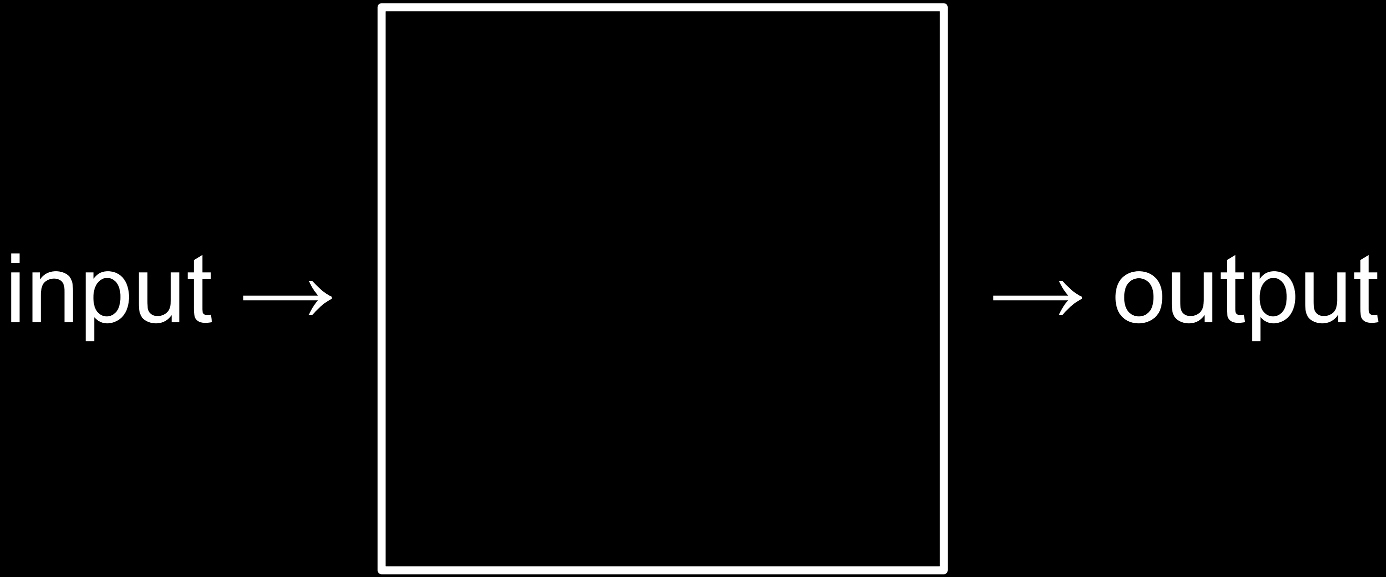
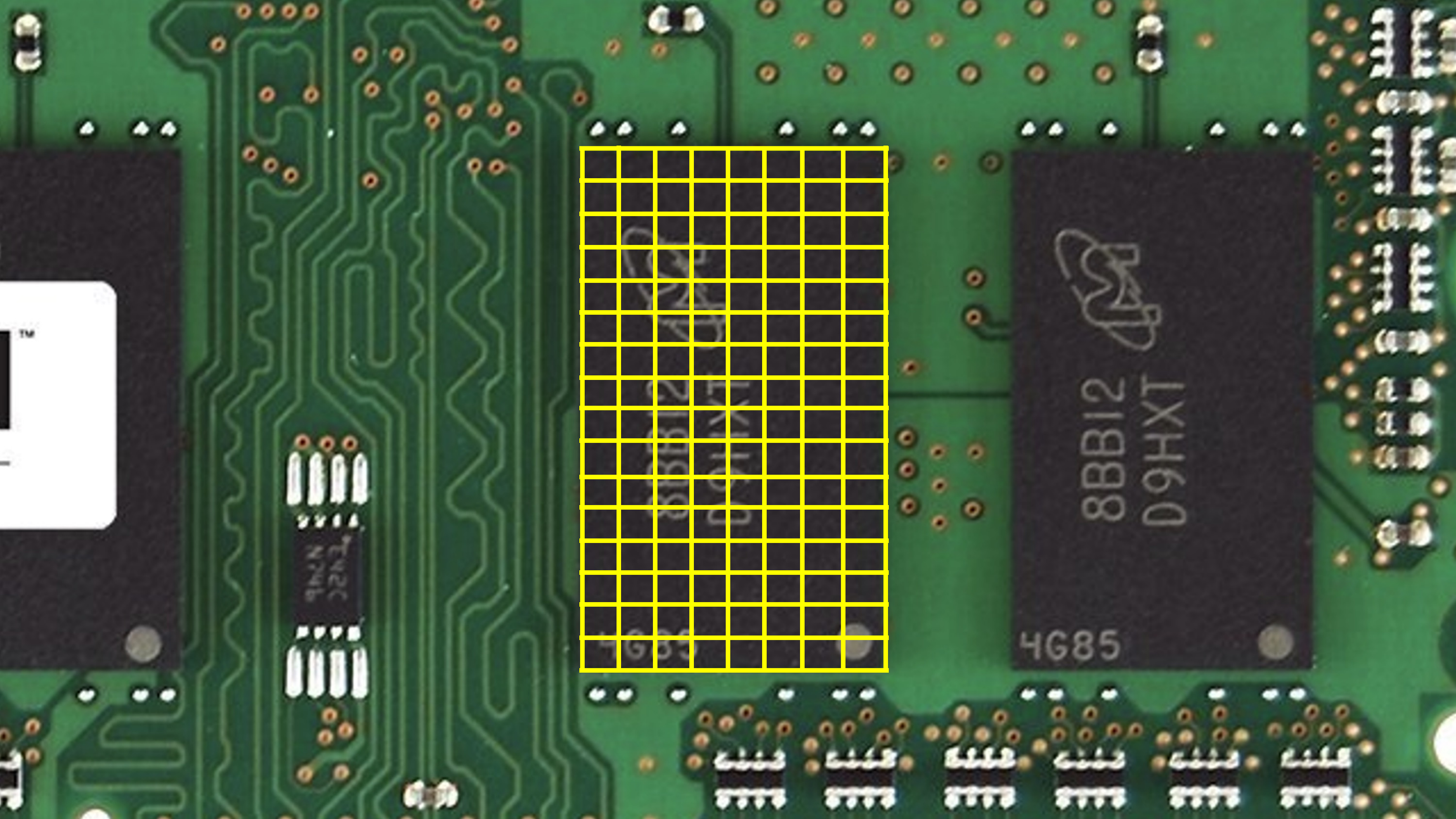
Introdução à Computação

