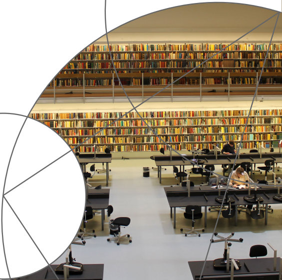
Recurrences II

Jurij Volčič

Institut for Matematiske Fag



Outline

Solving recurrences using

- Recursion-tree method
- Master theorem

Reading: CLRS 4.4. and 4.5

Recap

Goal: solve a given recurrence.

$$f_0 = 1, \quad f_1 = 1 \qquad\qquad T_1 = \Theta(1)$$
$$f_n = f_{n-1} + f_{n-2} \qquad\qquad T_n = T_{[n/2]} + T_{[n/2]} + \Theta(n)$$

Last lecture:

- Backtracking method
- Guess + Induction
- A method for homogeneous linear recurrences

Today: solve recurrences of the form $T_n = aT_{n/b} + f(n)$ by

- Recursion-tree method
- Master method

What does the recursion tree method get us?

- We will be a bit sloppy when using the tree method.
- The recursion tree method yields a **guess**.
- **Verify the guess** e.g. by induction.

Applying the recursion-tree method (example)

Goal: Find a good asymptotic upper bound for $T(n)$, where

$$\begin{cases} T(1), T(2), T(3) \in \Theta(1) \\ T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) & \text{for } n \geqslant 4 \end{cases}$$

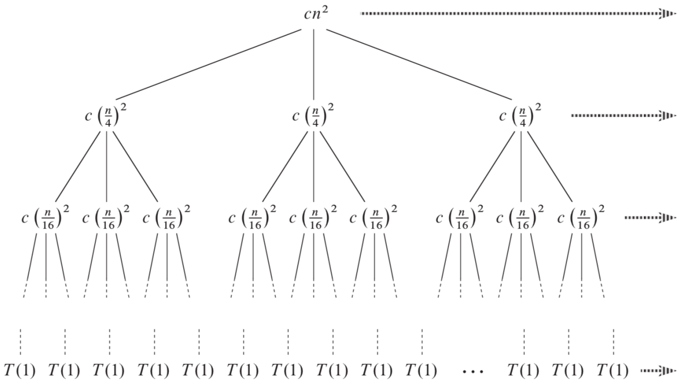
Simplifying assumptions:

- Assume that n is a power of 4. This lets us drop $\lfloor \cdot \rfloor$
- Replace the Θs with c' and cn^2, respectively.

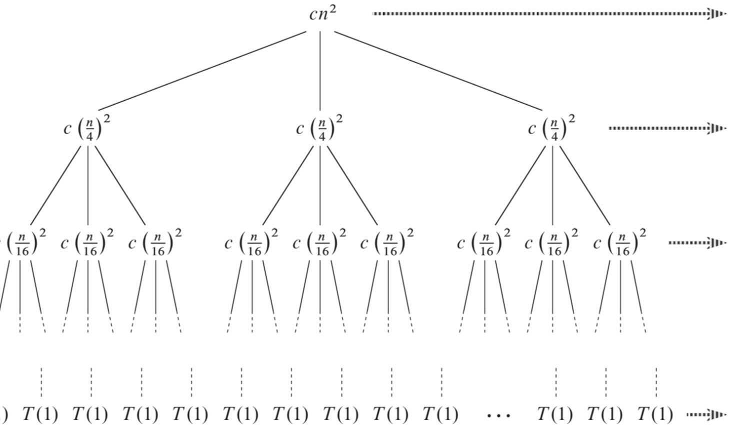
Converting a recurrence into a tree

$$\begin{cases} T(1) = T(2) = T(3) = c' \\ T(n) = 3T(n/4) + cn^2 \qquad \text{for } n \geqslant 2 \end{cases}$$

