

Divide and Conquer

Divide a problem into one or more instances of the same problem of smaller size.

Conquer the problem by using the solutions to the smaller problems to find the solution to the original problem.

Example: Binary Search:

to find an element in a sorted list

determine if list is empty | compare

determine which half of the list
should contain the key - | compare

recursively search the proper half

↳ problem of size $\frac{n}{2}$

$$f(n) = 2 + f\left(\frac{n}{2}\right) \quad O(\log n)$$

Example : Max and Min

split the list in half

find the max and min in left

find the max and min in right

Compare maxes and mins

$\frac{n}{2}$

$\frac{n}{2}$

2

$$f(n) = f\left(\frac{n}{2}\right) + f\left(\frac{n}{2}\right) + 2$$

$$= 2f\left(\frac{n}{2}\right) + 2 \quad O(n)$$

Example: Mergesort

split the list in half
mergesort the left half
mergesort the right half
merge the halves

$O(\frac{n}{2})$
 $O(\frac{n}{2})$
 $O(n)$

$$\begin{aligned} f(n) &= f\left(\frac{n}{2}\right) + f\left(\frac{n}{2}\right) + n \\ &= 2f\left(\frac{n}{2}\right) + n \quad O(n \log n) \end{aligned}$$

$f(n)$'s were divide and conquer
recurrence relations.

They have the form

$$f(n) = af\left(\frac{n}{b}\right) + g(n)$$

A D+C algorithm divides a problem into a subproblems of size $\frac{n}{b}$ and does $g(n)$ operations to conquer the problem (combine solutions)

Suppose f satisfies

$$f(n) = a f\left(\frac{n}{b}\right) + g(n)$$

and $n = b^k$

$$f(n) = a f\left(\frac{n}{b}\right) + g(n)$$

$$= a \left(a f\left(\frac{n}{b^2}\right) + g\left(\frac{n}{b}\right) \right) + g(n)$$

$$= a^2 \left(a f\left(\frac{n}{b^3}\right) + g\left(\frac{n}{b^2}\right) \right) + a g\left(\frac{n}{b}\right) + g(n)$$

...

$$= a^k \underbrace{f\left(\frac{n}{b^k}\right)}_{f(1)} + \sum_{i=0}^{k-1} a^i g\left(\frac{n}{b^i}\right)$$

If $g(n) = c$, a constant,

$$\begin{aligned} f(n) &= a^k f(1) + \sum_{i=0}^{k-1} a^i c \\ &= a^k f(1) + c \frac{a^k - 1}{a - 1} \end{aligned}$$

Theorem 1: Let f be an increasing function that satisfies

$$f(n) = a f\left(\frac{n}{b}\right) + c$$

n is divisible by b , $a \geq 1$, $b > 1$ integer, $c > 0$ real

Then

$$f(n) = \begin{cases} O(n^{\log_b a}) & \text{if } a > 1 \\ O(\log n) & \text{if } a = 1 \end{cases}$$

If $n = b^k$ and $a \neq 1$

then $f(n) = C_1 n^{\log_b a} + C_2$

$$C_1 = f(1) + \frac{c}{a-1}$$

$$C_2 = -\frac{c}{a-1}$$

Binary Search : $f(n) = f\left(\frac{n}{2}\right) + 2$ is $O(\log n)$
 $a=1, b=2, c=2$

Max and Min : $f(n) = 2f\left(\frac{n}{2}\right) + 2$ is $O(n^{\log_2 2}) = O(n)$
 $a=2, b=2, c=2$

More Power!!!

Master Theorem

let f be an increasing function that satisfies

$$f(n) = af\left(\frac{n}{b}\right) + cn^d$$

$n = b^k$, $k > 0$ integer, $a \geq 1$, $b > 1$ integer,
 $c > 0$ real, $d \geq 0$ real

Then $f(n)$ is $\begin{cases} O(nd) & \text{if } a < b^d \\ O(nd \log n) & \text{if } a = b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$

Mergesort : $f(n) = 2f\left(\frac{n}{2}\right) + n$

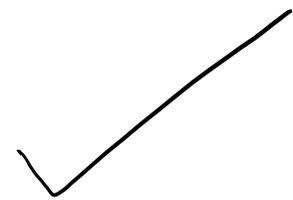
$$a=2, b=2, c=1, d=1$$

$$b^d = 2^1 = 2$$

is $O(n^d \log n)$

$$= O(n^1 \log n)$$

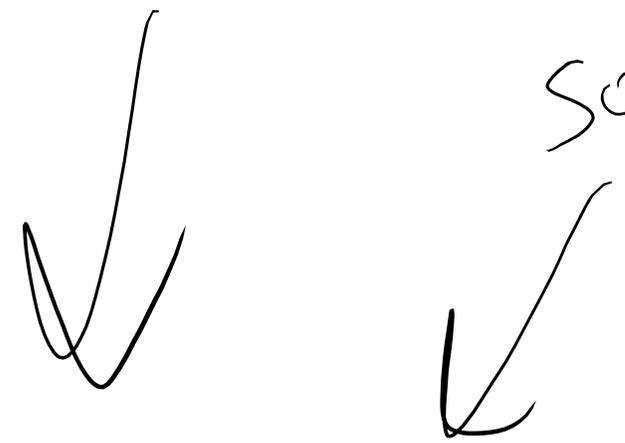
$$= O(n \log n)$$



my favorite

so fun!

Next Time:



formals languages
and automata!

