Divide-and-Conquer

Inge Li Gørtz

Thank you to Kevin Wayne for inspiration to slides

Divide-and-Conquer

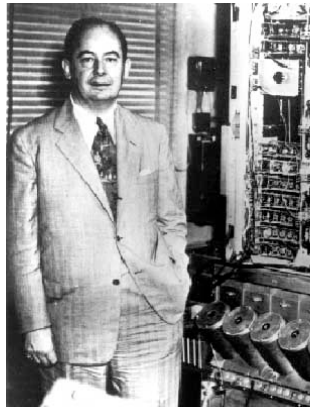
- Divide -and-Conquer.
  - Break up problem into several parts.
  - Solve each part recursively.
  - Combine solutions to subproblems into overall solution.

- Today
  - Mergesort (recap)
  - Recurrence relations
  - Integer multiplication

Mergesort

Mergesort

- Mergesort.
  - Divide array into two halves.
  - Recursively sort each half.
  - Merge two halves to make a sorted whole.



Jon von Neumann (1945)

A	L	G	O	R	I	T	H	M	S

Divide

A	L	G	O	R

I	T	H	M	S

Sort recursively

A	G	L	O	R

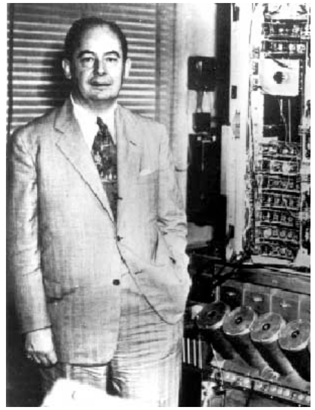
H	I	M	S	T

Merge

A	G	H	I	L	M	O	R	S	T

Mergesort

- Mergesort.
  - Divide array into two halves.
  - Recursively sort each half.
  - Merge two halves to make a sorted whole.
- T(n) = running time of mergesort on input of size n



Jon von Neumann (1945)

`A L G O R I T H M S`

`A L G O R`   `I T H M S`   Divide   O(1)

`A G L O R`   `H I M S T`   Sort recursively   2T(n/2)

`A G H I L M O R S T`   Merge   O(n)

Recurrence relations

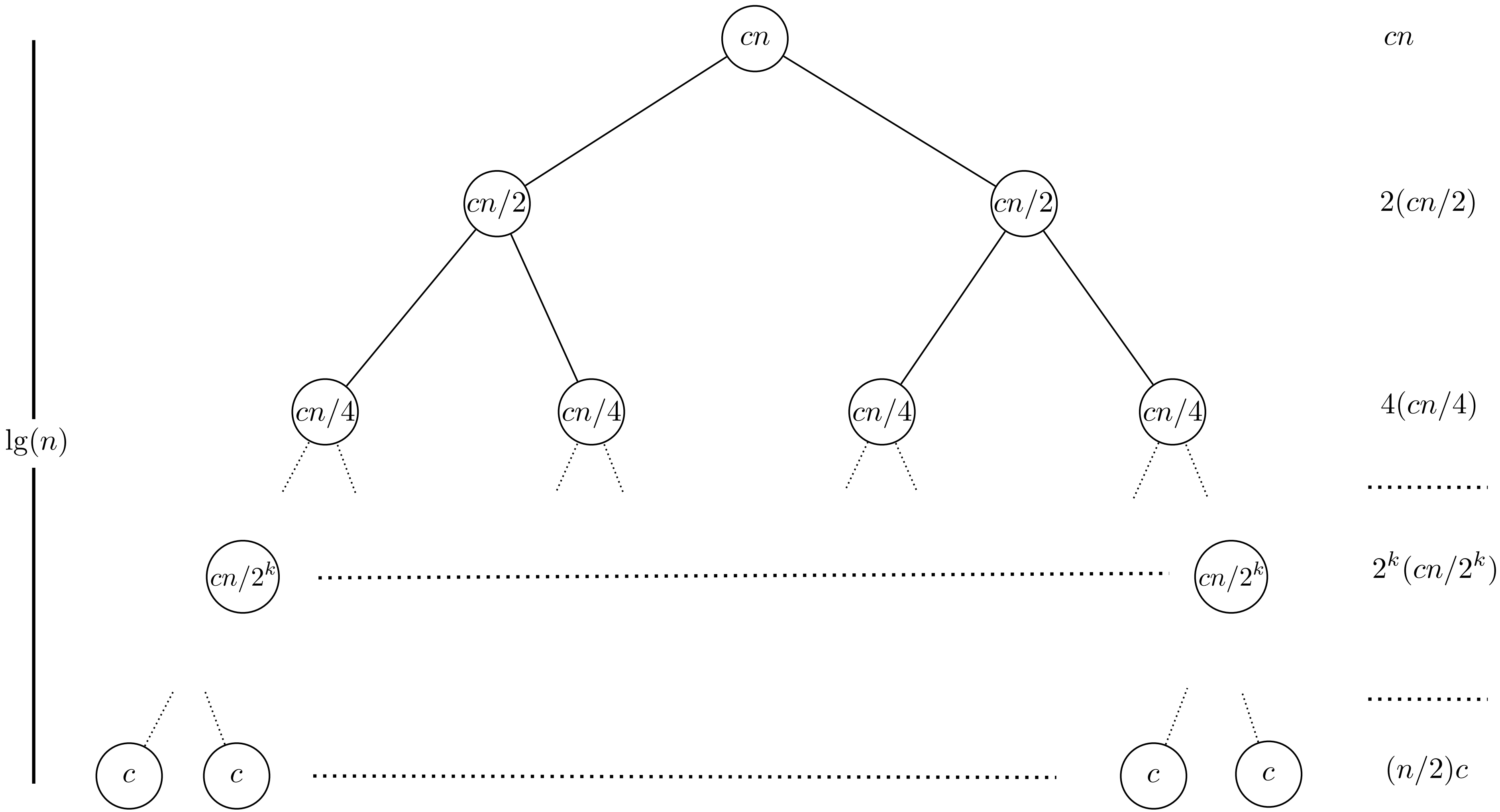
- T(n) = running time of mergesort on input of size n
- Mergesort recurrence:

