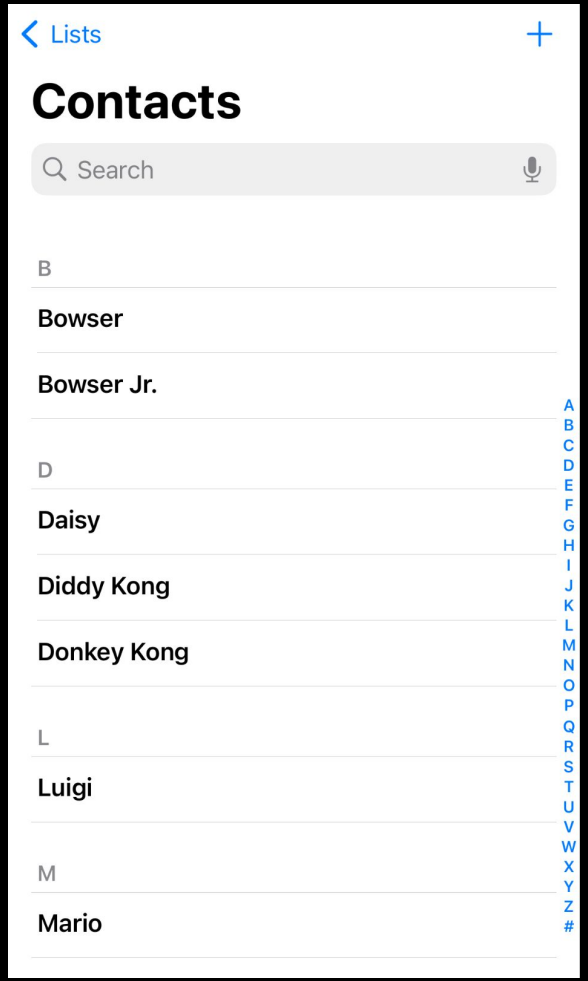
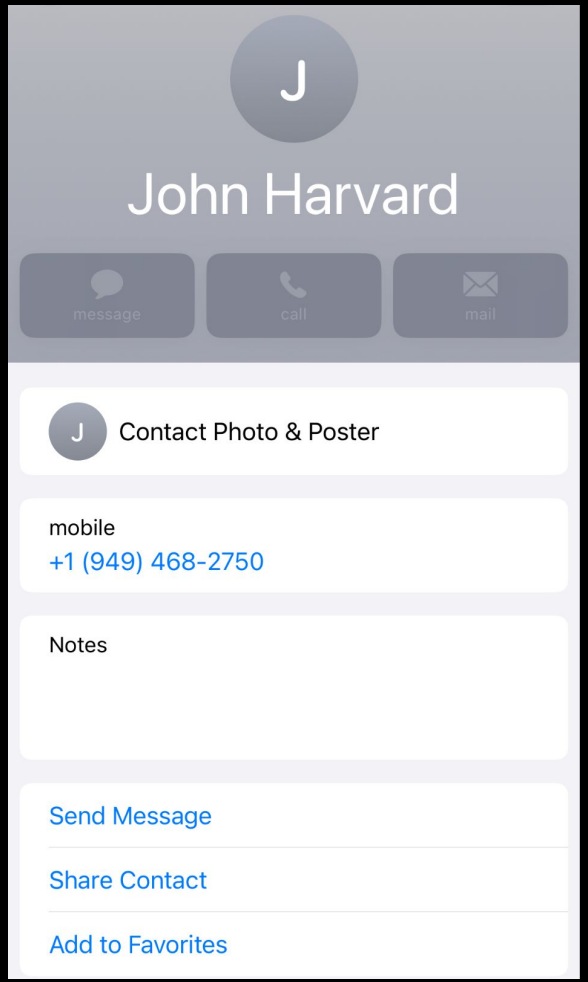
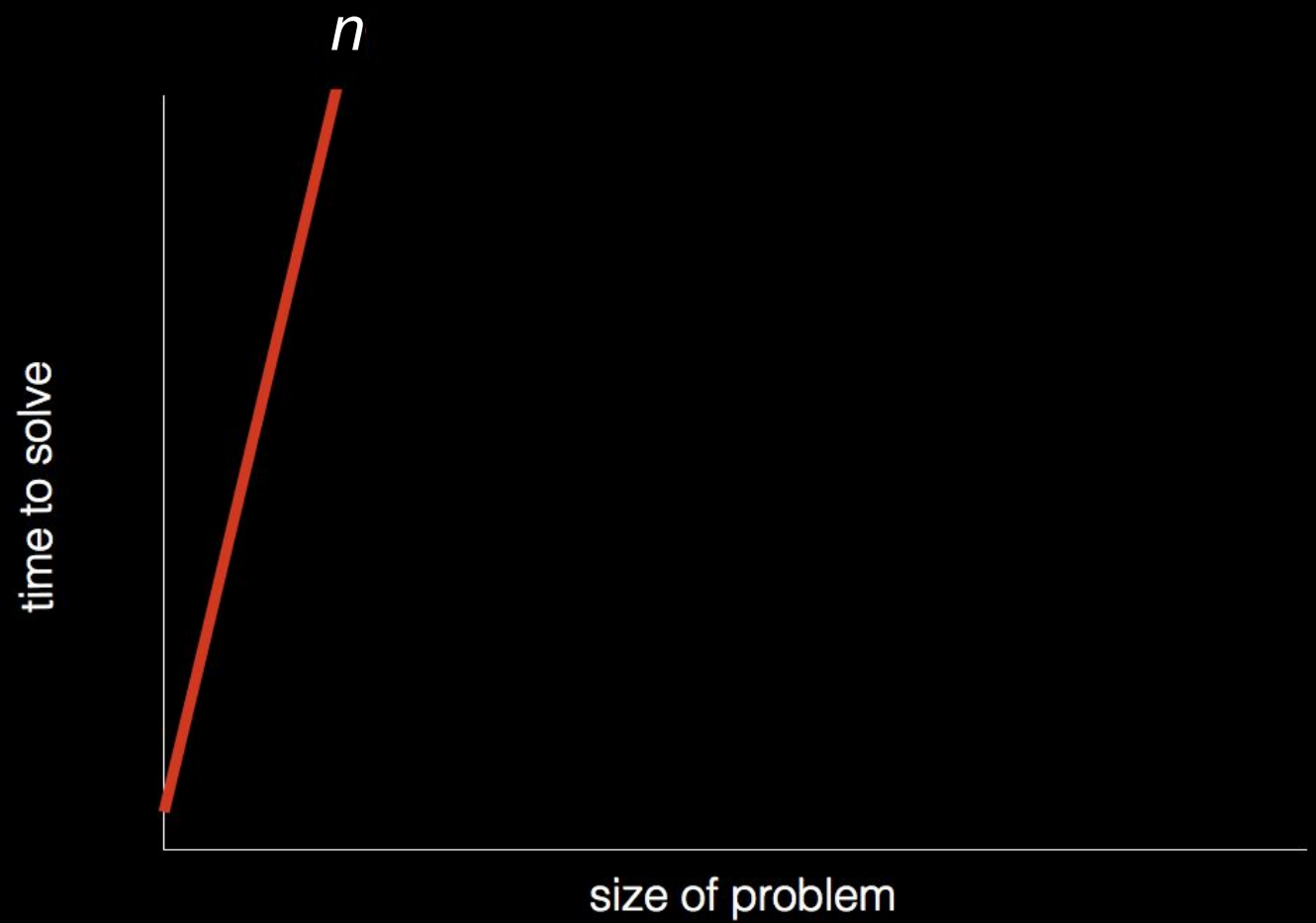
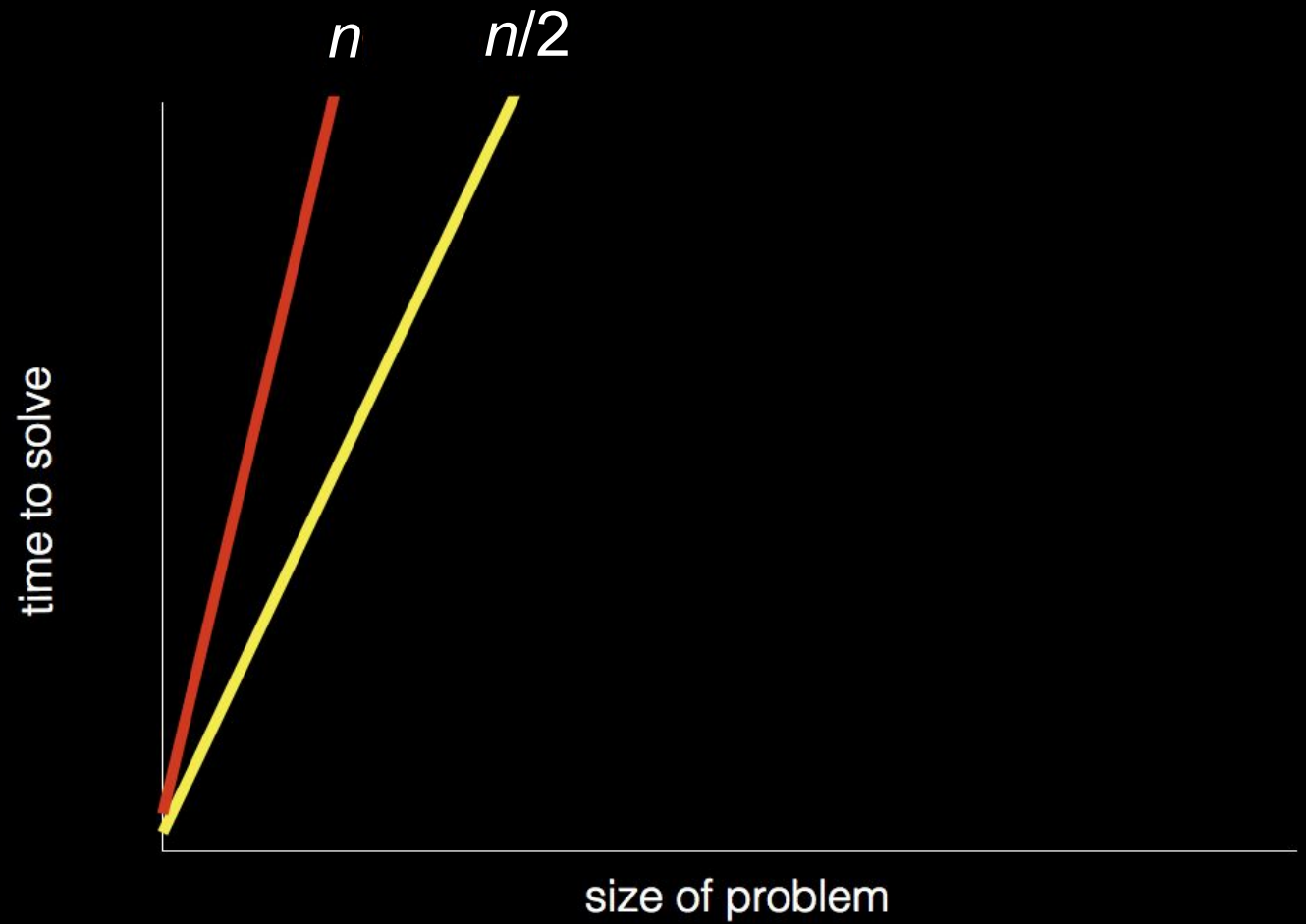
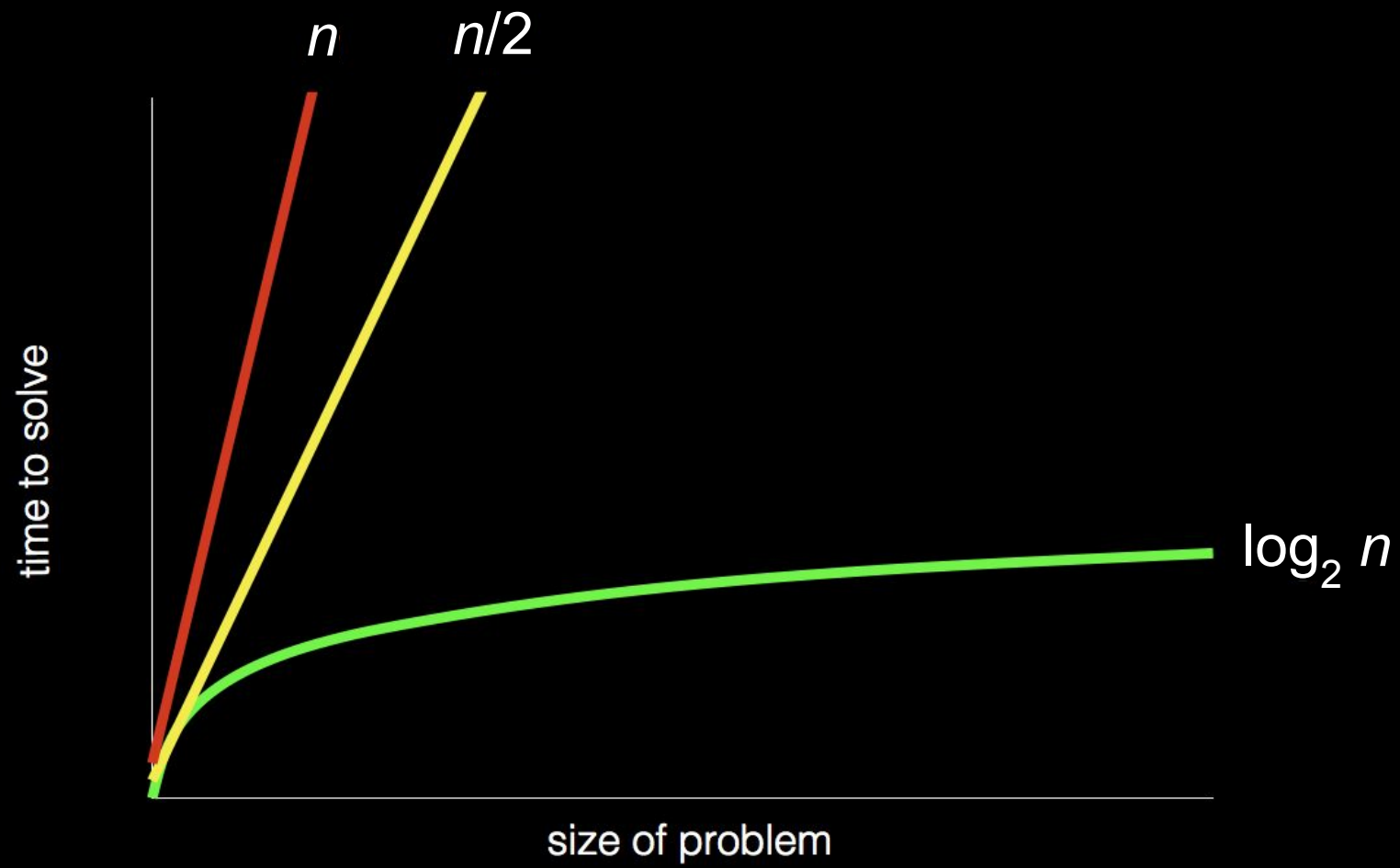
This is CS50


