

# TCSS 343 - Week 3 - Monday

Jake McKenzie

January 28, 2019

## Stating Problems Formally and Solving Recurrences

“The object of life is not to be on the side of the majority, but to escape finding oneself in the ranks of the insane”.

...

Marcus Aurelius

“Treat everyone kindly and light up the night.”

...

Peter Dinklage

“Computer science got us into this mess. Can computer science get us out of it?”

...

Latanya Sweeney re privacy breaches

For this problem consider the problem of finding the product of a list of integers.

Product of all integers in a list (PRODUCT)

Input: A set of integers  $A[a..b]$

Output:  $p = \prod_{i=a}^b A[i]$ .

Let  $P(A[a..b])$  represent the output of the PRODUCT problem on input  $A[a..b]$

0. State two different self-reductions for the PRODUCT problem. Use the self-reduction examples from lecture as a guide.

$$\text{Let } m = \lfloor \frac{a+b}{2} \rfloor$$

$$P(A[a..b]) = \begin{cases} A[a] & ; \text{ if } a=b. \\ A[a] \cdot P(A[a+1..b]); & ; \text{ if } a < b \end{cases}$$

$$P(A[a..b]) = \begin{cases} 1 & ; \text{ if } a > b \\ A[a] & ; \text{ if } a = b \\ A[m] \cdot P(A[a+1..m]) \cdot P(A[m+1..b]); & ; \text{ otherwise} \end{cases}$$

1. Give recursive algorithms based on your divide and conquer self-reduction to solve the PRODUCT problem.
2. What are the worst case runtimes of the solutions you have generated.  
(Just state the runtimes. You do not need to show your work.)

```

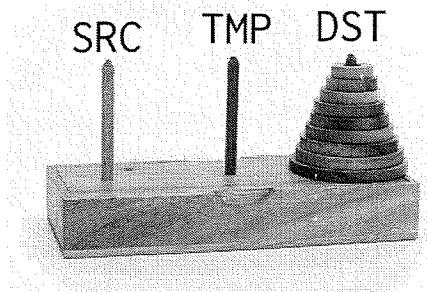
Product (A[a..b])
    if a=b return A[b]
    Let Z = Product (A[a+1... b])
    Return A[a] * Z
End Product

```

```

Product (A[a..b])
    if a>b return 1
    if a=b return A[a]
    Let mid = floor((a+b)/2)
    Let y = Product (A[a+1... mid])
    Let z = Product (A[mid+1... b])
    return A[a] * z * y;
End product

```



The **Tower of Hanoi** has 3 pegs *src*, *tmp* and *dst*. Some disks of different sizes are given which can slide onto any peg. Initially all of those are in *src* peg in order of size with largest disk at the bottom and smallest disk at the top. We have to move all the disks from *src* peg to *dst* peg. At the end, *dst* peg will have disks in the same order of size. there are some rules :

- 1: Only one disk can be moved from one peg to another peg at a time.
  - 2: A disk can be placed only on top of a larger one.
  - 3: A disk can be moved from top only.
3. Write a formal statement of this problem. That is, state the input and output criteria as tidy as possible.

Transferring disks on 3 pegs with largest disk on Bottom (Hanoi)

Input: A sorted set of disks by largest to smallest size  $[a, b]$ ,  
An empty list  $tmp[a, b]$ , and  $n$  the number of disks

Output:  $dst[a, b]$  such that no larger disk was ever placed  
on top of a smaller disk

<u>HANOI(<math>n, src, dst, tmp</math>):</u>			
if $n > 0$			
HANOI( $n-1, src, tmp, dst$ )	⟨⟨Recurse⟩⟩	↙	$T(n-1)$
move disk $n$ from $src$ to $dst$		↘	$O(1)$
HANOI( $n-1, tmp, dst, src$ )	⟨⟨Recurse⟩⟩	↖	$T(n-1)$

4. Note that the algorithm above does nothing when  $n = 0$  and only works for positive integers in  $n$ . The recursive algorithm above solves the Tower of Hanoi problem. Let  $T(n)$  denote the number of moves required to transfer  $n$  disks (the running time of this algorithm). The vacuous base case implies that  $T(0) = 0$  and I want you to write down the more general recurrence for this problem from the algorithm state, then solve for the running time of that algorithm.

