Algorithms

Recursion
(part 3)

Recap

- HW2 due Friday
    - by start of class
    - may work in groups
    - cite sources if you use them

- Office hours: Wed. 10-11am
  Thurs 1-2pm
  (or by arrangement)

Last week:

Recursion & recurrences
(& induction - same thing!)

Next One: Multiplication

In general, we say this is $O(1)$ time $\longrightarrow$ lies!

In reality:

$$
\begin{array}{r}
31415962 \\
\times\ 27182818 \\
\hline
251327696 \\
31415962 \\
251327696 \\
62831924 \\
251327696 \\
31415962 \\
219911734 \\
62831924 \\
\hline
853974377340916
\end{array}
$$

How to formalize?

nested for loops

Runtime? (2 n-bit #s)

$O(n^2)$

m & n:

$O(mn)$

Better: A trick:

$$(10^m a + b)(10^m c + d)$$
$$= 10^{2m} ac + 10^m (bc + ad) + bd$$

Example

$$\left.\begin{array}{r} 963{,}245 \\ \times\, 624{,}197 \end{array}\right\} \leftarrow \underline{m=3}:$$

$$= (10^3 \times 963 + 245)(10^3 \cdot 624 + 197)$$

$$= 10^{2\cdot 3}(963)(624) +$$
$$10^3 (245 \times 624 + 963 \times 197)$$

$\dots$

Make this an algorithm:

$\underline{\text{MULTIPLY}(x, y, n):}$ $M(n)$

if $n = 1$
  return $x \cdot y$?

else
  $O(1)$ $m \leftarrow \lceil n/2 \rceil$ +
  $a \leftarrow \lfloor x/10^m \rfloor$; $b \leftarrow x \bmod 10^m$
  $d \leftarrow \lfloor y/10^m \rfloor$; $c \leftarrow y \bmod 10^m$
  $M(\frac{n}{2})$ — $e \leftarrow \text{MULTIPLY}(a, c, m)$
  $M(\frac{n}{2})$ - $f \leftarrow \text{MULTIPLY}(b, d, m)$ "
  $M(\frac{n}{2})$ - $g \leftarrow \text{MULTIPLY}(b, c, m)$ "
  $M(n/2)$ $h \leftarrow \text{MULTIPLY}(a, d, m)$
  $O(1)$ return $10^{2m}e + 10^m(g + h) + f$

Runtime:

$$M(n) \le O(1) + O(1) + 4M\left(\frac{n}{2}\right) + O(1)$$

$$= \underbrace{4M\left(\frac{n}{2}\right)}_{aM\left(\frac{n}{b}\right)} + O(1) \rightarrow n^{\log_b a} = n^{\log_2 4} = n^2$$

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

Hrm - not better after all...

Another trick! 3 multiplications

$$\cancel{ac} + \cancel{bd} - \underbrace{(a-b)(c-d)}_{(ac - \cancel{bc} - \cancel{ad} + bd)} = bc + ad$$

Huh?

Recall:

$$(10^m a + b)(10^m c + d)$$
$$= 10^{2m} ac + 10^m (bc + ad) + bd$$

4 mult.

Conclusion: black magic:

using algebra, can do 3 multiplications & get result of 4!

New & improved pseudocode:

<u>FastMultiply(x, y, n):</u>
  if $n = 1$
    return $x \cdot y$
  else
    $m \leftarrow \lceil n/2 \rceil$
    $a \leftarrow \lfloor x/10^m \rfloor$; $b \leftarrow x \bmod 10^m$
    $d \leftarrow \lfloor y/10^m \rfloor$; $c \leftarrow y \bmod 10^m$
    $e \leftarrow$ FastMultiply(a, c, m)
    $f \leftarrow$ FastMultiply(b, d, m)
    $g \leftarrow$ FastMultiply$(a - b, c - d, m)$
    return $10^{2m}e + 10^m(e + f - g) + f$

O(1)

3 calls

Analysis:

$$M(n) = \overset{a}{3}M\left(\frac{n}{\underset{b}{2}}\right) + O(1)$$

$$f(n) = O(1)$$

to $n^{\log_b a} = n^{\log_2 3}$

between 1 & 2

$$M(n) = O\left(n^{\log_2 3}\right) \approx O\left(n^{1.7\ldots}\right)$$

Some comments

- In practice, done in base 2, not 10.