Lists
Contacts
Search
B
Bowser
Bowser Jr.
D
Daisy
Diddy Kong
Donkey Kong
L
Luigi
M
Mario
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
#

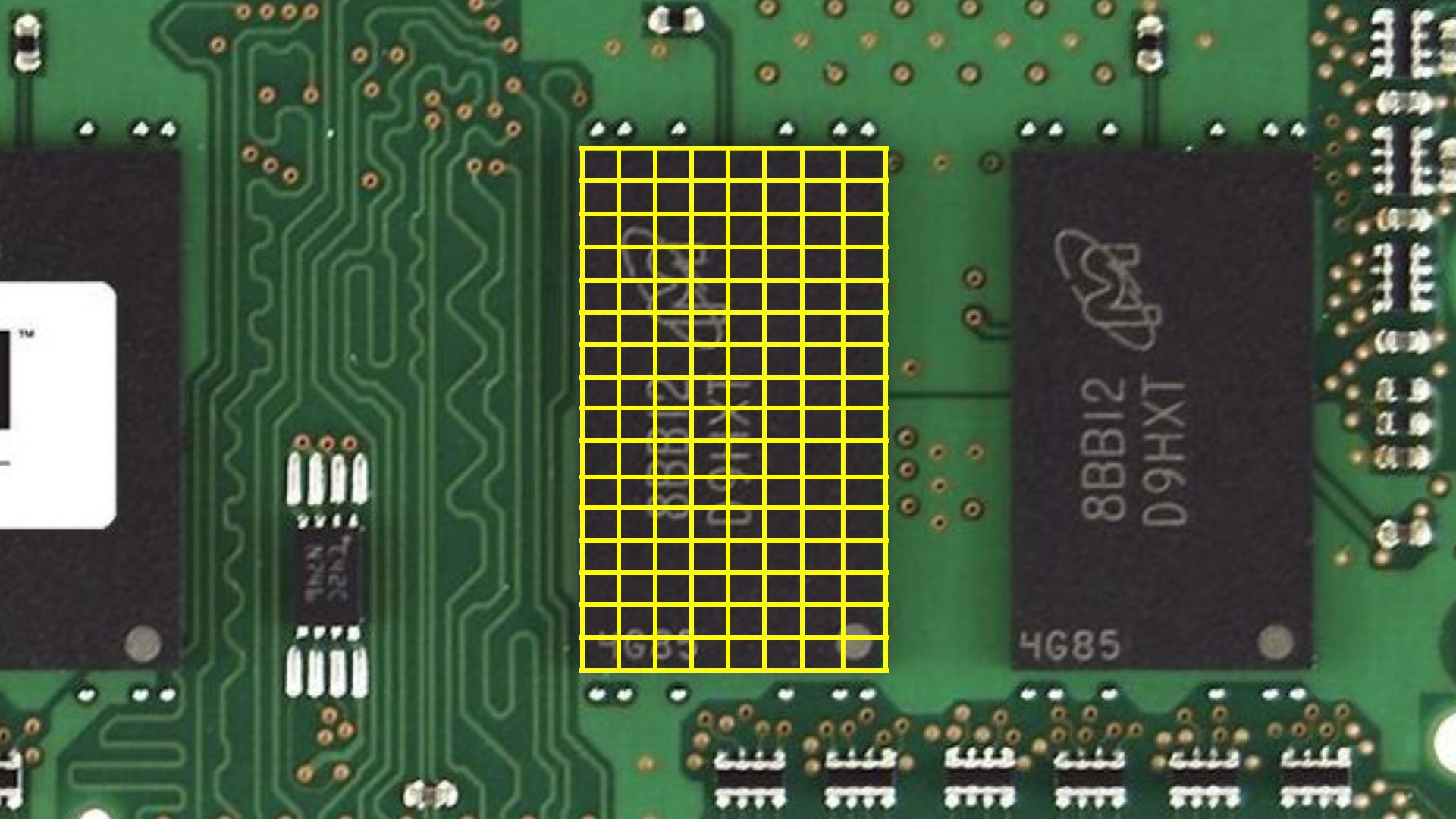

J
John Harvard
message
call
mail
J
Contact Photo & Poster
mobile
+1 (949) 468-2750
Notes
Send Message
Share Contact
Add to Favorites

