Divide and conquer algorithms

- binary search
- Merge Sort

Cormen et. al.
Introduction to Algorithms

$A[1..n]$ $\qquad$ $A[1]\ A[2]\ \dots\ A[n]$

$q := \left\lfloor \frac{n+1}{2} \right\rfloor$

$A[1..q]$ $\qquad$ $A[q+1..n]$ $\qquad$ Sort both subarrays recursively

sort $\qquad$ sort

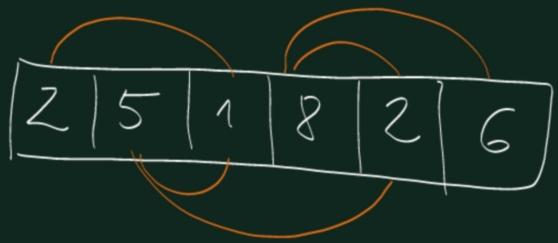
merge the sorted subarrays

$A[1..n]$

1. Count the number of inversions in an array $A[1..n]$ of natural numbers

Def.: $A[i]$ and $A[j]$ form an <u>inversion</u>

$i < j$ and $A[i] > A[j]$



Brute-force: check $(A[1], A[2]), (A[1], A[3]), \ldots$

$\ldots (A[1], A[n]), (A[2], A[3]), \ldots, (A[n-1], A[n])$

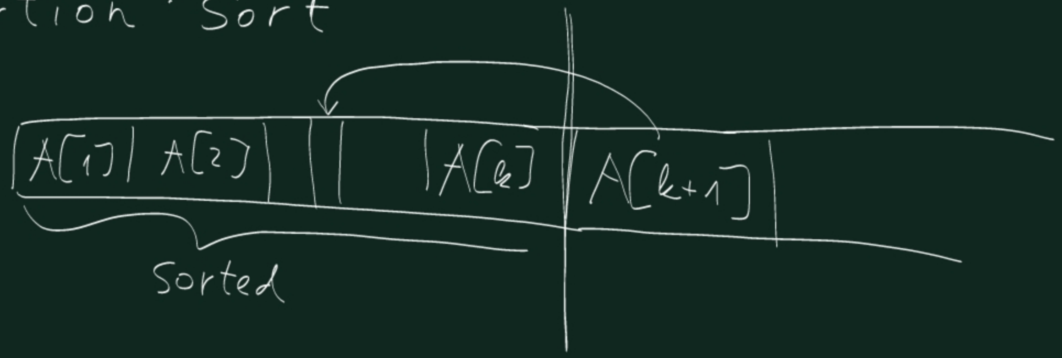
Check all pairs $\longrightarrow$ $\underbrace{n-1+n-2+\ldots+1}_{\frac{n(n-1)}{2}}$ pairs to check

$$\in O(n^2)$$

$$\in \Theta(n^2)$$

Idea: sort array $A[1..n]$ and count inversions along the way.

Insertion sort



Not much better, the time complexity of Insertion sort is also $O(n^2)$

Sort $A[1..n]$ with MergeSort and count inversions along the way.

$\longrightarrow$ next week

Time complexity : $T(n)$

for divide and conquer algorithms

$$T(n) = \boxed{a}\,T\left(\frac{n}{\boxed{b}}\right) + f(n)$$

→ divides the input into smaller sets of size $\frac{n}{\boxed{b}}$

→ and is recursively performed on $\boxed{a}$ of these

→ and the final calculations are done in $f(n)$ time

Binary Search :

$$T(n) = \begin{cases} T(\frac{n}{2}) + 1 & \text{if } n \geq 2 \\ 1 & \text{if } n = 1 \end{cases}$$

$\Theta(1)$, $c \leftarrow \text{const}$

$\Theta(n)$

$n \cdot c$

Merge Sort :

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + n & n \geq 2 \\ 1 & \text{if } n = 1 \end{cases}$$

