NOTE BOOK

Rokshana Ahmed

DIVIDE-ET-IMPERA / DIVIDE-AND-CONQUER ALGORITHMS

Rq ($q \geq 2$) (RECURRENCIES)

$$T_q(n) \leq \begin{cases} q \cdot T(\frac{n}{2}) + c \cdot n & \forall n \geq 3 \\ c & n \in \{0,1,2\} \end{cases}$$

(q: parameter)

THM: $T_2(n) \leq O(n \log n)$

THM: $T_q(n) \leq O(n^{\log_2 q})$ $\forall$ INTEGER $q \geq 3$ ← different behaviour then when $q < 3$

$(T_1(n) \leq O(n))$

Thus,

$T_4(n) \leq O(n^{\log_2 4}) = O(n^2)$

$T_3(n) \leq O(n^{\log_2 3}) = O(n^{1.5849\ldots})$ ← strange because we have been looking at combinatorial runtimes

$\log_2 3 = 1.5849$

INTEGER ADDITION

$n=4$

```
 1100 +
 0111 =
------
10011
```

Summing up two binary (decimal) numbers of "n" digits each takes $O(n)$ time.

INTEGER MULTIPLICATION

```
 1100 ·
 0111
------
 1100
 1100 ·
1100 ·
0000
------
```

Multiplying two binary (decimal) numbers of n digits each (using the primary school algo) takes $O(n^2)$ time.

HOW TO IMPROVE MULTIPLICATION?

Let us assume that our goal is to multiply two n digits numbers, x and y.

We would like to compute $M = x \cdot y$

To implement a divide-et-impera approach we will split x and y into two halves each.

$X = 100110$	$X_1 = 100$	$X_0 = 110$
$Y = 011010$	$Y_1 = 011$	$Y_0 = 010$

$x_1 \cdot 2^{n/2} = 100 \cdot 1000 = 100000$

$x_1 2^{n/2} + x_0 = 100000 + 110 = 100110$

Then, $x = x_1 \cdot 2^{n/2} + x_0$ and $Y = Y_1 \cdot 2^{n/2} + y_0$

$(a+b)(c+d)$

$ac + ad + bc + bd$

$$x \cdot y = (x_1 \cdot 2^{n/2} + x_0) \cdot (y_1 \cdot 2^{n/2} + y_0) =$$
$$= x_1 y_1 2^{n/2} 2^{n/2} + x_1 y_0 2^{n/2} + x_0 y_1 2^{n/2} + x_0 y_0$$
$$= x_1 y_1 2^{n} + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$$

Thus, we have split the problem of multiplying two binary numbers of n-digits each into the problems of

1. Computing 4 products of binary numbers of $n/2$ digits each;
2. Performing 3 sums of n-digits binary numbers;
3. Performing 2 multiplications with powers-of-2

Now, 1 and 2 can be easily performed in $O(n)$ time. What about 3?

$$T_4(n) \leq 4T\left(\frac{n}{2}\right) + cn$$

If $q = 4$, we have that $T_4(n) \leq O(n^{\log_2 4}) = O(n^2)$

We didn't gain anything so far! =

If we want to do better than n^2, we need to reduce the number of subproblems to $q \leq 3$.

We aim to compute:

$$M = X_1 Y_1 \, 2^n + (X_1 Y_1 + X_0 Y_0) 2^{n/2} + X_0 Y_0$$

$$M_2 = (X_1 + X_0)(Y_1 + Y_0) = X_1 Y_1 + X_1 Y_0 + X_0 Y_1 + X_0 Y_0$$

$$M_1 = X_1 Y_1$$

$$M_0 = X_0 Y_0$$

we are performing ONE SINGLE multiplication. We can see it gives 4 mult. later but it's still a single one.

$$M_2 - M_1 - M_0 = (X_1 + X_0)(Y_1 + Y_0) - X_1 Y_1 - X_0 Y_0 =$$

$$= \cancel{X_1 Y_1} + X_1 Y_0 + X_0 Y_1 + \cancel{X_0 Y_0} - \cancel{X_1 Y_1} - \cancel{X_0 Y_0} =$$

$$= X_1 Y_0 + X_0 Y_1$$

$$M = X_1 Y_1 \, 2^n + (X_1 Y_0 + X_0 Y_1) 2^{n/2} + X_0 Y_0 = M_1 \, 2^n + (M_2 + M_1 - M_0) 2^{n/2} + M_0$$

Thus, we can get the product of two n-digits numbers by taking 3 products of two $\frac{n}{2}$-digits numbers.