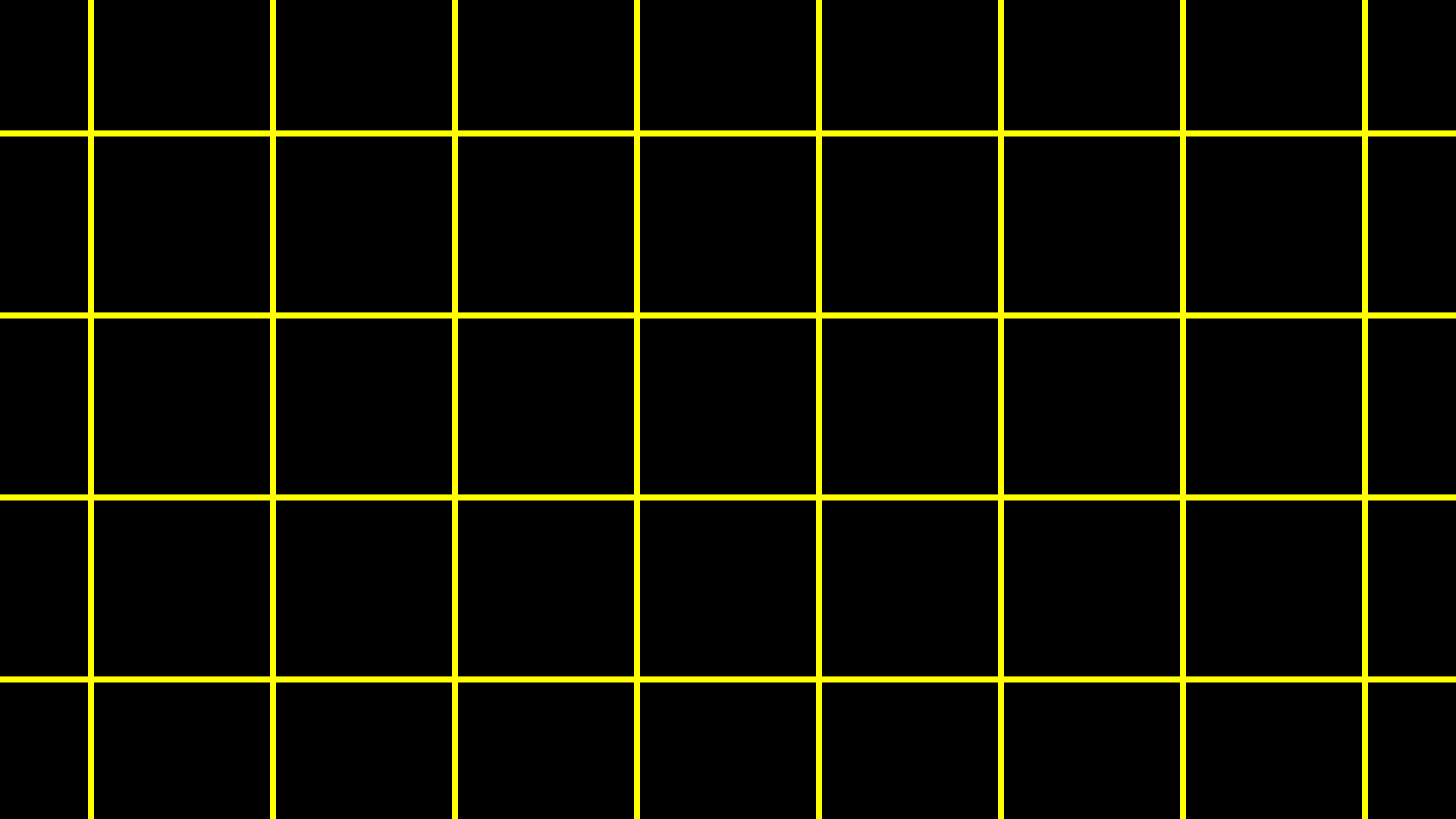

n
time to solve
size of problem

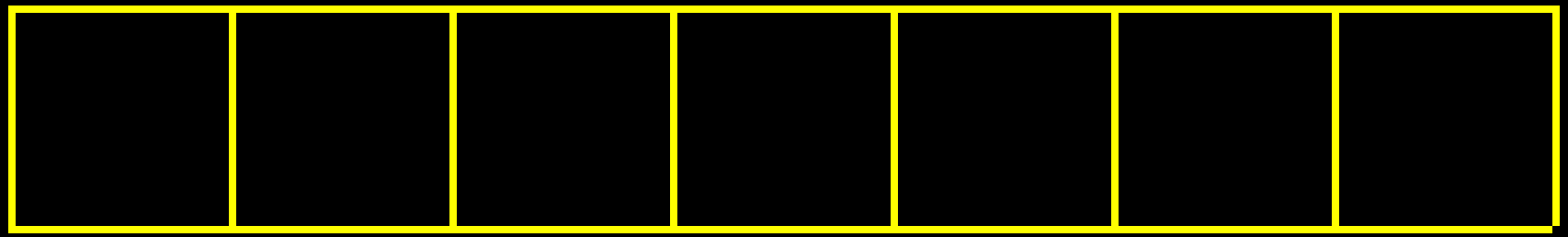

n
n/2
time to solve
size of problem

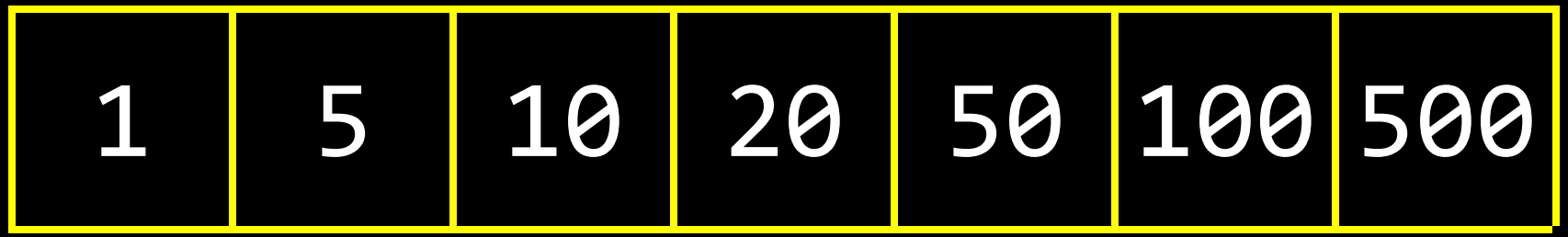

n
n/2
time to solve
log2 n
size of problem

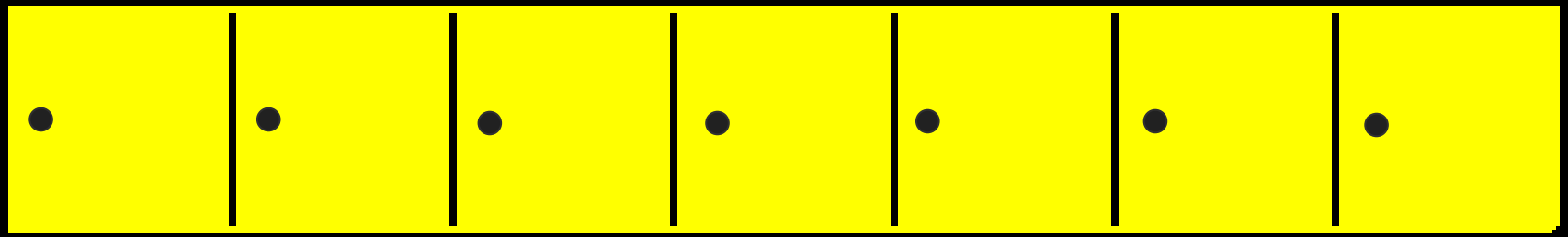

8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85

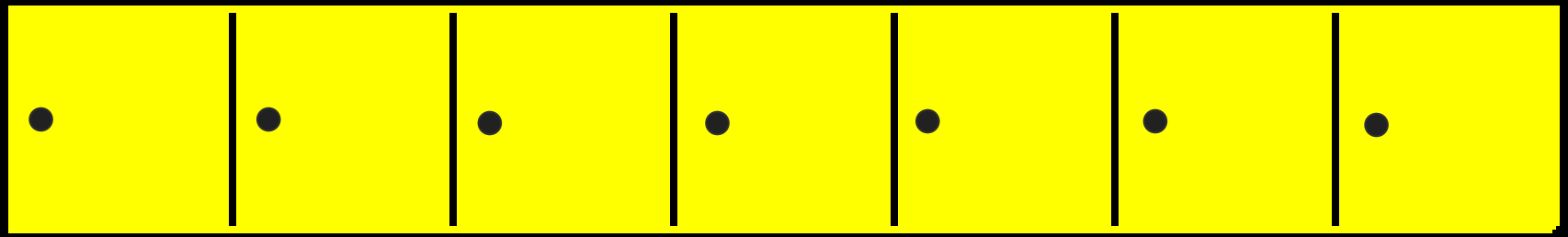

8BBI2
D9HXT
4G85

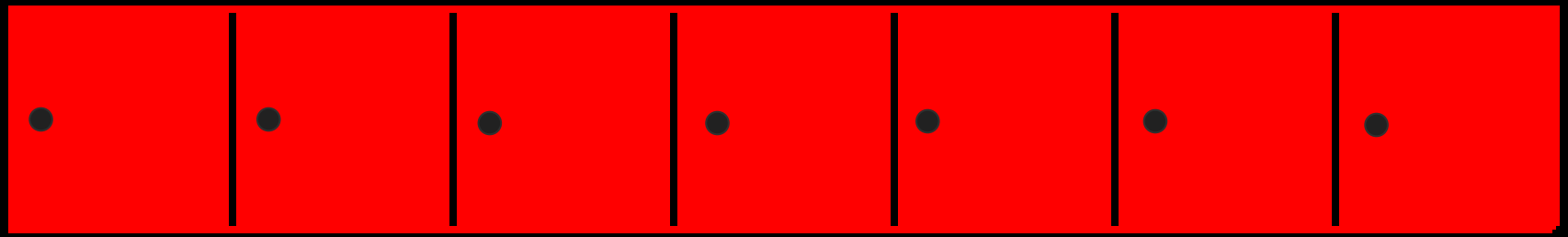




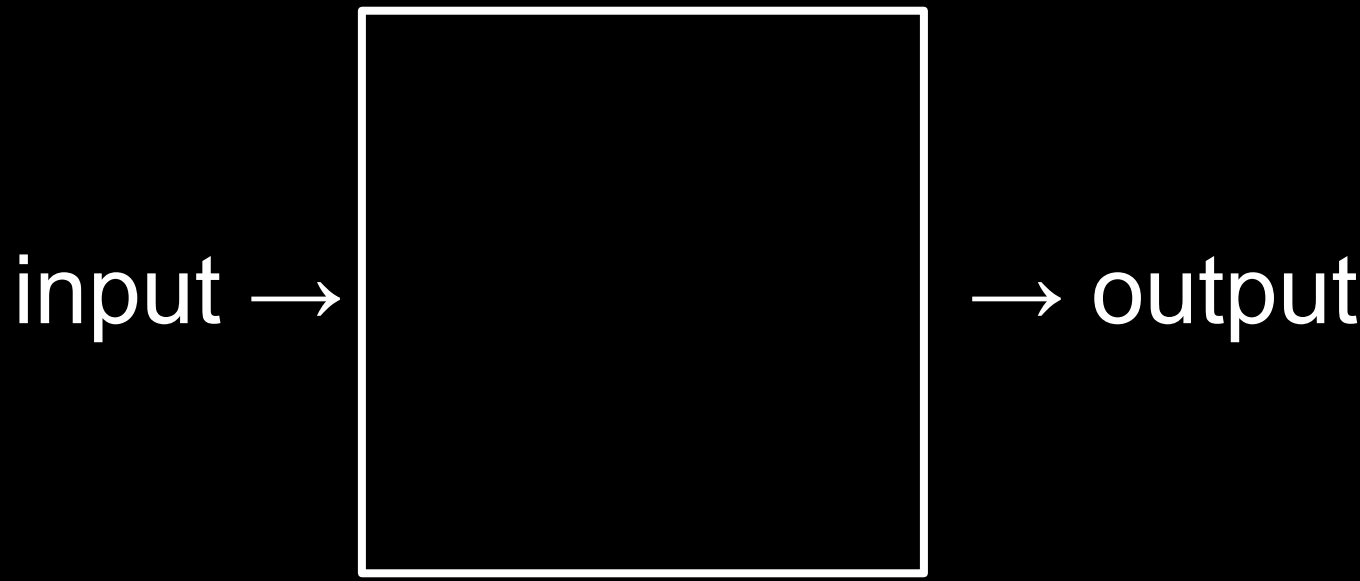

1
5
10
20
50
100
500

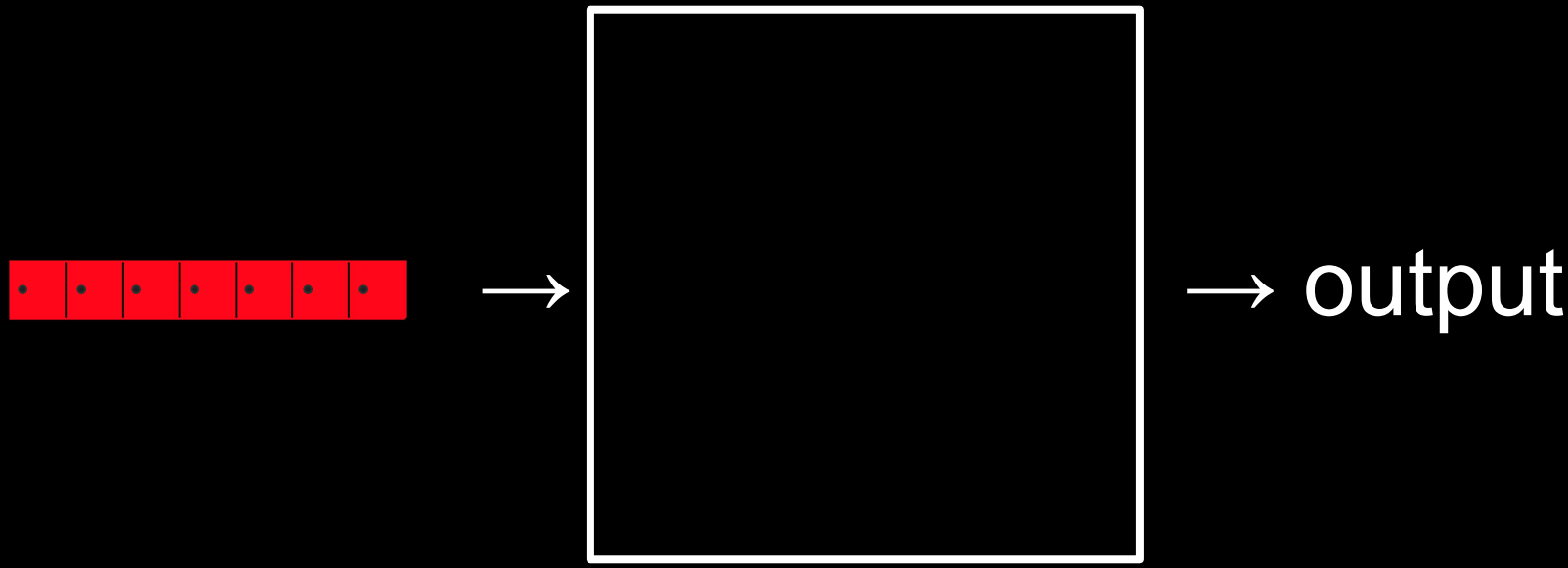


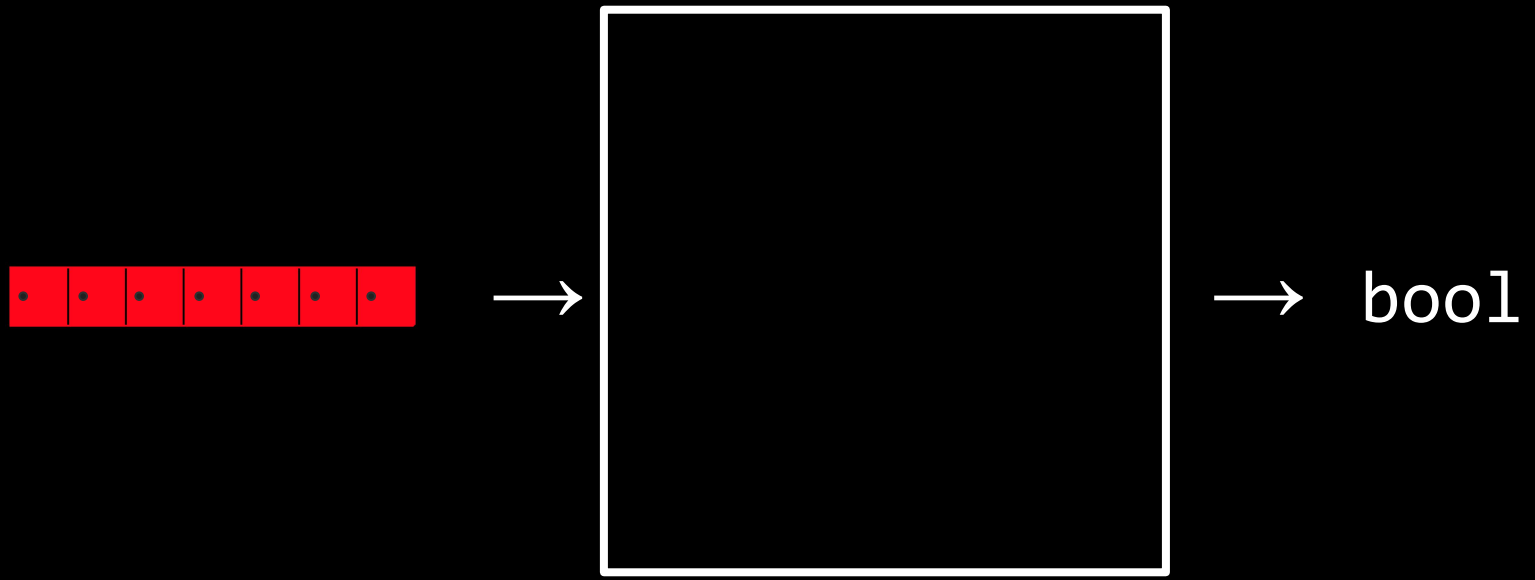

[0]
[1]
[2]
[3]
[4]
[5]
[6]