We can assume that $n=2^k$

$\boxed{T(m)=2T\left(\frac{m}{2}\right)+m}$

$\boxed{T(n)=\Theta(n\log n)}$

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

$n\in\mathbb{N}$

$$=2\left(2T\left(\frac{n/2}{2}\right)+\frac{n}{2}\right)+n=2^2\boxed{T\left(\frac{n}{2^2}\right)}+2n=$$

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

$$=\ldots=2^k\cdot T\left(\frac{n}{2^k}\right)+k\cdot n=2^k\cdot T(1)+k\cdot n=n+kn=$$

$k=\log_2 n$

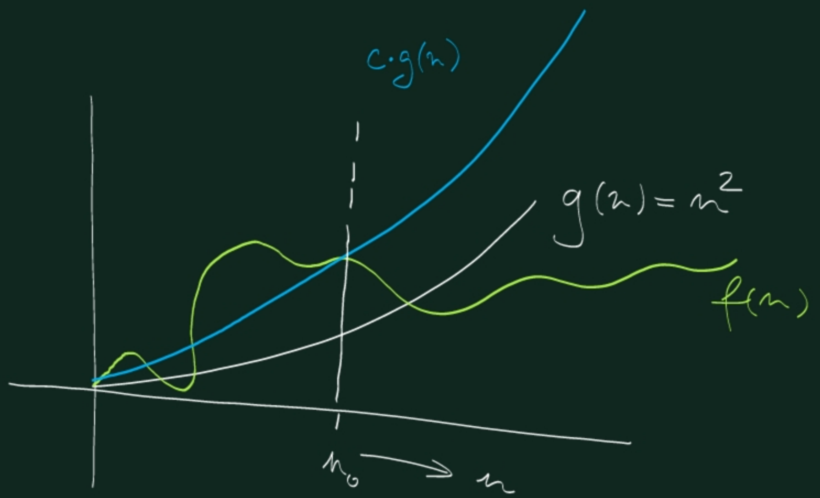
$\boxed{\begin{array}{c} n=2^k \\ T(1)=1 \end{array}}$

$$=n+n\cdot\log n\in O(n\log n)$$

$$\Theta(n\log n)$$

$$O(g) = \left\{ f : \text{there exist constants } c \text{ and } n_0 \text{ such that } f(n) \leq c \cdot g(n) \text{ for every } n \geq n_0 \right\}$$

