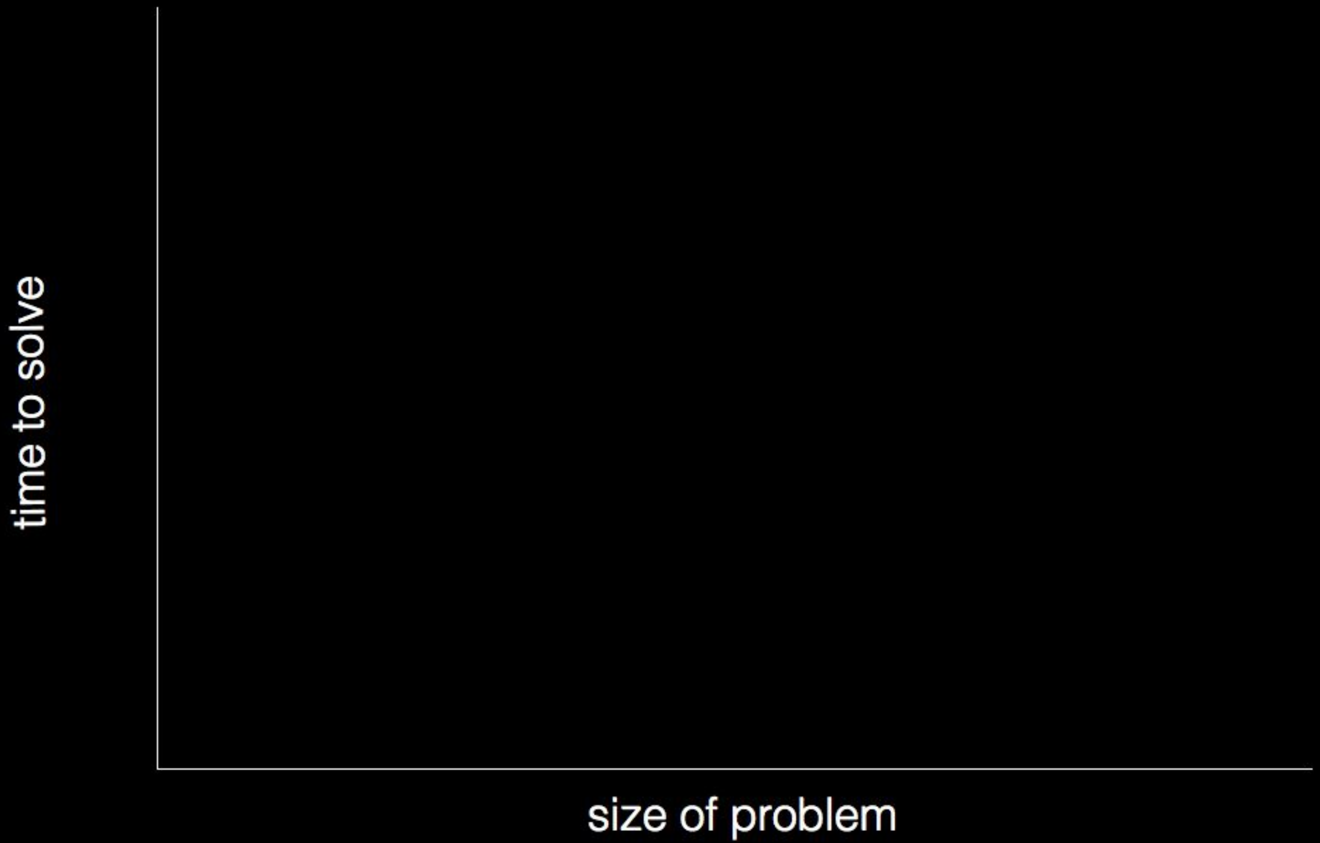
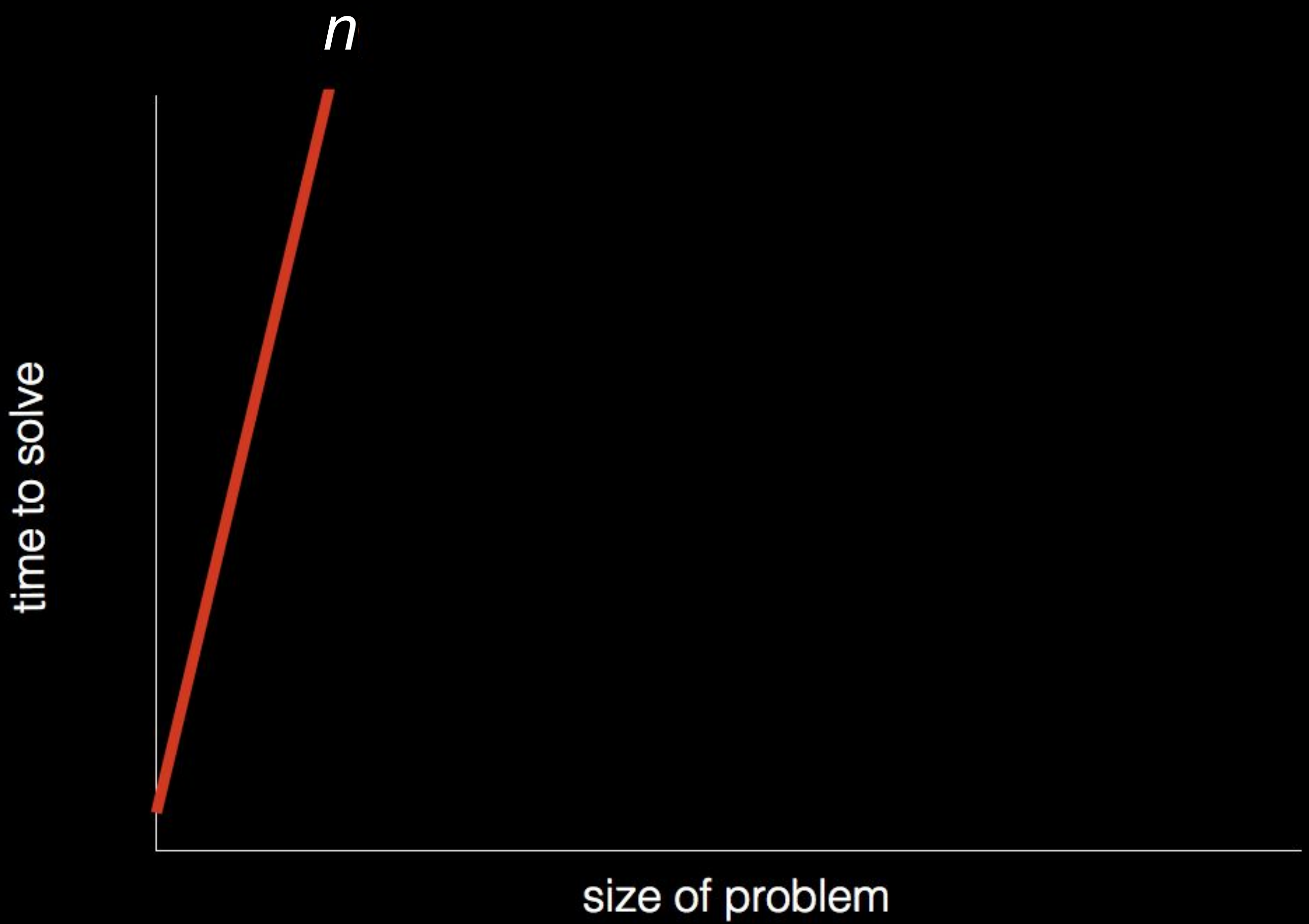
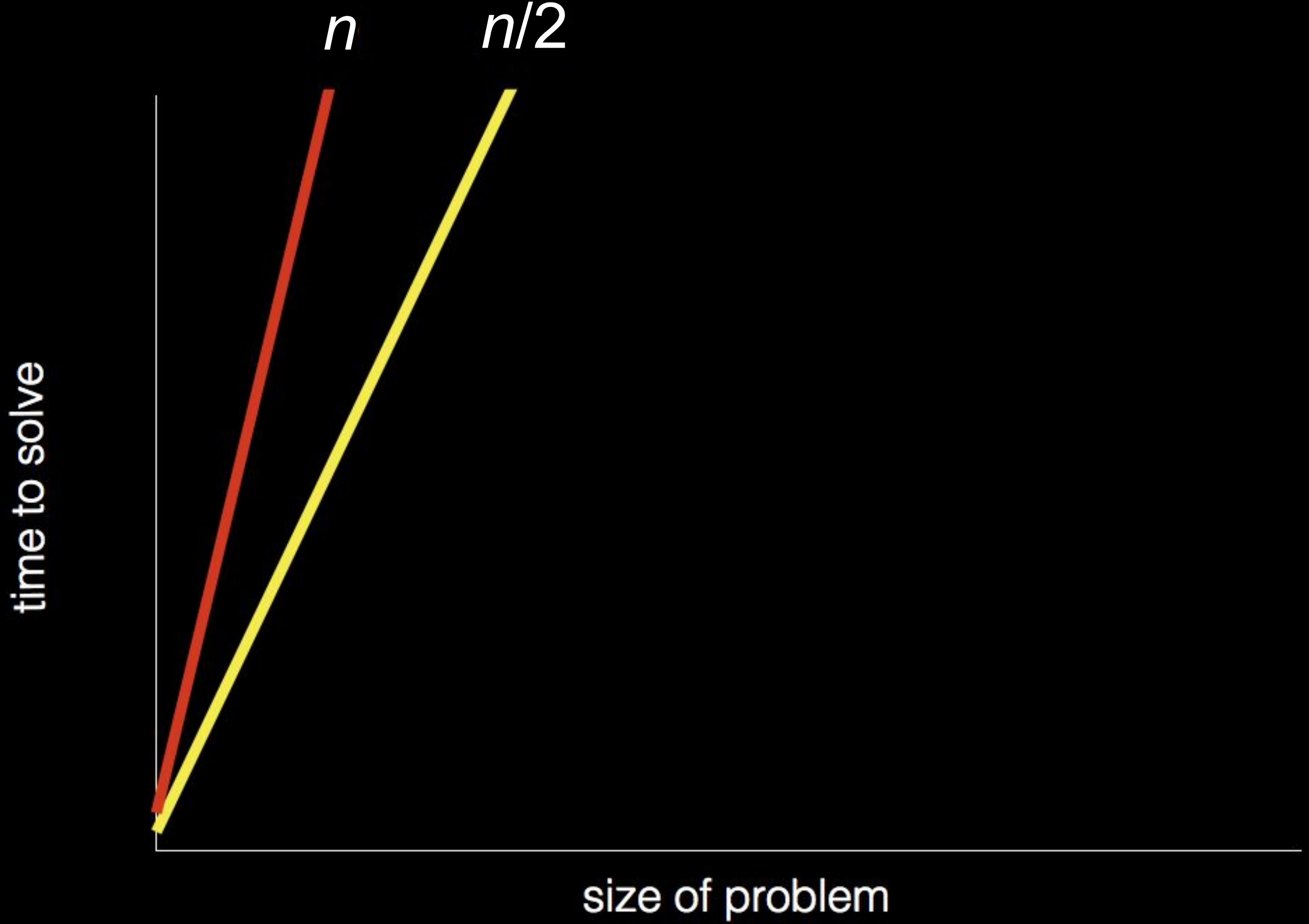
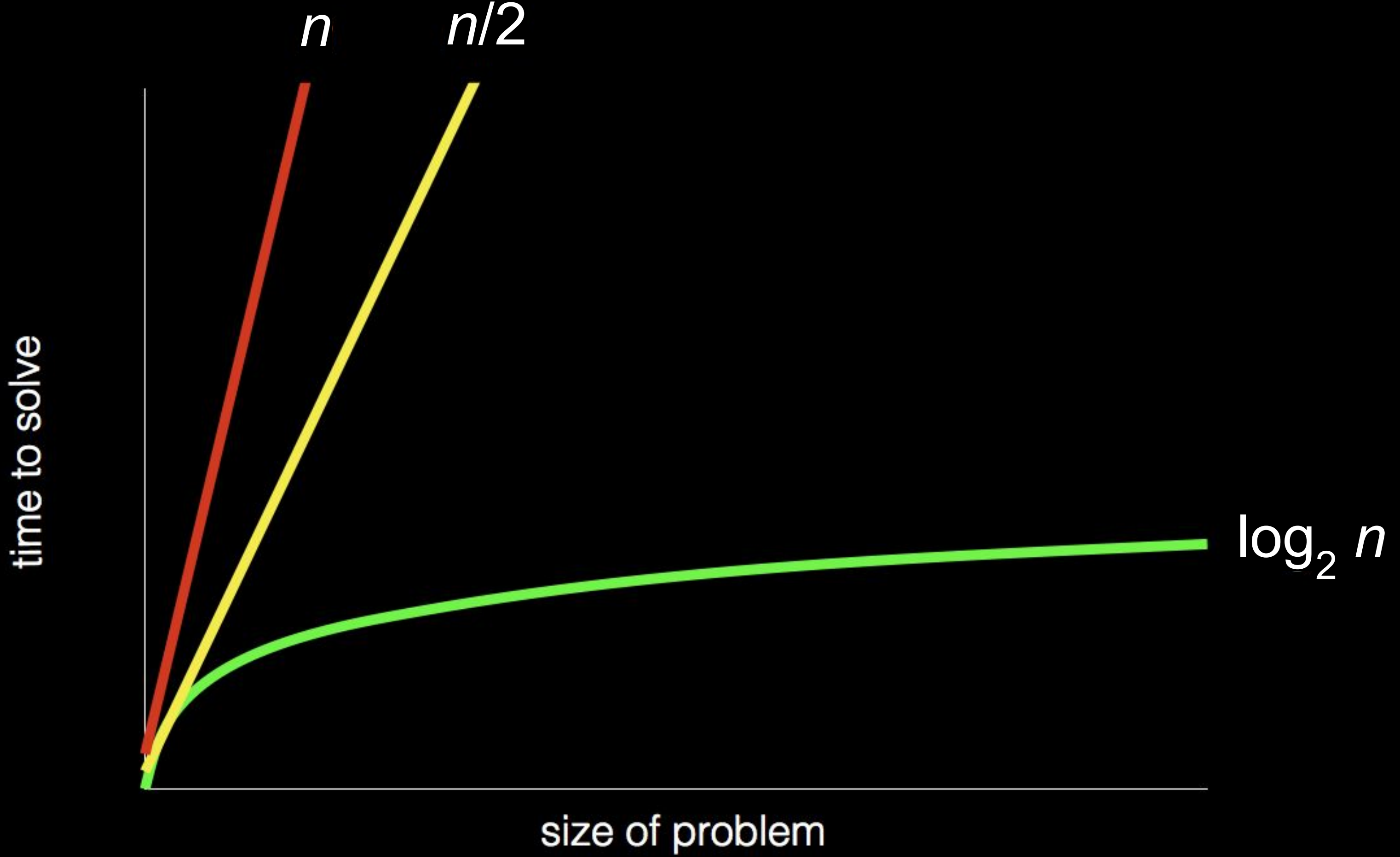
This is CS50


