

**Algorithms and Data Structures 2011/12**  
**Week 7 SOLUTION SHEET (Tues 1st Nov - Fri 4th Nov)**

1. Some tutorial groups will have covered this Matrix-chain multiplication problem in week 6, some not. The solution is given in `sols6.pdf`, not duplicating here.
2. *CLAIM:* The number of possible parenthesizations of a Matrix-chain sequence containing  $n$  matrices is  $\Omega(3^n)$ .

*This proof/justification is HORRIBLE.* In class we did a proof that the recursive algorithm for Matrix-Chain is  $\Omega(2^n)$  on the board - that one is nicer ...

I will be putting up a scan of handwritten "board-notes" of the  $\Omega(2^n)$  but cannot guarantee it will be before tomorrow (Tues of week 7). Will let you know if it is.

**Anyway:** For this one, I only want you to do the recurrence (which for the parenthesization-count has a product in it), and the Induction step. Tell them how high the base cases must go (and you can mention the weird nature of this recurrence, where taking a smaller  $c$  doesn't help at all in getting the proof - just due to that product ...)

**Discussion:** We can prove this claim by induction; however before proceeding it will make sense to explore which constants should be used. let  $P(n)$  denote the number of possible parenthesisations for a sequence of  $n$  matrices.

We can write a recurrence for  $P(n)$  as follows:

$$P(n) = \begin{cases} 1 & n \leq 2 \\ \sum_{i=1}^{n-1} P(i) \times P(n-i) & n \geq 3 \end{cases} \quad (1)$$

The justification is that for 1 or two matrices there is only one possible parenthesisation. For  $n > 2$  matrices, there are  $n - 1$  places where we can choose the top-level parenthesisation (ie, after the  $i$ th matrix), then we have  $P(i)$  options for the left-hand-side and  $P(n - i)$  options for right-hand-side.

Using (1), we know that  $P(1) = P(2) = 1$ ,  $P(3) = 2$ ,  $P(4) = 5$ . Therefore, to prove  $P(n) \geq c \cdot 3^n$ , we will have to use a smallish value of  $c$  - eg,  $\frac{1}{3^3}$  would work for  $1, \dots, 4$ .

Now imagine we had knew  $P(k) \geq 3^{k-3}$  for all  $k < n$ . We now make a first attempt at induction, to try to deduce  $P(n) \geq 3^{n-3}$ . Using (1), we find

$$\begin{aligned} P(n) &= \sum_{i=1}^{n-1} P(i) \times P(n-i) \\ &\geq \sum_{i=1}^{n-1} 3^{i-3} \times 3^{n-i-3} \\ &= \sum_{i=1}^{n-1} 3^{n-6} = \sum_{i=1}^{n-1} \frac{1}{3^3} 3^{n-3} = \frac{(n-1)}{27} 3^{n-3}. \end{aligned}$$

So in fact this approach will work perfectly, *if we have*  $n - 1 \geq 27$ . However, this would necessitate checking  $P(k) \geq 3^{k-3}$  for each of the cases  $k = 5, \dots, 27$ , which is a huge number.

Looking again, we could have substituted  $3^{n-2.5}$  for most of  $n = 1, 2, 3, 4$ . The only term which does not satisfy this was  $P(4)$ :  $P(4) = 5$ , but  $3^{4-2.5} = 3^{1.5} \sim 5.1$ . However,  $3^{n-2.6}$  would suffice for  $n = 4$ . We can use  $c = 3^{-2.6}$ , and prove that this works.

*This induction is very messy and the main part of the work is being done on the base cases, which is NOT typical. The only reason that I have created such a fiddly constant as  $3^{-2.6}$  is to reduce the number of base cases as far as possible ...*

*I WOULD NOT GIVE YOU SOMETHING LIKE THIS IN AN EXAM - at least if I did, I would tell you to BELIEVE the base cases  $n = 1, \dots, 27$  and do the induction step (for one thing, you won't have any calculator, or Google, in an exam).*

**Theorem:**  $P(n) \geq 3^{n-2.6}$  for all  $n \geq 1$ .

*Proof:* Our proof is by induction.