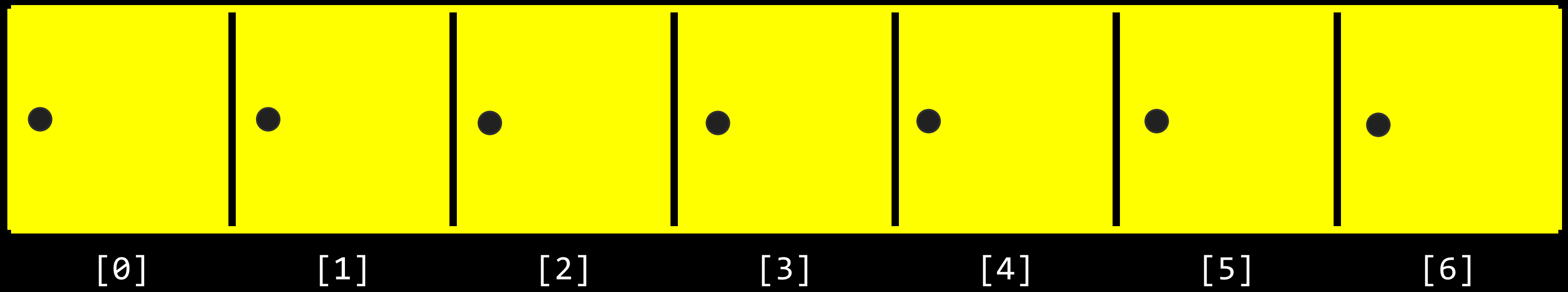


input →

→ output

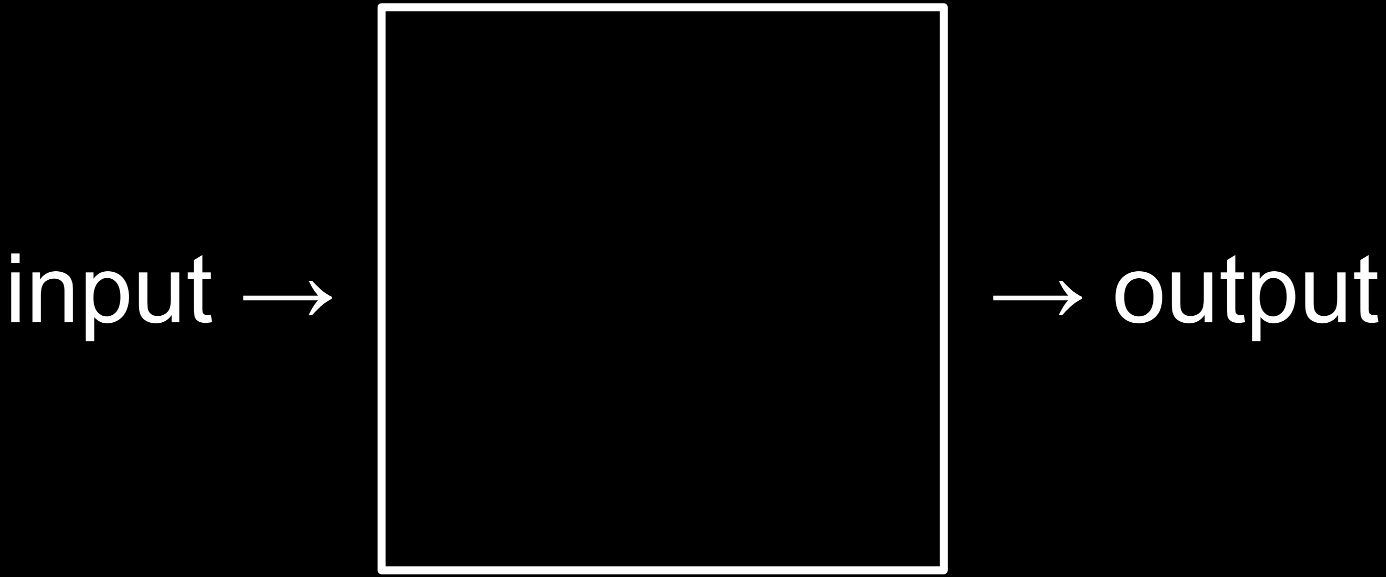

TM
8BB12
D9HXT
4G85
8BB12
D9HXT
4G85

4	6	8	2	7	5	0

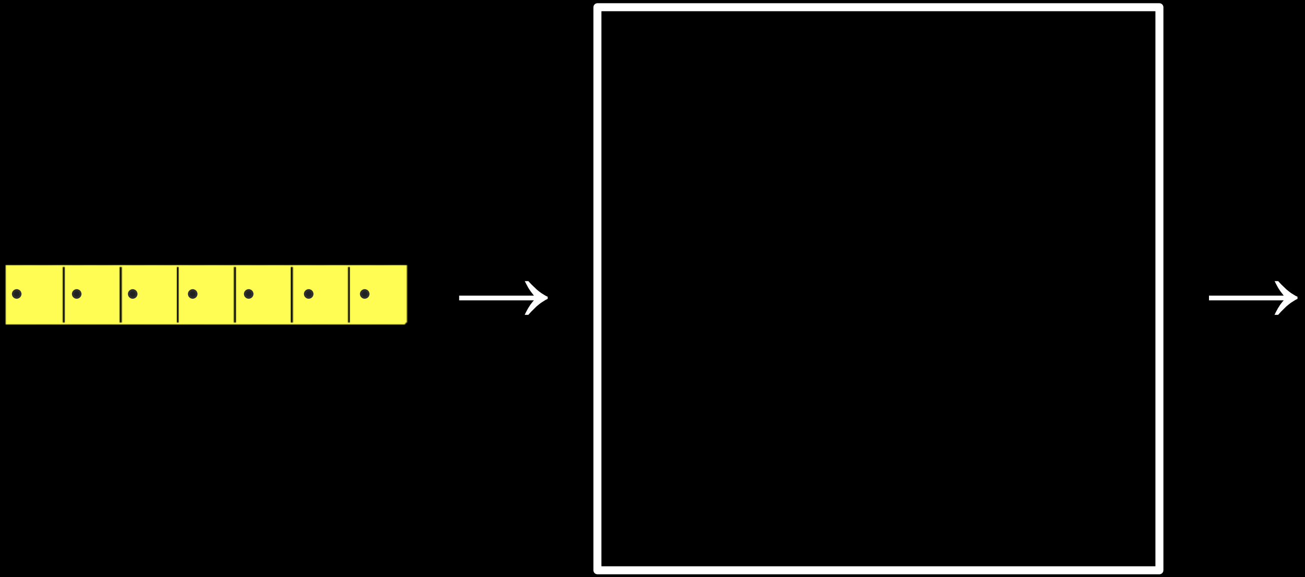


[0] [1] [2] [3] [4] [5] [6]