time to solve
size of problem


n
time to solve
size of problem

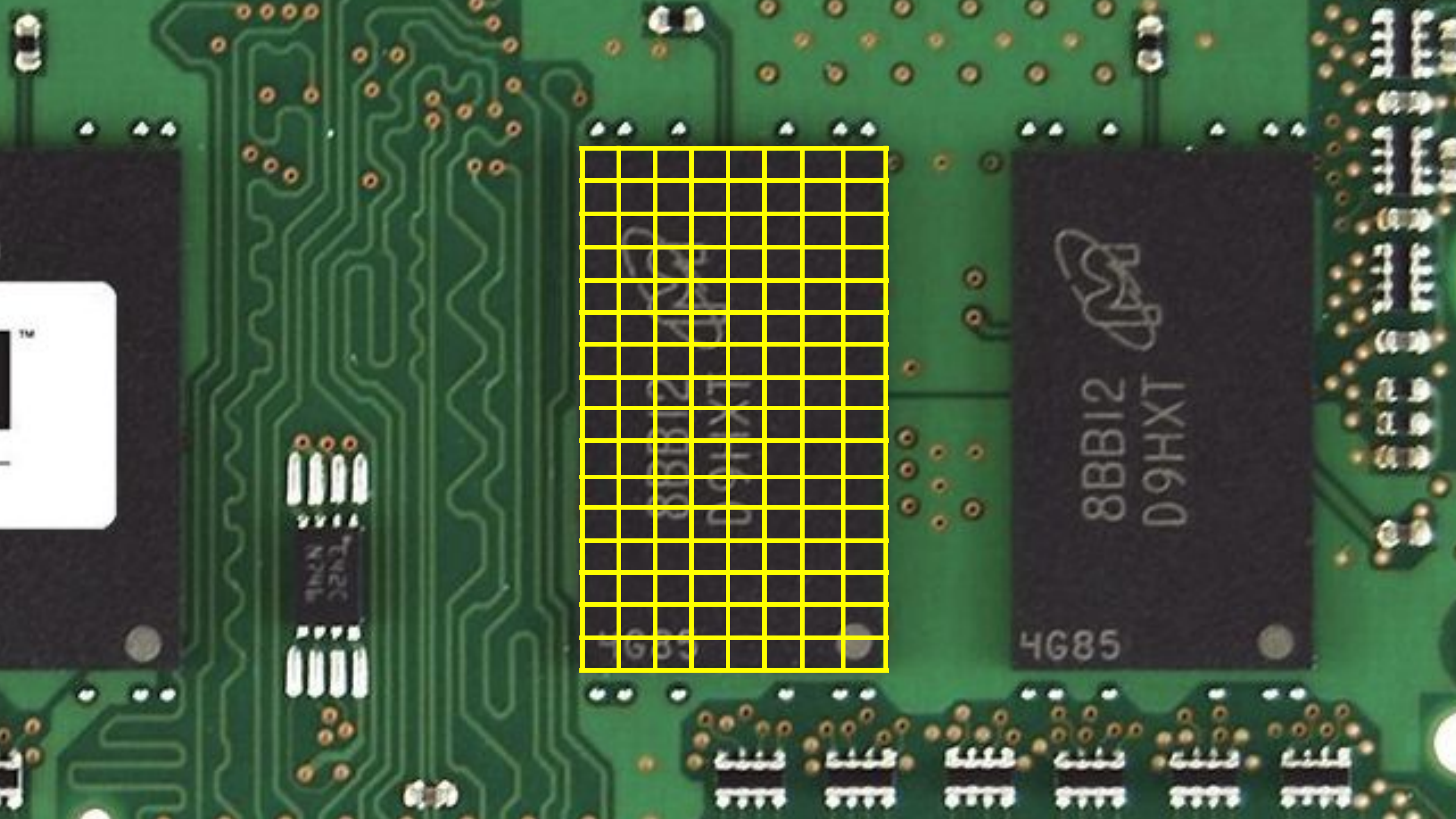


n
n/2
time to solve
size of problem

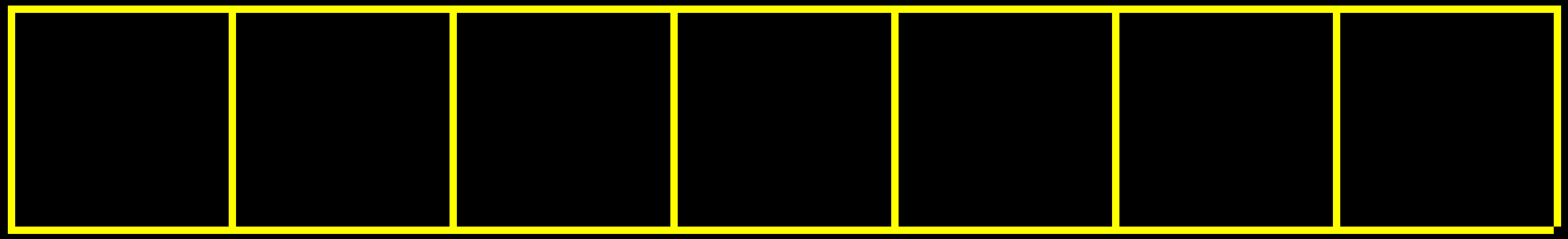


n
n/2
log2 n
time to solve
size of problem

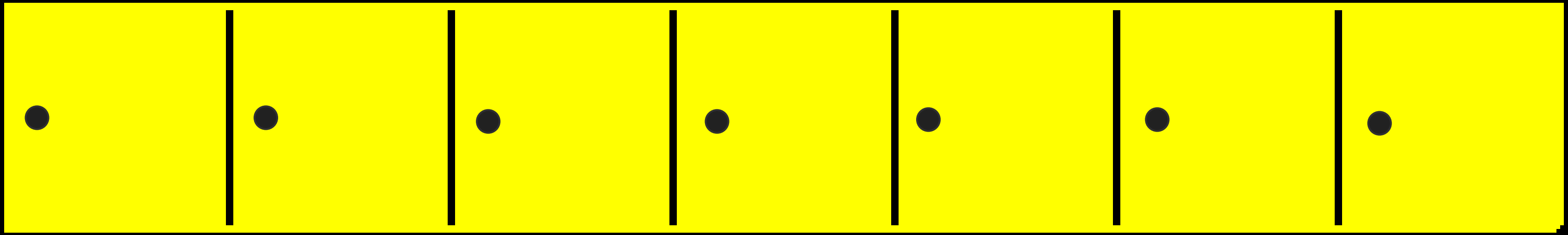

8BB12
D9HXT
4G85
8BB12
D9HXT
4G85

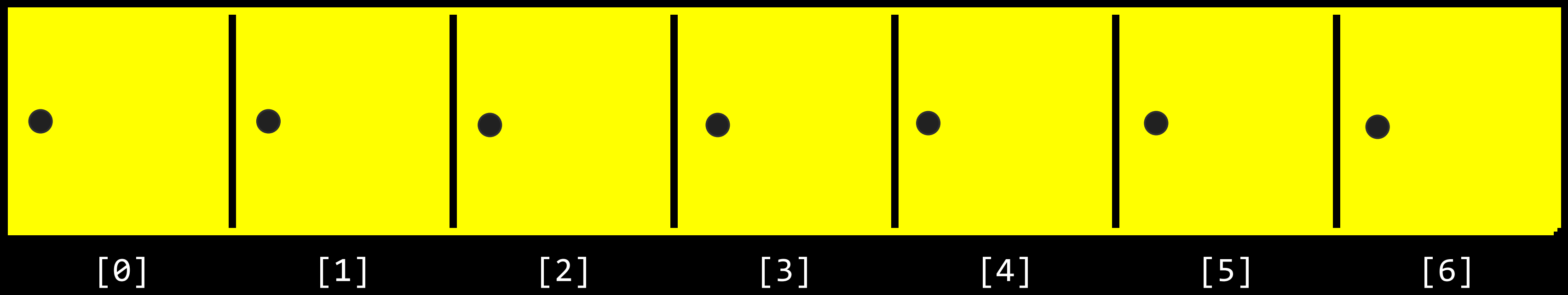


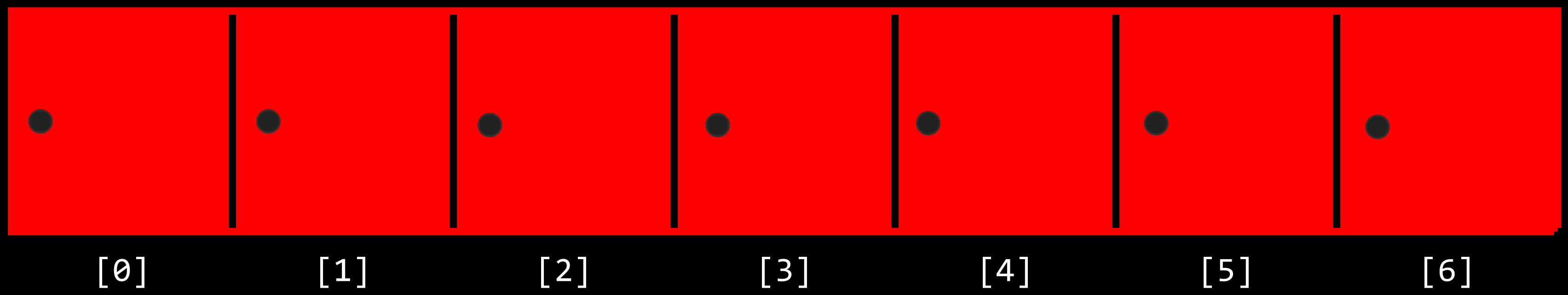
8BB12
D9HXT
4G85



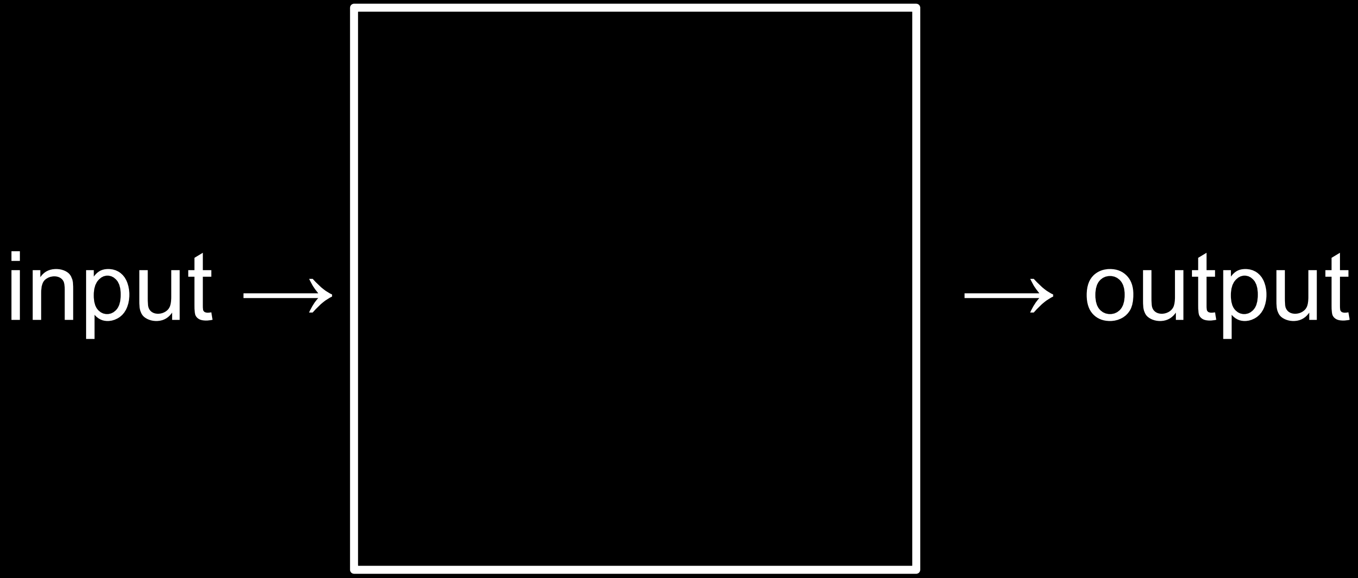
1	5	10	20	50	100	500

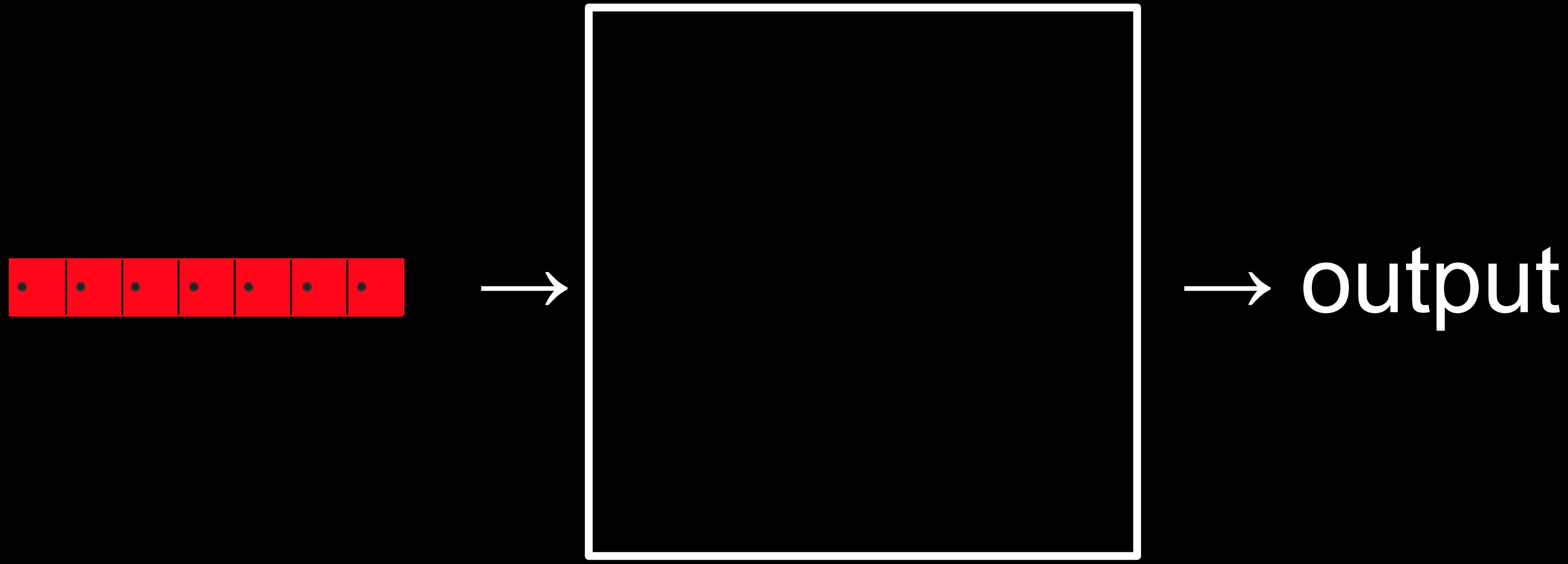