$$T(n) \leq \begin{cases} 2T(n/2) + cn & \text{if } n > 2 \\ c & \text{otherwise} \end{cases}$$

- Solving the recurrence:
  - Recursion tree
  - Substitution

Mergesort recurrence: recursion tree

$$T(n) \leq \begin{cases} 2T(n/2) + cn & \text{if } n > 2 \\ c & \text{otherwise} \end{cases}$$



Mergesort recurrence: substitution

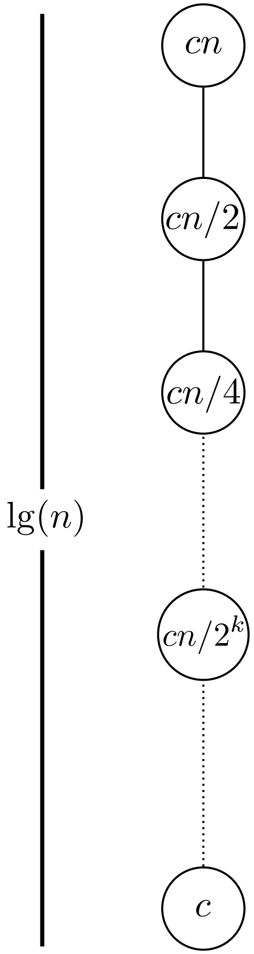
$$T(n) \leq \begin{cases} 2T(n/2) + cn & \text{if } n > 2 \\ c & \text{otherwise} \end{cases}$$

- Substitute $T(n)$ with $cn \lg n$ and use induction to prove $T(n) \leq cn \lg n$.
- Base case ($n = 2$):
  - By definition $T(2) = c$.
  - Substitution: $cn \lg n = c \cdot 2 \lg 2 = 2c \geq c = T(2) = T(n)$
- Induction: Assume $T(m) \leq cm \lg m$ for $m < n$.

$$\begin{aligned} T(n) &\leq 2T(n/2) + cn \\ &\leq 2c(n/2)\lg(n/2) + cn \\ &= cn(\lg n - 1) + cn \\ &= cn \lg n - cn + cn \\ &= cn \lg n. \end{aligned}$$

More recurrence relations: 1 subproblem

$$T(n) \leq \begin{cases} T(n/2) + cn & \text{if } n > 2 \\ c & \text{otherwise} \end{cases}$$



- Summing over all levels:

$$T(n) \leq \sum_{k=0}^{\lg n - 1} \frac{cn}{2^k} = cn \sum_{k=0}^{\lg n - 1} \frac{1}{2^k} \leq 2cn = O(n)$$

