Solutions for Homework 2

Maxwell Young

September 23, 2003

Question 1

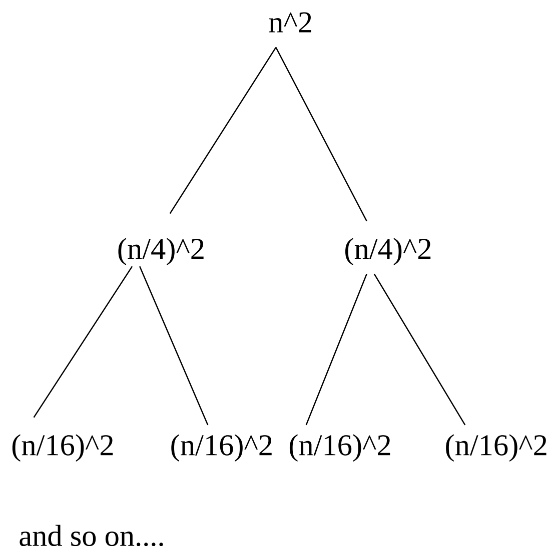

Figure 1: Recursion Tree for Question 1

(A) By looking at the recursion tree in Figure 1, we see that we need to look at the following series:

$$\Sigma_{i=0}^{lgn} n^2/8^i$$

Using the version of summing the geometric series where $|x| < 1$, we get that $T(n) = (8/7)n^2$ and so $T(n) = O(n^2)$.

(B) Let $n = 4^k$. Then we can write out the recurrence as $T(4^k) = 2T(4^{k-1}) + 4^{2k}$. We solve the homogenous component of the recurrence first; $T(4^k) - 2T(4^{k-1}) = 0$. The annihilator for this is (L-2). Now we deal with the inhomogenous part of the recurrence; the 4^{2k} component. The annihilator for this is (L-16). Putting these two annihilators together we see that we have:

$c_0 2^k + c_1 16^k$ but $n = 4^k$ so subtituting back in we get $c_0\sqrt{n} + c_1 n^2$. Hence $T(n) = O(n^2)$.

Question 2

(A) In this case, $a = 2$, $b = 2$, and $f(n) = (lgn)^2$. We can see that $(logn)^2 = O(n^{1-\epsilon})$ so by the Master Theorem, $T(n) = \Theta(n)$.

(B) We deal with the homogenous part first: $T(n) = 2T(n/2)$. Letting $n = 2^k$, we get that $T(2^k) = 2T(2^{k-1})$ which has the annihilator of (L-2). The inhomogenous part of the recurrence becomes k^2 and it has an annihilator of $(L-1)^3$. Therefore, we end up with $c_0 2^k + (c_1 + c_2 k + c_3 k^2)1^k$. And the largest term here, when we substitute back in with n, is n and so this recurrence is in $O(n)$.

Question 3

(A) The recurrence for the value of this code is:

$$f(n) = 6f(n-1) - 9f(n-2)$$

When we work out the annihilator for this recurrence we get $(L-3)^2$. We then end up with $(c_0 + c_1 n)3^n$. Using the base cases, we find that $c_0 = 0$ and that $c_1 = 1/3$. Hence, the recursion has solution $(n/3)3^n$.

(B) The recurrence for the running time of this code is $f(n) = f(n-1) + f(n-2) + 1$. We've seen this recurrence in class (for the runtime of the Fibonnaci numbers) and so we know that the complexity is $f(n) = c_1\phi^n + c_2\hat{\phi}^n + c_3 1^n$ where $\phi = ((1+\sqrt{5})/2)$ and $\hat{\phi} = (1-\sqrt{5})/2$. The final result is $T(n) = (1 + 1/\sqrt{5})\phi^n + (1 - 1/\sqrt{5})\hat{\phi}^n - 1$.

Question 4

Figure 2 gives the table that you should have constructed by following the DP algorithm. The number of different string alignments that you should have achieved is 10.

Question 5

Figure 3 gives the two tables that you should have constructed if you correctly followed the DP algorithm for matrix multiplication.

The optimal paranthesization is $((A_1 A_2)(A_3 A_4))$ and it has a cost of 11.