Looking back at the expansion of  $P(n)$  in the  $3^{n-3}$  case above, we remark that if we work with  $3^{n-2.6}$ , we will have  $\frac{1}{3^{2.6}} = \frac{1}{17.39\dots}$  in place of the  $1/27$ .

Therefore for  $c = 3^{-2.6}$ , we have to check base cases  $k = 1, \dots, 18$ .

*Base cases:* We have already discussed  $k = 1, \dots, 4$ . Here we check the details for  $k = 5, \dots$

We have the following:

$$P(5) = 2P(1) \cdot P(4) + 2P(2) \cdot P(3) = 2(1 \cdot 5 + 1 \cdot 2) = \mathbf{14}.$$

$$\text{Greater than } 3^{5-2.6} = 3^{2.4} \sim 13.9$$

$$P(6) = 2P(1) \cdot P(5) + 2P(2) \cdot P(4) + P(3)^2 = 2(1 \cdot 14 + 2 \cdot 5) + 2^2 = \mathbf{54}.$$

$$\text{Greater than } 3^{6-2.6} = 3^{3.4} \sim 41.9$$

$$P(7) = 2(P(1) \cdot P(6) + P(2) \cdot P(5) + P(3) \cdot P(4)) = 2(1 \cdot 54 + 1 \cdot 14 + 2 \cdot 5) = \mathbf{156}.$$

$$\text{Greater than } 3^{7-2.6} = 3^{4.4} \sim 125.7$$

$$P(8) = 2(P(1) \cdot P(7) + P(2) \cdot P(6) + P(3) \cdot P(5)) + P(4)^2 = 2(156 + 54 + 2 \cdot 14) + 5^2 = \mathbf{421}.$$

$$\text{Greater than } 3^{8-2.6} = 3^{5.4} \sim 377.0$$

$$P(9) = 2(P(1) \cdot P(8) + P(2) \cdot P(7) + P(3) \cdot P(6) + P(4) \cdot P(5))$$

$$= 2(421 + 156 + 2 \cdot 54 + 5 \cdot 14) = \mathbf{1510}.$$

$$\text{Greater than } 3^{9-2.6} = 3^{6.4} \sim 1131.3$$

$$P(10) = 2(P(1) \cdot P(9) + P(2) \cdot P(8) + P(3) \cdot P(7) + P(4) \cdot P(6)) + P(5)^2$$

$$= 2(1510 + 421 + 2 \cdot 156 + 5 \cdot 54) + 14^2 = \mathbf{5222}.$$

$$\text{Greater than } 3^{10-2.6} = 3^{7.4} \sim 3393.9$$

$$\begin{aligned}
P(11) &= 2(P(1) \cdot P(10) + P(2) \cdot P(9) + P(3) \cdot P(8) + P(4) \cdot P(7) + P(5) \cdot P(6)) \\
&= 2(5222 + 1510 + 2 \cdot 421 + 5 \cdot 156 + 14 \cdot 54) = \mathbf{18220} \\
&\quad \text{Greater than } 3^{11-2.6} = 3^{8.4} \sim 10181.6 \\
P(12) &= 2(P(1) \cdot P(11) + P(2) \cdot P(10) + P(3) \cdot P(9) + P(4) \cdot P(8) + P(5) \cdot P(7)) + P(6)^2 \\
&= 2(18220 + 5222 + 2 \cdot 1510 + 5 \cdot 421 + 14 \cdot 156) + 54^2 = \mathbf{64418} \\
&\quad \text{Greater than } 3^{12-2.6} = 3^{9.4} \sim 30544.9 \\
P(13) &= 2(P(1) \cdot P(12) + P(2) \cdot P(11) + P(3) \cdot P(10) + P(4) \cdot P(9) + P(5) \cdot P(8) + P(6) \cdot P(7)) \\
&= 2(64418 + 18220 + 2 \cdot 5222 + 5 \cdot 1510 + 14 \cdot 421 + 54 \cdot 156) = \mathbf{229900} \\
&\quad \text{Greater than } 3^{13-2.6} = 3^{10.4} \sim 91634.9 \\
P(14) &\quad \dots
\end{aligned}$$

I KNOW IT IS BAD NOT TO FINISH - but it's such a pain!

