Complexity IBC028, Lecture 2

H. Geuvers

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: spring 2021

Outline

Techniques to prove complexity

The Master Theorem

Techniques to prove $T(n) = \mathcal{O}(g(n))$ [or $T(n) = \Omega(g(n))$ or $T(n) = \Theta(g(n))$]

There a basically three techniques

1. **Substitution Method**:
   Choose (guess) g and c (and N_0) and prove $T(n) \leq c\, g(n)$ (for $n > N_0$) by induction on n.
2. **Recursion Tree method** :
   Method to find g. And then you still have to prove g is correct using (1)
3. **Master theorem method** :
   General theorem for patterns of the shape

   $$T(n) = aT(\frac{n}{b}) + f(n).$$

   Actually: casting the heuristic method of (2) into a general theorem.

Substitution method

Last week (MergeSort):

Theorem

If $T(n) \leq 2T(\lfloor \frac{n}{2} \rfloor) + \Theta(n)$, then

$$T(n) \in \mathcal{O}(n \log n).$$

In fact, the $n \log n$ was an educated guess, which we then proved by induction.

When proving something by induction, sometimes a trick is needed.

Substitution method: Example

Given $T(n) = 9T(\frac{n}{2}) + \Theta(n^3)$, prove that $T(n) = \mathcal{O}(n^3\sqrt{n})$.

Substitution method: Induction loading

$$T(n) = T(\left\lfloor \frac{n}{2} \right\rfloor) + T(\left\lceil \frac{n}{2} \right\rceil) + 1 \qquad \text{for } n \geq 2, \text{ and } T(1) = b$$

We guess that $T(n) = \mathcal{O}(n)$ and we try to show that $T(n) \leq c\,n$ for some appropriately chosen c.

$$\begin{array}{rcl} T(n) & \leq & c\left\lfloor \frac{n}{2} \right\rfloor + c\left\lceil \frac{n}{2} \right\rceil + 1 \\ & = & cn + 1 \quad \overset{??}{\leq} cn \quad \ldots \text{ no!} \end{array}$$

The trick is to add some constant: $T(n) \leq c\,n + d$.
Try the proof again and figure out what c and d could be.

$$\begin{array}{rcll} T(n) & \leq & c\left\lfloor \frac{n}{2} \right\rfloor + d + c\left\lceil \frac{n}{2} \right\rceil + d + 1 & \\ & = & cn + 2d + 1 & \\ & \leq & cn + d & \text{for } d = -1 \text{ and any } c. \end{array}$$

For the base case: $T(1) = b \leq c - 1$, so take $c := b + 1$.
We have $T(n) \leq (b+1)n - 1$ for all $n \geq 1$, so $T(n) \in \mathcal{O}(n)$.

Substitution method: Changing variables

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \log n$$

We rename variables and put $n = 2^m$ (and so $m = \log n$). Ignoring rounding off errors, we have

$$T(2^m) = 2T(2^{m/2}) + m$$

Consider this as a function in m: $S(m) = T(2^m)$ and we have

$$S(m) = 2S(\frac{m}{2}) + m$$

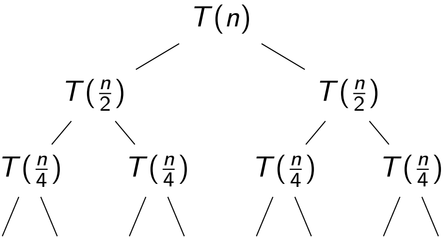
This is well-known and we have $S(m) = \mathcal{O}(m \log m)$.
We conclude that

$$T(n) = T(2^m) = S(m) \leq c(m \log m) = c(\log n \log \log n)$$

for some c.
So $T(n) = \mathcal{O}(\log n \log \log n)$.

Recursion Tree method (I)

Example $T(n) = 2T(\frac{n}{2}) + d\,n$.



$d\,n$

$2\,d\,\frac{n}{2} = d\,n$

$4\,d\,\frac{n}{4} = d\,n$

- The height is $\log n$, so there are $\log n + 1$ layers
- per layer: $d\,n$ contribution
- bottom: $\#\text{leaves} = 2^{\log n} = n$; cost per leaf $\Theta(1)$.
- So we conjecture: $T(n) = \Theta(n \log n)$

Some computation rules with log

For exponent: $(b^n)^m = b^{n \cdot m}$ and $b^n\, b^m = b^{n+m}$.