- Actually, this can break down even more!

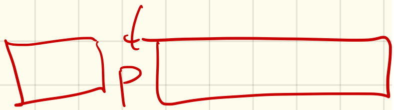
  If we apply another recursive layer, can get $O(n \log n)$ eventually.

  (Ever heard of Fast Fourier transforms?)

Linear Time Selection

First: Quicksearch

Idea: Modify quick sort, but don't recurse on both sides




```
QUICKSELECT(A[1..n], k):
  if n = 1
      return A[1]
  else
      Choose a pivot element A[p]
      r ← PARTITION(A[1..n], p)

      if k < r
          return QUICKSELECT(A[1..r − 1], k)
      else if k > r
          return QUICKSELECT(A[r + 1..n], k − r)
      else
          return A[r]
```

median of medians

Figure 1.12. Quickselect, or one-armed quicksort

Ex: Find 5th element (in sorted order) from:

3 5 11 2 6 9 1 7 4 8 0

p

2 1 0 3 5 11 6 9 7 4 8

Runtime:

Well, depends on pivot!

Ideal case:

divide in half

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

$$= n + \frac{n}{2} + \frac{n}{4} + \cdots$$

$$= \sum \frac{n}{2^i} = O(n)$$

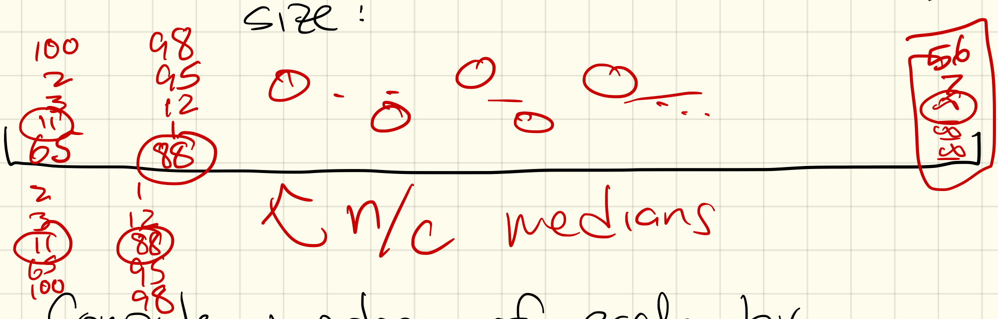
Worst case: pivot is 1st or last

$$Q(n) = \underbrace{O(n) + Q(n-1)}_{= n 2^n}$$

Improvement: Need a good enough pivot!

Median of medians: "MOM"

Divide into blocks that are $O(1)$ size:



Compute median of each by brute force

Runtime so far:

blocks of size c

n/c blocks

$$O(1) \times (\#\text{ blocks}) = O(n)$$

Then, recurse, & find median of medians.

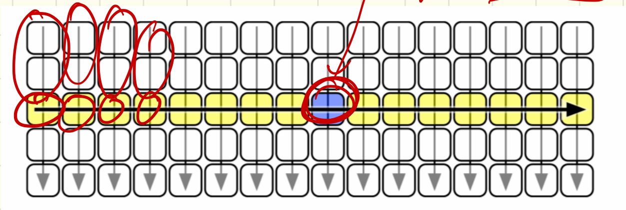
Why does this work!?

Claim: MoM is a good pivot.

Why? Well, we've chopped the n elements into n/c groups of c.

Say $c=5$:

median in his row



n/c blocks, each holding c
($n/5$, each w/ 5)

The median of medians is $\geq$ at least $\frac{n}{5}$ elements in large array

Runtime:

```
MomSelect(A[1..n], k):
  if n ≤ 25  ⟨⟨or whatever⟩⟩
      use brute force
  else
      m ← ⌈n/5⌉
      for i ← 1 to m
          M[i] ← MedianOfFive(A[5i−4..5i])  ⟨⟨Brute force!⟩⟩
      mom ← MomSelect(M[1..m], ⌊m/2⌋)       ⟨⟨Recursion!⟩⟩

      r ← Partition(A[1..n], mom)

      if k < r
          return MomSelect(A[1..r−1], k)      ⟨⟨Recursion!⟩⟩
      else if k > r
          return MomSelect(A[r+1..n], k−r)    ⟨⟨Recursion!⟩⟩
      else
          return mom
```

Next time: Backtracking

(reading due as usual over 2 intro sections)