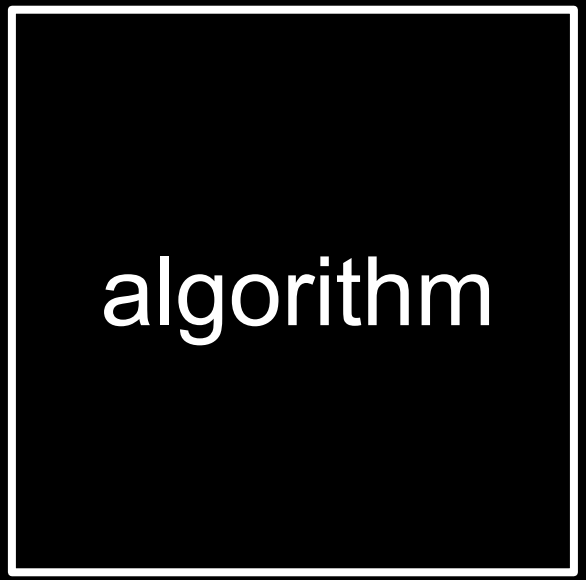

[0]
[1]
[2]
[3]
[4]
[5]
[6]

searching

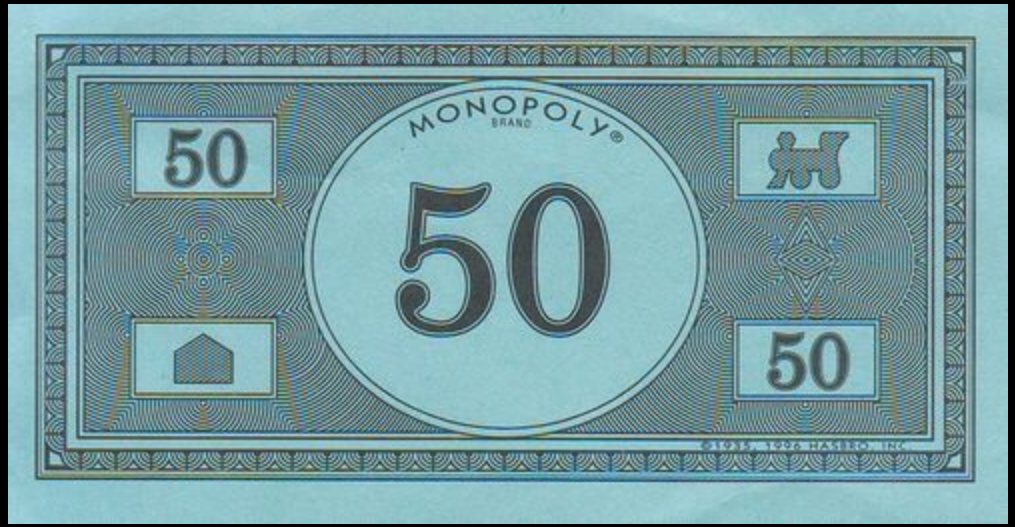

input →
→ output


output


bool


algorithm


MONOPOLY
20
50
10
1
100
500
5


MONOPOLY®
BRAND
50
50
50
©1935 1936 HASBRO INC

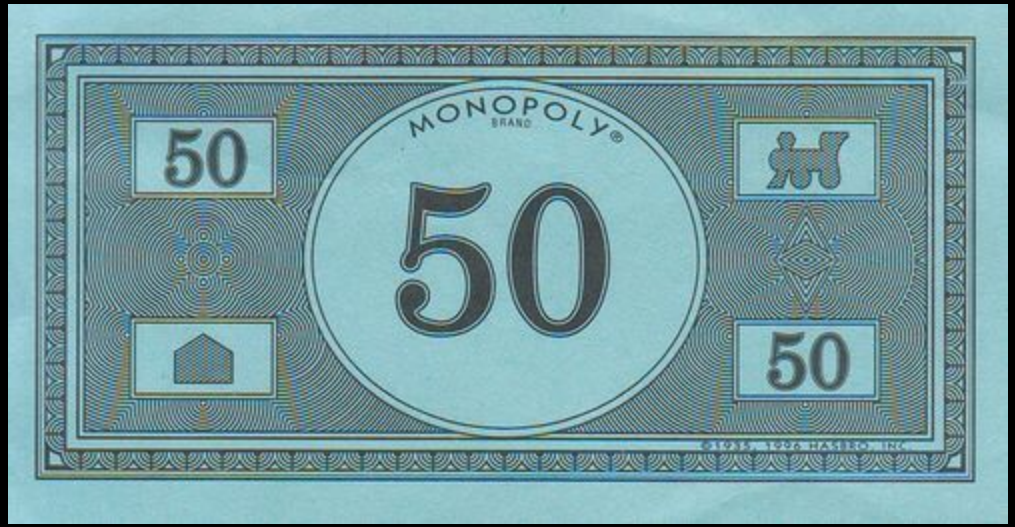
linear search

```
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 1936 HASBRO INC

binary search

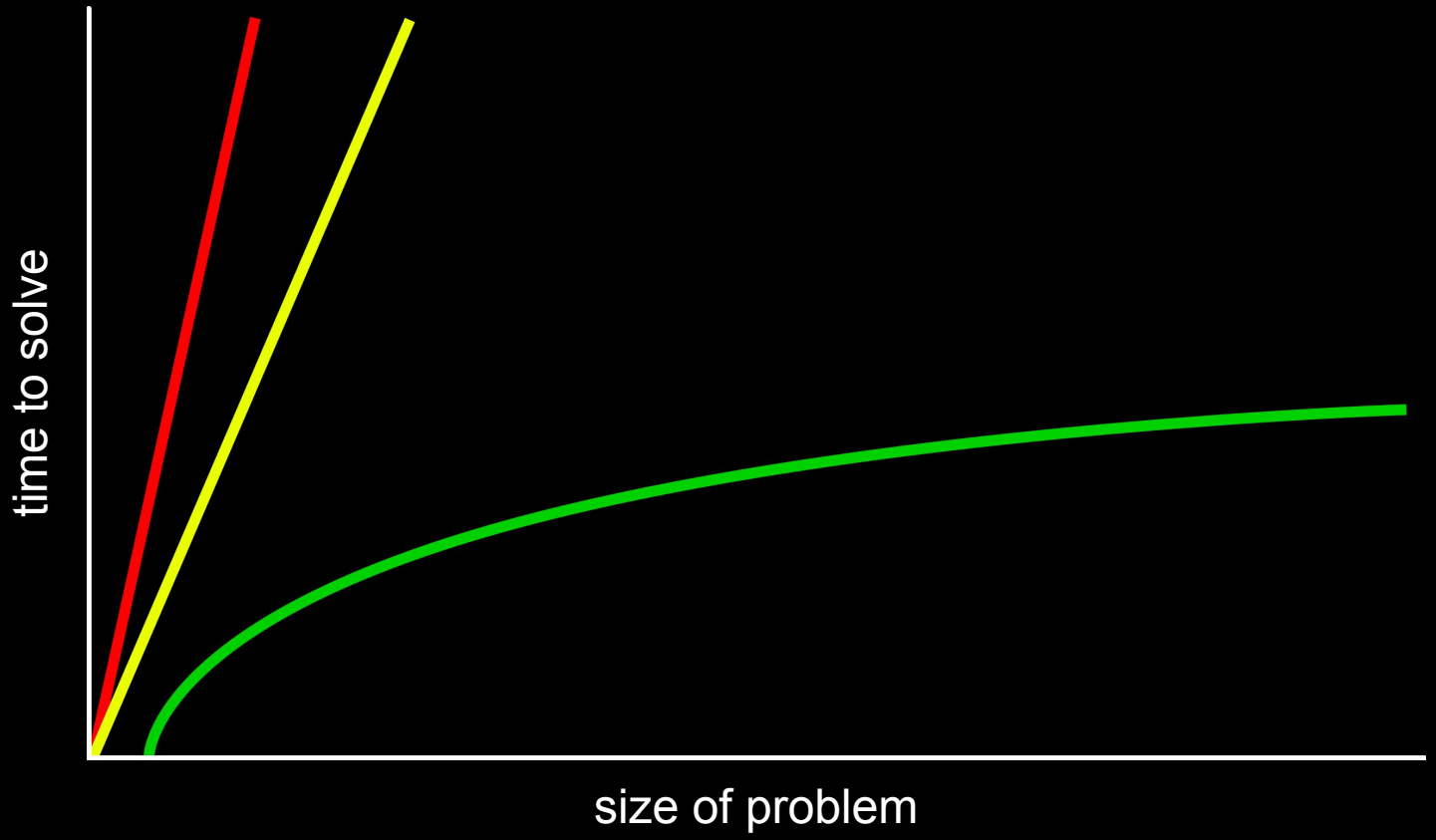
```
If 50 is behind middle door
    Return true
Else if 50 < middle door
    Search left half
Else if 50 > middle door
    Search right half
```

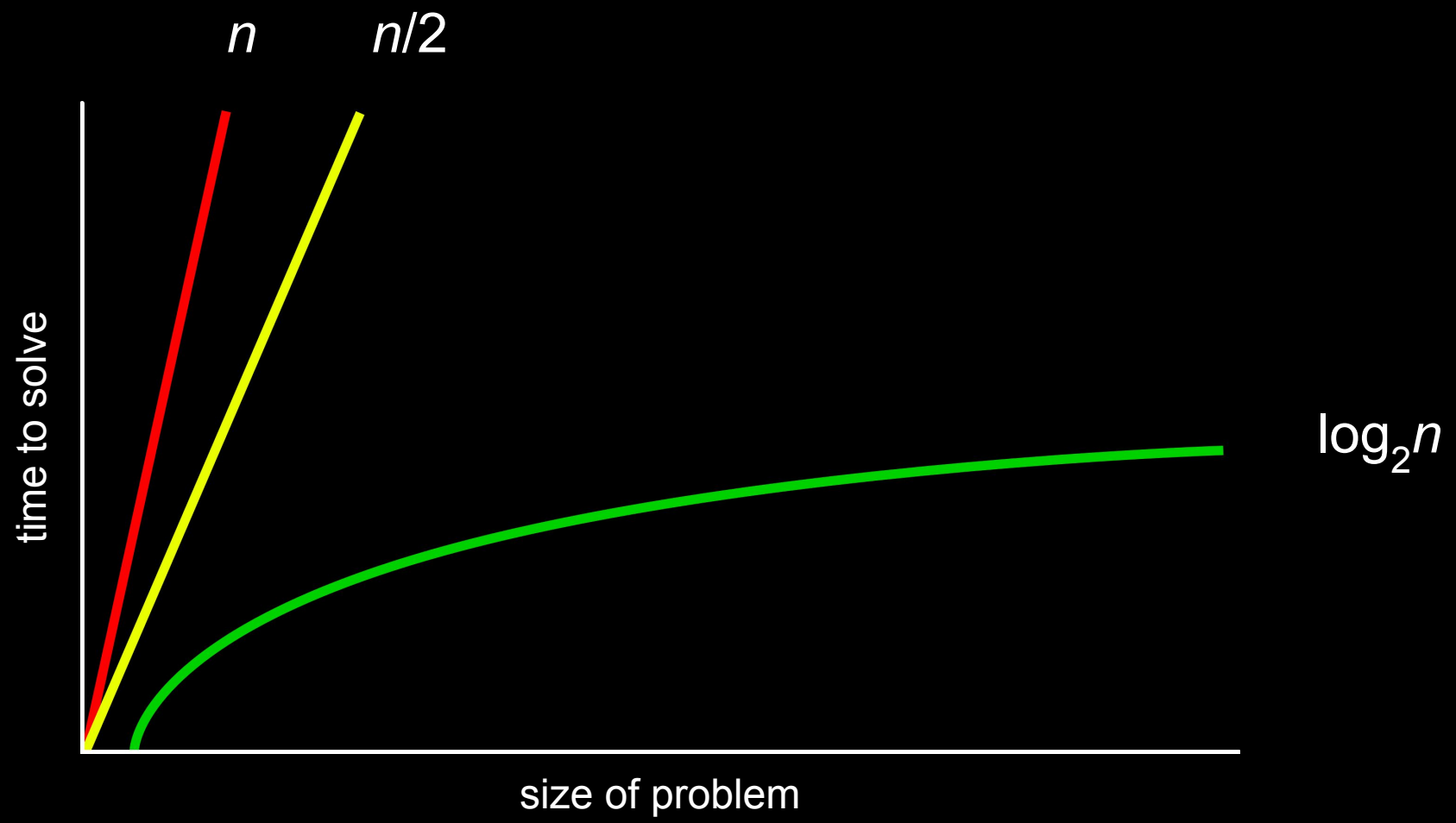
```
If no doors left