I told the students only to consider Ind Step (and the reason why)

**Induction step:** Now suppose that we have some value  $n > 18$ .

Our induction hypothesis (I.H.) is that  $P(k) \geq 3^{k-2.6}$  for every  $k < n$ .

We will show that assuming the I.H., we also get the result for  $P(n)$ . We use (1) and work with our I.H. (since each  $i$  satisfies  $i < n$ ):

$$\begin{aligned}
P(n) &= \sum_{i=1}^{n-1} P(i) \times P(n-i) \\
&\geq \sum_{i=1}^{n-1} 3^{i-2.6} \times 3^{n-i-2.6} \\
&= \sum_{i=1}^{n-1} \frac{1}{3^{2.6}} 3^{n-2.6} \\
&> \frac{18}{3^{2.6}} 3^{n-2.6} \sim \frac{18}{17.4} 3^{n-2.6} > 3^{n-2.6},
\end{aligned}$$

as required. Note the final step is due to having  $n > 18$ . QED.

### 3. Longest Common Subsequence

Really want this Q to be done collaboratively as a group.

Really want you to flag the “main steps” which are common wrt DP.

For a more detailed solution see [CLRS]. For this (and all dynamic programming) questions, there are a few main issues. (I write  $\text{lcs}$  to denote the Length of the lcs, rather than the actual *sequence* itself (which may anyhow change).)

- What is the generalization we look at?

For  $\text{lcs}$ , we will generalize to the problem of finding  $\text{lcs}(x[1 \dots k], y[1 \dots \ell])$ , for all  $0 \leq k \leq n$ , all  $0 \leq \ell \leq m$ .

You might want to mention that there’s no justification for this, YET (we could have considered generalising to computing  $\text{lcs}(x[k' \dots k], y[\ell' \dots \ell])$  for all  $k', k, \ell', \ell$ , fortunately we’ll see that’s not necessary).

- What size table do we need to store our solutions?

We will need a table of size  $(n + 1)(m + 1)$  (to store  $\text{lcs}$  for every  $k, \ell$ ).

- We need a recurrence to justify our choice of generalization (ie, why would it be possible to use ‘small’ solutions to build bigger ones). The recurrence is

$$\begin{aligned} & \text{lcs}(x[1 \dots k], y[1 \dots \ell]) \\ &= \begin{cases} 1 + \text{lcs}(x[1 \dots k - 1], y[1 \dots \ell - 1]) & \text{if } x_k = y_\ell \\ \max\{\text{lcs}(x[1 \dots k - 1], y[1 \dots \ell]), \text{lcs}(x[1 \dots k], y[1 \dots \ell - 1])\} & \text{otherwise} \end{cases} \end{aligned}$$

Where does this recurrence come from? First notice that we *realise* (make concrete) the presence of a common subsequence of two sequences using an alignment with –s inserted, where we only align characters *if they are the same*. For example, if we were given the two sequences  $x = \text{‘alignment’}$  and  $y = \text{‘tantalise’}$ , then a description of the common subsequence ‘alie’ is

```

- - - - a l i - g n m e n t
t a n t a l i s - - - e - -

```

With the above picture in mind, notice that any *common subsequence* of  $x[1 \dots k]$  and  $y[1 \dots \ell]$  has three possible options for the final column:

- It might be that  $x[k]$  and  $y[\ell]$  both appear in this column. This is only possible if  $x[k] = y[\ell]$ .
- It might be that  $x[k]$  appears in this column, aligned with a ‘-’ (in this scenario  $y[\ell]$  might \*possibly\*, but not necessarily, be matched against a character from  $x[1 \dots k - 1]$ )
- Alternatively, the third option is that  $y[\ell]$  appears in this column, aligned with a ‘-’ (in this scenario  $x[k]$  might \*possibly\*, but not necessarily, be matched against a character from  $y[1 \dots \ell - 1]$ )

