# We've done

- Solving Recurrences
  - The substitution method
    - Make a guess with iteration or recursion-tree
    - Prove correctness by induction
  - The Master theorem

# Now

- The "Divide and Conquer" method
  - Introduction to sorting, featuring Quicksort
  - Medians and order statistics
  - Introduction to probabilistic analysis, average-case analysis
  - Integer multiplication
  - Matrix multiplication

# Next

- Sorting networks

# Divide and Conquer

Basic idea:

1. Divide: Partition the problem into smaller ones
2. Conquer: Recursively solve the smaller problems (Remember to solve the base case)
3. Combine the partial solutions

Examples:

- Merge-sort (read on your own)
- Quicksort
- Order statistics, matrix multiplication, integer multiplication
- Bitonic & merging networks (parallel sorting)

One of my favorite motivating examples:

> Given an array $A[1, \dots, n]$ of real numbers. Report the largest sum of numbers in a (contiguous) sub-array of $A$

(If all elements are negative, report $0$: the sum of an *empty* sub-array)

# Sorting algorithms

Importance

- Many many practical applications require sorting
- Great problems to illustrate algorithm design and analysis techniques
- One of the very few problems which we can prove non-trivial lower-bound on running time

Classification

- In place: only a constant amount of extra memory needed
- Comparison based: only comparisons are used to gain order information

A few names

- Classic: insertion, merge, quick, shell, bucket, counting, radix
- More modern: no names yet

# On presenting an algorithm

1. Input & Output
2. A brief description of the idea
3. Pseudo code
4. Analysis of running time (worst case, average case, ...), memory usage, and possibly other practical measures

You will be asked to follow this convention in homework assignments

# Quicksort

- Input: array $A$, two indices $p, q$
- Output: same array with $A[p, \dots, q]$ sorted
- Idea: use divide & conquer
  - Divide: rearrange $A[p, \dots, q]$ such that for some $r$ in between $p$ and $q$,

$$
\begin{aligned}
A[i] &\leq A[r] \ \forall i = p, \dots, r-1 \\
A[r] &\leq A[j] \ \forall j = r+1, \dots, q
\end{aligned}
$$

    Compute $r$ as part of this step.
  - Conquer: Quicksort($A[p, \dots, r-1]$), and Quicksort($A[r+1, \dots, q]$)
  - Combine: Nothing

Note: I intentionally use different indices than in CLRS.

# Quicksort: Pseudo code

**Quicksort**$(A, p, q)$

**if** $p < q$ **then**
    $r \leftarrow \text{Partition}(A, p, q)$
    $\text{Quicksort}(A, p, r - 1)$
    $\text{Quicksort}(A, r + 1, q)$
**end if**

Let's "Analyze this"

Note:

- Robert de Niro was not the inventor of Quicksort,
- neither was Billy Crystal.
- Definitely not Al Gore (-ithm), the self-proclaimed "inventor" of the Internet

# The key: partitioning

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | i | p,j | | | | | | | q | | |
| ... | | 3 | 1 | 8 | 5 | 6 | 2 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p,i | j | | | | | | q | | |
| ... | | 3 | 1 | 8 | 5 | 6 | 2 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | i | j | | | | | q | | |
| ... | | 3 | 1 | 8 | 5 | 6 | 2 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | i | | j | | | | q | | |
| ... | | 3 | 1 | 8 | 5 | 6 | 2 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | i | | | j | | | q | | |
| ... | | 3 | 1 | 8 | 5 | 6 | 2 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | i | | | | j | | q | | |
| ... | | 3 | 1 | 8 | 5 | 6 | 2 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | | i | | | | j | q | | |
| ... | | 3 | 1 | 2 | 5 | 6 | 8 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | | i | | | | | q,j | | |
| ... | | 3 | 1 | 2 | 5 | 6 | 8 | 7 | 4 | | ... |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | | | i | | | | q,j | | |
| ... | | 3 | 1 | 2 | 4 | 6 | 8 | 7 | 5 | | ... |

# Partitioning: pseudo code

Partition around $A[q]$:

**Partition**$(A, p, q)$

$x \leftarrow A[q]$<br>
$i \leftarrow p - 1$<br>
**for** $j \leftarrow p$ **to** $q$ **do**<br>
    **if** $A[j] \leq x$ **then**<br>
        swap $A[i+1]$ and $A[j]$<br>
        $i \leftarrow i + 1$<br>
    **end if**<br>
**end for**<br>
return $i$

Question: how would you partition around $A[m]$ for some $m$: $p \leq m \leq q$ ?