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

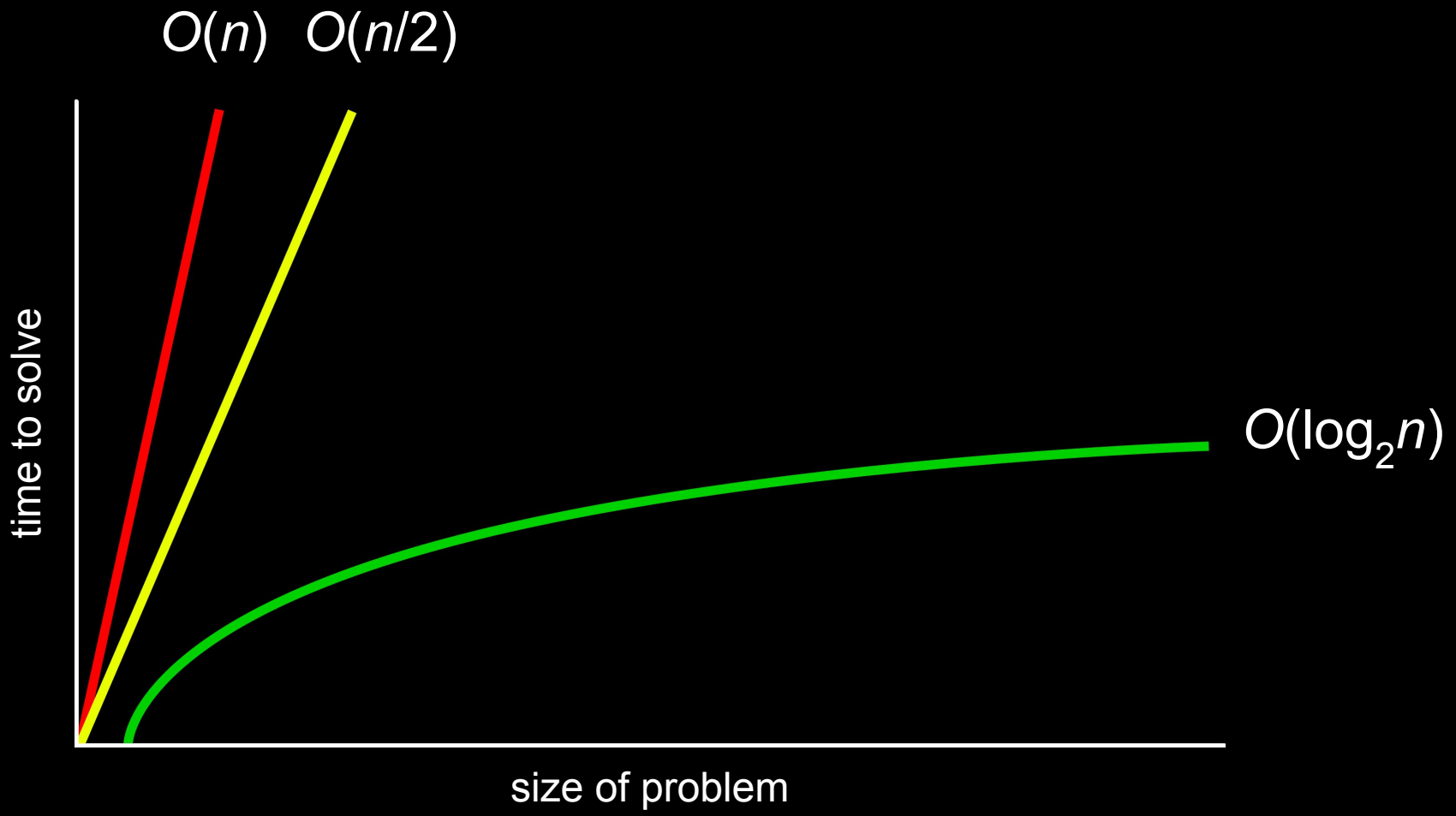
```
If no doors left
    Return false
If 50 is behind middle door
    Return true
Else if 50 < middle door
    Search left half
Else if 50 > middle door
    Search right half
```

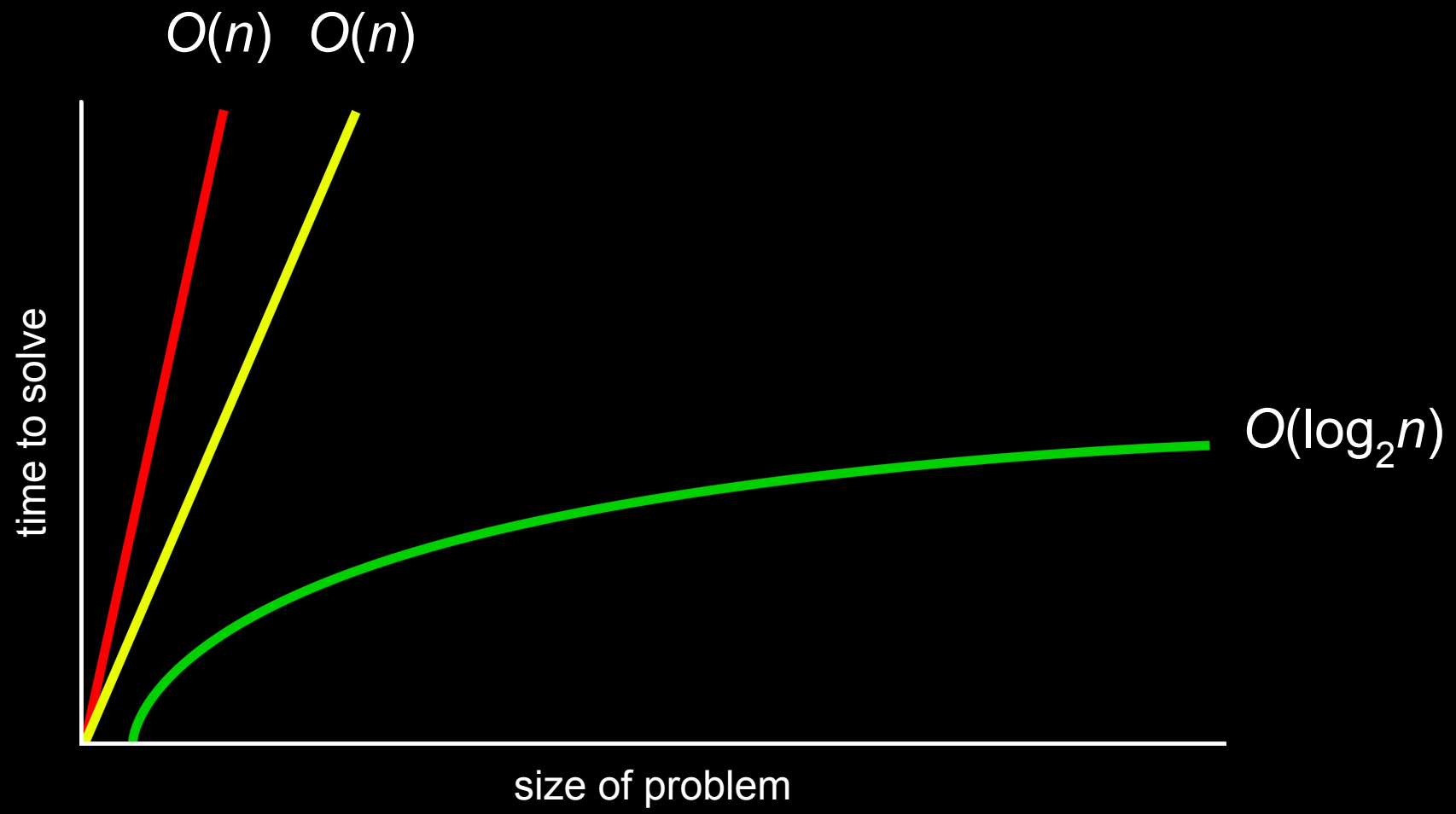
```
If no doors left
    Return false
If 50 is behind doors[middle]
    Return true
Else if 50 < doors[middle]
    Search doors[0] through doors[middle - 1]
Else if 50 > doors[middle]
    Search doors[middle + 1] through doors[n - 1]
```

