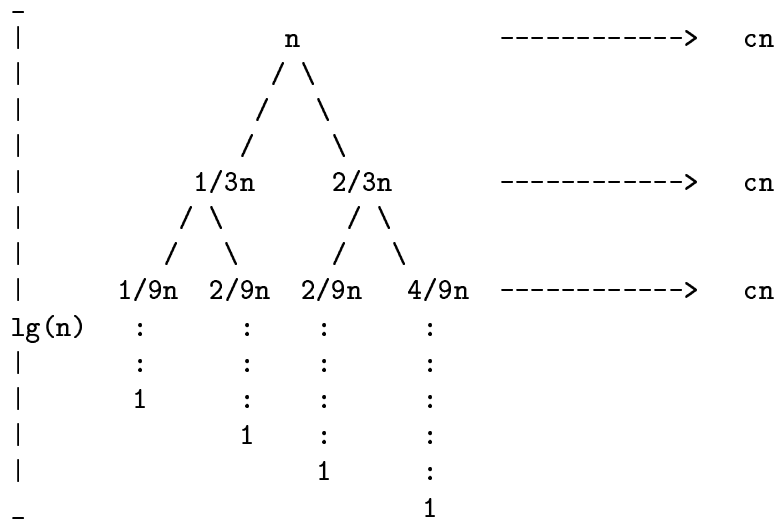
From the recursion tree one can see that the tree has $\lg n$ levels, and that at the i -th level the total work performed is $2^i n$. Thus, $T(n) = \sum_{i=0}^{\lg n} 2^i n$.

To obtain a tight bound one can either evaluate this sum or apply the Master method. Since $a = 4$, $b = 2$, and $f(n) = n$, by setting $\varepsilon = 1$ we can see that case 1 is applicable, and hence $T(n) = \Theta(n^{\log_2 4}) = \Theta(n^2)$.

Problem 5

Drawing the recursion tree (shown on the next page) we see that the cost at every level is $\leq cn$, so the total cost is that multiplied by the height of the tree. At worst this height is $\log_{3/2} n$, for which we have the identity: $\log_{3/2} n = \frac{\lg n}{\lg 3/2} = d \lg n$, where $d = \lg 3/2$. This gives us an upper bound on the runtime of $(cd)n \log n = O(n \lg n)$.

To obtain the lower bound we consider the shortest path in the recursion tree from the root to a leaf. The depth of this shortest path is $\log_3 n$. Since all the levels up to this leaf are full and have a cost of cn , using a similar base conversion as above, we obtain a lower bound on the runtime of $\Omega(n \log n)$ as well.



For a better illustration see page 151 of the book.