- Substitution:
  - Base case:

$$2c \cdot 2 = 4c \geq c = T(2).$$

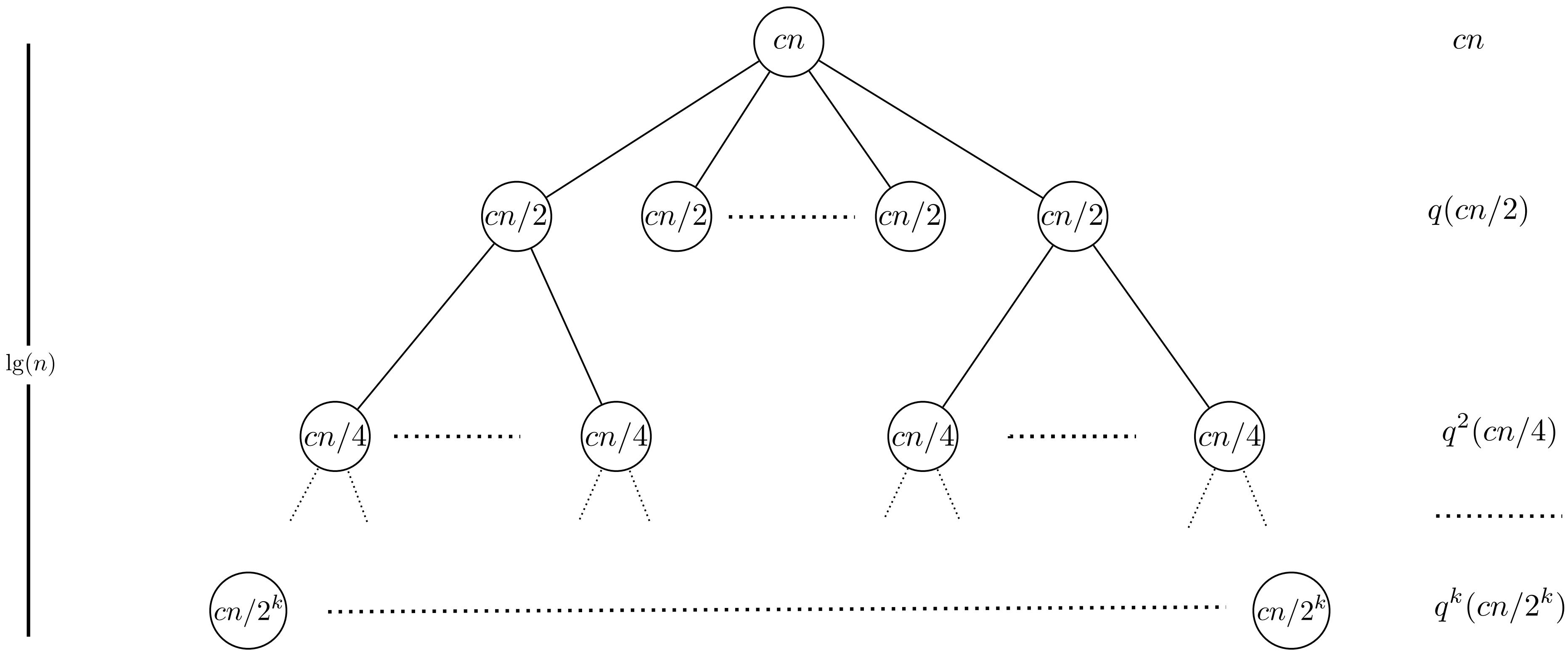
  - Assume $T(m) \leq 2cm$ for $m < n$.

$$T(n) \leq T(n/2) + cn \leq 2c(n/2) + cn = 2cn$$

More than 2 subproblems

- q subproblems of size n/2.

$$T(n) \leq \begin{cases} qT(n/2) + cn & \text{if } n > 2 \\ c & \text{otherwise} \end{cases}$$



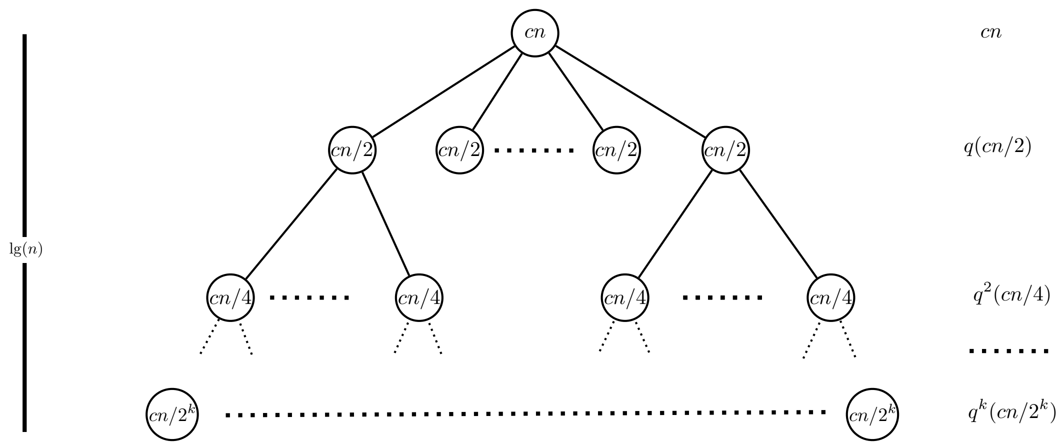
More than 2 subproblems

- q subproblems of size n/2.