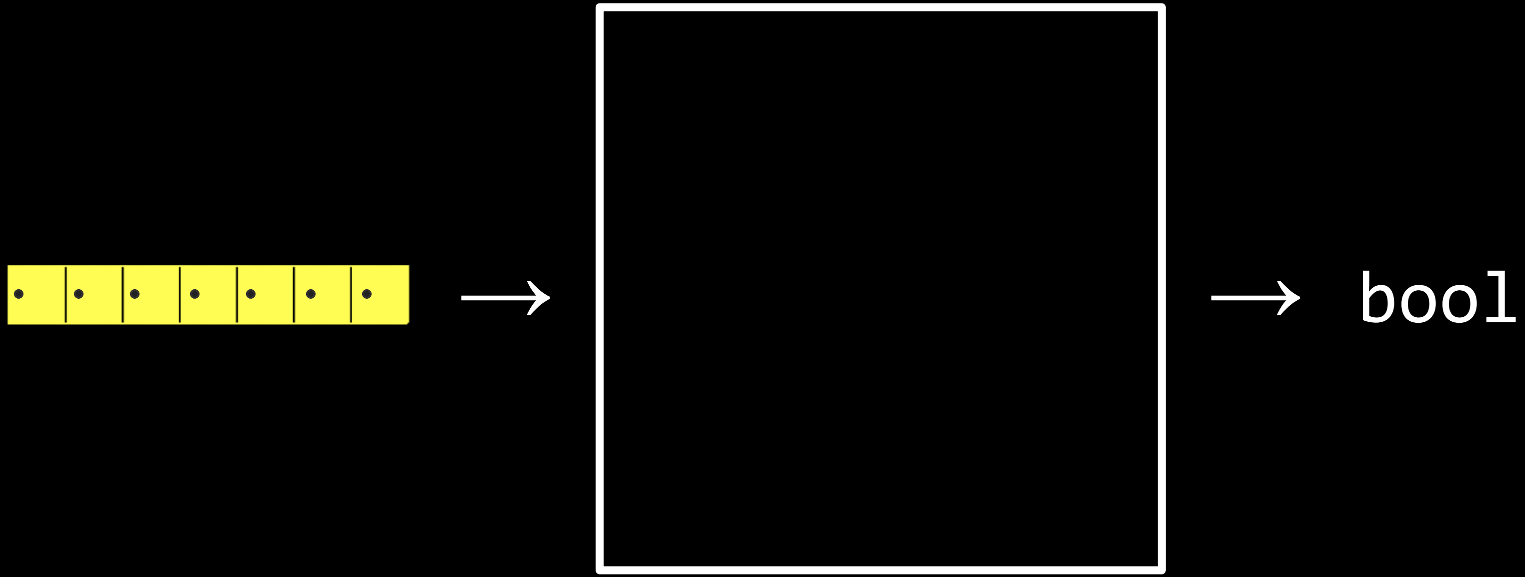
searching



input → → output



→ output

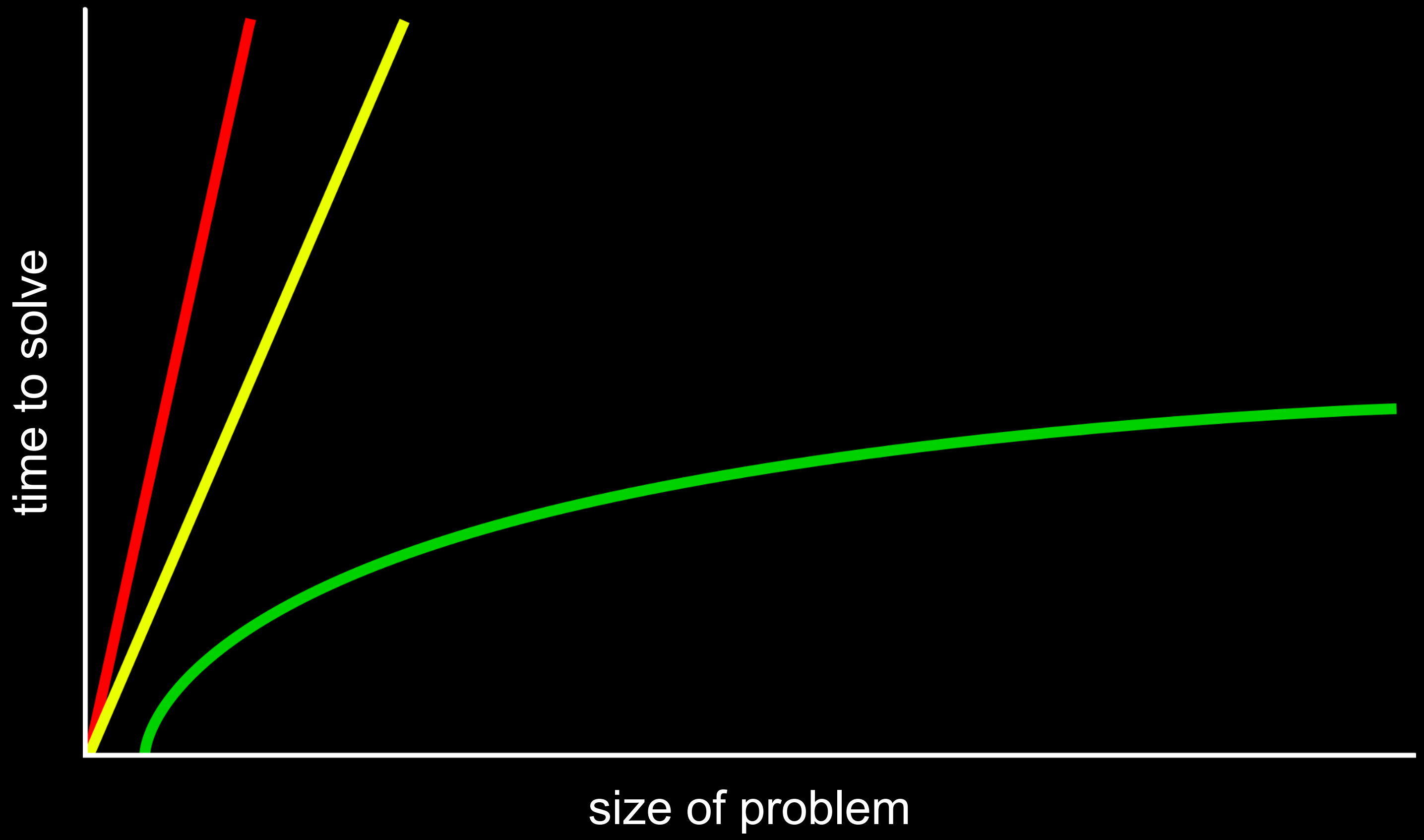


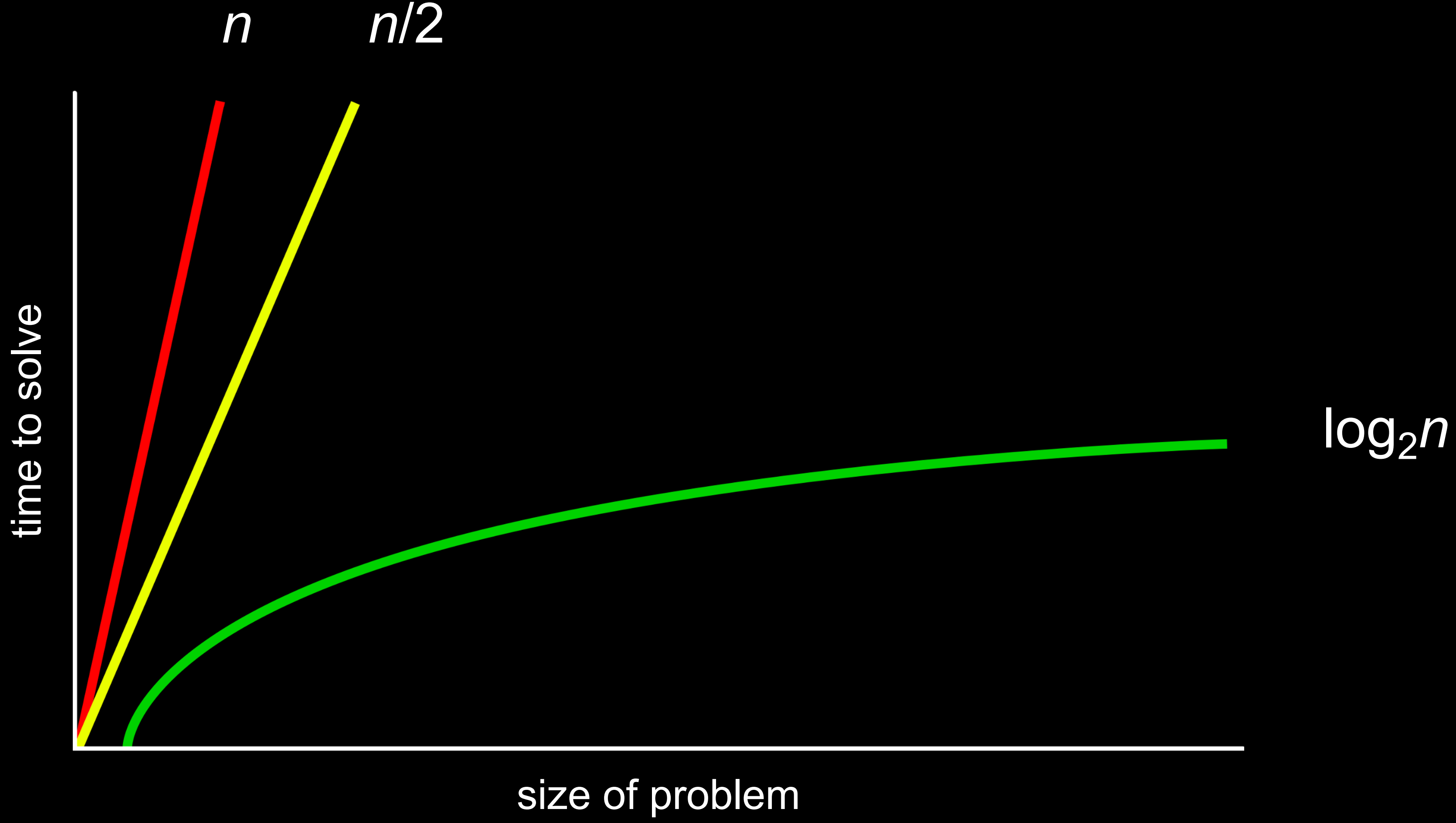
bool

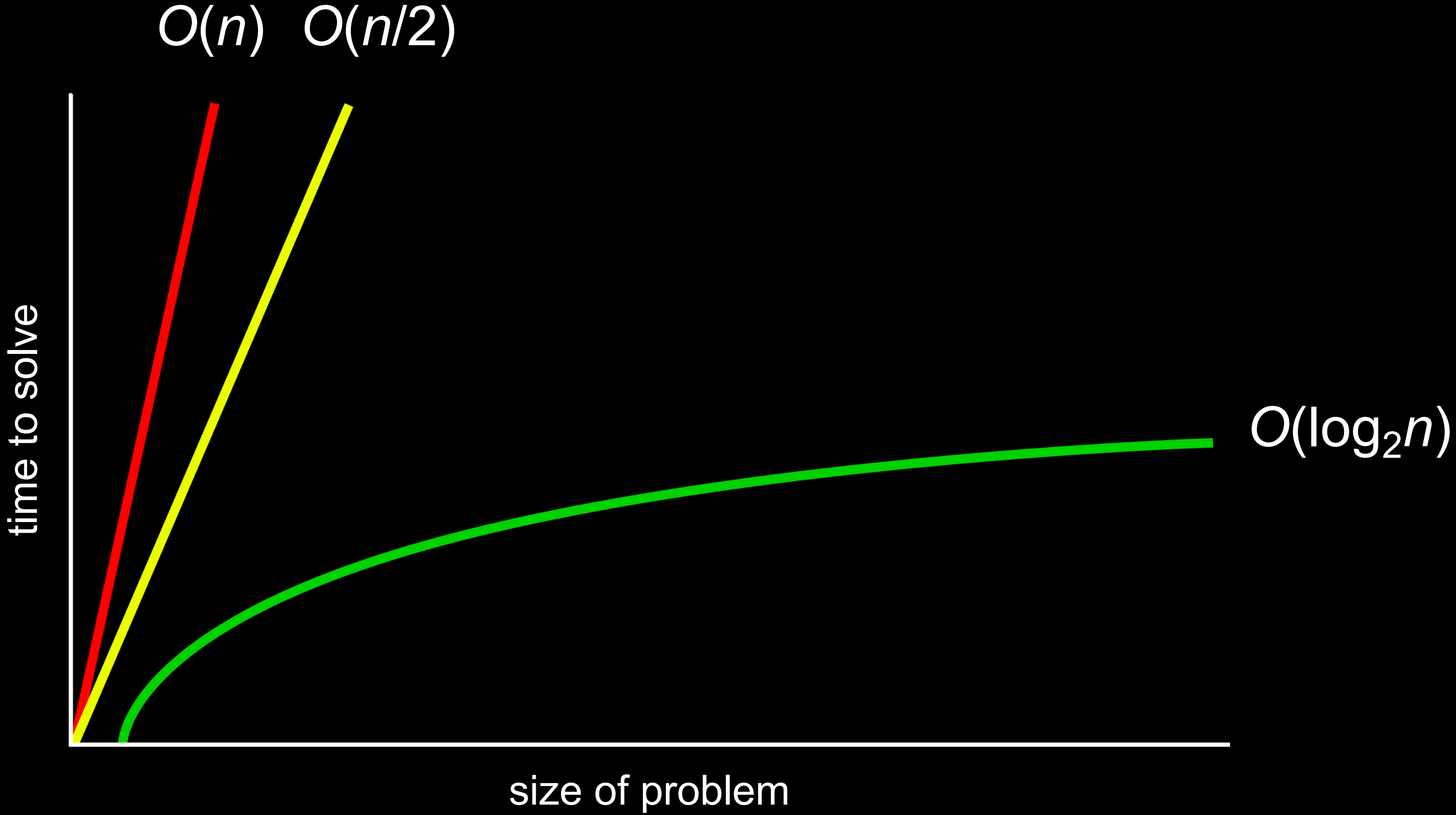
algorithms

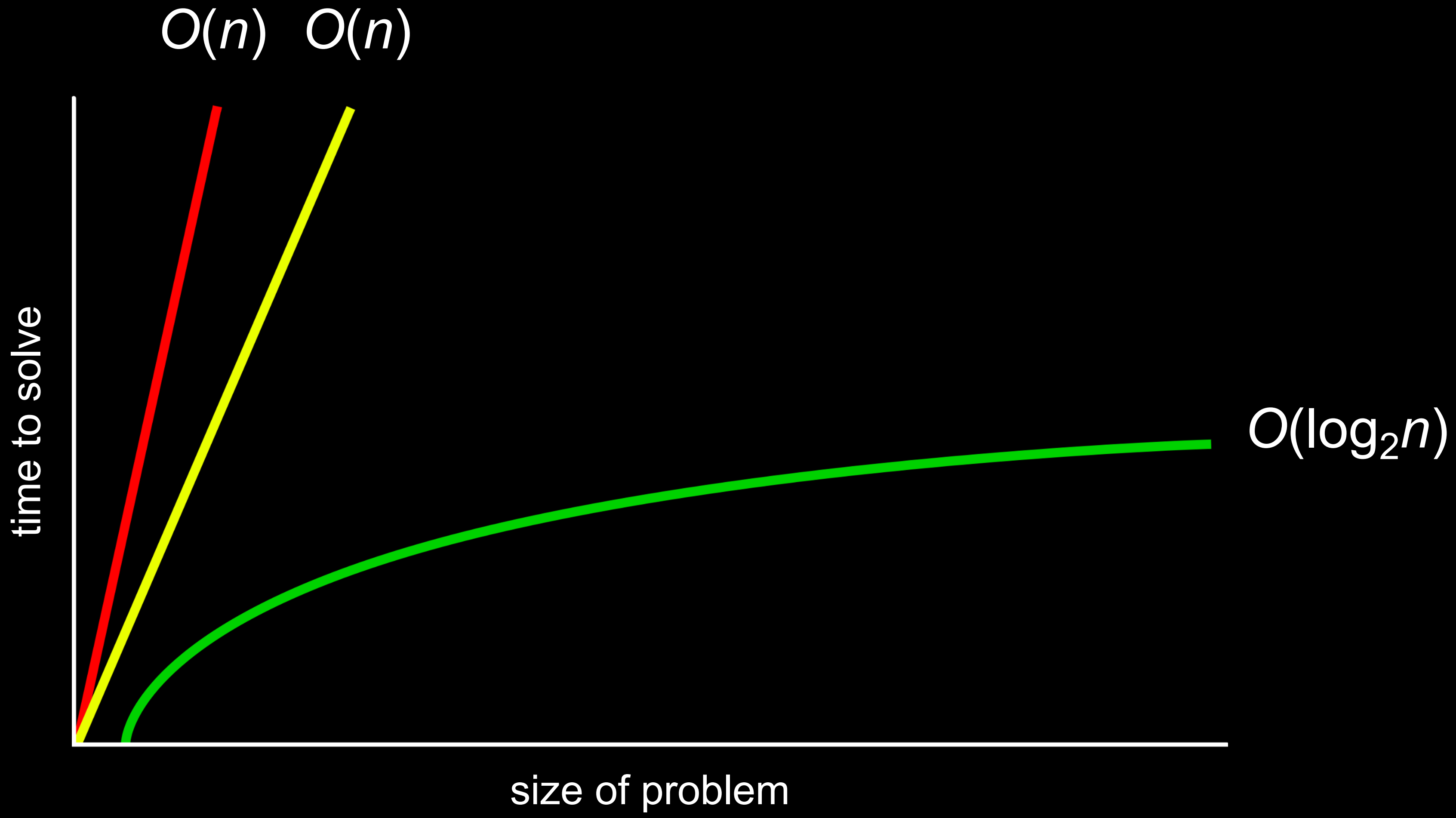
running times

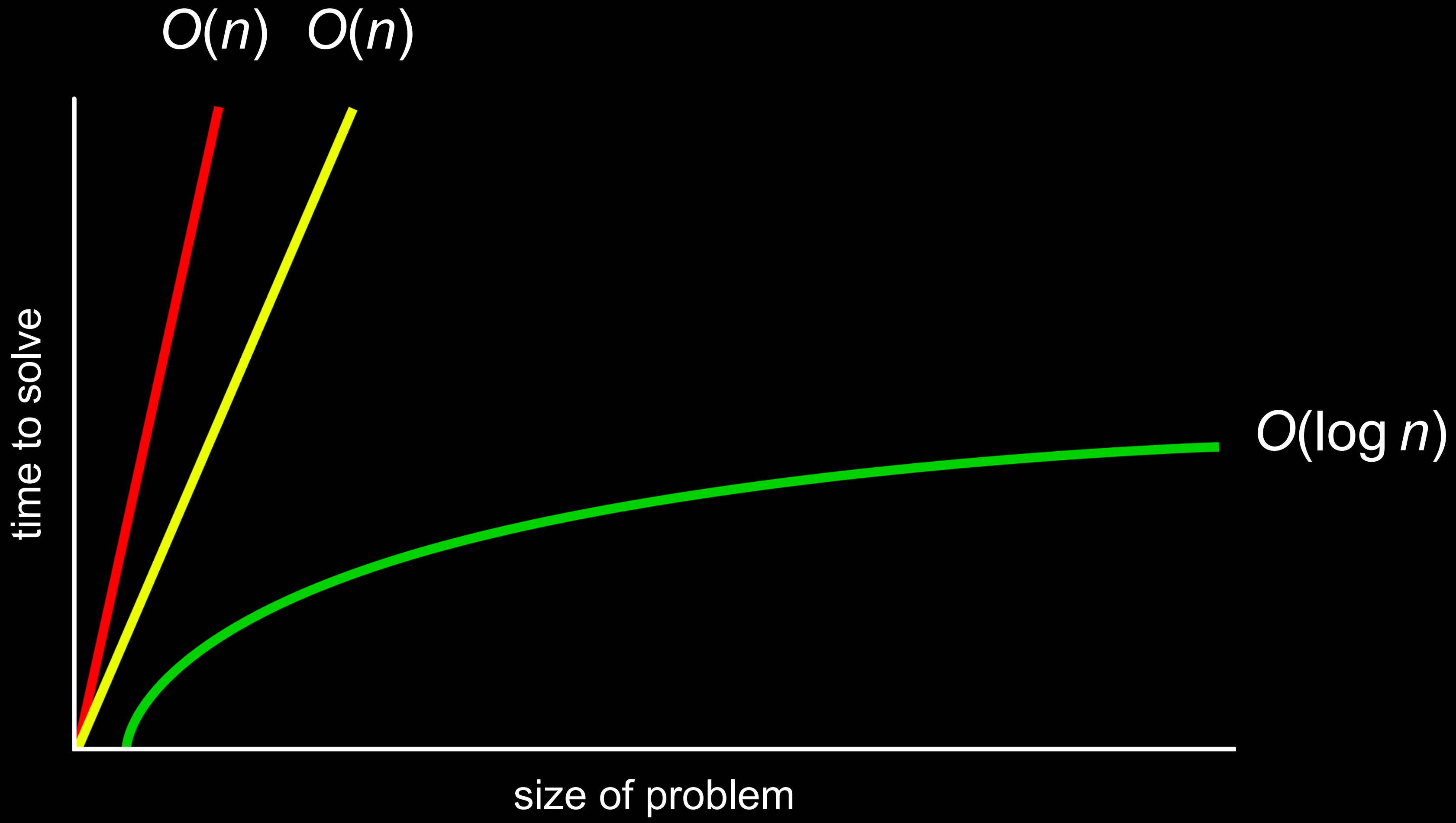
O

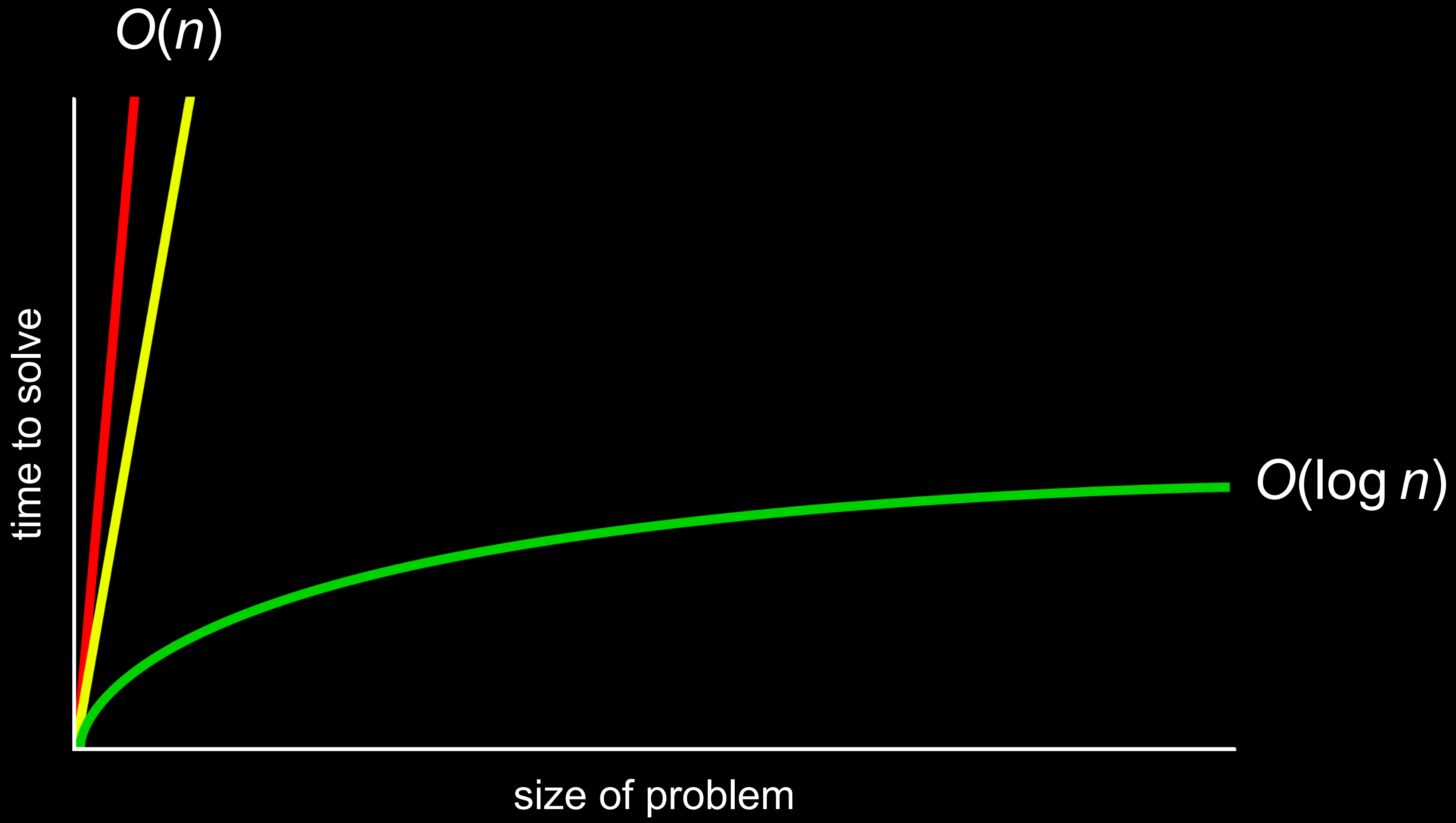

time to solve
size of problem


n
$n/2$
$\log_2 n$
time to solve
size of problem


$O(n)$
$O(n/2)$
$O(\log_2 n)$
time to solve
size of problem


$O(n)$
$O(n)$
$O(\log_2 n)$
time to solve
size of problem


$O(n)$
$O(n)$
$O(\log n)$
time to solve
size of problem


O(n)
O(log n)
time to solve
size of problem

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

Ω e Θ

linear search

```
For each door from left to right
    If number is behind door
        Return true
Return false
```

```
For i from 0 to n-1
    If number behind doors[i]
        Return true
Return false
```

$O(n^2)$	
$O(n \log n)$	
$O(n)$	linear search
$O(\log n)$	
$O(1)$	

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search

binary search

```
If no doors
    Return false
If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
```

```
If no doors
    Return false
If number behind doors[middle]
    Return true
Else if number < doors[middle]
    Search doors[0] through doors[middle - 1]
Else if number > doors[middle]
    Search doors[middle + 1] through doors[n - 1]
```

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$ binary search

$O(1)$

$\Omega(n^2)$	
$\Omega(n \log n)$	
$\Omega(n)$	
$\Omega(\log n)$	
$\Omega(1)$	binary search

```
int numbers[]
```