Per definition:

$$\log_b n = x \iff b^x = n \qquad \text{and so } b^{\log_b n} = n$$

Rules for log

$\log_b(n \cdot m)$	$=$	$\log_b n + \log_b m$	$\log_b(n^k)$	$=$	$k \log_b n$
$\log_b(\frac{n}{m})$	$=$	$\log_b n - \log_b m$	$\log_b(\frac{1}{n})$	$=$	$-\log_b n$

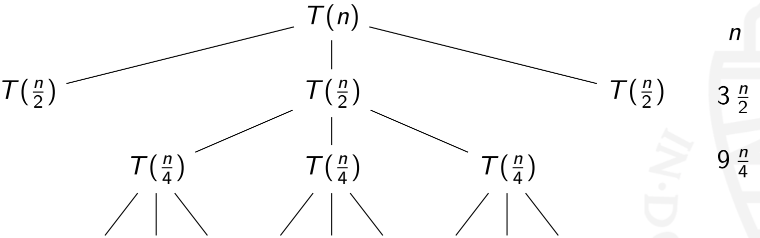
Changing base:

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$b^{\log_c a} = a^{\log_c b}$$

Recursion Tree method (II)

Exercise 4.4-1: $T(n) = 3T(\lfloor \frac{n}{2} \rfloor) + n$.
Question: find a "good" f with $T(n) = \mathcal{O}(f(n))$.



- The height is $\log n$. At layer i we have $3^i \frac{n}{2^i}$ contribution.
- Total:
  $\Sigma_{i=0}^{\log n} (\frac{3}{2})^i\, n = n\frac{(\frac{3}{2})^{\log n+1} - 1}{\frac{3}{2} - 1} \approx 2n(\frac{3}{2})^{\log n} = 2 \cdot 3^{\log n} = 2 \cdot n^{\log 3}$.
- So we conjecture: $T(n) = \mathcal{O}(n^{\log 3})$.

Substituion method

Exercise 4.4-1: $T(n) = 3T(\lfloor \frac{n}{2} \rfloor) + n$.
Conjecture: $T(n) = \mathcal{O}(n^{\log 3})$.

Proof. $T(n) \leq cn^{\log 3}$ for appropriately chosen c

$$
\begin{aligned}
T(n) &= 3T(\left\lfloor \frac{n}{2} \right\rfloor) + n \\
&\overset{IH}{\leq} 3c(\frac{n}{2})^{\log 3} + n \\
&= \frac{3c\, n^{\log 3}}{2^{\log 3}} + n = cn^{\log 3} + n \overset{??}{\leq} cn^{\log 3}
\end{aligned}
$$

The induction fails, so we add a linear factor: $T(n) \leq cn^{\log 3} + dn$.
We notice that it works for $d = -2$, because we have

$$T(n) = 3T(\left\lfloor \frac{n}{2} \right\rfloor) + n \overset{IH}{\leq} 3(c(\frac{n}{2})^{\log 3} - 2\frac{n}{2}) + n = cn^{\log 3} - 3n + n = cn^{\log 3} - 2n$$

Computing the median of an unsorted list

Problem: Given an unsorted list of elements, how to compute the median? (book: pp. 220-222)
(Median of A = element that has half of the elements of A below it and the other half above it.)
Possible solution:

- First sort the list A, with $|A| = n$.
- Then take the $\lfloor \frac{n}{2} \rfloor$-th element

This takes $\mathcal{O}(n \log n)$ time.
But it can be done in linear time!

General:

$$M(A, k) := \text{the } k\text{-th element of the sorted version of } A.$$

Then the median of A is $M(A, \frac{|A|}{2})$.

Computing the median of a list in linear time (I)

$$M(A, k) := \text{the } k\text{-th element of the sorted version of } A.$$

Let $n = |A|$. For purpose of exposition, we assume $n = 5^p$ for some p. (The book treats the general case.)

1. Split A randomly in $\frac{n}{5}$ groups of 5 elements
2. Determine the median of each group of 5 elements.
3. Determine recursively the median of these $\frac{n}{5}$ medians, say m
4. Count the number of elements in A that are $\leq m$, say ℓ.
    - If $\ell = k$, we are done and m is the output.
    - If $\ell > k$, then m is larger than the number we are looking for, so we continue recursively with $M(A \setminus A_{\mathrm{high}}, k)$
    - If $\ell < k$, then m is smaller than the number we are looking for, so we continue recursively with $M(A \setminus A_{\mathrm{low}}, k - 3\lceil \frac{n}{10} \rceil)$.
    - Until n is "very small", say $n \leq 10$, then compute the k-th element directly

Q. What exactly are A_{high} and A_{low} and how large are they?

Computing the median of a list in linear time (II)

$M(A, k) :=$ the k-th element of the sorted version of A.