$$T(n) \le \begin{cases} qT(n/2) + cn & \text{if } n > 2 \\ c & \text{otherwise} \end{cases}$$

- Summing over all levels:

$$T(n) \;\le\; \sum_{j=0}^{\lg n - 1} \left(\frac{q}{2}\right)^j cn \;=\; cn \sum_{j=0}^{\lg n - 1} \left(\frac{q}{2}\right)^j \;=\; O(n^{\lg q})$$



Geometric series.

for $x \neq 1 : \sum_{i=0}^{m} x^i = \frac{x^{m+1} - 1}{x - 1}$

for $x < 1 : \sum_{i=0}^{\infty} x^i = \frac{1}{1 - x}$

More than 2 subproblems

Proof of $cn \sum_{j=0}^{\lg n-1} \left(\frac{q}{2}\right)^j = O(n^{\lg q})$

Geometric series.

for $x \neq 1 : \sum_{i=0}^{m} x^i = \frac{x^{m+1}-1}{x-1}$

for $x < 1 : \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$

Use geometric series: $cn \sum_{j=0}^{\lg n-1} \left(\frac{q}{2}\right)^j = cn \frac{\left(\frac{q}{2}\right)^{\lg n} - 1}{\frac{q}{2} - 1}$

Reduce $\left(\frac{q}{2}\right)^{\lg n} = \frac{q^{\lg n}}{2^{\lg n}} = \frac{q^{\lg n}}{n^{\lg 2}} = \frac{q^{\lg n}}{n}$

Now:

$$cn \frac{\left(\frac{q}{2}\right)^{\lg n} - 1}{\frac{q}{2} - 1} = cn \frac{\frac{q^{\lg n}}{n} - 1}{\frac{q-2}{2}} = \frac{2c}{q-2} n \left(\frac{q^{\lg n}}{n} - 1\right) = \boxed{\frac{2c}{q-2}} \left(q^{\lg n} - n\right) = O(q^{\lg n})$$

constant

Integer Multiplication

Integer multiplication

- Add. Given two n-bit integers a and b, compute a + b.
- School method. $\Theta(n)$ bit operations.

1	0	1	1	1	
	1	0	0	1	1
+	1	0	1	1	1
1	0	1	0	1	0

- Multiply. Given two n-bit integers a and b, compute a × b.
- School method. $\Theta(n^2)$ bit operations.

1	1	0	×	1	1	1
				0	0	0
+			1	1	1	0
+		1	1	1	0	0
	1	0	1	0	1	0

Integer multiplication: warmup

- Divide-and-conquer: divide the n-bit integers into two.

$$x = \underbrace{1000}_{x_1}\underbrace{1101}_{x_0} \qquad y = \underbrace{1110}_{y_1}\underbrace{0001}_{y_0}$$

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

- First try:

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

  - Multiply four n/2-bit integers (recursively)
  - Add two n/2-bit integers
  - Shift and add to obtain result.

$$T(n) = \underbrace{4T(n/2)}_{\text{recursive calls}} + \underbrace{cn}_{\text{add, shift}}$$

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

Integer multiplication: Karatsuba

- Divide-and-conquer: divide the n-bit integers into two.

$$x = \underbrace{1000}_{x_1}\underbrace{1101}_{x_0} \qquad y = \underbrace{1110}_{y_1}\underbrace{0001}_{y_0}$$

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

$$\begin{aligned} x \cdot y &= 2^n \cdot x_1 y_1 + 2^{n/2} \cdot (x_1 y_0 + x_0 y_1) + x_0 y_0 \\ &= 2^n \cdot \underbrace{x_1 y_1}_{1} + 2^{n/2} \cdot \big(\underbrace{(x_1 + x_0)(y_1 + y_0)}_{2} - \underbrace{x_1 y_1}_{1} - \underbrace{x_0 y_0}_{3}\big) + \underbrace{x_0 y_0}_{3} \end{aligned}$$

- Karatsuba:
  - Recursively compute *three* products of n/2-bit integers:
    - $x_1 y_1,\ (x_1 + x_0)(y_1 + y_0),\ x_0 y_0$
  - Shift, add, and subtract to obtain result.

$$(x_1 + x_0)(y_1 + y_0) =$$
$$x_1 y_1 + x_1 y_0 + x_0 y_1 + x_0 y_0$$
$$\Rightarrow$$
$$x_1 y_0 + x_0 y_1 =$$
$$(x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0$$

$$T(n) = \underbrace{3T(n/2)}_{\text{recursive calls}} + \underbrace{cn}_{\text{add, shift}}$$

$$T(n) = O(n^{\lg 3}) = O(n^{1.59})$$