g is a function



$O(n^2)$

$5n + 3 \in O(n^2)$ ✓

$5n + 3 \leq 5 n^2$

$n_0 = 2$ is good

$c = 5$ is good

$8n^2 + 120n \in O(n^2)$

$8n^2 + 120n \leq 200 n^2$

$n_0 = 1$ is good

$$5n+3 \in O(n)$$

$$5n+3 \le 8 \cdot n$$

$c=8$
$n_0=1$ is good

$$n + n\log n \in O(n\log n)$$

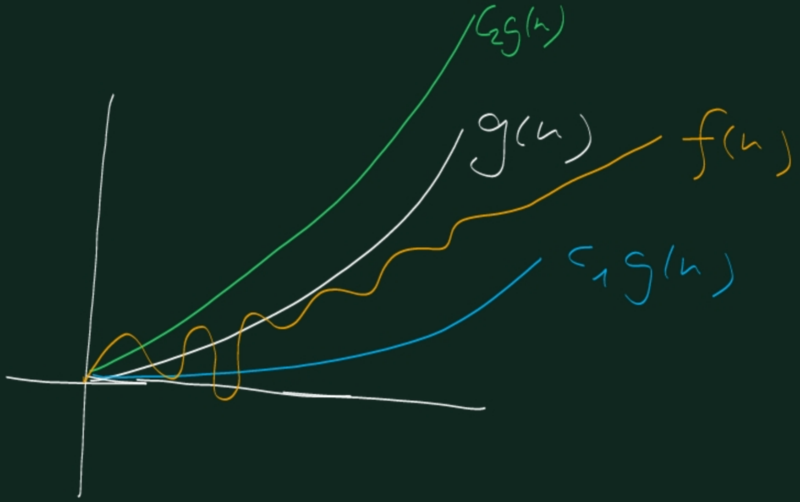
$$n+n\log n \le 2n\log n$$

$c=2$
$n_0=2$

$$n^7+8n^2 \in O(n^7)$$

$$n^7+8n^2 \in O(n^{25})$$

$$\Theta(g) = \left\{ f : \begin{array}{l} \text{there exist constants } c_1, c_2, n_0 \\ \text{such that} \\ \qquad c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \\ \text{for every } n \ge n_0 \end{array} \right\}$$



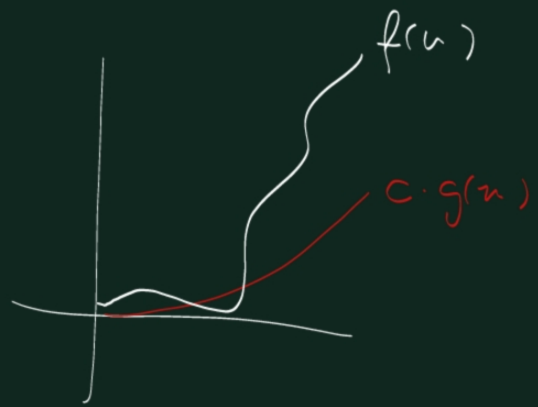
$$3n+5 \notin \Theta(n^2)$$

$$3n+5 \in \Theta(n)$$

$$n + n\log n \in \Theta(n \log n)$$

$$n \log n \le n + n \log n \le 2 \cdot n \cdot \log n$$

$\Omega(g) = \{ f :$ there exist constants c and n_0 such that $c\, g(n) \le f(n)$ for all $n \ge n_0 \}$



$$\boxed{\Theta(g) = O(g) \cap \Omega(g)}$$

We have proved that for $T(n) = 2T(\frac{n}{2}) + n \Longrightarrow T(n) = \Theta(n \log n)$

Similar calculations for $T(n) = 2T(\frac{n}{2}) + 1$ give $T(n) = \Theta(n)$

(HomeWork)

If the general case $\boxed{T(n) = aT(\frac{n}{b}) + f(n)}$ it is harder

$\longrightarrow$ Master theorem

Master Theorem

$$\boxed{T(n) = a\,T\left(\frac{n}{b}\right) + f(n)} \qquad \begin{matrix} a \geq 1 \\ b > 1 \end{matrix} \qquad \boxed{\boxed{n^{\log_b a}} \;?\; \boxed{f(n)}}$$

1. If $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ for some $\varepsilon > 0$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

2. If $f(n) = \Theta\left(n^{\log_b a}\right)$ then $T(n) = \Theta\left(n^{\log_b a} \cdot \log n\right)$

3. If $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ for some $\varepsilon > 0$ then $T(n) = \Theta\left(f(n)\right)$

   ↳ if $f(n)$ meets regularity condition

$$a f\left(\frac{n}{b}\right) \leq c \cdot f(n) \text{ for some } \boxed{c < 1}$$

1. $T(n) = 9 \cdot T\left(\frac{n}{3}\right) + n$

$a = 9$

$b = 3$

$\log_3 9 = 2$

$n^{\log_b a} = n^2$

$\boxed{n^2}$ $\boxed{f(n) = n}$

$f(n) = n \in O(n^{2-\varepsilon})$ for example $\varepsilon = \frac{1}{2}$ is good

We are in case 1. of MT

$\Longrightarrow T(n) = \Theta(n^2)$

2. $T(n) = T\left(\frac{2n}{3}\right) + 1$ $\longleftarrow$ $a = 1 \quad b = \frac{3}{2}$

case 2

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

case 3
(regularity)

4. $T(n) = 2T\left(n/2\right) + n\log n$

NEITHER