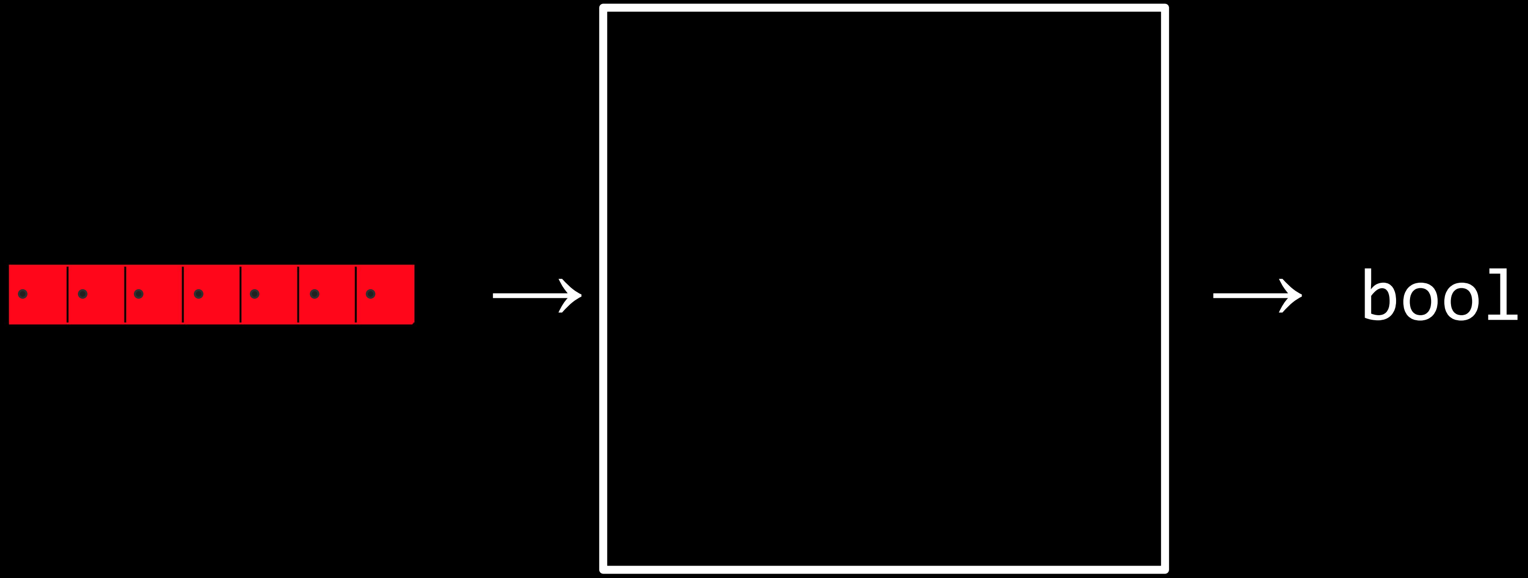

[0]
[1]
[2]
[3]
[4]
[5]
[6]


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

searching

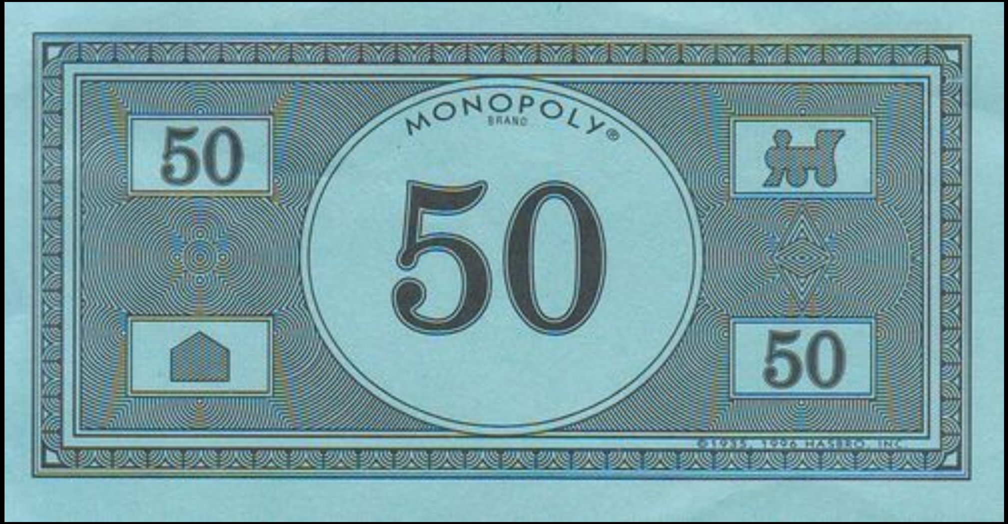

input →
→ output


output


bool

algorithm

MONOPOLY ®
20
MONOPOLY ®
10
MONOPOLY ®
5
MONOPOLY ®
50
MONOPOLY ®
500
MONOPOLY ®
100
MONOPOLY ®
1


MONOPOLY®
BRAND
50
50
50

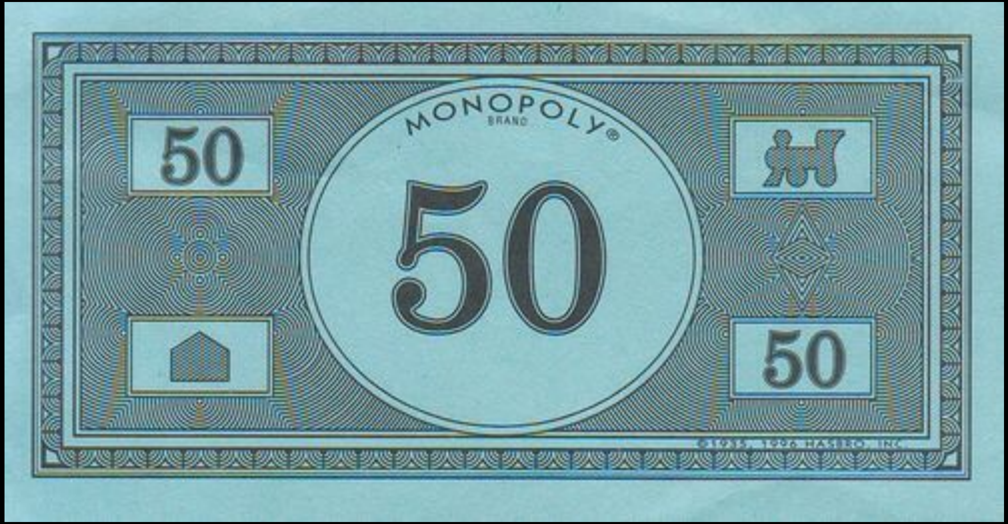
linear search

```
For each door from left to right
    If 50 is behind door
        Return true
Return false
```

```
For each door from left to right