With $q = 3$, I get:

$$T_3(n) \leq O(n^{\log_2 3}) = O(n^{1.5849\ldots})$$

This improves significantly the primary school algo.

RECURSIVE-MULTIPLY (x, y):

LET $x = x_1 2^{n/2} + x_0$ and $Y = Y_1 2^{n/2} + Y_0$	$O(n)$
COMPUTE $S_x = X_1 + X_0$ and $S_Y = Y_1 + Y_0$	$O(n)$
M_2 = RECURSIVE-MULTIPLY (S_x, S_Y)	$T_3(\frac{n}{2})$
M_1 = RECURSIVE-MULTIPLY (X_1, Y_1)	$T_3(\frac{n}{2})$
M_0 = RECURSIVE-MULTIPLY (X_0, Y_0)	$T_3(\frac{n}{2})$
RETURN $M_1 \, 2^n + (M_2 - M_1 - M_0)$	$O(n)$

$$T_3(n) \leq 3T_3(\tfrac{n}{2}) + O(n) \implies T_3(n) \leq O(n^{\log_2 3})$$

The best known algorithm for multiplying two n-digits numbers takes time $O(n \log n \cdot 2^{O(\log^* n)})$

n of logs to take to make result < 2

$\log^* n \leq 10$ IF $n \leq 2^{2^{2^{2^{2^{2^{2^{2^{2^2}}}}}}}}$ (10 times)

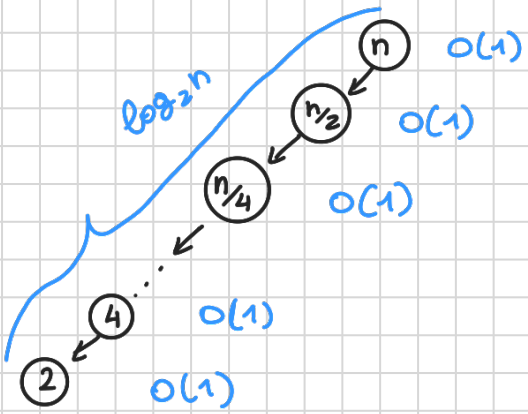
2^{50}

$((2^2)^2)^2 = 2^{2^3} = 2^8 = 256$

$< 2^{273}$ atoms in the universe

$\log^* 2^{273} < 5$

$$S(n) \leq \begin{cases} S(n/2) + c & \forall n \geq 3 \\ c & n \in \{0,1,2\} \end{cases}$$



TOTAL RUNTIME

$S(n) \leq O(\log n)$

THM: $S(n) \leq c \log_2 n, \quad \forall n \geq 2$

P: $S(0) = S(1) = S(2) = c$ (BASE CASE HOLDS)

$$S(n) \leq S\left(\frac{n}{2}\right) + c \leq c \log_2 \left(\frac{n}{2}\right) + c$$

$$= c(\log_2 n - \log_2 2) + c$$

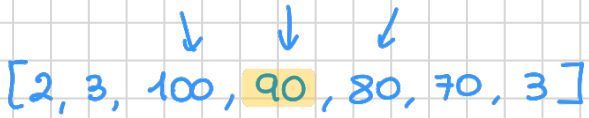
$$= c(\log_2 n - 1) + c$$

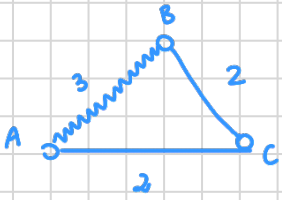
$$= c \log_2 n - \cancel{c} + \cancel{c} = c \log_2 n \quad \blacksquare$$

EX: Let x be an unimodal array of size n, that is, $\exists\, i \in \{0, 1, \ldots, n-1\}$, such that $x[i]$ is sorted increasingly, and $x[i]$ is sorted decreasingly

$x = [2, 3, 10, 100, 94, 20, 3, 1]$

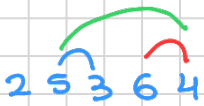
Find the largest value in x as fast as you can.





Let V be a n-dimensional array.

We say that $0 \leq i \leq j \leq n-1$ form an inversion IF $V[i] > V[j]$



1,2
1,4
3,4

```
C=0
FOR i=0,...,n-2
  FOR j=i+1,...,n-1
    IF V[i] > V[j]:
      C+=1
RETURN C
```

} O(1)