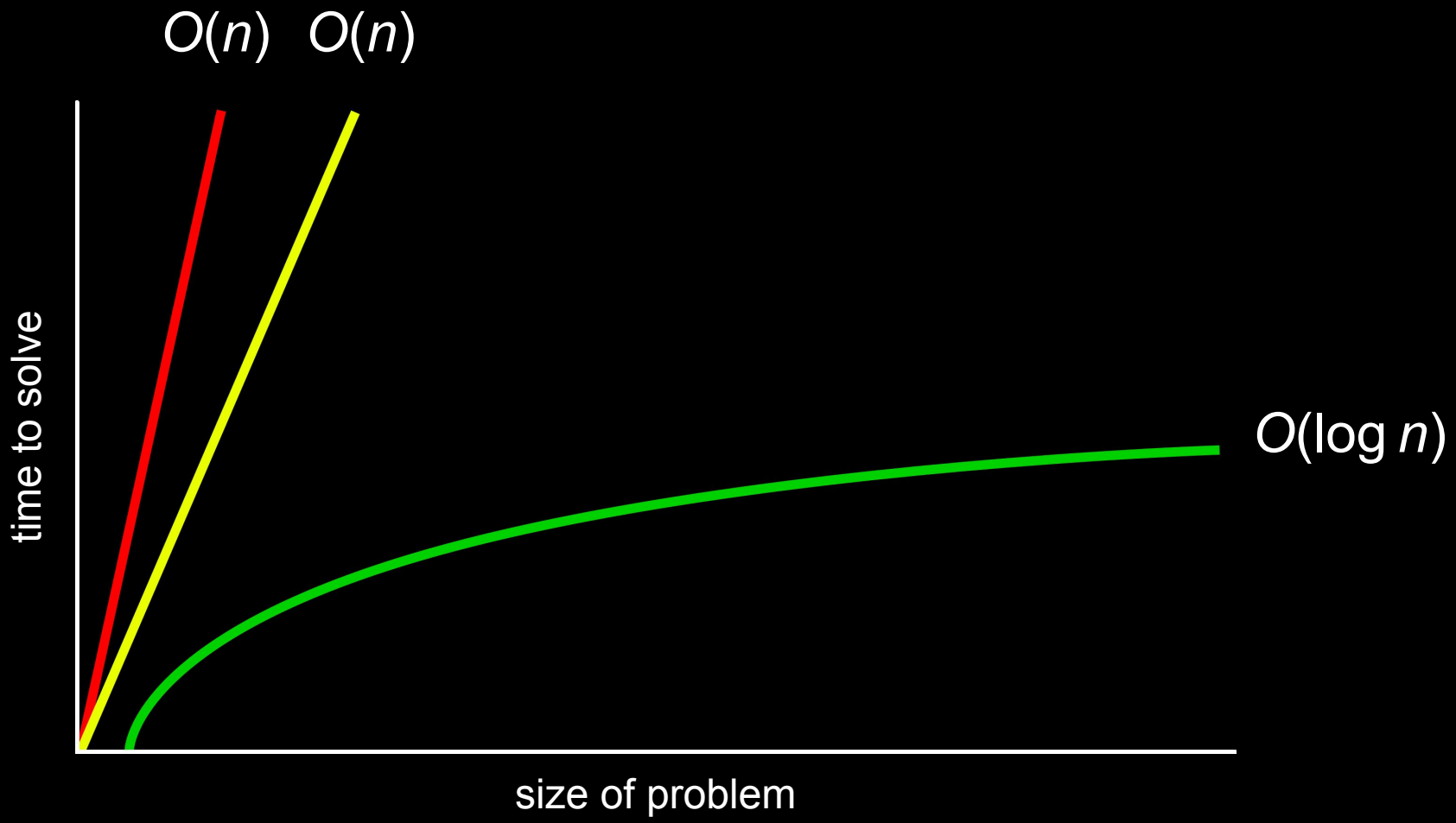
running time

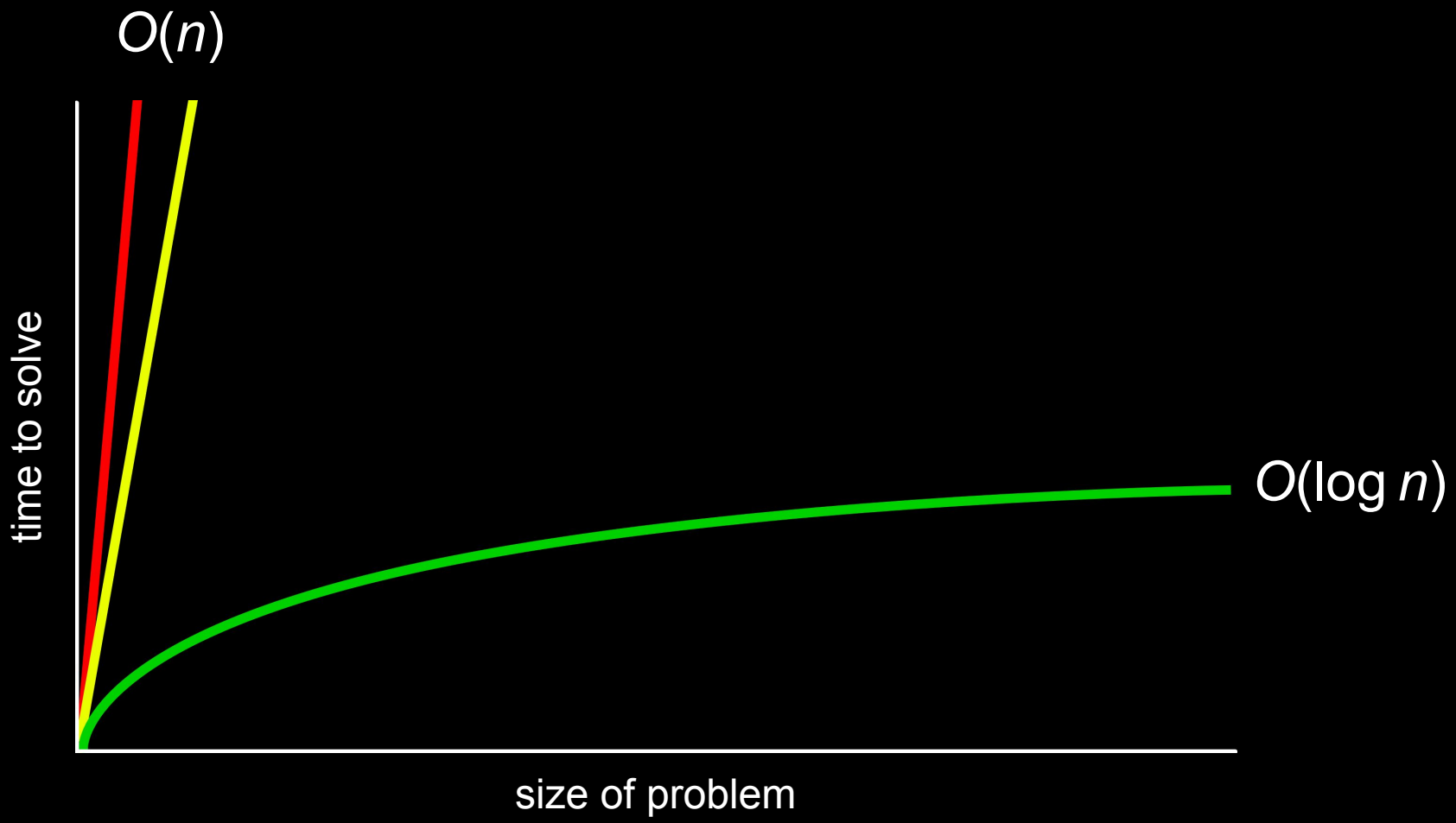

time to solve
size of problem


n
n/2
time to solve
log2n
size of problem


O(n)
O(n/2)
O(log2n)
time to solve
size of problem


O(n)
O(n)
O(log2n)
time to solve
size of problem


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


O(n)
time to solve
O(log n)
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