    If 50 is behind door
        Return true
    Else
        Return false
```

```
For each door from left to right
    If 50 is behind door
        Return true
Return false
```

```
For i from 0 to n-1
    If 50 is behind doors[i]
        Return true
Return false
```


MONOPOLY®
BRAND
50
50
50
©1935, 1996 HASBRO, INC.

binary search

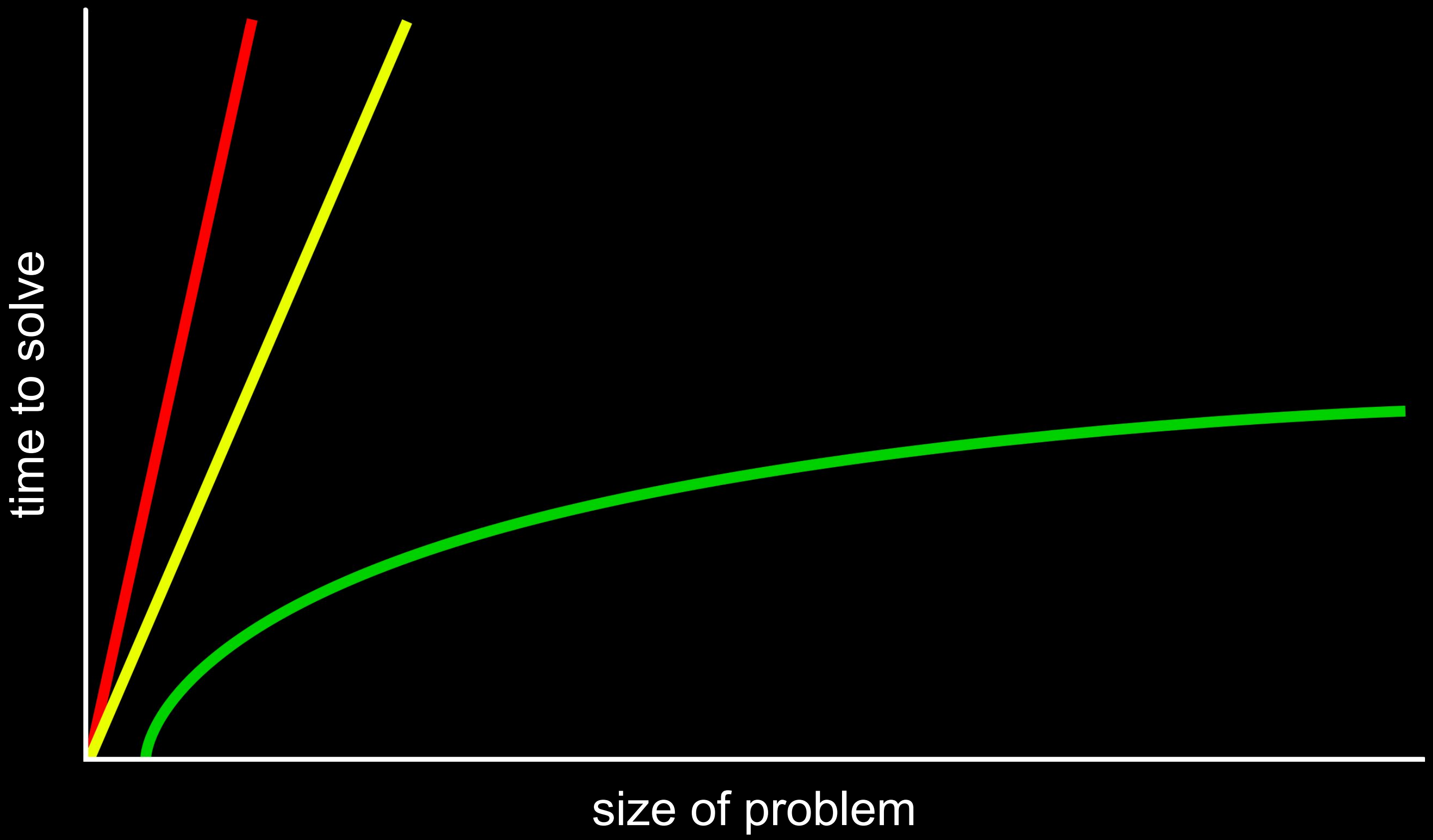
```
If 50 is behind middle door
    Return true
Else if 50 < middle door
    Search left half
Else if 50 > middle door
    Search right half
```

