DIVIDE-AND-CONQUER ALGORITHMS, PART 2

Marcus Ritt

CMP 601 – Algorithms and Theory of Computation —

<2020-04-14 ter>

Outline

1. Solving recurrences

2. Counting inversions

3. Matrix multiplication

- We can unroll

$$T(n) = \begin{cases} f(1), & \text{if } n = 1, \\ T(n-1) + f(n), & \text{otherwise.} \end{cases}$$

- to get $T(n) = \sum_{i\in[n]} f(i)$.

- We can show by induction that

$$T(n) = \begin{cases} O(1), & \text{if } \boxed{n \le n_0}, \\ T(bn) + f(n), & \text{otherwise,} \end{cases}$$

- for $b \in (0, 1)$
- is $T(n) = \sum_{0 \le i < \bar{ß}} f(b^i n) + O(1)$
- where $\bar{ß} = log_{1/b} n / n_0$.

Base case: $b^i n \le n_0$
$\Leftrightarrow b^i n/n_0 \le 1 \quad | : n_0$
$\Leftrightarrow n/n_0 \le (1/b)^i \quad | : b^i$
$\Leftrightarrow \log_{1/b} n/n_0 \le i \quad | \log_{1/b}$

$\bar{ß} \; \bar{i}$

$i \ge \bar{ß} \Rightarrow$ base case
$i < \bar{ß} \Rightarrow$ recurrence
$\bar{i} \equiv \bar{ß}$

b

	i
n	0
bn	1
$b^2 n$	2
$b^i n$	i

$f(n)$
$f(bn)$
$f(b^2 n)$
$\vdots$

$b^i n \le n_0$
$n/n_0 \le (1/b)^i$
$\log_{1/b} n/n_0 \Leftrightarrow i = \bar{i} = \bar{ß}$

Binary search

$T(n) = T(\tfrac{1}{2} \cdot n) + O(1) = \sum_{0 \le i < \log_2 n} c + O(1) = O(\log n)$.

An interesting recurrence

$T(n) = 2T(n/2) + O(n)$
$T(n) = 3T(n/2) + O(n)$
$T(n) = 7T(n/2) + O(n^2)$

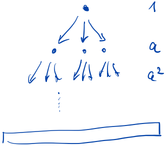
- By working harder we can reduce

$a^{\bar{\beta}}$

$$T(n) = \begin{cases} O(1), & \text{if } n \le n_0, \\ aT(bn) + f(n), & \text{otherwise,} \end{cases}$$

- where $a \in \mathbb{N}$ e $b \in (0,1)$ to a previous case, and find
- $T(n) = \sum_{0 \le i < \bar{ß}} a^i f(b^i n) + O(n^{\alpha})$
- where $\bar{ß} = \log_{1/b} n/n_0$ and $\alpha = \log_{1/b} a$.

internal nodes $>$ $=$ $<$? leafs



$1 \quad a^0$ $\quad a = 3$
$a \quad a^1$
$a^2 \quad a^2$
a^i

$\bar{\beta} = \lceil \log_{1/b} n/n_0 \rceil$

leafs: $a^{\bar{\beta}}$; cost $O(a^{\bar{\beta}}) = O(a^{\log_{1/b} n/n_0}) = O(a^{\log_{1/b} n}) = O(n^{\alpha})$.

$\log_{1/b} n/n_0 = \log_{1/b} n - \underbrace{\log_{1/b} n_0}_{\text{const.}}$ $\quad \alpha = \log_{1/b} n$

$f(n) = \Omega(n)$

$f(n) \geq cn$

- We were mainly interested in upper bounds
- You can use the methods also for proving lower bounds
- Substitution: guess a lower bound $f(n)$, then prove $T(n) \geq f(n)$.
- Expansion: assume a lower bound at the nodes, summing gives a lower bound for the total cost

$\Theta(n^{1.58})$ $\quad f(n) \stackrel{?}{=} O(n^{\alpha})$

The solution of

$T(n) = 3T(n/2) + O(n)$

$\alpha = \log_2 3 \approx 1.58$

$$T(n) = \begin{cases} O(1), & \text{if } n \le n_0, \\ aT(bn) + f(n), & \text{otherwise,} \end{cases}$$

$\alpha = 1 \quad \Theta(n^{\alpha}) = O(n)$

$T(n) = 2T(n/2) + n\log n$

$T(n) = 2T(n/2) + n/\log n$

$O(n^{1.58})$ leafs $\gg$ $O(n)$ merge

for $a \in \mathbb{N}$, $b \in (0,1)$ and $f(n)$ ~~assymtotically~~ asymptotically positive, is

a) $T(n) = \Theta(n^{\alpha})$, if $f(n) = O(n^{\alpha-\epsilon})$ for some $\epsilon > 0$; $\quad n^{\epsilon}$

3 b) $T(n) = \Theta(n^{\alpha}\log n)$, if $f(n) = \Theta(n^{\alpha})$;

c) $T(n) = \Theta(f(n))$, if $f(n) = \Omega(n^{\alpha+\epsilon})$ for some $\epsilon > 0$, and if $af(nb) \le cf(n)$ for some $c < 1$ and all sufficiently large n. $\quad n^{\epsilon}$

with $\alpha = \log_{1/b} a$.