$O(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

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

$\Theta(n^2)$

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$

asymptotic notation

linear search

string.h

manual.cs50.io/#string.h

strcmp

data structures

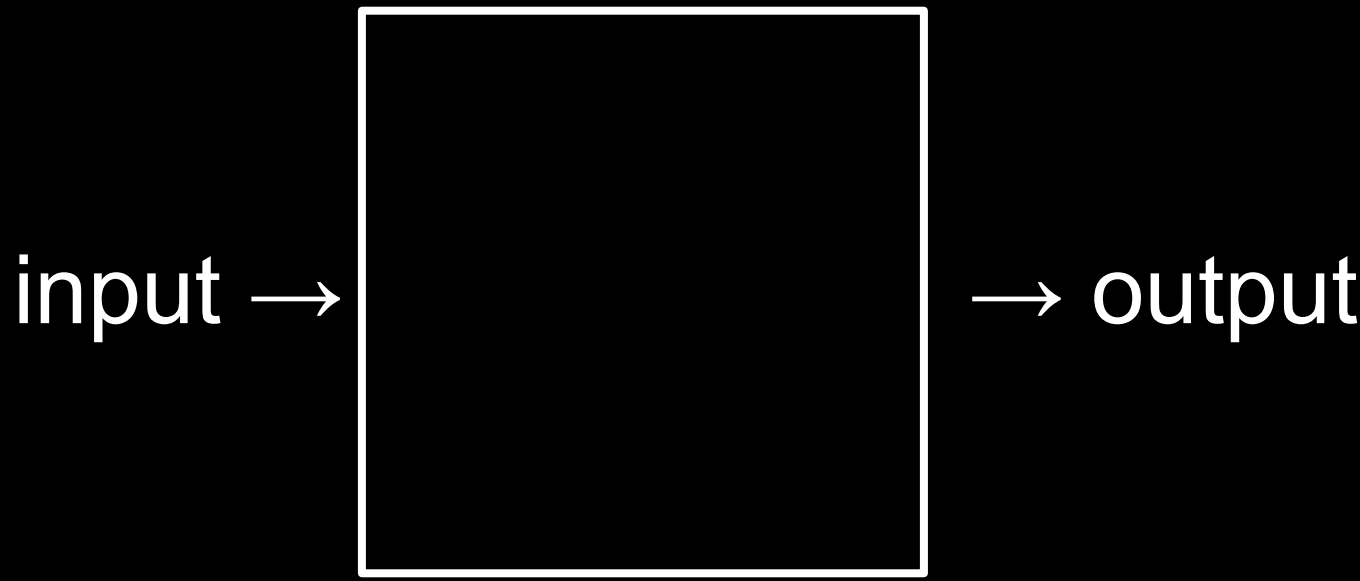
```
person people[]
```

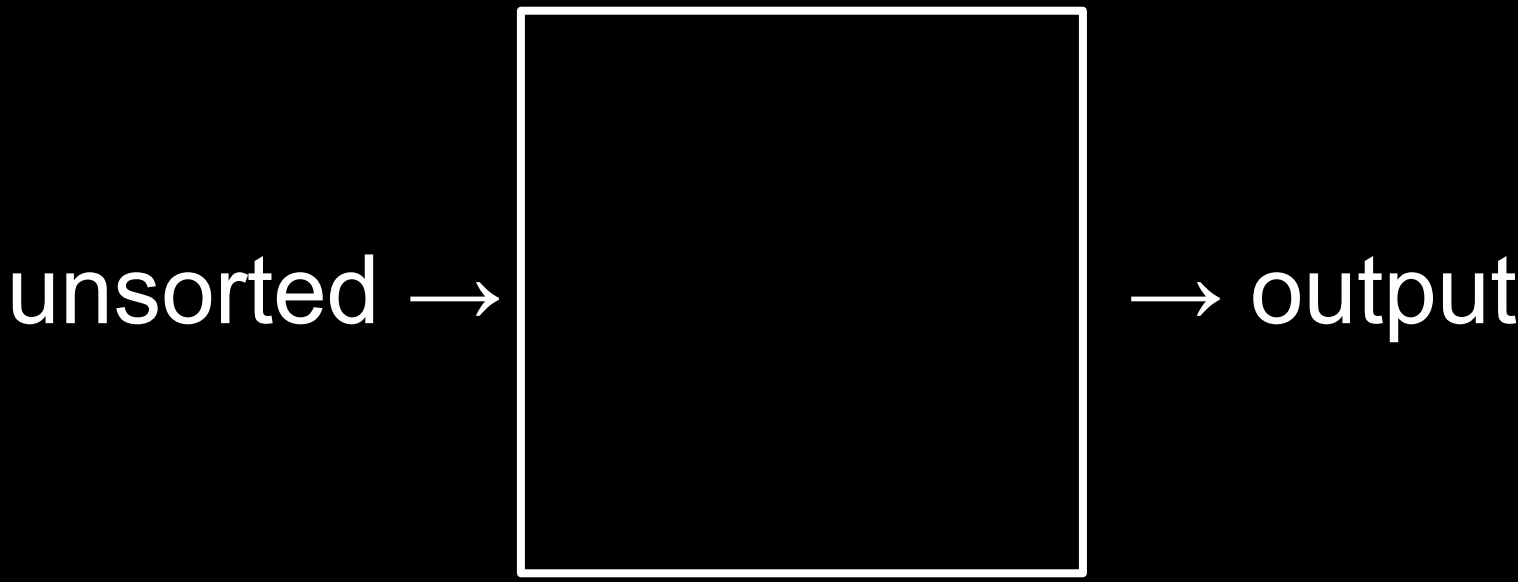
```
string name;
string number;
```

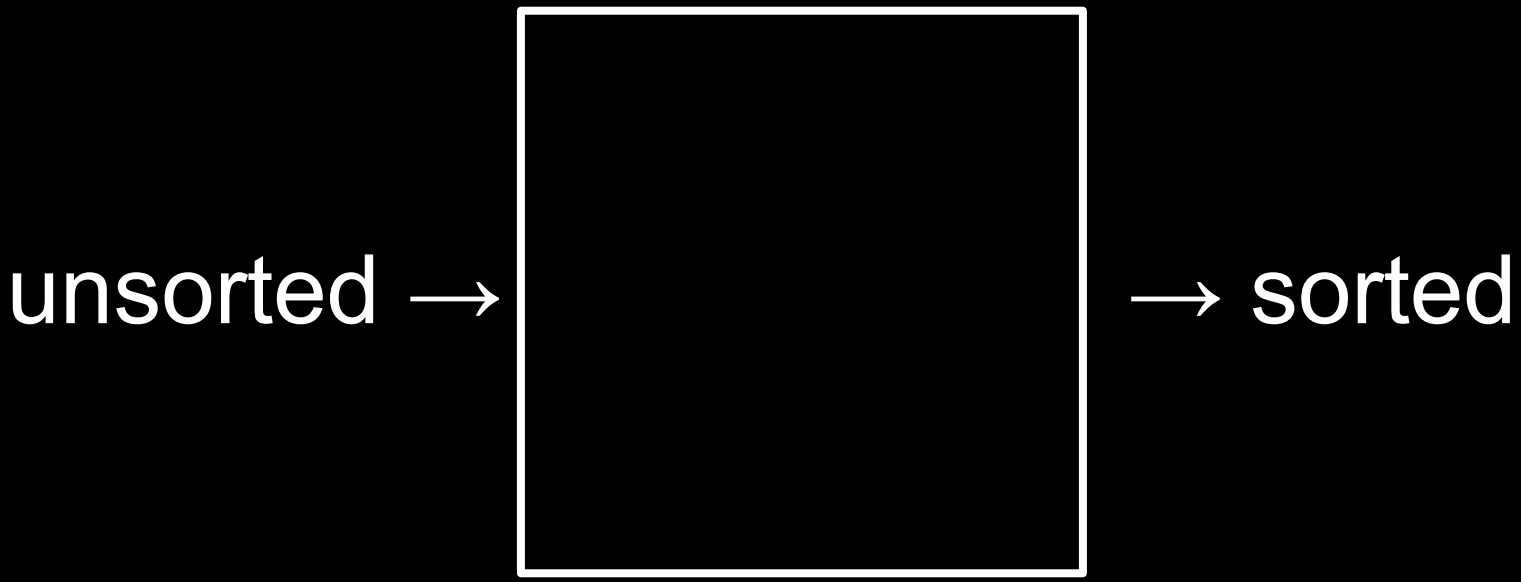
```
typedef struct
{
    string name;
    string number;
}
person;
```

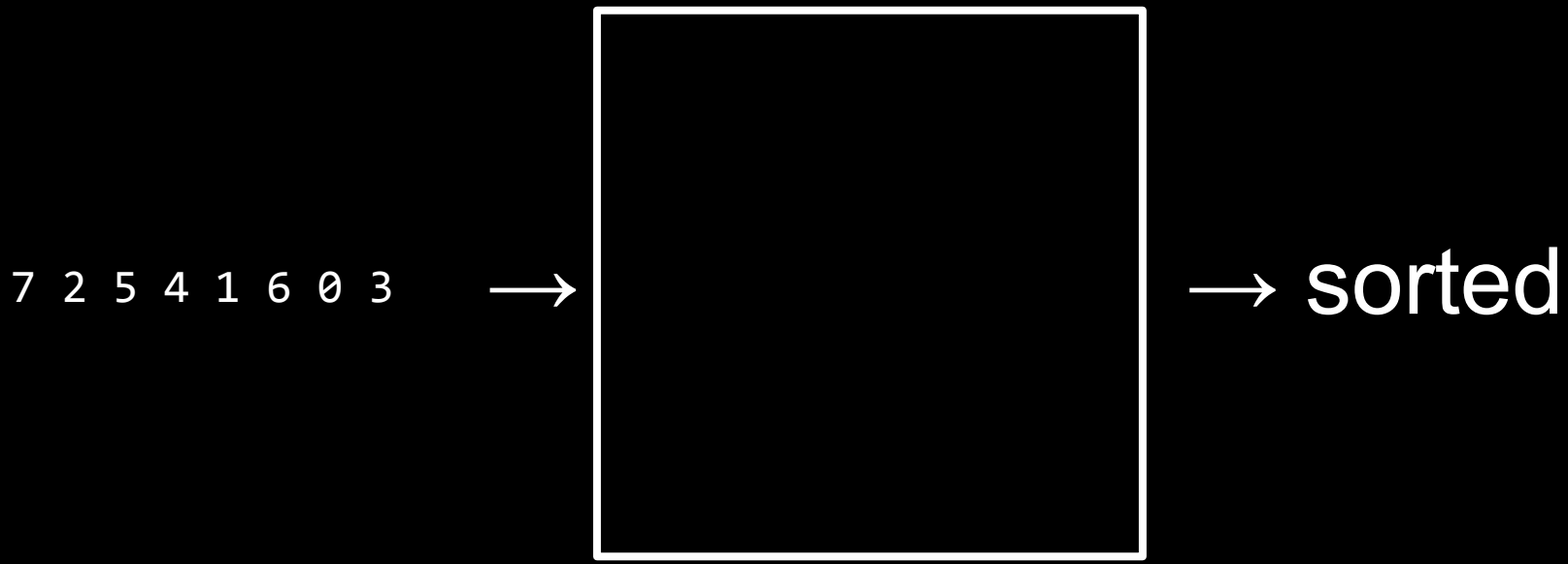
```
typedef struct
{
    string name;
    string number;
} person;
```

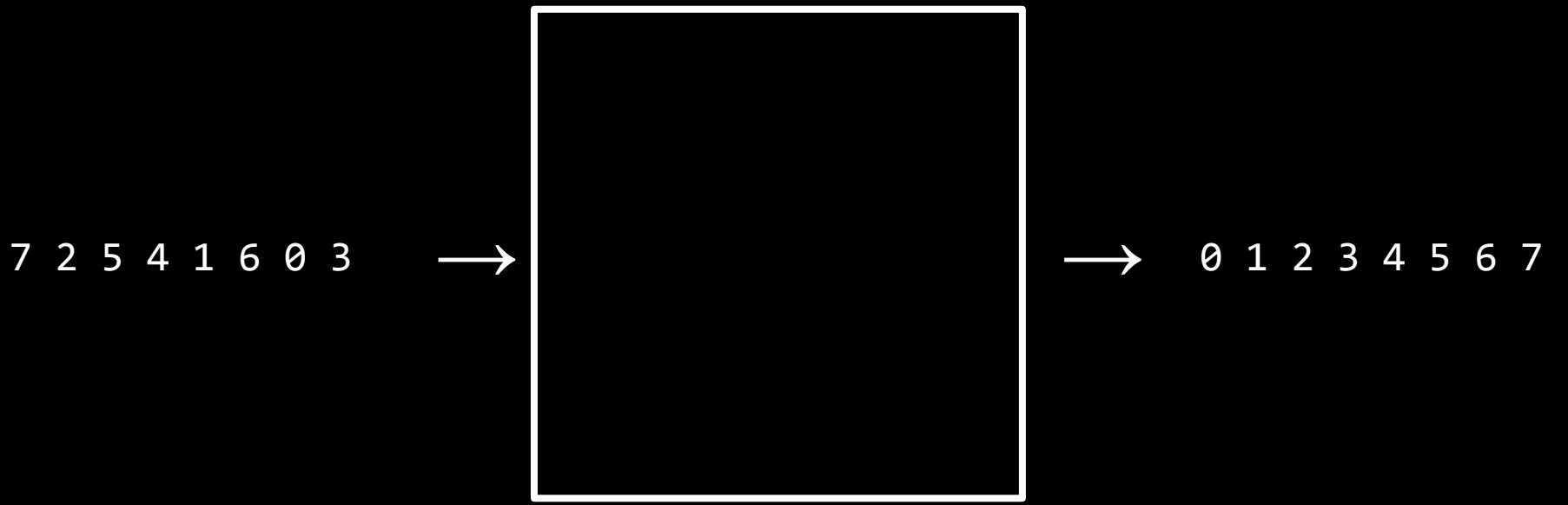
sorting


input →
→ output


unsorted →
→ output


unsorted →
→ sorted

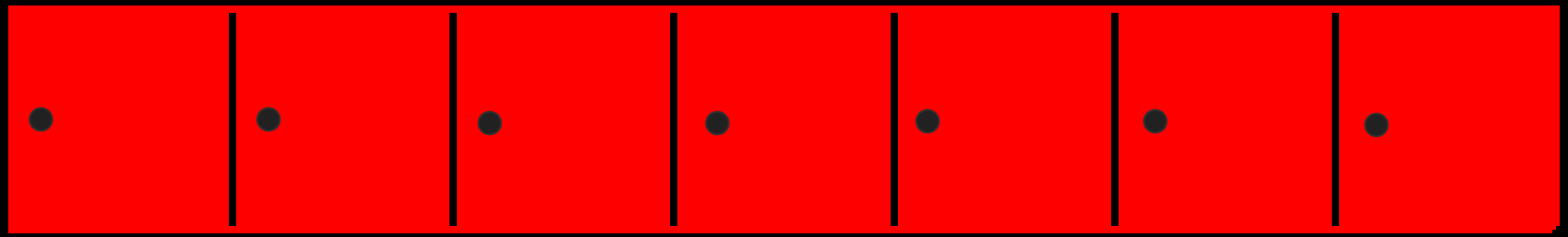

7 2 5 4 1 6 0 3
sorted


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

7 2 5 4 1 6 0 3


30 BAGS
30 BAGS
NO HFCS
NO Artificial Colors or Flavors
meiji
Hello PANDA
Chocolate
WITH OTHER NATURAL FLAVORS
Bite Size
Crème Center with Crunchy Shell
CHOCOLATE FUN WITH A CRACKER CRUNCH.
Made in USA
30 - 0.75 OZ (21g) BAGS
NET WT 22.5 OZ (1 LB 6.5 OZ) 638g
Each Bag is Packaged for Resale

selection sort

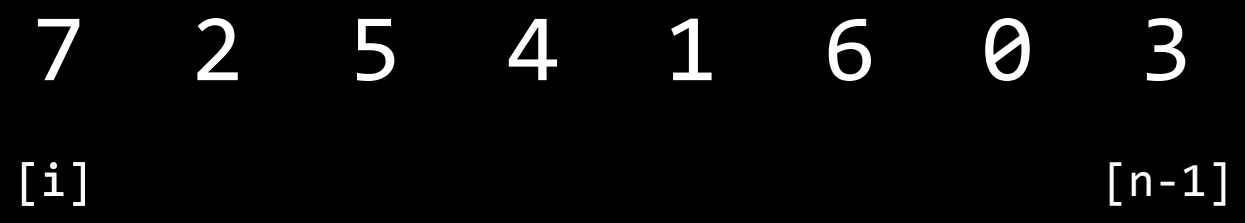

[0]
[1]
[2]
...
[n-3]
[n-2]
[n-1]

