Recurrences

A quantity that is defined recursively in terms of smaller value of itself.

i.e: $T(n) = 3T(n/2) + 4(n/5)$

Mergesort

Mergesort (A, p, r) $\quad\quad$ $T(n)$ = total time of Mergesort

if $p < r$

$q = \frac{p+r}{2}$

Mergesort $(A, q+1, r) \rightarrow T(n/2)$

Mergesort $(A, p, q) \rightarrow T(n/2) \Rightarrow T(n) = T(n/2) + T(n/2) + \Theta(n)$

Merge $(A, p, q, r) \rightarrow \Theta(n)$ $\quad\quad$ $= 2T(\frac{n}{2}) + \Theta(n)$

Merge:

A_1: $\cancel{3}$ $\cancel{7}$ $\cancel{11}$ $\cancel{15}$ $\quad O(n)$

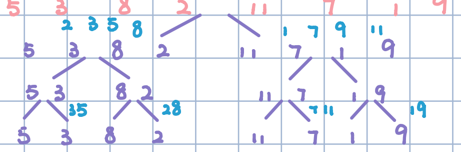
A_2: $\cancel{1}$ $\cancel{8}$ $\cancel{9}$ $\cancel{12}$ $\underbrace{17 \quad 20}$ $\quad O(n)$

1 3 7 8 9 11 12 15 17 20 (final sorted A_1 & A_2 Array)

total time $= O(n)$

1 2 3 5 7 8 9 11

Example: 5 3 8 2 11 7 1 9



(Merge)

Master Method (Cookbook)

Let $a \geq 1$ and $b > 1$ and $f(n)$ some function. Recurrance $T(n) = aT(n/b) + f(n)$ has solution:

Case 1:

upper bounded

If $f(n) = O(n^{\log_b^a - \epsilon})$ for $\epsilon > 0$ then $T(n) = \Theta(n^{\log_b^a})$

Case 2:

tightly bounded

If $f(n) = \Theta(n^{\log_b^a})$ then $T(n) = \Theta(n^{\log_b^a} \cdot \log n)$

Case 3:

lower bounded

If $f(n) = \Omega(n^{\log_b^a + \epsilon})$ for $\epsilon > 0$ and $af(n/b) \leq cf(n)$ for $0 < c < 1$ then $T(n) = \Theta(f(n))$

Example:

Mergesort $T(n) = 2T(n/2) + \Theta(n)$

$\Rightarrow a = 2 \quad b = 2$

$\log_2^2 = 1$

Application of case 2 : $T(n) = \Theta(n \log^n)$

$T(n) = 9T(n/3) + n$

$\Rightarrow a = 9 \quad b = 3 \quad f(n) = n = O(n)$

$\Rightarrow f(n) = O(n^{\log_3^9 - \epsilon}) \Rightarrow \epsilon = 1$

Application of case 1: $T(n) = \Theta(n^2)$

$T(n) = 3T(n/4) + n\log n$

$\Rightarrow a = 3,\ b = 4,\ f(n) = n\log n$

We got $f(n) = \Omega(n^{\log_4^3 + \epsilon}) \Rightarrow \epsilon = 0.2$ because $\log_4^3 = 0.793$

We got $3f(n/4) = \frac{3}{4} n \log\frac{n}{4} \leq \frac{3}{4} n \log n$ condition holds

Hence, $T(n) = \Theta(n\log^n)$