Notes:

- Slightly different from the textbook. Idea is the same.
- $A[p], \dots, A[i] \leq x$
- $A[i+1], \dots, A[j-1] > x$
- $A[j], \dots, A[q-1]$: elements not examined yet

# Worst case running time

Let $T(n)$ be the worst-case running time of Quicksort.

It's easy to see that $T(n) = \Omega(n^2)$

We shall show $T(n) = O(n^2)$, implying $T(n) = \Theta(n^2)$.

$$T(n) = \max_{0 \leq r \leq n-1} (T(r) + T(n - r - 1)) + \Theta(n)$$

$T(n) = O(n^2)$ follows by induction.

# Informal analysis

**Worse-case partitioning**:

$$T(n) = T(n-1) + T(0) + \Theta(n) = T(n-1) + \Theta(n)$$

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

**Best-case partitioning**:

$$T(n) \approx 2T(n/2) + \Theta(n)$$

yielding $T(n) = O(n \lg n)$.

**Somewhat balanced partitioning**:

$$T(n) \approx T\left(\frac{n}{10}\right) + T\left(9\frac{n}{10}\right) + \Theta(n)$$

yielding $T(n) = O(n \lg n)$ (recursion-tree).

# Average-case running time: a sketch

*Claim.* The running time of Quicksort is proportional to the number of comparisons

Let $M_n$ be the expected number of comparisons (what's the sample space?).

Let $X$ be the random variable counting the number of comparisons.

$$\begin{aligned} M_n = E[X] &= \sum_{j=1}^{n} E[X \mid A[q] \text{ is the } j\text{th least number}]\frac{1}{n} \\ &= \frac{1}{n}\sum_{j=1}^{n}\big(n-1+M_{j-1}+M_{n-j}\big) \\ &= n-1+\frac{2}{n}\sum_{j=0}^{n-1}M_j \end{aligned}$$

Hence,

$$M_n = \frac{2(n-1)}{n} + \frac{n+1}{n}M_{n-1},$$

which yields $M_n = \Theta(n \lg n)$.

# Randomized Quicksort

**Randomized-Quicksort**$(A, p, q)$

**if** $p < q$ **then**
    $r \leftarrow$ Randomized-Partition$(A, p, q)$
    Randomized-Quicksort$(A, p, r - 1)$
    Randomized-Quicksort$(A, r + 1, q)$
**end if**

**Randomized-Partition**$(A, p, q)$

pick $m$ at random between $p, q$
swap $A[m]$ and $A[q]$
// The rest is unchanged
$x \leftarrow A[q]$
$i \leftarrow p - 1$
**for** $j \leftarrow p$ **to** $q$ **do**
    **if** $A[j] \leq x$ **then**
        swap $A[i + 1]$ and $A[j]$
        $i \leftarrow i + 1$
    **end if**
**end for**
return $i$

Question: why bother?

# Basic concepts from Probability Theory - a quick reminder

(Please also scan Chapter 5 and Appendix C1-4.)

- Sample space $S$, e.g. all pairs of outcomes when rolling 2 dice
- $E \subseteq S$ are events, e.g. the event that the first die is $4$
- $E, F$ are events, then $E \cap F$ and $E \cup F$ are events
- A function $Pr$ assigning each event a number in $[0, 1]$
- Probability of an event $E$ is $Pr(E)$
- $Pr$ must satisfy
  - $0 \leq Pr(E) \leq 1, \forall E \subseteq S$
  - $P(S) = 1$
  - If $\{E_i \mid i \in I\}$ is mutually exclusive, then

$$Pr\left(\cup_{i \in I} E_i\right) = \sum_{i \in I} Pr(E_i)$$

For example, let $E_i, i \in \{1, 2, \dots, 6\}$ be the events that the first die gives $i$, then the $E_i$ are mutually exclusive

# Conditional Probability

Probability of $E$ given $F$ is

$$Pr[E \mid F] = \frac{Pr(E \cap F)}{Pr(F)}$$

(Note: this is only valid when $Pr(F) = 0$.)

