DP vs. D&C:

Dynamic Programming:

- write recursive formula but often express in terms of slightly smaller subproblems, e.g. $T(i)$ in terms of $T(i-1)$ & $T(i-2)$.
- Hence recursive algorithm will blow-up. small subproblems solved too many times.
- Thus use iterative algorithm to solve bottom-up.

Divide & conquer:

- Express solution to problem of size n in terms of subproblems of size $\frac{n}{2}$ (or of size cn for $c<1$)
- Use recursion to solve subproblems & "Combine/merge" to get solution to original.

Fast multiplication alg. from last class:

FastMultiply(x, y):

input: n-bit integers x & y where n is a power of 2
output: $z = xy$

x_L = 1st $\frac{n}{2}$ bits of x & x_R = last $\frac{n}{2}$ bits of x
y_L = 1st $\frac{n}{2}$ bits of y & y_R = last $\frac{n}{2}$ bits of y

$A = \text{FastMultiply}(x_L, y_L)$
$B = \text{FastMultiply}(x_R, y_R)$
$C = \text{FastMultiply}(x_L + x_R, y_L + y_R)$
Return$(A \times 2^n + (C - A - B) \times 2^{n/2} + B)$

Running time:
We'll show now:

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n) = O(n^{\log_2 3}) \approx O(n^{1.59})$$

Note, consider example: $x = 13 = (1101)_2$ & $y = 11 = (1011)_2$
to compute 13×11 we use: $x_L = 3$, $x_R = 1$
$y_L = 2$, $y_R = 3$
& $A = 3 \times 2$, $B = 1 \times 3$, $C = (3+1) \times (2+3)$.

Key fact for solving recurrences:

Understanding <u>geometric</u> series:

Examples:

$$1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^n} = O(1)$$

$$1 + 3 + 3^2 + 3^3 + \cdots + 3^n = O(3^n)$$

for constant $\alpha > 0$,

$$\sum_{i=0}^{k} \alpha^i = 1 + \alpha + \alpha^2 + \cdots + \alpha^k$$

if $\alpha < 1$, then first term dominates

if $\alpha > 1$, then last term dominates.

<u>Lemma</u>: For $\alpha > 0$,

$$\sum_{i=0}^{k} \alpha^i = \begin{cases} O(1) & \text{if } \alpha < 1 \\ O(k) & \text{if } \alpha = 1 \\ O(\alpha^k) & \text{if } \alpha > 1 \end{cases}$$

Easy Multiply had the following recurrence:

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n)$$

note this means there exists a constant $c > 0$ so that:

$$T(n) \leq 4T\left(\frac{n}{2}\right) + cn$$

& the base case is always $T(1) = O(1) \leq c$.

Take & expand it out:

$$T(n) \leq cn + 4T\left(\frac{n}{2}\right)$$

note $T\left(\frac{n}{2}\right) \leq 4T\left(\frac{n}{2^2}\right) + c\left(\frac{n}{2}\right)$

Thus,

$$T(n) \leq cn + 4\left[4T\left(\frac{n}{2^2}\right) + \frac{cn}{2}\right]$$

$$= cn + \left(\frac{4}{2}\right)cn + 4^2 T\left(\frac{n}{2^2}\right)$$

$$\leq cn + \left(\frac{4}{2}\right)cn + 4^2\left[4T\left(\frac{n}{2^3}\right) + \frac{cn}{2^2}\right]$$

$$\leq cn\left(1 + \left(\frac{4}{2}\right) + \left(\frac{4}{2}\right)^2\right) + 4^3 T(n/2^3)$$

$$\leq cn\left(1 + \left(\frac{4}{2}\right) + \left(\frac{4}{2}\right)^2 + \cdots + \left(\frac{4}{2}\right)^{i-1}\right) + 4^i T(n/2^i)$$

Stop when $i = \log_2 n$ so that $\frac{n}{2^i} = \frac{n}{n} = 1$

$$T(n) \leq cn\left(1 + \left(\frac{4}{2}\right) + \left(\frac{4}{2}\right)^2 + \cdots + \left(\frac{4}{2}\right)^{\log_2 n - 1}\right) + 4^{\log_2 n} c$$

$T(1)$

$$= O(n) \times O\left(2^{\log_2 n}\right) + O\left(4^{\log_2 n}\right)$$

$$= O(n) \times O(n) + O(n^2)$$

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

Recall:

$$4^{\log_2 n} = 2^{2\log_2 n} = n^2$$

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$$\leq cn + 3T\left(\frac{n}{2}\right)$$

$$\leq cn + 3\left(3T\left(\frac{n}{2^2}\right) + \frac{cn}{2}\right)$$

$$= cn\left(1 + \frac{3}{2}\right) + 3^2 T\left(\frac{n}{2^2}\right)$$

$$\leq cn\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^{i-1}\right) + 3^i T\left(\frac{n}{2^i}\right)$$

$$\leq cn\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \cdots + \left(\frac{3}{2}\right)^{\log_2 n - 1}\right) + 3^{\log_2 n} c$$

$$= O\cancel{(n)} \times O\left(\left(\frac{3}{\cancel{2}}\right)^{\log_2 n}\right) + \cancel{3}\, O\left(3^{\log_2 n}\right)$$

$$= O\left(3^{\log_2 n}\right)$$

$$= O\left(n^{\log_2 3}\right)$$

recall: $3^{\log_2 n} = \left(2^{\log_2 3}\right)^{\log_2 n} = \left(2^{\log_2 n}\right)^{\log_2 3} = n^{\log_2 3}.$

$$T(n) = 2T\left(\frac{n}{3}\right) + O(n)$$

$$\leq 2T\left(\frac{n}{3}\right) + cn$$

$$\leq cn + 2\left(2T\left(\frac{n}{3^2}\right) + cn/3\right)$$

$$= cn\left(1 + \left(\frac{2}{3}\right)\right) + 2^2 T\left(\frac{n}{3^2}\right)$$