Code

- numbers.c
- names.c
  - strcmp
- phonebook0.c
  - Telefones como números vs strings?

```
string names[]
string names[]
string numbers[]
```

data structures

```
person people[]
```

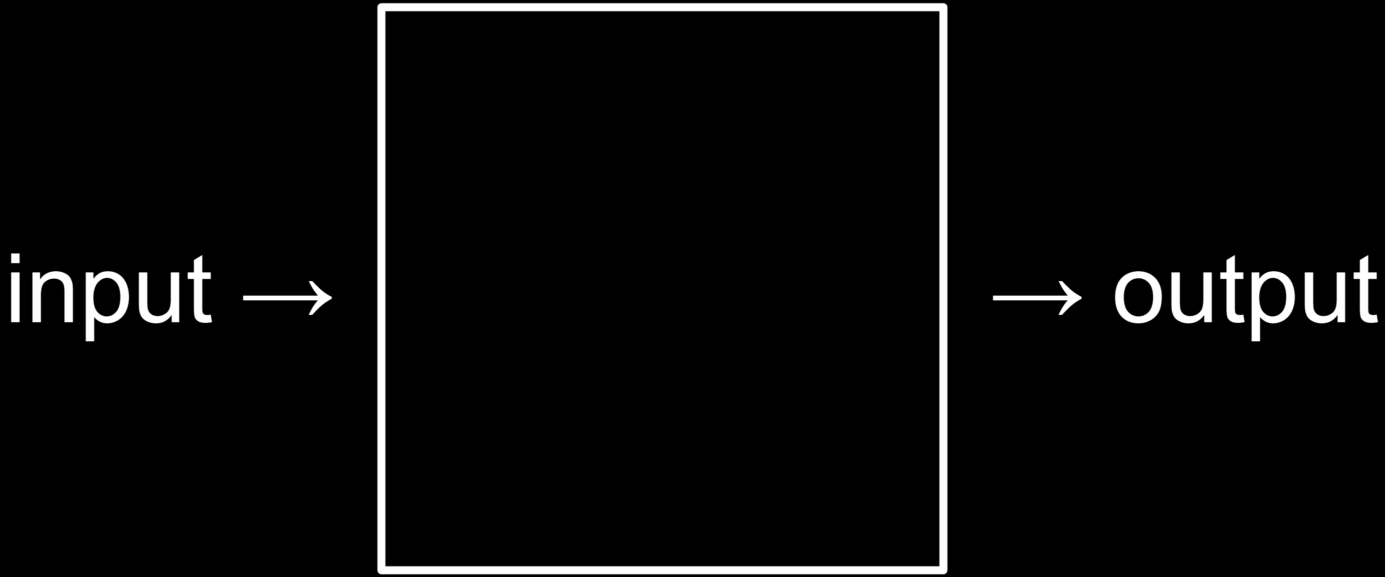
```
string name;
string number;
```

```
typedef struct
{
    string name;
    string number;
}
person;
```