Example:

- $E$ the event the first die is $4$
- $F$ the event the second die is $4$
- Intuitively, what's $Pr(E \mid F)$ ?
- Does the definition of conditional probability agree with your intuition?

# Random variables

- We're often more interested in some function on events
- E.g., what's the probability that the sum of two dice is 7?
- These quantities of interest are random variables
- Formally, a random variable $X$ is a function:

$$X : 2^S \to \mathbb{R}$$

  which means each event $E$ has a real value assigned by $X$

Example:

- $X$ the number of coin tosses until a head turns up
- What's the probability that $X = 5$?
- What's the probability that $X = k$?

A random variable $X$ is discrete if it only takes countably many values.

$X$ and $Y$ are independent if for all $a$, $b$,

$$Pr[X \leq a, Y \leq b] = Pr[X \leq a]Pr[Y \leq b]$$

# Expectation

The expected value of a discrete random variable $X$ is:

$$E[X] = \sum_a a Pr[X = a]$$

where the sum ranges over all possible values of $X$.

Example:

- Roll a fair die
- Let $X$ be the number on the face up
- Then

$$\begin{aligned} E[X] &= 1\frac{1}{6} + 2\frac{1}{6} + 3\frac{1}{6} + 4\frac{1}{6} + 5\frac{1}{6} + 6\frac{1}{6} \\ &= 3.6777 \end{aligned}$$

# Further properties

*Linearity of expectation*

$$E[a_1X_1 + \cdots + a_nX_n] = a_1E[X_1] + \cdots + a_nE[X_n]$$

When $X$ and $Y$ are independent,

$$E[XY] = E[X]E[Y]$$

# Conditional Probability and Expectation

Definitions:

$$Pr[X = a \mid Y = b] := \frac{Pr[X = a, Y = b]}{Pr[Y = b]}$$

$$E[X \mid Y = b] := \sum_a a Pr[X = a \mid Y = b]$$

Properties:

$$Pr[X = a] = \sum_b Pr[X = a \mid Y = b] Pr[Y = b]$$

$$E[X] = \sum_b E[X \mid Y = b] Pr[Y = b]$$

Note: these only hold for discrete random variables

# Average-case analysis of Quicksort revisited

(See the textbook for another way to do this analysis.)

Let $M_n$ be the expected number of comparisons.

Let $X$ be the random variable counting the number of comparisons.

$$\begin{aligned} M_n = E[X] &= \sum_{j=1}^{n} E[X \mid A[q] \text{ is the } j\text{th least number}]\frac{1}{n} \\ &= \frac{1}{n}\sum_{j=1}^{n}\bigl(n-1+M_{j-1}+M_{n-j}\bigr) \\ &= n-1+\frac{2}{n}\sum_{j=0}^{n-1} M_j \end{aligned}$$

Hence,

$$M_n = \frac{2(n-1)}{n} + \frac{n+1}{n}M_{n-1},$$

which yields $M_n = \Theta(n \lg n)$.

# The selection problem

- The $i$th order statistic of a set of $n$ numbers is the $i$th smallest number
- The median is the $\lfloor n/2 \rfloor$th order statistic
- Selection problem: find the $i$th order statistic as fast as possible

Examples:

- Conceivable that the running time is proportional to the number of comparisons
- Find a way to determine the 2nd order statistic using as few comparisons as possible
- How about the 3rd order statistic?