$$\leq cn\left(1 + \left(\frac{2}{3}\right)\right) + 2^2\left(2T\left(\frac{n}{3^3}\right) + cn/3^2\right)$$

$$= cn\left(1 + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right)^2\right) + 2^3 T\left(\frac{n}{3^3}\right)$$

$$\leq cn\left(1 + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right)^2 + \cdots + \left(\frac{2}{3}\right)^{i-1}\right) + 2^i T\left(\frac{n}{3^i}\right)$$

stop when $i = \log_3 n$ so $\frac{n}{3^i} = 1$

$$\leq cn\Big(\underbrace{1 + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right)^2 + \cdots + \left(\frac{2}{3}\right)^{\log_3 n - 1}}_{= O(1) \text{ since } \alpha = \frac{2}{3} < 1}\Big) + 2^{\log_3 n} c$$

$$= O(n) \times O(1) + O\left(2^{\log_3 n}\right)$$

$$= O(n) + O\left(n^{\log_3 2}\right) \qquad 2^{\log_3 n} = \left(3^{\log_3 2}\right)^{\log_3 n} = n^{\log_3 2}$$

$$= O(n) \quad \text{since } \log_3 2 < 1.$$

In general, for $T(n) \le aT\left(\frac{n}{b}\right) + O(n)$, $T(1) = O(1)$
for constants $a > 0$, $b > 1$.

$$T(n) \le aT\left(\frac{n}{b}\right) + cn$$

$$\le cn + a\left[aT\left(\frac{n}{b^2}\right) + \frac{cn}{b}\right]$$

$$= cn\left(1 + \left(\frac{a}{b}\right)\right) + a^2 T\left(\frac{n}{b^2}\right)$$

$$\le cn\left(1 + \left(\frac{a}{b}\right)\right) + a^2\left[aT\left(\frac{n}{b^3}\right) + \frac{cn}{b^2}\right]$$

$$\le cn\left(1 + \left(\frac{a}{b}\right) + \left(\frac{a}{b}\right)^2\right) + a^3 T\left(\frac{n}{b^3}\right)$$

$$\le cn\left(1 + \left(\frac{a}{b}\right) + \left(\frac{a}{b}\right)^2 + \cdots + \left(\frac{a}{b}\right)^{i-1}\right) + a^i T\left(\frac{n}{b^i}\right)$$

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

$$\le cn\Bigg(\underbrace{1 + \left(\frac{a}{b}\right) + \left(\frac{a}{b}\right)^2 + \cdots + \left(\frac{a}{b}\right)^{\log_b n - 1}}\Bigg) + a^{\log_b n} c$$

$= O(1)$ if $\frac{a}{b} < 1$

$= O(\log n)$ if $a = b$

$= O\left(\left(\frac{a}{b}\right)^{\log_b n}\right)$ if $\frac{a}{b} > 1$

$a^{\log_b n} = n^{\log_b a}$

if $\underline{a < b}$: $T(n) \le O(n) \times O(1) + O(n^{\log_b a}) = O(n)$

if $\underline{a = b}$: $T(n) = O(n \log n) + O(n^{\log_b a}) = O(n \log n)$

if $\underline{a > b}$: $T(n) = O(\cancel{n}) \times O\left(\left(\frac{a}{\cancel{b}}\right)^{\log_b n}\right) + O(n^{\log_b a}) = O(n^{\log_b a})$

Master Theorem:

For constants $a>0, b>1, d \geq 0$, the recurrence:

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

Solves to:

$$T(n) = \begin{cases} O(n^d) & \text{if } \frac{a}{b^d} < 1 \\ O(n^d \log n) & \text{if } \frac{a}{b^d} = 1 \\ O(n^{\log_b a}) & \text{if } \frac{a}{b^d} > 1 \end{cases}$$

Proof idea:

expanding out we get:

$$T(n) \leq cn^d\left(1 + \left(\frac{a}{b^d}\right) + \left(\frac{a}{b^d}\right)^2 + \dots + \left(\frac{a}{b^d}\right)^{\log_b n}\right)$$

if $\frac{a}{b^d} < 1$ then $T(n) = O(n^d)$

if $\frac{a}{b^d} = 1$ then $T(n) = O(n^d \log n)$

if $\frac{a}{b^d} > 1$ then $T(n) = O(a^{\log_b n})$

$$= O(n^{\log_b a})$$

Example recurrences:

Binary search: $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$$a=1,\ b=2,\ d=0$$

$$\frac{a}{b^d} = \frac{1}{1} = 1$$

$$\text{so } T(n) = O(\log n)$$

MergeSort: $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

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

$$\frac{a}{b^d} = \frac{2}{2} = 1$$

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

Multiplication: $T(n) = 4T\left(\frac{n}{2}\right) + O(n)$

$$a=4,\ b=2,\ d=1 \text{ so } \frac{a}{b^d} > 1$$

$$T(n) = O(n^{\log_2 4}) = O(n^2)$$

Note, $T(n) = 2T\left(\frac{n}{2}\right) + O(1)$

$$a=2,\ b=2,\ d=0 \text{ so } \frac{a}{b^d} = \frac{2}{1} > 1$$

$$T(n) = O(n^{\log_2 2}) = O(n)$$

$$T(n) = T\left(\frac{3}{4}n\right) + O(n)$$

$a=1,\ b=\frac{4}{3},\ d=1$ so $\frac{a}{b^d} = \frac{1}{4/3} < 1$

$$T(n) = O(n)$$

Note, $T(n) = T(\alpha n) + O(n)$

for any $\alpha < 1$ solves to $T(n) = O(n)$.