BC:

$$T(0) = 0 \quad \checkmark$$

IH:

$$T(n) = T(n-1) + 1 + T(n-1) = 2T(n-1) + 1$$

$$T(k) = 2T(k-1) + 1 = 2^k - 1$$

IS:

$$T(k+1) = 2T(k) + 1 = 2^{k+1} - 1$$

$$2T(k) + 1 = 2^{k+1} - 1$$

$$2(2^k - 1) + 1 = 2^{k+1} - 1$$

$$2^{k+1} - 1 = 2^{k+1} - 1$$

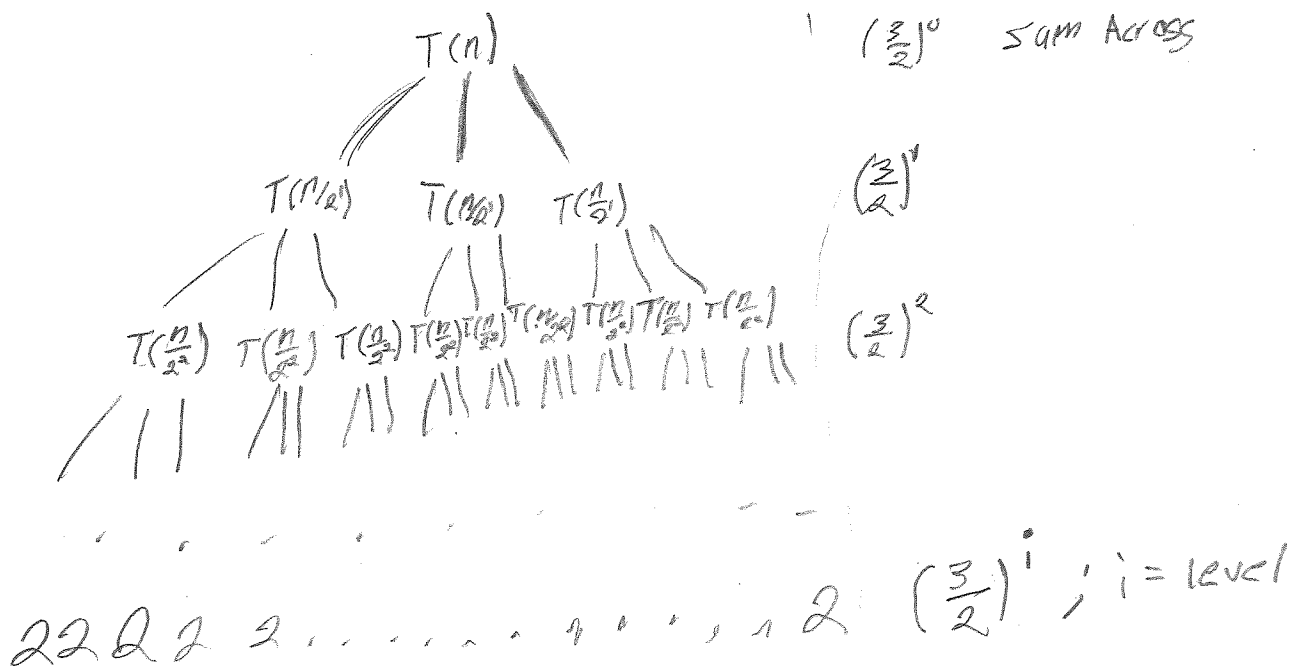
QED

5. Consider the following recurrence:

$$T(n) = \begin{cases} 2, & \text{if } n = 1 \\ 3T(\frac{n}{2}) + n, & n > 1 \end{cases} \quad (1)$$

$$(2)$$

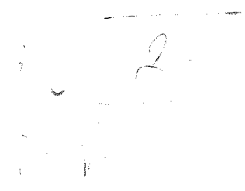
Draw out a visualization of what this recurrence looks like as a tree.



$$\frac{n}{2^i} = 1$$

$$n = 2^i \quad \boxed{i = \log_2 n}$$

For the following problems use the recurrence from problem 5 to answer the following problems.

6. How much work is done on level  $i$ ? 
7. How many recursive levels are there in the tree?
8. How much work is done at the leaf level?
9. Construct a non-recursive expression equivalent to the recurrence. Your solution may use a summation
10. Use the master theorem to find the big- $\Theta$  bound for the recurrence.

6. We do  $(\frac{3}{2})^i n$  work - there are  $3^i$  nodes per level

where  $\frac{n}{2^i}$  each node does  $\frac{n}{2^i}$  work

7. Terminates  $T(\overset{==1}{\sqrt[n]{n}}) \therefore (\frac{n}{2^i}) = 1 \therefore n = 2^i \therefore \boxed{i = \log_2 n}$

8.  $2 \cdot 3^{\log_2 n}$  work at the bottom level

$$\begin{aligned}
 9. \quad \sum_{i=0}^{\log_2 n - 1} n \left(\frac{3}{2}\right)^i + 2 \cdot 3^{\log_2 n} &= n \sum_{i=0}^{\log_2 n - 1} \left(\frac{3}{2}\right)^i + 2 \cdot 3^{\log_2 n} \\
 &= n \left( \frac{\left(\frac{3}{2}\right)^{\log_2 n} - 1}{\frac{3}{2} - 1} \right) + 2 \cdot 3^{\log_2 n} \\
 &= \frac{n}{2} \left( n^{\log_2 \left(\frac{3}{2}\right)} - 1 \right) + 2 \cdot n^{\log_2 3} \\
 &= \frac{n}{2} (n^{\log_2 3} - n^0 - 1) + 2 \cdot n^{\log_2 3} \\
 &\in \Theta(n^{\log_2 3})
 \end{aligned}$$