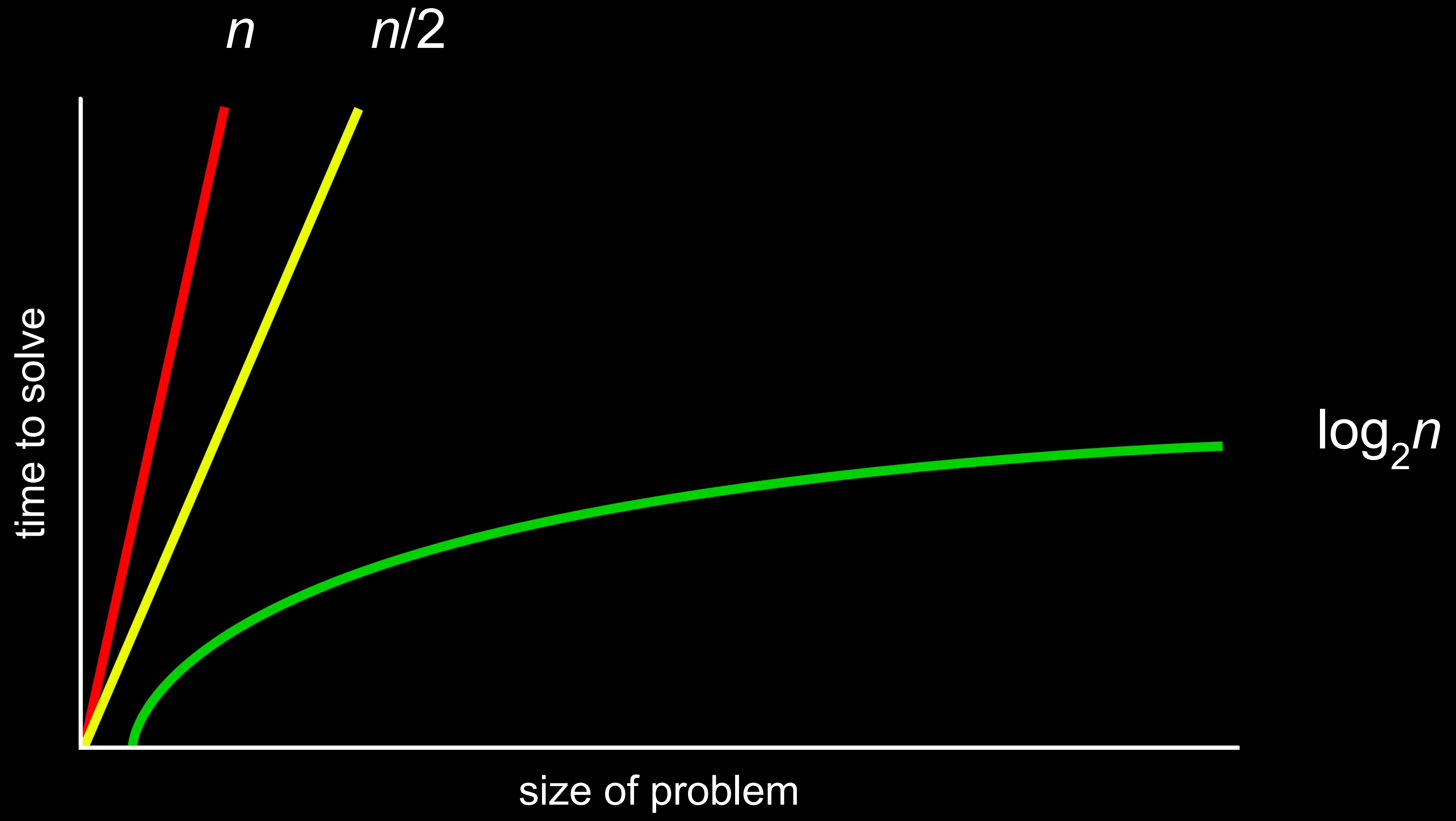
```
If no doors left

If 50 is behind middle door
    Return true
Else if 50 < middle door
    Search left half
Else if 50 > middle door
    Search right half
```

```
If no doors left
    Return false
If 50 is behind middle door
    Return true
Else if 50 < middle door
    Search left half
Else if 50 > middle door
    Search right half
```

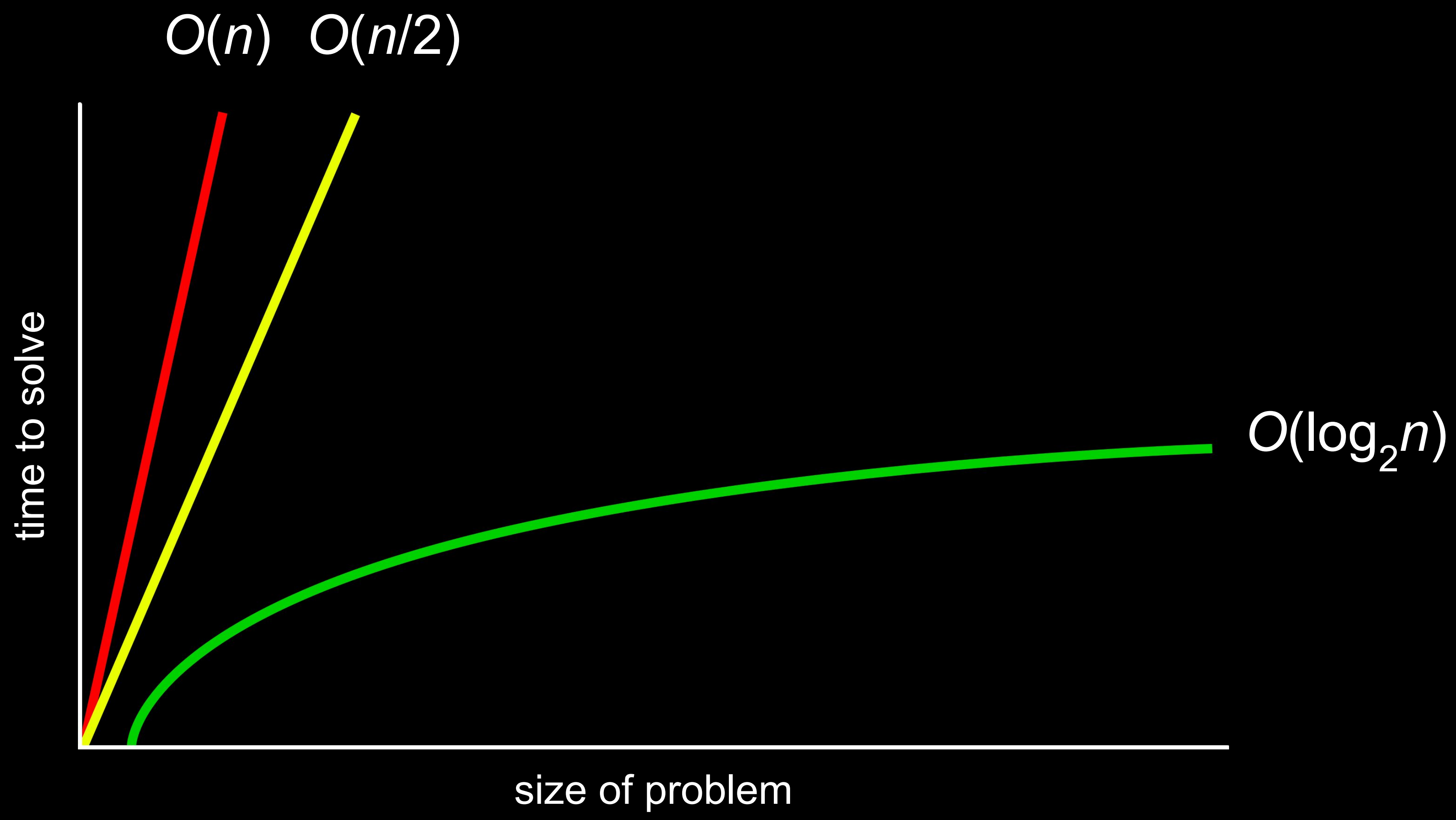
```
If no doors left
    Return false
If 50 is behind doors[middle]
    Return true
Else if 50 < doors[middle]
    Search doors[0] through doors[middle - 1]
Else if 50 > doors[middle]
    Search doors[middle + 1] through doors[n - 1]
```