# Randomized selection

- Input: $A[p, \dots, q]$ and $i$, $1 \leq i \leq q - p + 1$
- Output: the $i$th order statistic of $A[p, \dots, q]$
- Idea: use "Partition" from Quicksort

  If "Partition" return the right O-STAT, then accept it

  If not, go left or right correspondingly

# Randomized selection: pseudo code

**Randomized-Select**$(A, p, q, i)$

$r \leftarrow$ Partition$(A, p, q)$<br>
$k = r - p + 1$ // the O-STAT order of $A[r]$<br>
**if** $i = k$ **then**<br>
    return $A[r]$<br>
**end if**<br>
**if** $i < k$ **then**<br>
    return Randomized-Select$(A, p, r - 1, i)$<br>
**else**<br>
    return Randomized-Select$(A, r + 1, q, i - r)$<br>
**end if**

Of course, we could replace "Partition" by "Randomized-Partition"

# Randomized-Selection: Analysis

- Let $X_k$ be the indicator that $A[r]$ is the $k$th O-STAT
- Let $T(n)$ be the expected running time

Then,

$$\begin{aligned} T(n) &\leq \sum_{k=1}^{n} X_k \left(T(\max(k-1, n-k)) + O(n)\right) \\ &= \sum_{k=1}^{n} \left(X_k T(\max(k-1, n-k))\right) + O(n) \end{aligned}$$

Hence,

$$\begin{aligned} E[T(n)] &\leq E\left[\sum_{k=1}^{n} X_k T(\max(k-1, n-k))\right] + O(n) \\ &= \sum_{k=1}^{n} E[X_k] E[T(\max(k-1, n-k))] + O(n) \end{aligned}$$

Consequently,

$$E[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n} E[T(k)] + O(n)$$

# Selection in Worst-case Linear Time

- Input: $A[p, \dots, q]$ and $i$, $1 \leq i \leq q - p + 1$
- Output: the $i$th order statistic of $A[p, \dots, q]$
- Idea: same as Randomized-Selection, but also try to guarantee a good split.

  Find $A[m]$ which is not too far left nor too far right

  Then, split around $A[m]$

The idea is from the following paper:

Manuel Blum, Vaughan Pratt, Robert E. Tarjan, Robert W. Floyd, and Ronald L. Rivest, **"Time bounds for selection."** Fourth Annual ACM Symposium on the Theory of Computing (Denver, Colo., 1972). Also, J. Comput. System Sci. 7 (1973), 448–461.

# Linear-Selection: Pseudo-Pseudo Code

**Linear-Select**$(A, i)$

```
"Divide" n elements into ⌈n/5⌉ groups,
    • ⌊n/5⌋ groups of size 5, and
    • ⌈n/5⌉ − ⌊n/5⌋ group of size n − 5⌊n/5⌋
Find the median of each group
Find x: the median of the medians by calling Linear-Select
recursively
Swap A[m] with A[n], where A[m] = x
r ← Partition(A, 1, n)
if r = i then
    return A[r]
else
    recursively go left or right accordingly
end if
```

# Linear-Select: Analysis

- $T(n)$ denotes running time
- Lines 1 & 2: $\Theta(n)$
- Line 3: $T(\lceil \frac{n}{5} \rceil)$
- Lines 4, 5: $\Theta(n)$
- Lines 6-10: at most $T(f(n))$, where $f(n)$ is the larger of two numbers:
  - number of elements to the left of $A[r]$,
  - number of elements to the right of $A[r]$

$f(n)$ could be shown to be at most $\frac{7n}{10} + 6$, hence

$$T(n) \leq \begin{cases} \Theta(1) & \text{if } n \leq 71 \\ T(\lceil \frac{n}{5} \rceil) + T(\lfloor \frac{7n}{10} + 6 \rfloor) + \Theta(n) & \text{if } n > 71 \end{cases}$$

Induction gives $T(n) = O(n)$

# Long integer multiplication

- Let $i$ and $j$ be two $n$-bit integers, $n$ is huge, compute $ij$.
- Straightforward multiplication takes $\Theta(n^2)$ (please convince yourself of this fact)
- Naive D&C:

$$\begin{aligned} i &= a2^{2n/3} + b2^{n/3} + c \\ j &= x2^{2n/3} + y2^{n/3} + z \end{aligned}$$

Hence,

$$\begin{aligned} ij = ax2^{4n/3} + (ay + bx)2^{n} + \\ (az + cx + by)2^{2n/3} + (bz + cy)2^{n/3} + cz. \end{aligned} \quad (1)$$

Naive D&C gives

$$T(n) = 9T(n/3) + \Theta(n),$$

which is again $\Theta(n^2)$ by Master Theorem.