Total merge cost next level is less than the merge cost of the current level

$$\begin{aligned} f(n) \quad & f(n) \\ af(bn) \quad & \le cf(n) \\ a^2 f(b^2 n) \quad & \le c^2 f(n) \\ \vdots \quad & \le \\ a^i f(b^i n) \quad & \le c^i f(n) \end{aligned}$$

$\sum c^i f(n) = f(n)\left(\sum c^i\right) \le d = \frac{1}{1-c}$

$O(f(n))$

(Empty)

$T(n) = T(n/2) + T(n/3) + \alpha(n)$

Given

$T(n) = 2T(n/2) + n^{0.8}) + g(n)$ $\qquad$ $T(n) = 2T(n/2) + n^{\log n}$

$$T(x) = \begin{cases} \Theta(1), & \text{if } x \le x_0, \\ \sum_{i\in[k]} a_i T(b_i x + h_i(x)) + g(x), & \text{otherwise,} \end{cases}$$

$g(x) = n^{\log n}$

with constants $a_i > 0$, $0 < b_i < 1$ e funções g, h such that

$|h_i(x)| \le \text{const}$ ✓

$$c_1 \; |g'(x)| \in O(x^c); \qquad c_2 \; |h_i(x)| \le x/\log^{1+\epsilon} x$$

for a $\epsilon > 0$ and constant x_0 sufficiently large we have

$$T(x) \in \Theta\left(x^p\left(1 + \int_1^x \frac{g(u)}{u^{p+1}}du\right)\right) \quad ②$$

$\log n$

$\int x^{-1} = [\log x]_1^x$

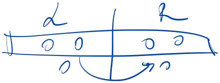
with p such that $\sum_{i\in[k]} a_i b_i^p \overset{!}{=} 1$. ①

x^{k-p}

$\int \frac{c u^k}{u^{p+1}} = \int c\, u^{k-p-1} = \left[\frac{c}{k-p} u^{k-p}\right]_1^x$

$\boxed{k=p}$

(Empty)

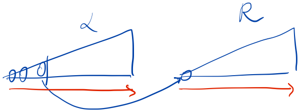


$n/2 \times n/2 = \Theta(n^2)$

1 2 3 4 : 0

4 3 2 1 : $\Theta(n^2)$

1 2 4 3 : 1



$O(n)$

- Split the sequence, count inversion on the left and the right
- Problem: counting *~~intra~~* inter *-inversions* between elements on the left and right
- Naïve: $(n/2)^2 = O(n^2)$ comparisons. But then $T(n) = 2T(n/2) + O(n^2) = \cancel{????}$. $O(n^2)$.
- $O(n)$ $\log_2 2 = 1$
  Idea: if we have sorted sequences, we can count in linear time!
- And: there's no problem in recursively counting and sorting at the same time.
- So we get: $T(n) = 2T(n/2) + O(n) = O(n \log n)$.

(Empty)

Multiplying blocks

Strassen

$$C_{ik} = \sum_{j \in [n]} a_{ij} b_{jk}$$

$O(n)$

$O(n^2 n) = O(n^3)$

$\Omega(n^2)$

Computing $C = AB$ by

$$\begin{aligned}
C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\
C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\
C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\
C_{22} &= A_{21}B_{12} + A_{22}B_{22}
\end{aligned}$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \overset{A}{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}} \cdot \overset{B}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}}$$

costs $T(n) = \boxed{8}\,T(n/2) + O(1) = \Theta(n^3)$

$8 \downarrow 7$

6

A better way

7

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22})B_{11}$$
$$M_3 = A_{11}(B_{12} - B_{22})$$
$$M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12})B_{22}$$

$n/2 \times n/2$ $\quad O(n^2)$

$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$
$$C_{11} = M_1 + M_4 - M_5 + M_7$$

$18\, n^2 = O(n^2)$

$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

$O(n^{2.5})$

$O(n^2)$

const.

This costs only $T(n) = 7T(n/2) + O(1) = O(n^{\log_2 7}) = O(n^{2.81})$

$\alpha = \log_2 7 \approx 2.81$ $\omega \leq 2.81.$

$\omega \overset{?}{=} 2?$ $\omega \geq 2.$