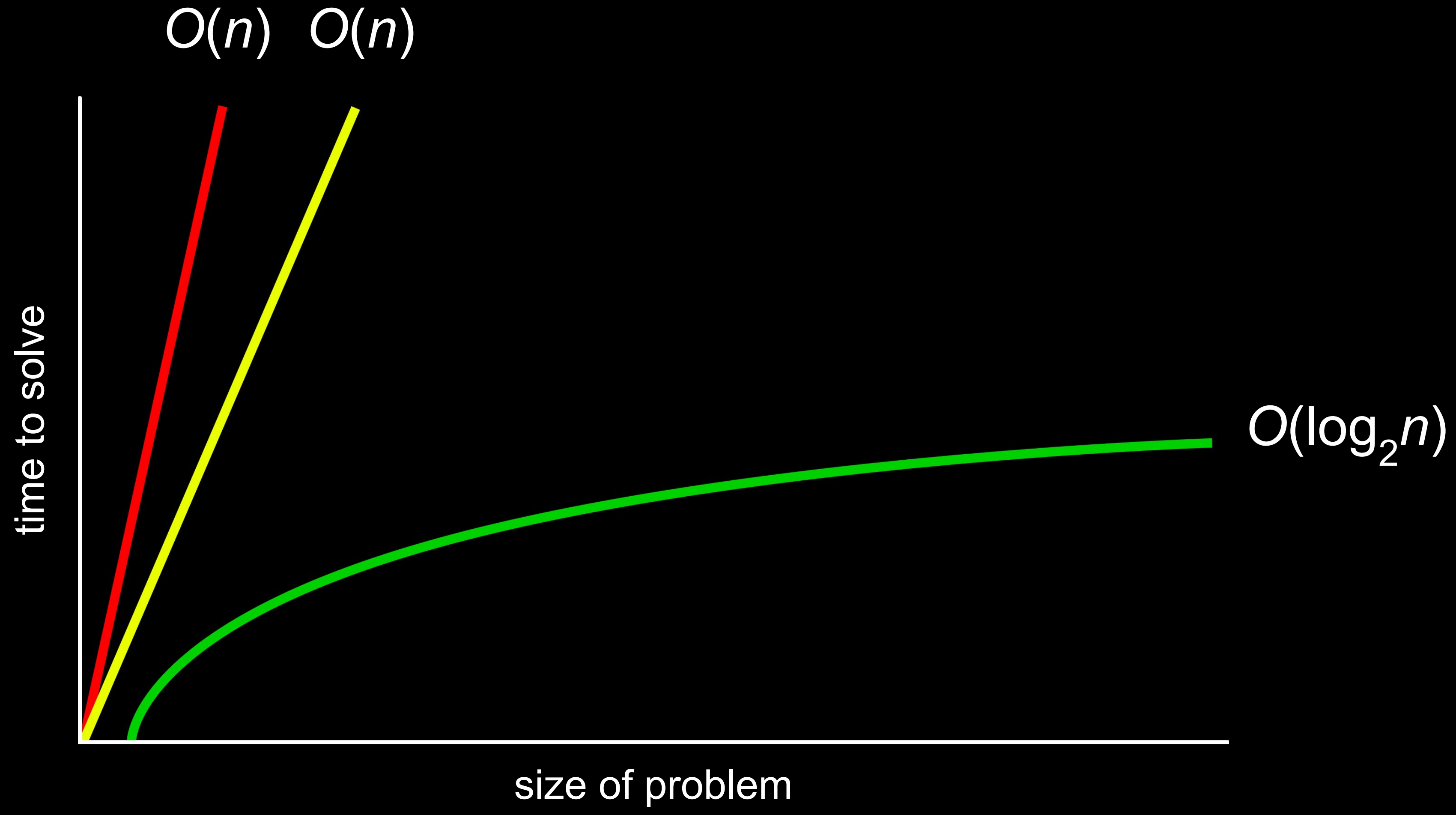
running time


time to solve
size of problem

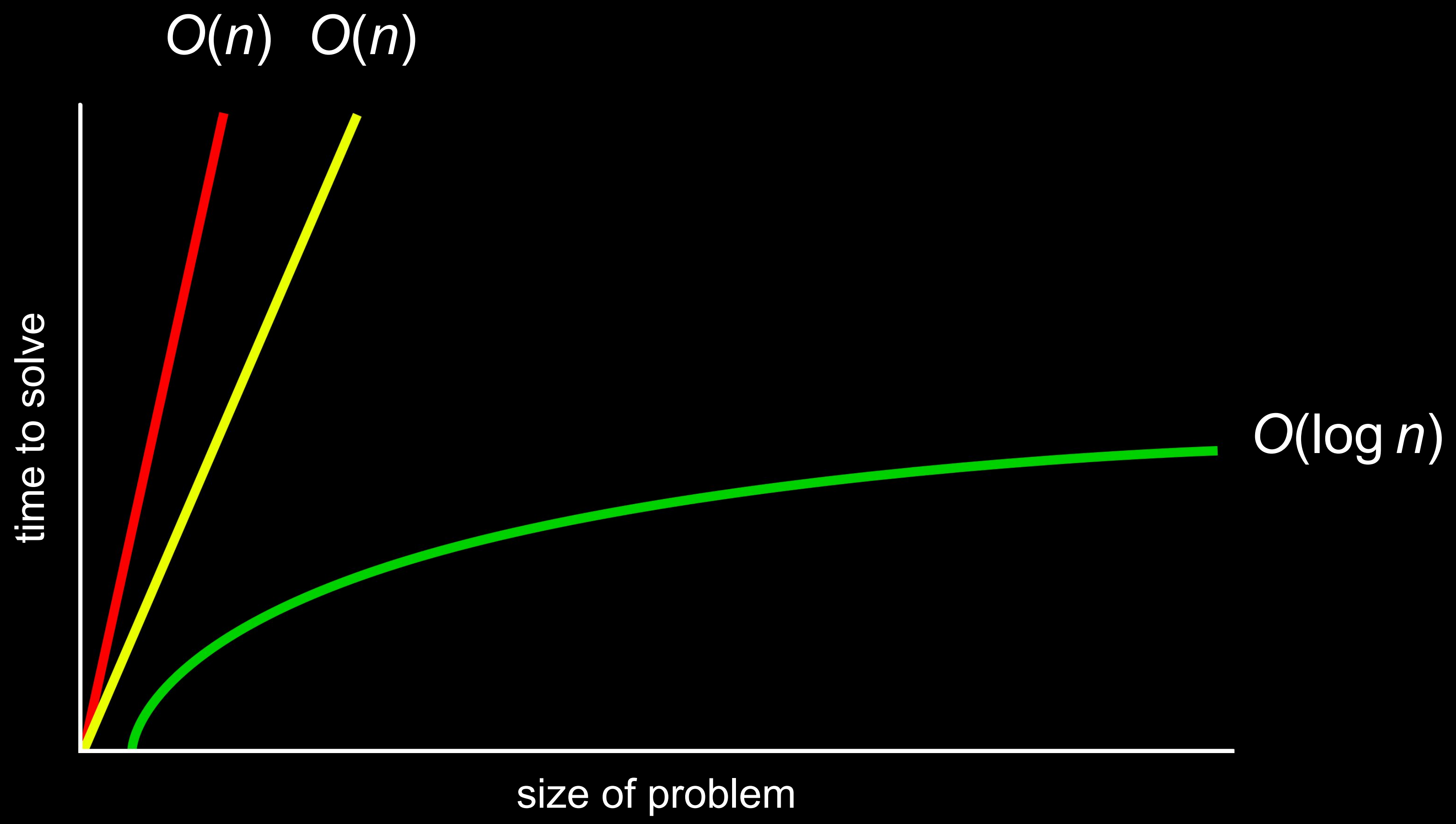


n
$n/2$
$\log_2 n$
time to solve
size of problem

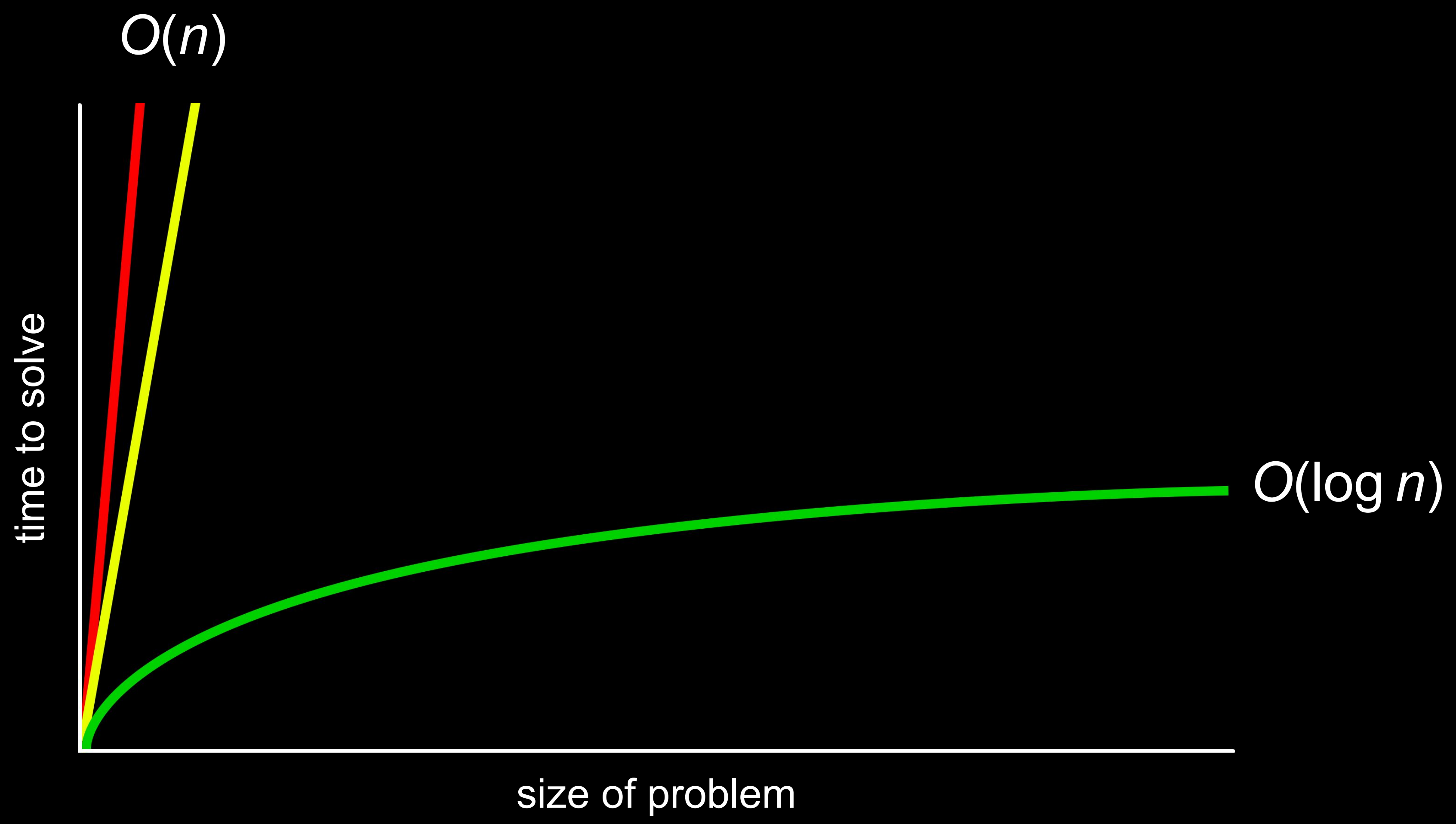

$O(n)$
$O(n/2)$
$O(\log_2 n)$
time to solve
size of problem



$O(n)$
$O(n)$
$O(\log_2 n)$
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