Code

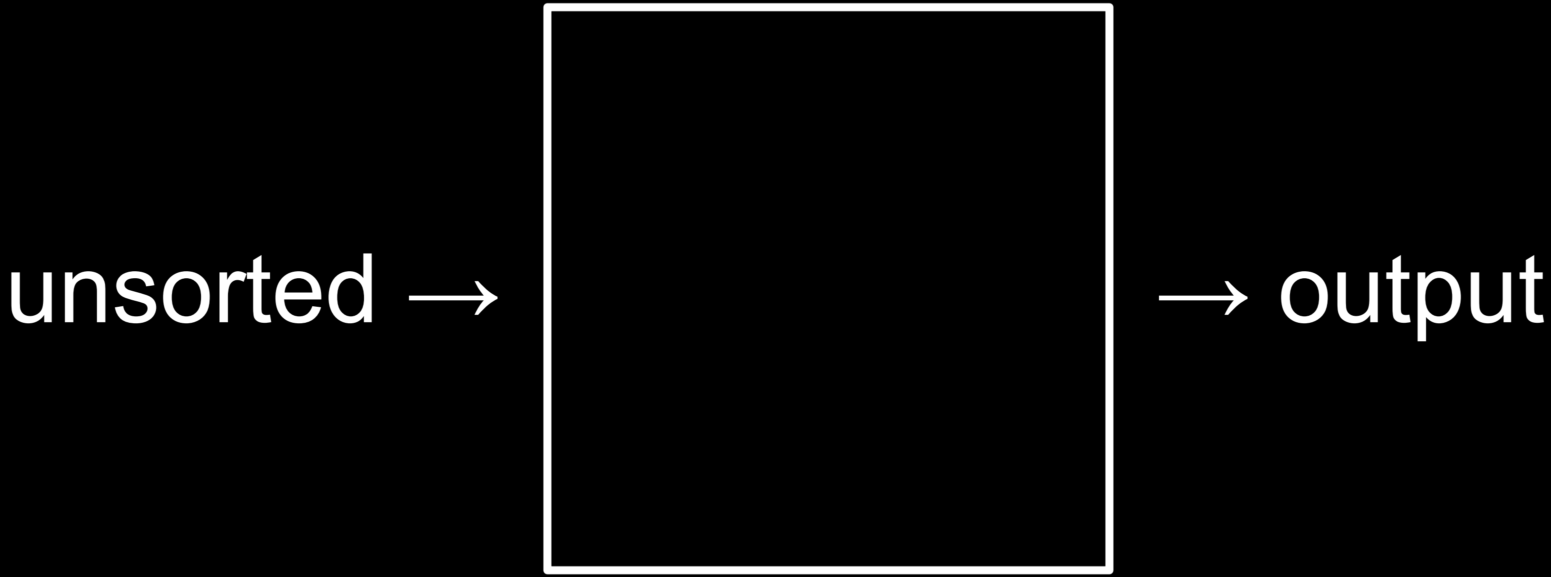
- phonebook1.c
    - struct em C

sorting



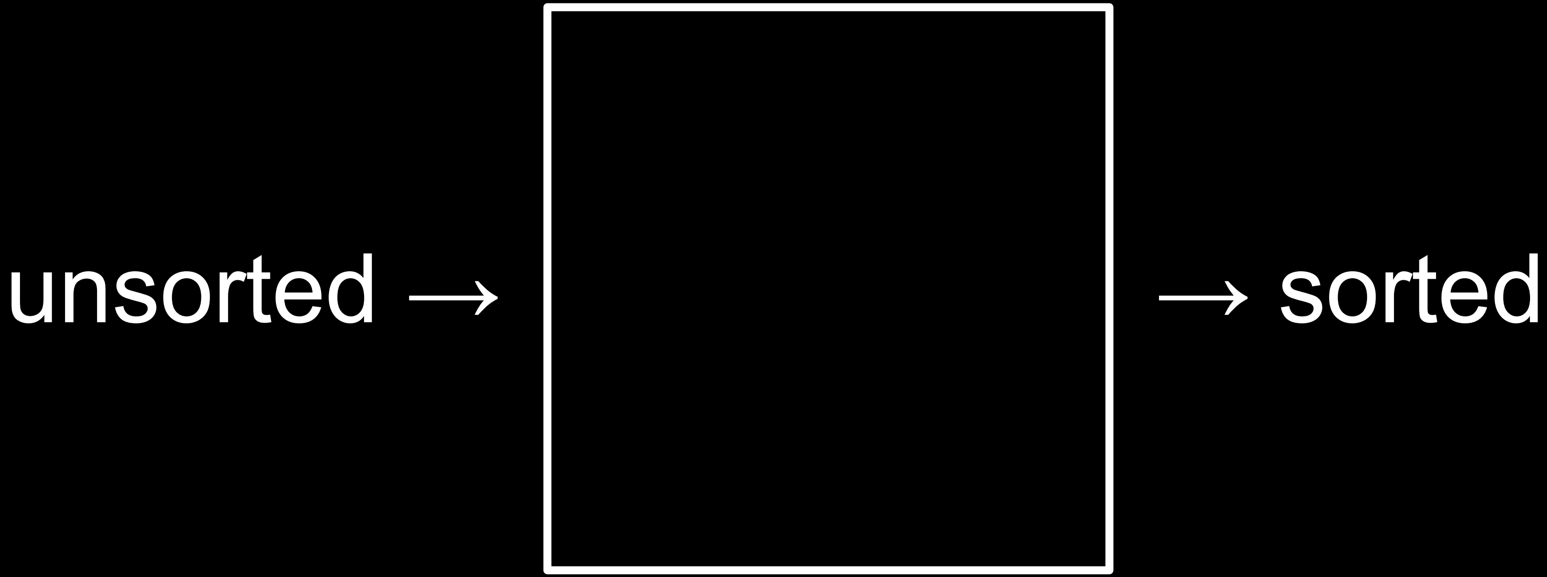
input →

→ output



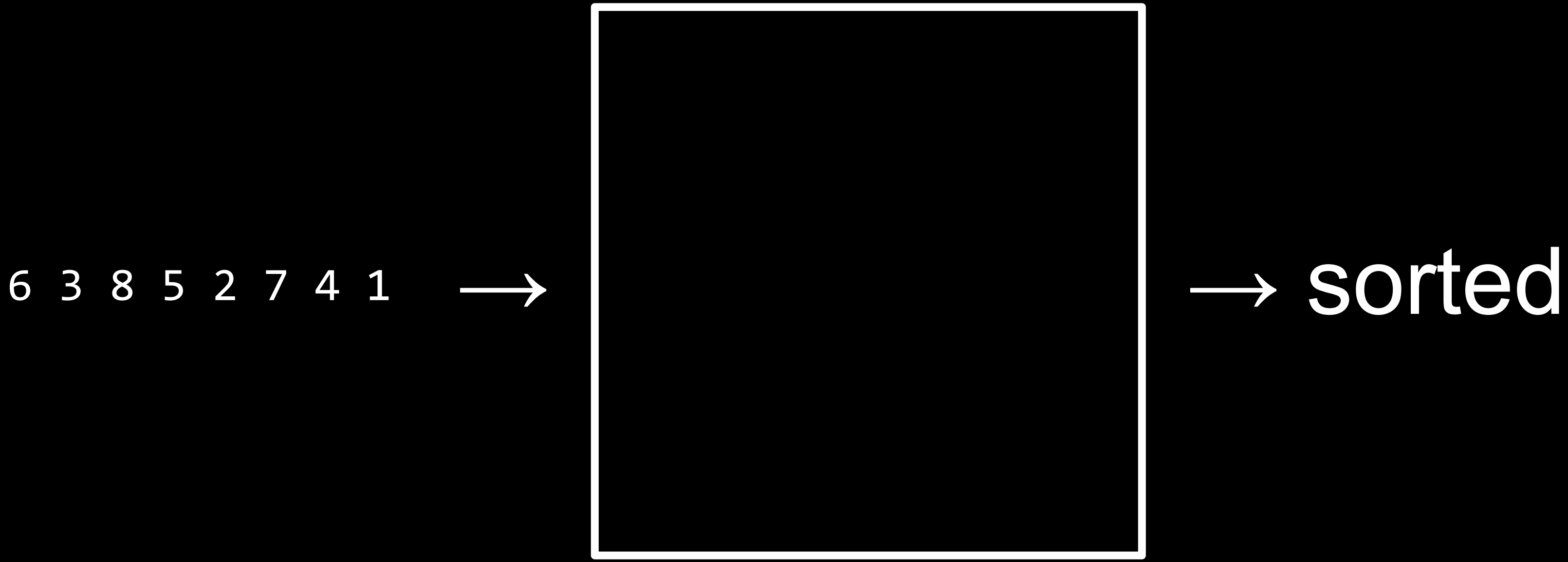
unsorted →

→ output

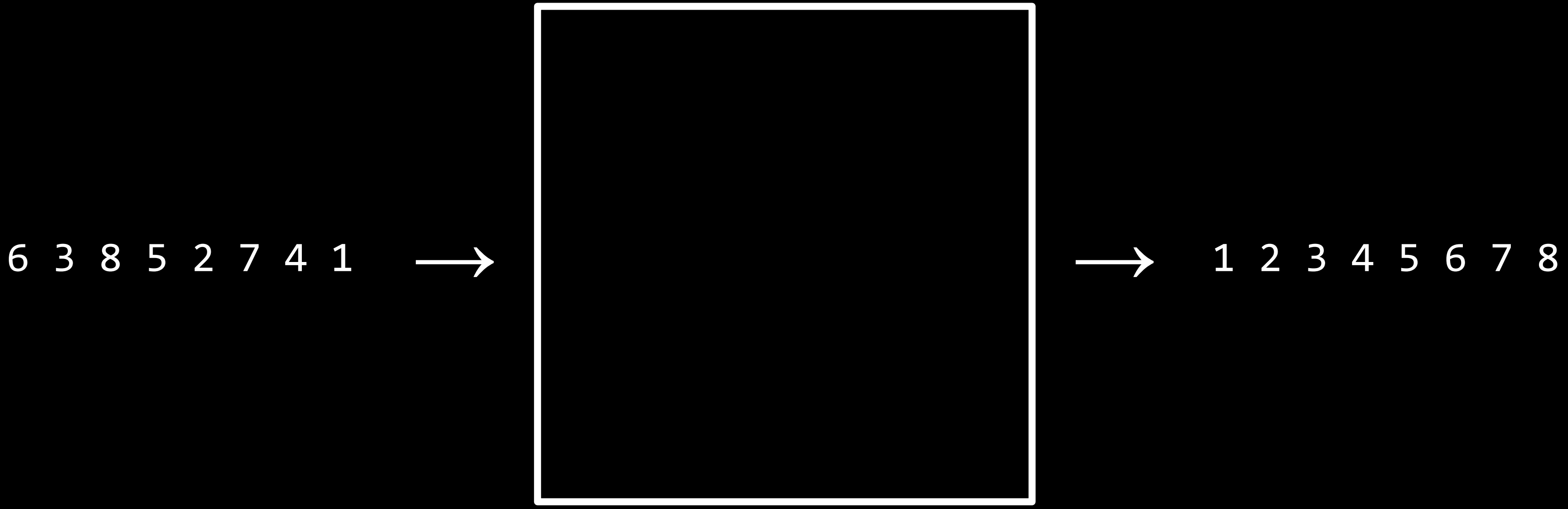


unsorted →

→ sorted



6 3 8 5 2 7 4 1 → → sorted



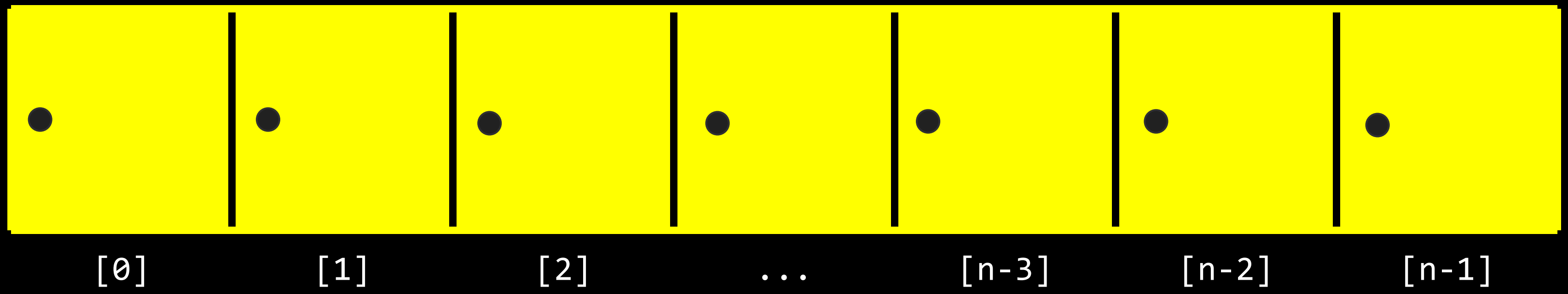
6 3 8 5 2 7 4 1

1 2 3 4 5 6 7 8

selection sort

5 2 7 4 1 6 3 0

```
For i from 0 to n-1
    Find smallest number between numbers[i] and numbers[n-1]
    Swap smallest number with numbers[i]
```



[0] [1] [2] ... [n-3] [n-2] [n-1]

n

$n + (n - 1)$

$n + (n - 1) + (n - 2)$

$n + (n - 1) + (n - 2) + ... + 1$

$n + (n - 1) + (n - 2) + ... + 1$

$n(n + 1)/2$

$n + (n - 1) + (n - 2) + ... + 1$

$n(n + 1)/2$

$(n^2 + \mathrm{n})/2$

$n + (n - 1) + (n - 2) + \ldots + 1$

$n(n + 1)/2$

$(n^2 + \mathrm{n})/2$

$n^2/2 + n/2$

$n + (n - 1) + (n - 2) + \ldots + 1$

$n(n + 1)/2$

$(n^2 + \mathrm{n})/2$

$n^2/2 + n/2$

$O(n^2)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$	selection sort
$O(n \log n)$	
$O(n)$	
$O(\log n)$	
$O(1)$	

```
For i from 0 to n-1
    Find smallest number between numbers[i] and numbers[n-1]
    Swap smallest number with numbers[i]
```

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

$\Omega(n^2)$	selection sort
$\Omega(n \log n)$	
$\Omega(n)$	
$\Omega(\log n)$	
$\Omega(1)$	

$\Theta(n^2)$

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

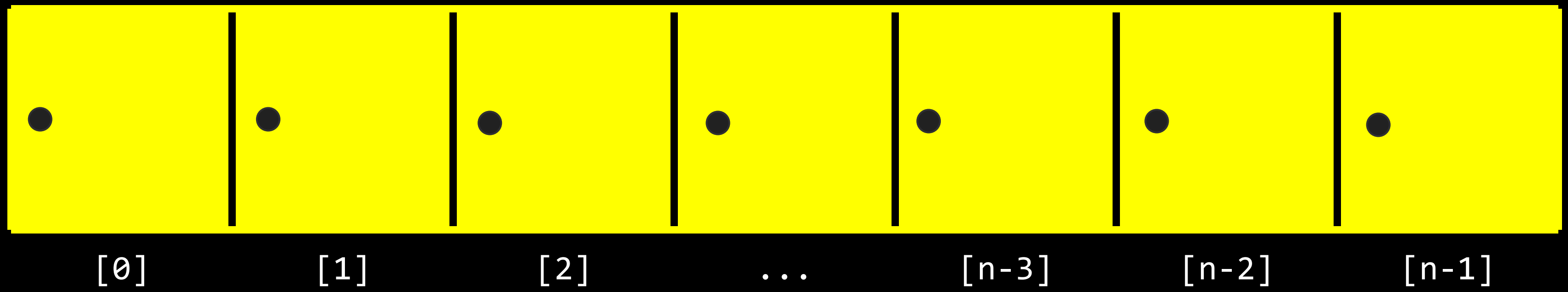
$\Theta(1)$

$\Theta(n^2)$	selection sort
$\Theta(n \log n)$	
$\Theta(n)$	
$\Theta(\log n)$	
$\Theta(1)$	

bubble sort

5 2 7 4 1 6 3 0

```
Repeat n-1 times
    For i from 0 to n-2
        If numbers[i] and numbers[i+1] out of order
            Swap them
```


[0]
[1]
[2]
...
[n-3]
[n-2]
[n-1]

$(n - 1) \times (n - 1)$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$n^2 - 2n + 1$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$n^2 - 2n + 1$

$O(n^2)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$	bubble sort
$O(n \log n)$	
$O(n)$	
$O(\log n)$	
$O(1)$	

```
Repeat n-1 times
    For i from 0 to n-2
        If numbers[i] and numbers[i+1] out of order
            Swap them
    If no swaps
        Quit
```

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

$\Omega(n^2)$	
$\Omega(n \log n)$	
$\Omega(n)$	bubble sort
$\Omega(\log n)$	
$\Omega(1)$	

recursion

```
If no doors
    Return false
If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
```

```
If no doors
    Return false
If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
```