Computing the median of a list in linear time (III)

1. Split A randomly in $\frac{n}{5}$ groups of 5 elements
2. Determine the median of each group of 5 elements.
3. Determine recursively the median of these $\frac{n}{5}$ medians, say m
4. Count the number of elements in A that are $\leq m$, say ℓ.
   - If $\ell = k$, we are done and m is the output.
   - If $\ell > k$, then m is larger than the number we are looking for, so we continue recursively with $M(A \setminus A_{\mathrm{high}}, k)$
   - If $\ell < k$, then m is smaller than the number we are looking for, so we continue recursively with $M(A \setminus A_{\mathrm{low}}, k - 3\lceil\frac{n}{10}\rceil)$.
   - Until n is "very small", say $n \leq 10$, then compute the k-th element directly

Complexity:

$$T(n) \leq T(\frac{n}{5}) + T(\frac{7n}{10}) + cn,$$

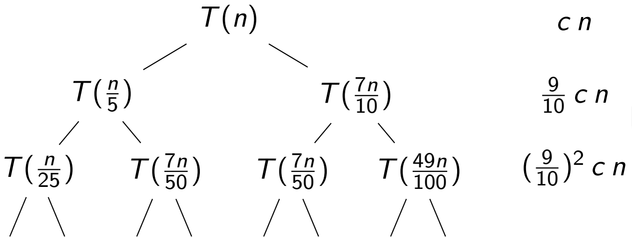
for some c.

Note that steps (1), (2) and the first part of (4) are linear in n.

Computing the median of a list in linear time (III)

$$T(n) \leq T(\frac{n}{5}) + T(\frac{7n}{10}) + cn.$$

To find T we can make a recursion tree;



So $T(n) = \Sigma_{i=0}^{??}(\frac{9}{10})^i\,c\,n \leq \Sigma_{i=0}^{\infty}(\frac{9}{10})^i\,c\,n = c\,n\,\Sigma_{i=0}^{\infty}(\frac{9}{10})^i = 10\,c\,n$

Computing the median of a list in linear time (IV)

$$T(n) \leq T(\frac{n}{5}) + T(\frac{7n}{10}) + cn.$$

From the recursion tree method we conjecture that $T(n) \leq 10\,c\,n$.

Proof by induction on n

- For small n, it is correct. (Possibly choose a larger c.)
- For larger n:

$$\begin{aligned} T(n) &\leq T(\frac{n}{5}) + T(\frac{7n}{10}) + cn \\ &\overset{\text{IH}}{\leq} 10\,c\,(\frac{n}{5}) + 10\,c\,(\frac{7n}{10}) + c\,n \\ &= 2\,c\,n + 7\,c\,n + c\,n \\ &= 10\,c\,n \end{aligned}$$

So T is linear in n, and so M is linear in the length of the input list.

Master Theorem

Theorem

Suppose $a \geq 1$ and $b > 1$ and

$$T(n) = aT(\frac{n}{b}) + f(n).$$

Then

1. $T(n) = \Theta(n^{\log_b a})$ if $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$.
   f is "relatively small" compared to $n^{\log_b a}$
2. $T(n) = \Theta(n^{\log_b a} \log n)$ if $f(n) = \Theta(n^{\log_b a})$.
   E.g. the Mergesort case
3. $T(n) = \Theta(f(n))$ if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$ and for sufficiently large n, we have $a\, f(\frac{n}{b}) \leq c\, f(n)$ for some $c < 1$.
   f is "relatively large" compared to $n^{\log_b a}$

Using the Master Theorem (I)

$$T(n) = 9T(\frac{n}{3}) + n.$$

Theorem

1. $T(n) = \Theta(n^{\log_b a})$ if $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$.
2. $T(n) = \Theta(n^{\log_b a} \log n)$ if $f(n) = \Theta(n^{\log_b a})$.
3. $T(n) = \Theta(f(n))$ if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$ and, for sufficiently large n, we have $a\,f(\frac{n}{b}) \leq c\,f(n)$ for some $c < 1$.

Now, $a = 9$ and $b = 3$, so $n^{log_b a} = n^{\log_3 9} = n^2$.
So $f(n) = n = \mathcal{O}(n) = \mathcal{O}(n^{log_b a - \varepsilon})$ with $\varepsilon = 1$.
So case (1) of the Master Theorem applies and we have

$$T(n) = \Theta(n^2).$$

Using the Master Theorem (II)

Theorem

1. $T(n) = \Theta(n^{\log_b a})$ if $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$.
2. $T(n) = \Theta(n^{\log_b a} \log n)$ if $f(n) = \Theta(n^{\log_b a})$.
3. $T(n) = \Theta(f(n))$ if $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$ and, for sufficiently large n, we have $a\, f(\frac{n}{b}) \leq c\, f(n)$ for some $c < 1$.

$$T(n) = 9\,T(\frac{n}{4}) + n^2.$$

Now, $a = 9$ and $b = 4$, so $n^{log_b a} = n^{\log_4 9} \approx n^{1.584}$.
So $f(n) = n^2 = \Omega(n^2) = \Omega(n^{log_b a + \varepsilon})$ for some $\varepsilon > 0$.
So case (3) of the Master Theorem applies and we have

$$T(n) = \Theta(n^2).$$

!!We need an extra check: $\exists c < 1\, \exists N_0\, \forall n \geq N_0 (a\, f(\frac{n}{b}) \leq c\, f(n)$??
That is: $9(\frac{n}{4})^2 \leq cn^2$, so take $c := \frac{9}{16}$ and this is ok.