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$ linear search

$O(\log n)$

$O(1)$

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

Ω

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search, binary search

Θ

$\Theta(n^2)$

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$

linear search

data structures

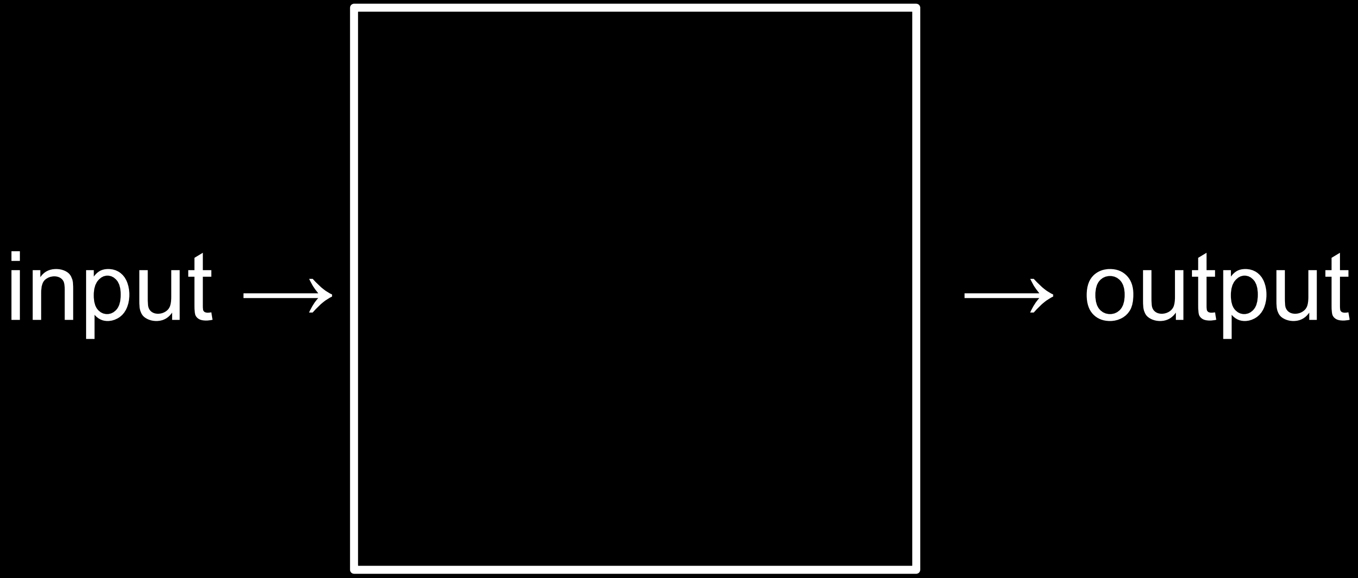
```
person people[]
```

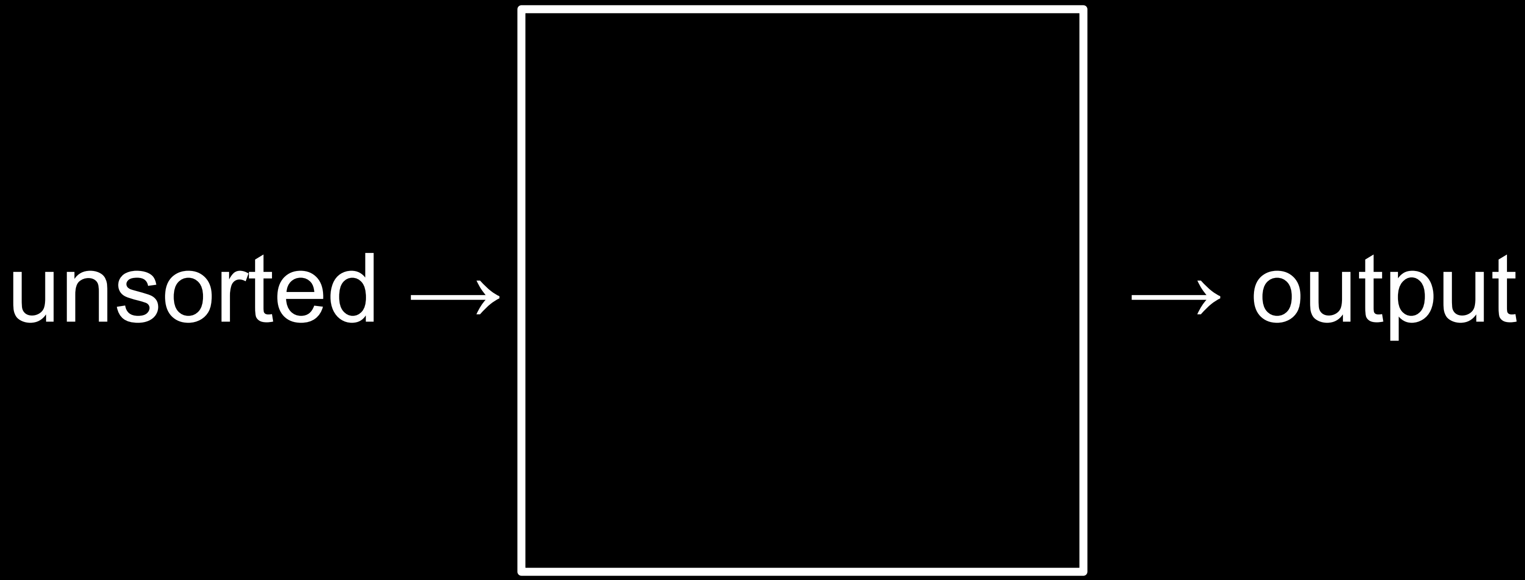
```
string name;
string number;
```

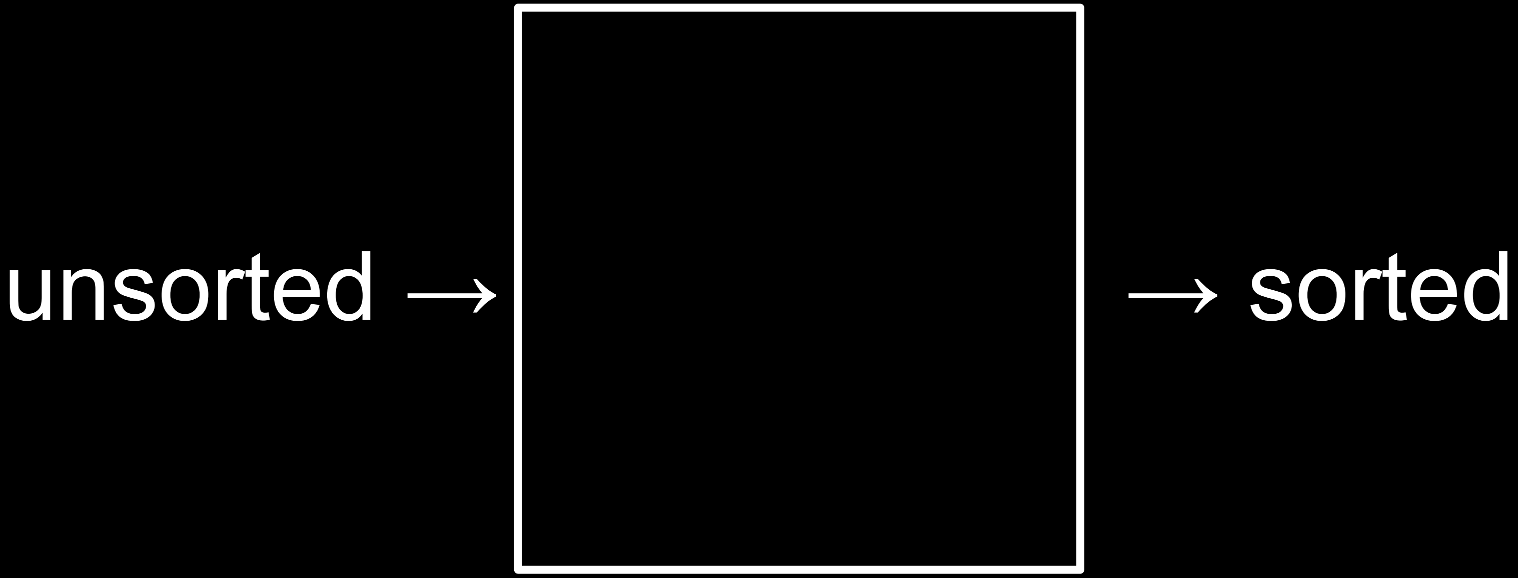
```
typedef struct
{
    string name;
    string number;
}
person;
```