Now in solving  $\text{lcs}(x[1 \dots k], y[1 \dots \ell])$ , we are looking for the alignment with the highest value. In doing this we can partition all possible alignments according to the three options above - we take the maximum of

- the value of the best alignment of  $x[1 \dots k-1]$  and  $y[1 \dots \ell-1]$ , with 1 added (this option is only available if  $x[k] = y[\ell]$ )
- the value of the best alignment of  $x[1 \dots k-1]$  and  $y[1 \dots \ell]$
- the value of the best alignment of  $x[1 \dots k]$  and  $y[1 \dots \ell-1]$

These three “best alignments” of the (slightly) shorter prefixes are written as  $\text{lcs}(x[1 \dots k-1], y[1 \dots \ell-1])$ ,  $\text{lcs}(x[1 \dots k-1], y[1 \dots \ell])$  and  $\text{lcs}(x[1 \dots k], y[1 \dots \ell-1])$ .

- What are the rules for filling-in the table?

This is more delicate than for the case of matrix-chain multiplication - the point being that in the recurrence above two of the smaller  $\text{lcs}$  problems are only smaller on one sequence.

For  $k = 0$  we fill the values of this row directly, setting  $\text{lcs}(x[1 \dots 0], y[1 \dots \ell]) = 0$  for every  $0 \leq \ell \leq m$ . We also fill in column 0 the same way.

Then for  $k \leftarrow 1$  to  $n$  (in increasing order) we fill in row  $k$  in one go as follows: we generate the values for  $\text{lcs}(x[1 \dots k], y[1 \dots \ell])$  in terms of increasing  $\ell$ , using the recurrence above.

This method of doing the rows in increasing order of  $k$ , and within each row, in increasing order of  $\ell$ , ensures that the  $\text{lcs}$  values from the right-hand side of the recurrence above are ALWAYS available in advance.

- Running time?  
 $\Theta(nm)$ .

To actually recover the \*sequence\* rather than just the value of the sequence, we need a second table (of size  $(n+1)(m+1)$ ) where we store a flag to say which of the three options was the optimum for  $\text{lcs}(x[1 \dots k], y[1 \dots \ell])$ , for every  $k, \ell$ . This is updated dynamically in partnership with the main table. Then after we are finished, we can ‘trace-back’ on this second table to recover the sequence.

4. Draw an example of a weighted graph which has 2 MSTs.

There are lots of these. For example, take a triangle whose edges have values 1, 2, 2. Then there are 2 MSTs, obtained by dropping either of the “2”-edges.

5. Let  $G, W$  be a weighted graph in which all edge weights are distinct.

Prove that the MST of  $G, W$  is unique.

**proof:** By contradiction.

Suppose there are two different MSTs,  $T_1$  and  $T_2$ . Choose  $(u, v) \in T_1 \setminus T_2$ . Let  $T_1(u), T_1(v)$  be the two sub-MSTs obtained by deleting  $(u, v)$ . Consider the path  $p_{u,v}(T_2)$

between  $u$  and  $v$  in  $T_2$ . Let  $E'$  be the edges of  $p_{u,v}(T_2)$  that cross the cut  $T_1(u), T_1(v)$ . We are guaranteed that  $|E'| \geq 1$ .

Two cases:

(1) Suppose  $\mathcal{W}(e) > \mathcal{W}(u, v)$  for some  $e \in E'$ . Then define  $T_2' = T_2 \setminus \{e\} \cup \{(u, v)\}$ . This is a spanning tree of cost strictly less than  $T_2$ . Contradiction!

(2) Suppose  $\exists e \in E'$  with  $\mathcal{W}(e) < \mathcal{W}(u, v)$ . Then define  $T_1' = (T_1 \setminus \{(u, v)\}) \cup \{e\}$ . This is a spanning tree of  $G$  with cost strictly less than  $T_1$ . Contradiction!

In either case we prove that one of  $T_1, T_2$  was not a MST.

Mary Cryan