Master Theorem for Recurrences

Recall recurrences from last day
- recursion tree
- proof by induction

Example – changing variables

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

Let $m = \log n$ so $n = 2^m$

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

Let $S(m) = T(2^m)$

so $S(m/2) = T(2^{m/2})$

Then $S(m) = 2S(m/2) + m$ which we know

$S(m) \in O(m \log m)$ so $T(2^m) = O(m \log m)$

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

We often get recurrences of the form

$$T(n) = a\,T\left(\frac{n}{b}\right) + c \cdot n^k$$

Note: the recurrence relations you studied in MATH 239 were homogenous i.e. last term was 0.

This arises if we divide problem of size n into a subproblems of size $\frac{n}{b}$ and do $c \cdot n^k$ extra work.

e.g. $k = 1$

- $a = b = 2$

  $T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n$ mergesort

  $O(n \log n)$

- $a = 1$ $b = 2$

  $T(n) = T\left(\frac{n}{2}\right) + c \cdot n$

  $O(n)$

- $a = 4$ $b = 2$

  $T(n) = 4\,T\left(\frac{n}{2}\right) + c \cdot n$

EX. $O(n^2)$

<u>Theorem</u> ("Master Theorem")

$$T(n) = aT(\tfrac{n}{b}) + c \cdot n^k$$

← floor/ceiling allowed

$a \geq 1, b > 1, c > 0, k \geq 1$

Then

$$T(n) \in \begin{cases} \Theta(n^k) & \text{if } a < b^k \quad \text{i.e. } \log_b a < k \\ \Theta(n^k \log n) & \text{if } a = b^k \\ \Theta(n^{\log_b a}) & \text{if } a > b^k \end{cases}$$

Notes

- CLRS has more general version with $f(n)$ in place of cn^k
- You are not responsible for the proof but must know & apply the theorem

A rigorous proof is by induction.

We'll just make sense of it using recursion tree (written out)

$$T(n) = \underbrace{aT(\tfrac{n}{b})} + c \cdot n^k$$

$$= a[aT(\tfrac{n}{b^2}) + c(\tfrac{n}{b})^k] + c \cdot n^k$$

$$= a^2 T(\tfrac{n}{b^2}) + a \cdot c(\tfrac{n}{b})^k + c \cdot n^k$$

$$= \cdots = a^3 T(\tfrac{n}{b^3}) + a^2 \cdot c \cdot (\tfrac{n}{b^2})^k + a \cdot c(\tfrac{n}{b})^k + c \cdot n^k$$

$$= \cdots \quad = a^{\log_b n} T(1) + \sum_{i=0}^{\log_b n - 1} a^i \cdot c \cdot \left(\frac{n}{b^i}\right)^k \qquad \boxed{a^{\log_b n} = n^{\log_b a}}$$

change base of log

how far?

$\frac{n}{b^i} = 1$ so $\boxed{i = \log_b n}$

$$= n^{\log_b a} T(1) + c \cdot n^k \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^k}\right)^i$$

- If $a < b^k$ i.e. $\log_b a < k$
  then the second term dominates.
  Also Σ is constant. So get $O(n^k)$
- If $a = b^k$ then Σ is $O(\log n)$
  so get $O(n^k \log n)$
- If $a > b^k$ then the first term dominates
  $O(n^{\log_b a})$.

More general Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + c \cdot n^k \log^{\ell} n$$

$a \geq 1$, $b > 1$, $k \geq 0$

Then

$$T(n) \in \begin{cases} \Theta(n^k \log^{\ell} n) & \text{if } a < b^k \ (\log_b a < k) \\ \Theta(n^k \log^{\ell+1} n) & \text{if } a = b^k \\ \Theta(n^{\log_b a}) & \text{if } a > b^k \end{cases}$$

Divide and Conquer Examples

Counting Inversions

Some web sites try to match your preferences (for music, movies, books) with others.

How do you compare two rankings?

e.g. I like   best B D C A worst

You like   A D B C

Count: how many pairs do we rank differently?

i.e. how many pairs of lines cross (out of 6 pairs)
$= 4 \cdot 3/2$

BD BA DA CA

(the pairs that don't cross are BC DC)

Equivalently   my ranking 1 2 3 4

your ranking 4 2 1 3

and we count # <u>inversions</u> — # pairs out of order in 2nd list.

Brute Force: check all $\binom{n}{2}$ pairs $O(n^2)$

Does sorting help? Doesn't seem to.

Better with divide and conquer

List $a_1 \cdots a_n$. count # inversions.

Divide list in two $m = \lceil \frac{n}{2} \rceil$

$A = a_1 \cdots a_m$ $B = a_{m+1} \cdots a_n$

Recursively count # inversions in each half, r_A, r_B

Combine: answer $\leftarrow r_A + r_B + r$

r = # inversions with one element in A, one in B

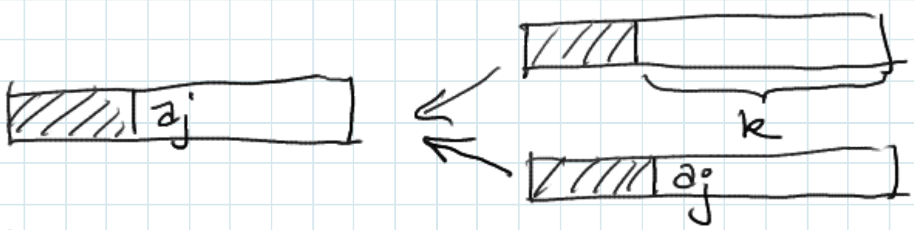
= # pairs a_i, a_j $a_i \in A$ $a_j \in B$, $a_i > a_j$

How do we find r ?

Can we count, for each $a_j \in B$, how many larger elements are there in A? — r_j

$$r = \sum_{a_j \in B} r_j$$

Think about mergesort: sort A, sort B, merge



when a_j is output to merged list $r_j \leftarrow k$

Whole algorithm

Sort-and-Count (L) – returns sorted L, #inversions

- divide L into A, B
  first half, second half
- $(r_A, A) \leftarrow$ Sort-and-Count (A)
- $(r_B, B) \leftarrow$ sort-and-count (B)
- $r \leftarrow 0$
- Do merge of A and B
  When element is moved from B to output
  $r \leftarrow r$ + #elements remaining in A
- return $(r_A + r_B + r$, merged list$)$

$T(n) = 2T(\frac{n}{2}) + O(n)$

solution: $T(n) = O(n \log n)$

Question: Is there a better algorithm?

$O(n \log n / \log\log n)$ '89

$O(n \sqrt{\log n})$ 2010 Timothy Chan et al.

using techniques/model where sorting is $o(n \log n)$