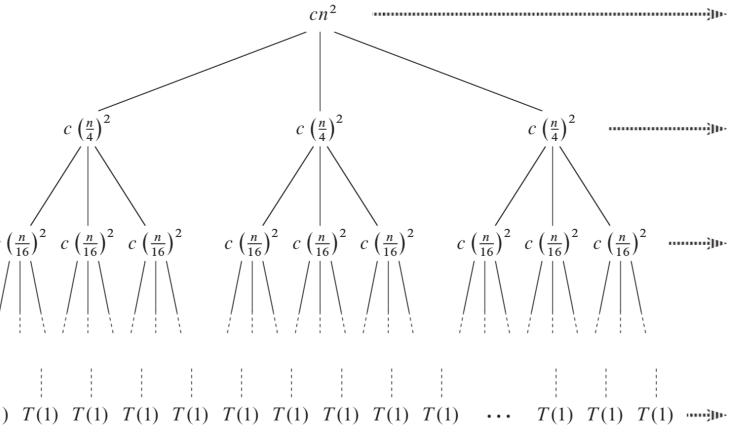
- Each node represents a recursive call (label is the incurred cost)
- Children represent the recursive calls made



Warm-up

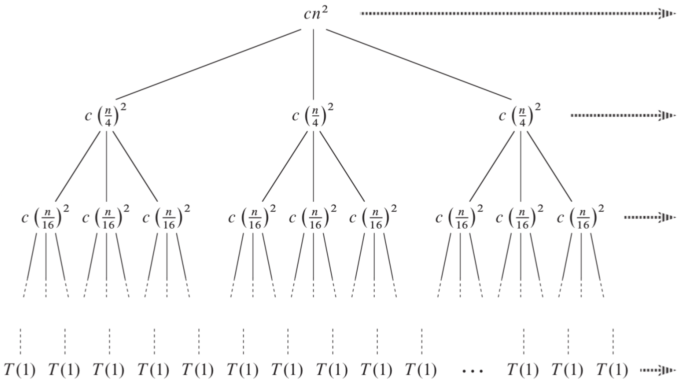


- Find the level cost at levels 0,1,2
- Determine the number of vertices at level k



- Find the level cost at level k
- Find the height of the tree
- Find the number of leaves

Finding the total cost $T(n)$



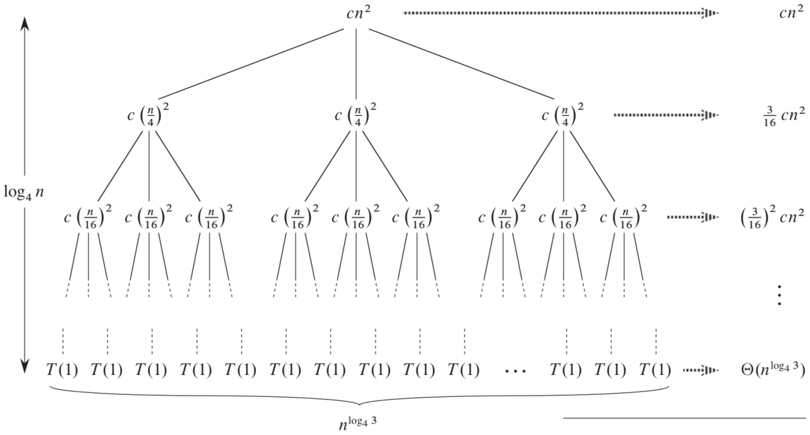
Finite geometric progression $(q \neq 1)$

$$1 + q + q^2 + \ldots + q^{h-1} = \frac{1 - q^h}{1 - q}$$

Cont'd

$\mathrm{cost}(k) = 3^k \cdot c \cdot (\frac{n}{4^k})^2, \quad h = \log_4 n, \quad \ell = n^{\log_4 3}$

Recursion tree method: overview



Total: $O(n^2)$

1. Draw the recursion tree.
2. Find the level costs and the height.
3. Add up the level costs to get the total cost $T(n)$.

Checking the guess[1] using induction

$$\begin{cases} T(1), T(2), T(3) \in \Theta(1) \\ T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2) & \text{for } n \geqslant 4 \end{cases}$$

Thm. For some $d > 0$, we have $T(n) \leqslant dn^2$ for all $n \geqslant 1$.