Question 6

Here we go with the induction:

Base Case: $P(1) = 1 \geq c(2^1)$ if $c \leq 1/2$.
Induction Hypothesis: assume that $P(n) \geq c2^n$.
Induction Step: $P(n+1) = \Sigma_{k=1}^{n} P(k)P(n+1-k)$

$= P(1)P(n) + \Sigma_{k=2}^{n-1} P(k)P(n+1-k) + P(n)P(1)$
$= 2(cP(1)P(n)) + \Sigma_{k=2}^{n-1} P(k)P(n+1-k)$
$\geq c2^{n+1} + \Sigma_{k=2}^{n-1} P(k)P(n+1-k)$
$\geq c2^{n+1}$ which is the desired result in the correct form.

Question 7

We can do this by using induction.

Base Case: For $n = 1$, A_1 requires n-1=0 parantheses, so it works.

IH: Given a sequence of n numbers (or matrices, as I will deal with here), there are exactly n-1 pairs of parentheses needed to completely parenthisize this sequence.

IS: Consider the sequence (of matrices) of length $n+1$; $A_1A_2A_3...A_nA_{n+1}$. We automatically have to place one pair of parantheses starting at A_1 and ending at A_{n+1}. Now, without loss of generality, we can consider that we start our next parentheses pair at A_1 and finish at A_i where $i \leq n$. So we have $((A_1A_2A_3...A_i)(A_{i+1}...A_n))$. Now consider $(A_1A_2A_3...A_i)$; this sequence is shorter than n and so the IH holds ie. we require $i-1$ pairs of parantheses to fully paranthesize it. In fact, if we are keeping a tally, this will require i-1-1 pairs of parantheses since we already added a pair starting at A_1 and finishing at A_i. The remaining matrices, $(A_{i+1}...A_n)$ require $(n+1)-i-1-1$ pairs of parentheses to fully paranthesize them (taking into account the one we have already put around them). Hence, we have already included 3 pairs of parantheses and the remaining groups require $i-2+n-i+1-2$ so the sequence of $n+1$ matrices requires $i-2+n-i+1-2+3 = n$ as desired.

Question 8

Figure 4 gives the table you should have constructed by following the DP algorithm for LCS. The longest possible subsequence is of length 6 and, if we trace back along the arrows, we can find one such string: 101011.

Question 9

You are given a sequence of n numbers, call this sequence A. Then copy A and sort it in increasing order to get B; this takes $O(nlgn)$ time. Then use the DP LCS algorithm on A and B; this will give back the longest monotonically increasing subsequence of the original sequence A in $O(n^2)$ time.

Bonus Questions

Please come and talk to me during office hours.

	null	a	b	a	b	b	a	b
null	0	1	2	3	4	5	6	7
a	1	0	1	2	3	4	5	6
b	2	1	0	1	2	3	4	5
b	3	2	1	1	1	2	3	4
a	4	3	2	1	2	2	2	3
b	5	4	3	2	1	2	3	2
a	6	5	4	3	2	2	2	3

```
a b a b b a b          a b a b b a b _
a b _ _ b a _          a b _ b _ a b a

a _ b a b b a b        a b a b b a b _
a b b a b _ a _        a b _ _ b a b a

a b _ a b b a b        a b a b b a b _
a b b a b _ a _        a _ _ b b a b a

a b a b _ b a b        a _ b a b b a b
a b _ b a b a _        a b b a _ b a _

a b a b b a b _        a b _ a b b a b
a b   b   a b a        a b b a   b a
```

Figure 2: DP Table for Question 4

	1	2	3	4
1	0	6	12	11
2	—	0	9	6
3	—	—	0	3
4	—	—	—	0

costs

	1	2	3	4
1	0	1	2	2
2	—	0	2	2
3	—	—	0	3
4	—	—	—	0

k−values

Figure 3: DP Tables for Question 5

	null	0	1	0	1	1	0	1	1	0
null	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1
0	0	1	1	2	2	2	2	2	2	2
0	0	1	1	2	2	2	3	3	3	3
1	0	1	2	2	3	3	3	4	4	4
0	0	1	2	3	3	3	4	4	4	5
1	0	1	2	3	4	4	4	5	5	5
0	0	1	2	3	4	4	5	5	5	6
1	0	1	2	3	4	5	5	6	6	6

Figure 4: DP Table for Question 8