Recursion and self reduction

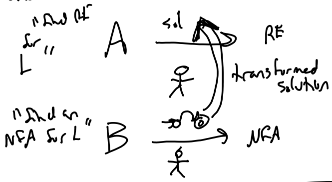
Tuesday, February 25, 2020 3:29 PM

Today:
- recursive algorithms
  self reduction
- analysis of runtime through
  recurrence
- hanoi puzzle
- merge sort

Reduction

NFA → DFA → RE

Problem A reduces to problem B
iff we can take a solution for B
and transform it to a solution for A.

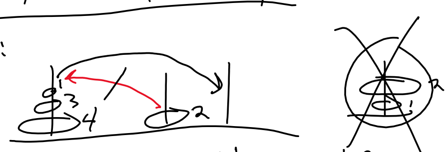


"Self reduction"
reducing to a smaller instance of
the same problem.
- design
- Correctness — by induction + base case
- runtime $T(n) = f(T(n-1))$
  analyze with recurrence

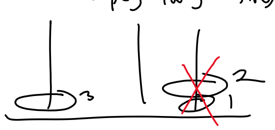
Towers of Hanoi

Goal

Moves:



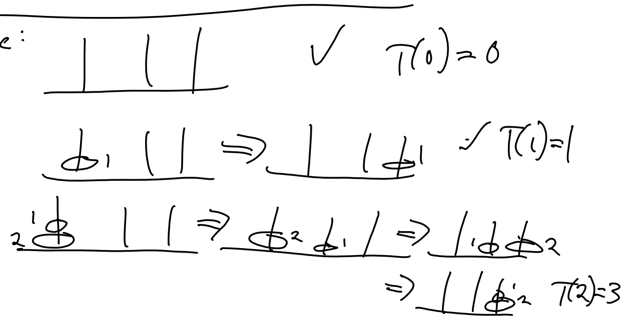
Rule: Can only take a tile from top,
and can only place on empty peg
of a peg larger tiles



Base Case: ✓ $T(0)=0$
n=0

n=1 ✓ $T(1)=1$

n=2 $T(2)=3$



$n=3$

How to generalize:

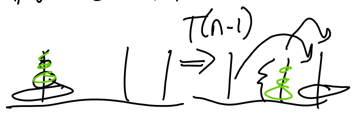
- Assume I have $\underline{H(n')}$ is a solution to moves n' discs to an empty peg.

To prove $H(n)$ has a solution for $n = n'+1$

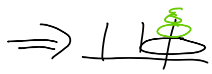
$n-1$

n

- Moving to right peg $\Rightarrow$ Can move to center peg too
- ⊙ Adding a larger tile at bottom doesn't affect moves.



1. Use $H(n-1)$ to move $n-1$ tiles to middle peg
2. Move n tile to right peg.
3. Use $H(n-1)$ to move



✓

$$T(n) = T(n-1) + 1 + T(n-1)$$
$$= 2T(n-1) + 1$$
$$T(0) = 0$$

- Guess and verify by induction

Guess $T(n) = \underline{2^n - 1}$

base: $T(0) = 2^0 - 1 = 0$ ✓

IH $\underline{T(n-1)} = 2^{n-1} - 1$

Prove $T(n) = 2^n - 1$

$$T(n) = 2\overline{T(n-1)} + 1$$
$$= 2(2^{n-1} - 1) + 1$$
$$= (2^n - 2) + 1$$
$$= 2^n - 1 \quad ✓$$

Merge Sort.

| 3 7 8 1 | 2 4 0 9 |

$7+9+2+2 \quad 8$

37 81 24 109

37 4 18 24 4 09 4+4 8

2^3

3 layers

1378 0249 8 8

0123 4789

analysis: - each layer is n comparisons,

- how many layers? $\log_2 n$

$$n(\log_2 n + 1)$$

$$T(n) = \underset{L}{T(n/2)} + \underset{R}{T(n/2)} + n$$

$T(0) = 0$

$T(1) = 0$

$T(2) = \cancel{1}\ 2$

IH $T(n/2) = \frac{n}{2}(\log_2 \frac{n}{2} + 1)$

$$T(n) = 2\left(\frac{n}{2}\left(\log_2 \frac{n}{2} + 1\right)\right) + n$$

$$n \log_2 n/2 + n + n$$

$$n(\log_2 n - \underbrace{\log_2 2}_{1}) + n + n$$

$$n \log_2 n - n + n + n$$

$$n \log_2 n + n \checkmark$$

Asymptotic analysis.

exact. e.g. $T(n) = n \log_2 n + n$

$$T(n) \in O(n \log n)$$

asymptotic.

- for some constant factor slack.
- only has to hold for large n.

$$T(n) = O(f(n))$$

$\exists\, c > 0,\ n_0,\ \forall n \geq n_0 \quad T(n) \leq c \cdot f($

"slack" "large enough"

— worst case — over all inputs of size n

— upper bound

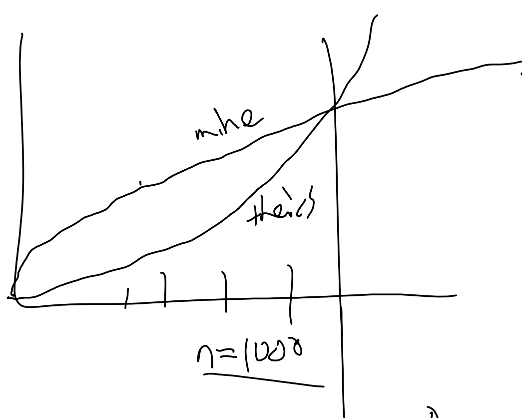
$T(n) = \Theta(f(n))$ "~~more precise~~"

both upper bound and lower bound

Model of computation:

- primitive math operations are $O(1)$ step
- Comparisons 1 step
- array accesses 1 step.

Why do asymp. analysis?



— existing $O(n^2$

— my goal $O(n^2 l$