Proof. (by strong induction on n)
Base case: $T(1) = c_1, T(2) = c_2, T(3) = c_3$.
Need that $c_1 \leqslant d \cdot 1^2$, $c_2 \leqslant d \cdot 2^2$, $c_3 \leqslant d \cdot 3^2$
Need to choose $d \geqslant \max\{c_1, c_2/4, c_3/9\}$

Induction step:
Assume $T(n) \leqslant dn^2$ for all $n \in \{1, 2, \ldots, m\}$, where $m \geqslant 3$.
Need to show: $T(m+1) \leqslant d(m+1)^2$

[1] $T(n)$ is $O(n^2)$

Cont'd

$d \geqslant \max\{c_1, c_2/4, c_3/9\}$,
$T(n) \leqslant dn^2$ for $n \leqslant m \implies T(m+1) \leqslant d(m+1)^2$

Recursion tree: another example

DIY recursion tree

$$\begin{cases} T(1) = \Theta(1) \\ T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & \text{for } n \geqslant 2 \end{cases}$$

1. Draw the recursion tree.
   - Recall: we can afford to be a bit sloppy!
2. Find the level costs.
3. Add up the level costs to get the total cost.
4. Check the guess using induction.

1. Draw the recursion tree.
2. Find the level costs. To this end, find
   - number of vertices at level k
   - the problem size at level k
   - height of the tree
3. Add up the level costs to get the total cost $T(n)$.
4. Check the guess using induction.

Master theorem

Master theorem: warm up

We will use Master theorem to solve recurrence relations of the form:

$$T_n = aT_{n/b} + f(n) \tag{1}$$

We will need to compare the asymptotic order of growth of

$$n^{\log_b(a)} \quad \text{vs} \quad f(n)$$

Q: How many leaves does the recursion tree for (1) have?

Master theorem

Let $a \geqslant 1, b > 1, f(n) : \mathbb{N} \to \mathbb{R}^+$ and consider a recurrence

$$T_1, \ldots, T_{b-1} = \Theta(1)$$
$$T_n = aT_{n/b} + f(n) \qquad (n \geqslant b)$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then $\mathbf{T_n} = \mathbf{\Theta}(n^{\log_b a})$
2. If $f(n) = \Theta(n^{\log_b a})$, then $\mathbf{T_n} = \mathbf{\Theta}(n^{\log_b a} \log n)$
3. If $n^{\log_b a + \varepsilon} = O(f(n))$ for some $\varepsilon > 0$, and if $af(n/b) \leqslant cf(n)$ for some $c < 1$ and all sufficiently large n, then $\mathbf{T_n} = \mathbf{\Theta}(f(n))$

Notes

- $n^{\log_b a}$ is the number of leaves in the recursion tree.
- Master theorem does not cover all possible cases.

Examples

$$T_n = 2T_{n/2} + \Theta(n)$$
$$T_n = 10T_{n/3} + n^2 + 2$$
$$T_n = 2T_{n/2} + n\log_2 n$$

Upper bounding the n^{th} term with induction

$$f_0 = 1, \quad f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2} \qquad\qquad (n \geqslant 2)$$

Thm. We have $f_n \leqslant \left(\frac{5}{3}\right)^n$ for all $n \geqslant 0$.

Proof. (by strong induction on n)

Base case: $1 = f_0 \overset{?}{\leqslant} \left(\frac{5}{3}\right)^0 = 1$ ✓

$1 = f_1 \overset{?}{\leqslant} \left(\frac{5}{3}\right)^1 = \frac{5}{3}$ ✓

Inductive step: Assume $f_j \leqslant \left(\frac{5}{3}\right)^j$ for all $j \leqslant k$.

Need to show: $f_{k+1} \leqslant \left(\frac{5}{3}\right)^{k+1} \qquad (k \geqslant 1)$

Test yourself

You should be able to:

- Use recursion tree method to produce a guess for the asymptotic runtime of simple recursions
- Use induction to verify your guess
- Apply Master Theorem to recurrences and recognize cases when it cannot be applied.