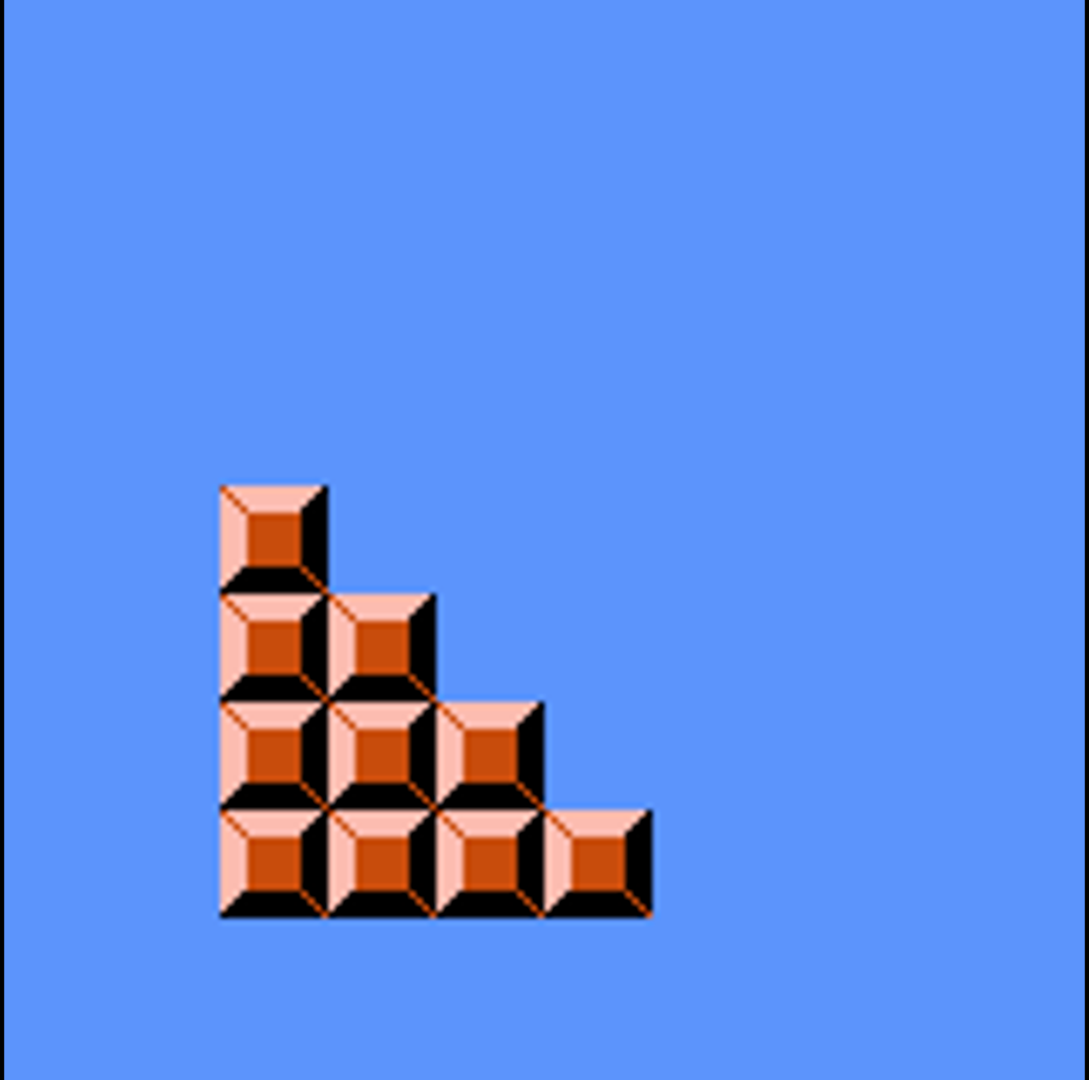
```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Open to middle of left half of book
8       Go back to line 3
9   Else if person is later in book
10      Open to middle of right half of book
11      Go back to line 3
12  Else
13      Quit
```

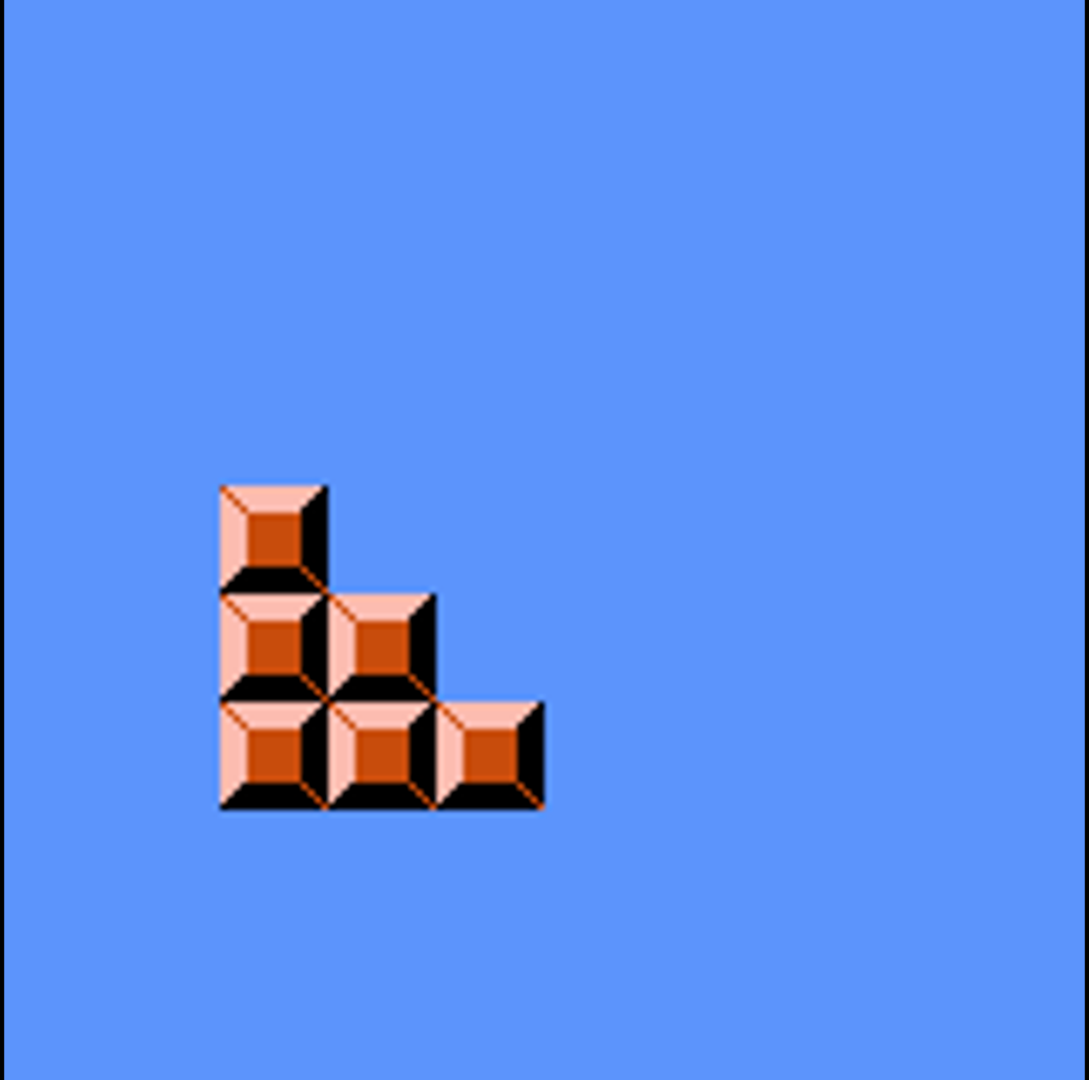
```
1    Pick up phone book
2    Open to middle of phone book
3    Look at page
4    If person is on page
5        Call person
6    Else if person is earlier in book
7        Open to middle of left half of book
8        Go back to line 3
9    Else if person is later in book
10       Open to middle of right half of book
11       Go back to line 3
12   Else
13       Quit
```

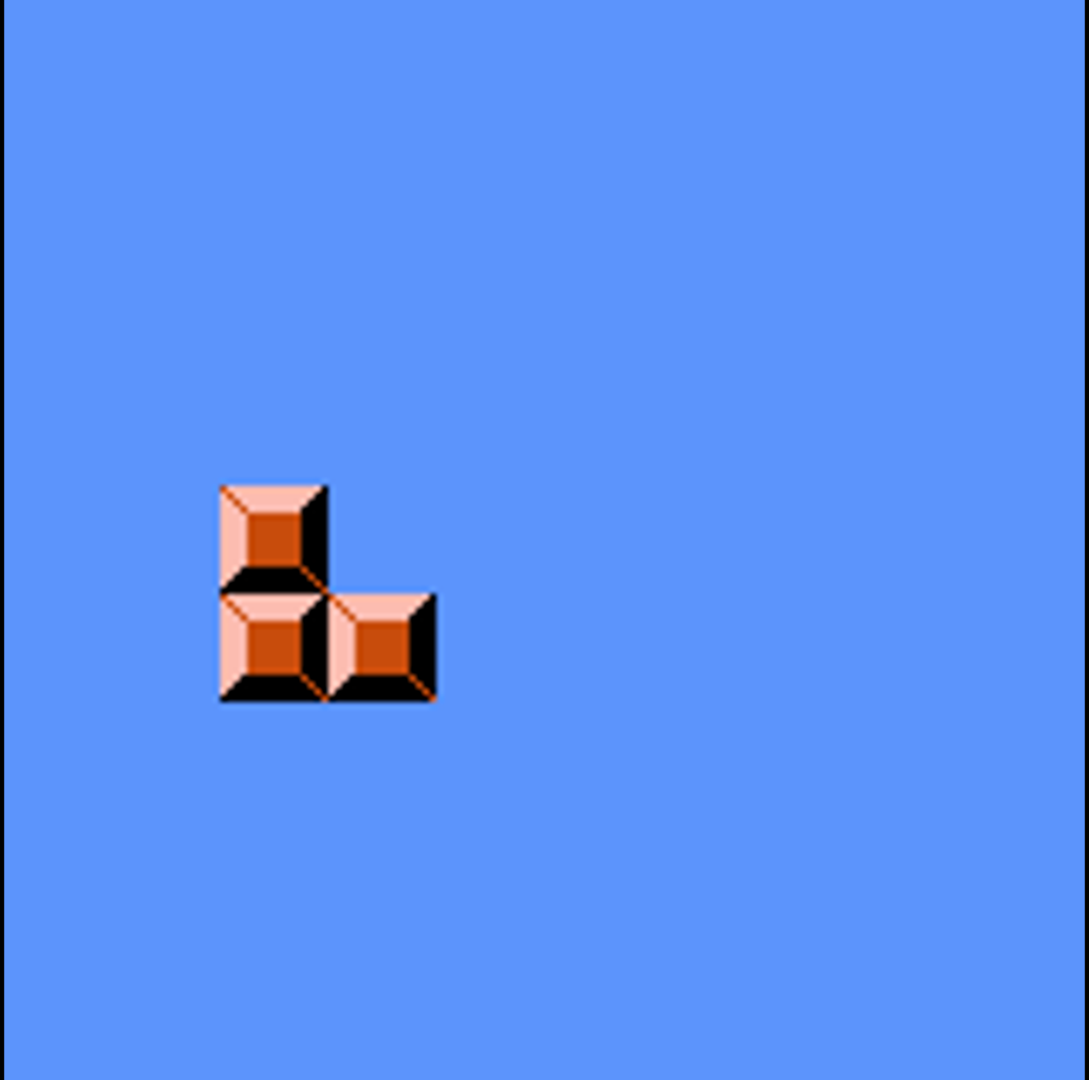
```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Open to middle of left half of book
8       Go back to line 3
9   Else if person is later in book
10      Open to middle of right half of book
11      Go back to line 3
12  Else
13      Quit
```

