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 — 1 compare

should contain the key

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	$\frac{n}{2}$
find the max and min in right	$\frac{n}{2}$
compare maxes and mins	2

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

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

Example: Mergesort

Split the list in half

mergesort the left half $\frac{n}{2}$

mergesort the right half $\frac{n}{2}$

merge the halves $O(n)$

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

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

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 <u>divides</u> a problem into <u>a</u> subproblems of size $\underline{\frac{n}{b}}$ and does $\underline{g(n)}$ operations to conquer the problem (combine solutions)

Suppose f satisfies

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

and $n = b^k$

$$\begin{aligned} f(n) &= af\left(\frac{n}{b}\right) + g(n) \\ &= a\left(af\left(\frac{n}{b^2}\right) + g\left(\frac{n}{b}\right)\right) + g(n) \\ &= a^2\left(af\left(\frac{n}{b^3}\right) + g\left(\frac{n}{b^2}\right)\right) + ag\left(\frac{n}{b}\right) + g(n) \\ &\vdots \\ &= a^k \underbrace{f\left(\frac{n}{b^k}\right)}_{f(1)} + \sum_{i=0}^{k-1} a^i g\left(\frac{n}{b^i}\right) \end{aligned}$$

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

$$f(n) = a^k f(1) + \sum_{i=0}^{k-1} a^i c$$

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

<u>Theorem</u> 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 : $f(n) = f(\frac{n}{2}) + 2$ is $O(\log n)$

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

Max and Min : $f(n) = 2f(\frac{n}{2}) + 2$ 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 $\begin{cases} O(n^d) & \text{if } a < b^d \\ O(n^d \log n) & \text{if } a = b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$

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

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

$b^d = 2^1 = 2$

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!