Note: *addition and shifting take* $\Theta(n)$, *hence we want to reduce the number of (recursive) multiplications*

# Smart D&C

Want only 5 terms: $ax$, $ay + bx$, $az + cx + by$, $bz + cy$, $cz$.

$$
\begin{aligned}
p_1 &= (a+b)(x+y) = (\mathbf{ay} + \mathbf{bx}) + ax + by \\
p_2 &= (b+c)(y+z) = (\mathbf{bz} + \mathbf{cy}) + by + cz \\
p_3 &= (a+c)(x+z) = (\mathbf{az} + \mathbf{cx} + \mathbf{by}) + ax + cz - by \\
p_4 &= ax \\
p_5 &= by \\
p_6 &= cz
\end{aligned}
$$

$$
\begin{aligned}
ax &= p_4 \\
ay + bx &= p_2 - p_4 - p_5 \\
az + cx + by &= p_3 - p_4 - p_6 + p_5 \\
bz + cy &= p_2 - p_5 - p_6 \\
cz &= p_6
\end{aligned}
$$

$$
\begin{aligned}
ij = {} & p_4 2^{4n/3} + (p_2 - p_4 - p_5)2^n + \\
& (p_3 - p_4 - p_6 + p_5)2^{2n/3} + (p_2 - p_5 - p_6)2^{n/3} + p_6. \qquad (2)
\end{aligned}
$$

$$T(n) = 6T(n/3) + \Theta(n)$$

Hence, $T(n) = \Theta(n^{\log_3 6}) = o(n^2)$.

# Questions

- What if $n$ is not an exact power of $3$? What's the running time in terms of $n$ in that case?
- Would dividing things into $2$ pieces work?
- Would dividing things into $4$ pieces work?
- Would they be better than $3$?

# Matrix Multiplication

- $X$ and $Y$ are two $n \times n$ matrices. Compute $XY$.
- Straightforward method takes $\Theta(n^3)$.
- Naive Divide & Conquer:

$$\begin{aligned} XY &= \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} S & T \\ U & V \end{bmatrix} \\ &= \begin{bmatrix} AS+BU & AT+BV \\ CS+DU & CT+DV \end{bmatrix} \end{aligned}$$

$$T(n) = 8T(n/2) + \Theta(n^2)$$

Thus,

$$T(n) = \Theta(n^3)$$

# Smart D&C: Strassen Algorithm

Idea: somehow reduce the number of multiplications to be less than 8. E.g., $T(n) = 7T(n/2) + \Theta(n^2)$ gives

$$T(n) = n^{\log_2 7} = o(n^3)$$

Want: 4 terms (in lower-case letters for easy reading)

$$as + bu$$

$$at + bv$$

$$cs + du$$

$$ct + dv$$

# Seven

(Again, this is Strassen's algo, not Brat Pitt's)

$$
\begin{aligned}
p_1 &= (a-c)(s+t) &&= \mathbf{as} + \mathbf{at} - \mathbf{cs} - \mathbf{ct} \\
p_2 &= (b-d)(u+v) &&= \mathbf{bu} + \mathbf{bv} - \mathbf{du} - \mathbf{dv} \\
p_3 &= (a+d)(s+v) &&= \mathbf{as} + \mathbf{dv} + av + ds \\
p_4 &= a(t-v) &&= \mathbf{at} - av \\
p_5 &= (a+b)v &&= \mathbf{bv} + av \\
p_6 &= (c+d)s &&= \mathbf{cs} + ds \\
p_7 &= d(u-s) &&= \mathbf{du} - ds
\end{aligned}
$$

Write the following in terms of the $p_i$'s. Homework 2!

$$
\begin{aligned}
as + bu &= ??? \\
at + bv &= ??? \\
cs + du &= ??? \\
ct + dv &= ???
\end{aligned}
$$