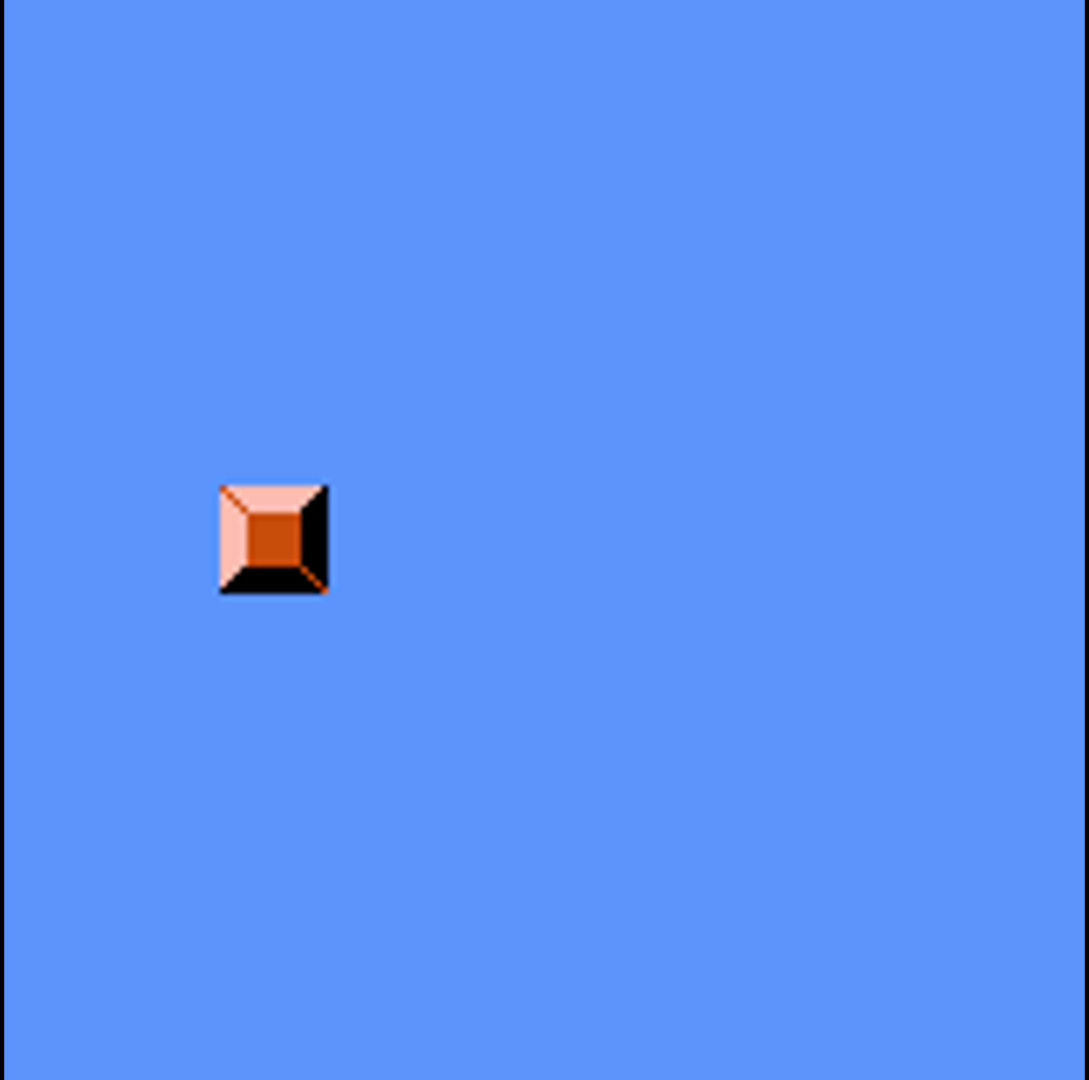
```
Pick up phone book
Open to middle of phone book
Look at page
If person is on page
    Call person
Else if person is earlier in book
    Search left half of book

Else if person is later in book
    Search right half of book

Else
    Quit
```

```
Pick up phone book
Open to middle of phone book
Look at page
If person is on page
    Call person
Else if person is earlier in book
    Search left half of book
Else if person is later in book
    Search right half of book
Else
    Quit
```









Lembrando de Fibonacci

- https://en.wikipedia.org/wiki/Fibonacci_number

$$F_n = \begin{cases} F_{n-1} + F_{n-2} & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

- Solução recursiva direta da formula acima para Fibonacci não é eficiente!

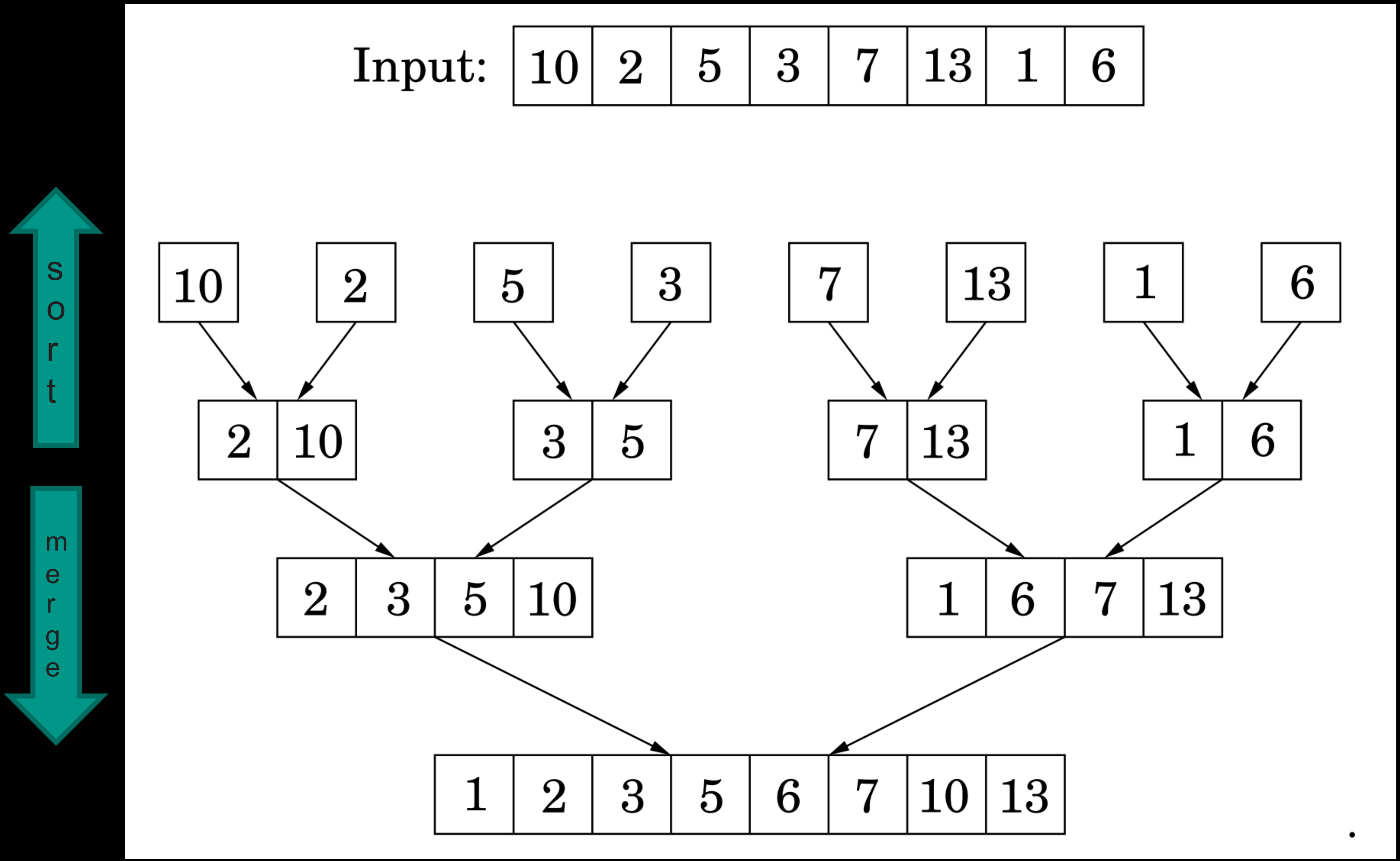
merge sort

```
Sort left half of numbers
Sort right half of numbers
Merge sorted halves
```

```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge sorted halves
```

```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge sorted halves
```

```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge sorted halves
```



Input: | 10 | 2 | 5 | 3 | 7 | 13 | 1 | 6 |

10 2 5 3 7 13 1 6

| 2 | 10 | | 3 | 5 | | 7 | 13 | | 1 | 6 |

| 2 | 3 | 5 | 10 | | 1 | 6 | 7 | 13 |

| 1 | 2 | 3 | 5 | 6 | 7 | 10 | 13 |

5 2 7 4 1 6 3 0

$O(n^2)$

$O(n \log n)$ merge sort

$O(n)$

$O(\log n)$

$O(1)$

$\Omega(n^2)$

$\Omega(n \log n)$ merge sort

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

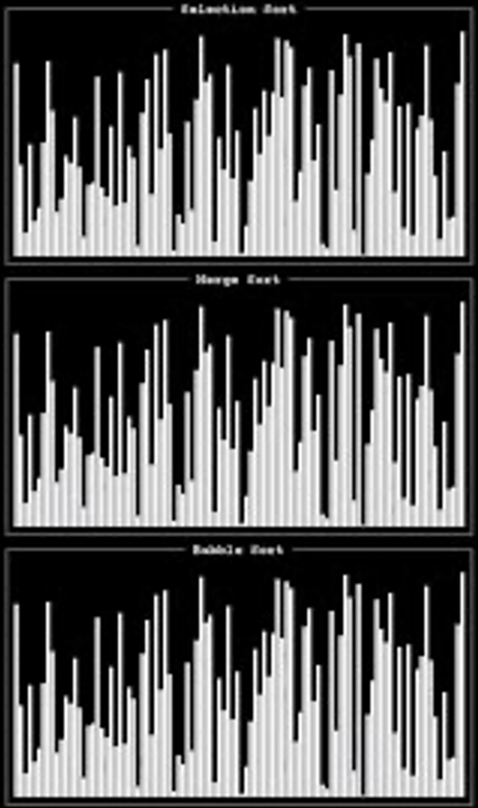
$\Theta(n^2)$

$\Theta(n \log n)$ merge sort

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$


Selection Sort
Merge Sort
Bubble Sort