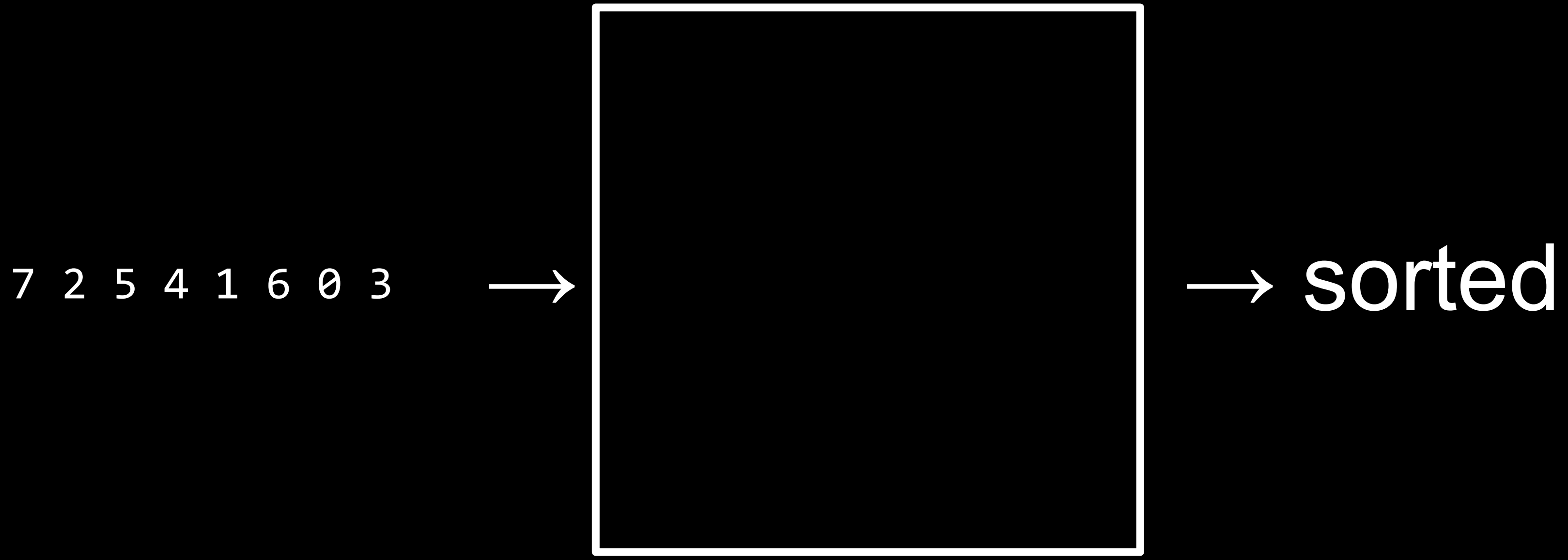

32
BROWNIES
WELLSLEY FARMS
Decadent
BROWNIE
Bites
100%
NET WT 30 OZ (1 LB 14 OZ) 850g
MADE IN A NUT FREE FACILITY

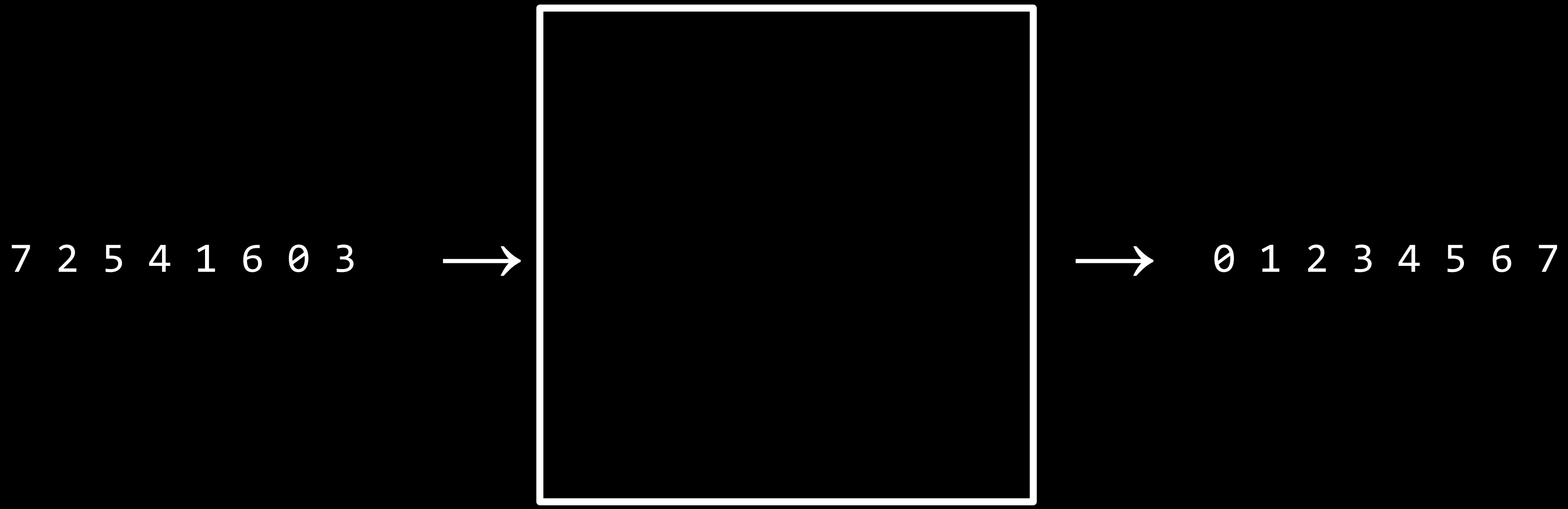
sorting


input →
→ output


unsorted →
→ output


unsorted →
→ sorted


7 2 5 4 1 6 0 3
sorted

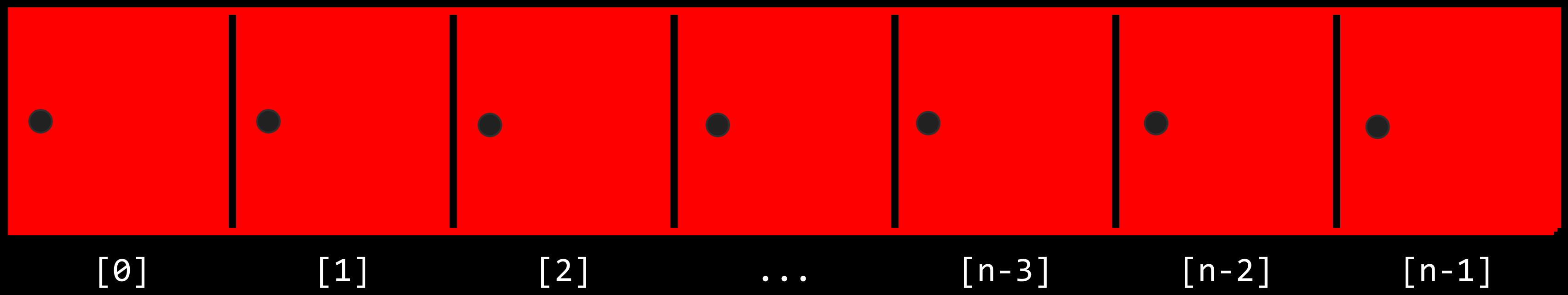

7 2 5 4 1 6 0 3
0 1 2 3 4 5 6 7

7 2 5 4 1 6 0 3

selection sort

7 2 5 4 1 6 0 3

```
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]

$(n - 1)$

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

$$(n - 1) + (n - 2) + (n - 3)$$

$$(n - 1) + (n - 2) + (n - 3) + \ldots + 1$$

$$(n - 1) + (n - 2) + (n - 3) + \ldots + 1$$

$$n(n - 1)/2$$

$(n - 1) + (n - 2) + (n - 3) + ... + 1$

$n(n - 1)/2$

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

$(n - 1) + (n - 2) + (n - 3) + ... + 1$

$n(n - 1)/2$

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

$n^2/2 - n/2$

$(n - 1) + (n - 2) + (n - 3) + ... + 1$

$n(n - 1)/2$

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

$n^2/2 - n/2$

$O(n^2)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$ selection sort

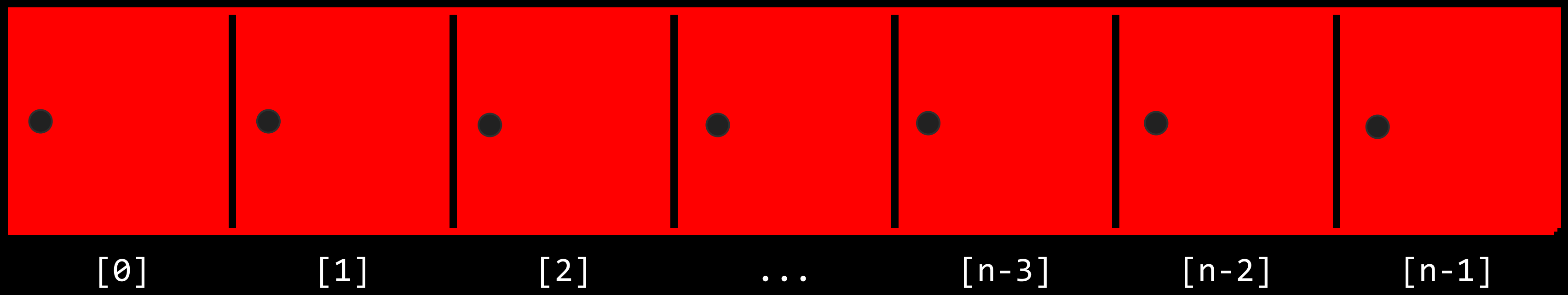
$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

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


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

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

$\Omega(n^2)$ 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

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

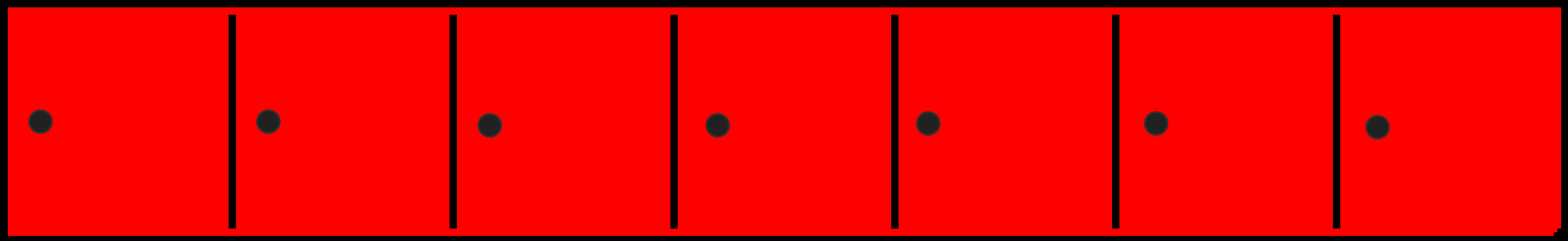
$\Theta(1)$

bubble sort

7 2 5 4 1 6 0 3

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

```
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
```

```
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 left
    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 left
    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
```

google.com/search?q=recursion

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
```

2 4 5 7 0 1 3 6

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

7 2 5 4 1 6 0 3

7	2	5	4	1	6	0	3
2	7	4	5	1	6	0	3
2	4	5	7	0	1	3	6
0	1	2	3	4	5	6	7

$O(n^2)$

$O(n \log n)$ 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

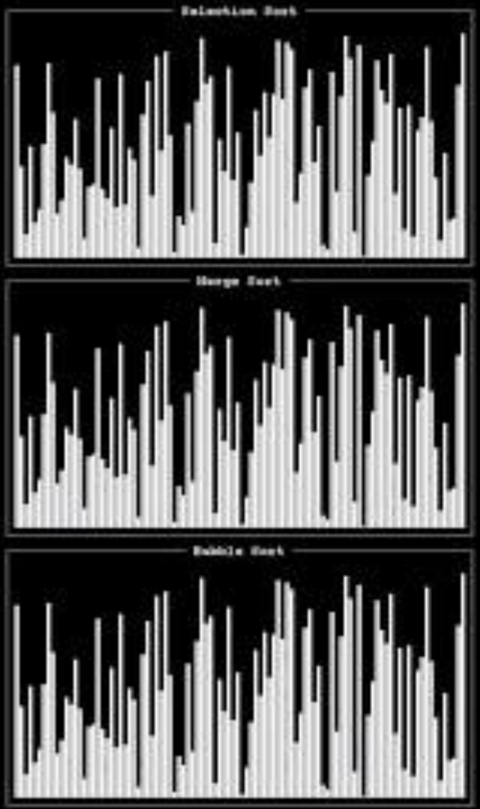
$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$

time

space


Selection Sort
Merge Sort
Bubble Sort

This is CS50