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

7 2 5 4 1 6 0 3

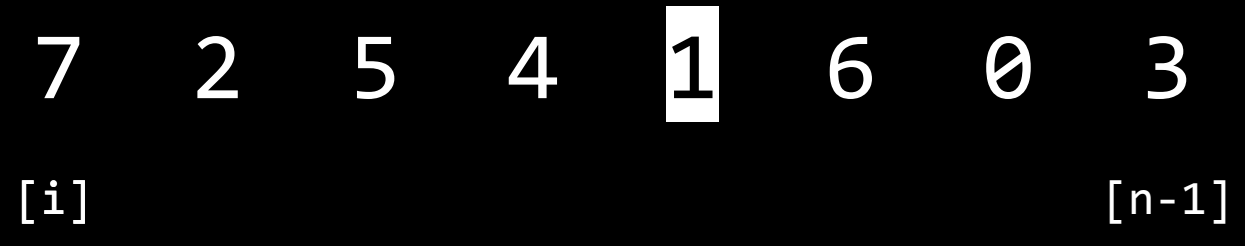
7 2 5 4 1 6 0 3

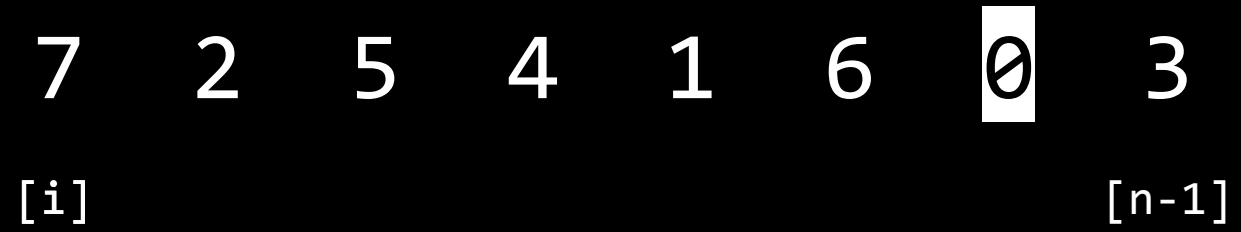
[i]


7 2 5 4 1 6 0 3
[i]
[n-1]


7 2 5 4 1 6 0 3
[i]
[n-1]


7 2 5 4 1 6 0 3
[i]
[n-1]


7 2 5 4 1 6 0 3
[i]
[n-1]

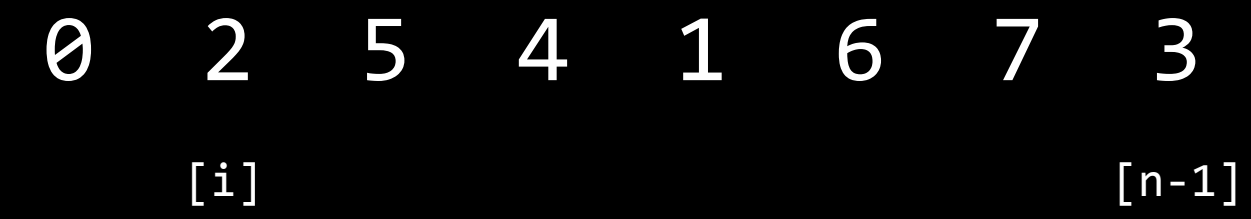

7 2 5 4 1 6 0 3
[i]
[n-1]


2 5 4 1 6 0 3
[i]
[n-1]


0 2 5 4 1 6 3
[i]
[n-1]


0 2 5 4 1 6 7 3
[i]
[n-1]


0 2 5 4 1 6 7 3
[i]
[n-1]


0 2 5 4 1 6 7 3
[i]
[n-1]

$(n - 1)$

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

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

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

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

$n(n - 1)/2$

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

$n(n - 1)/2$

$(n^2 - n)/2$

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

$n(n - 1)/2$

$(n^2 - n)/2$

$n^2/2 - n/2$

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

$n(n - 1)/2$

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

$n^2/2 - n/2$

$O(n^2)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$ selection sort

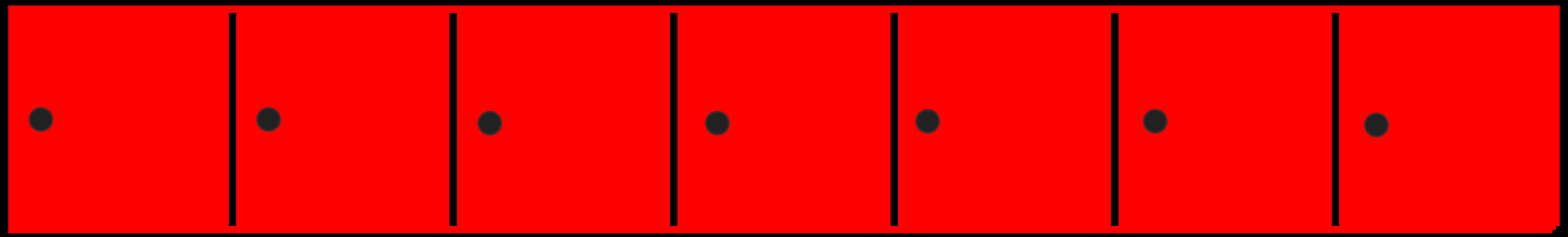
$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

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


[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

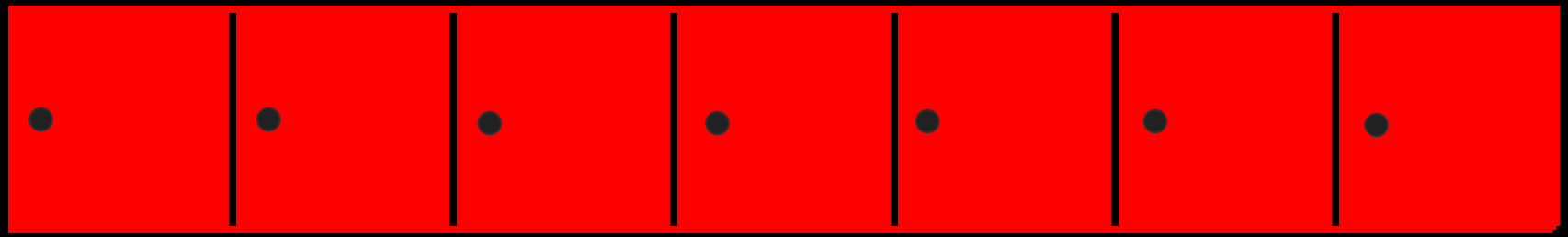
$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$

bubble sort


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

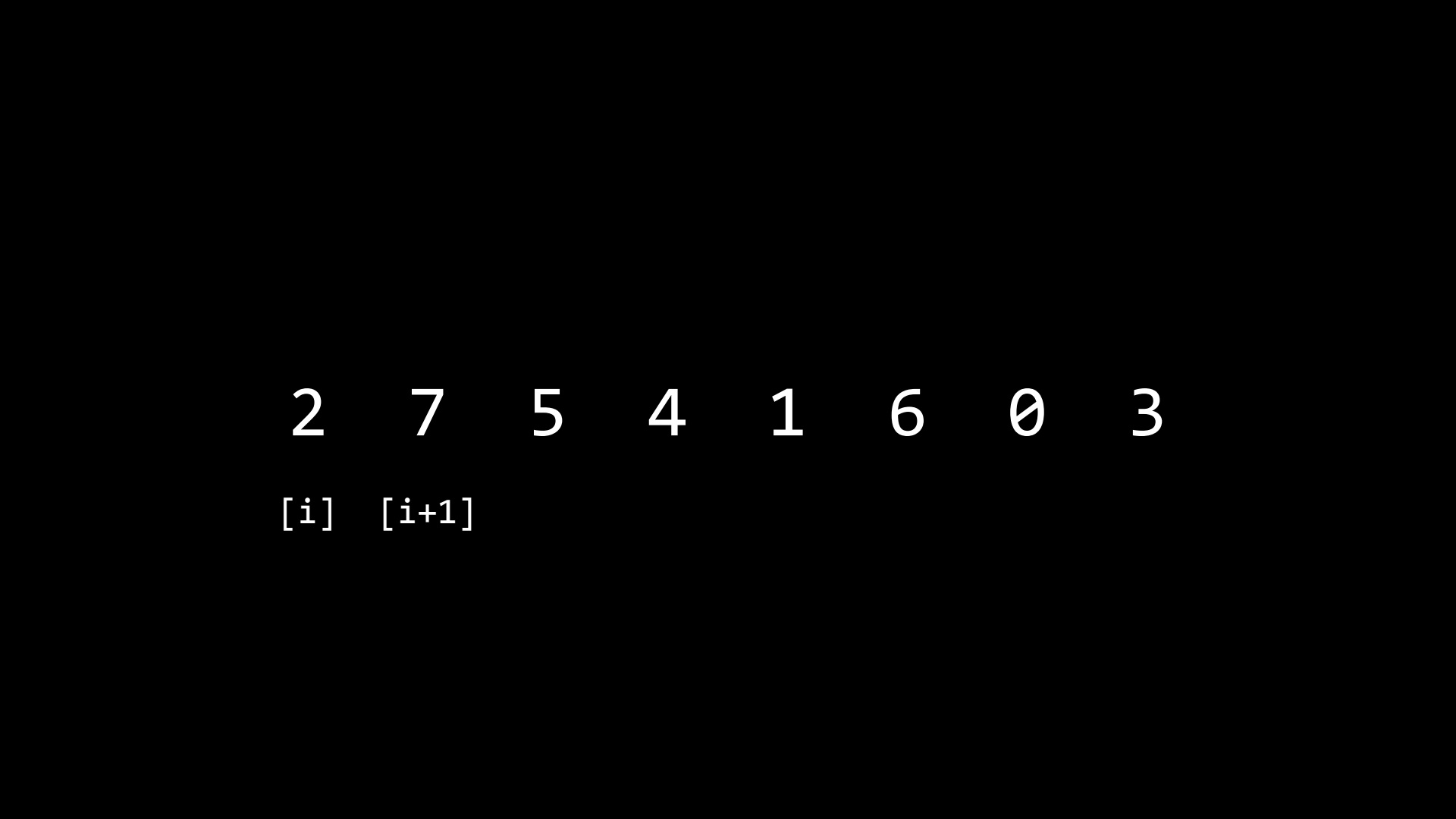
```
Repeat n times
    For i from 0 to n-2
        If numbers[i] and numbers[i+1] out of order
            Swap them
```

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

7 2 5 4 1 6 0 3

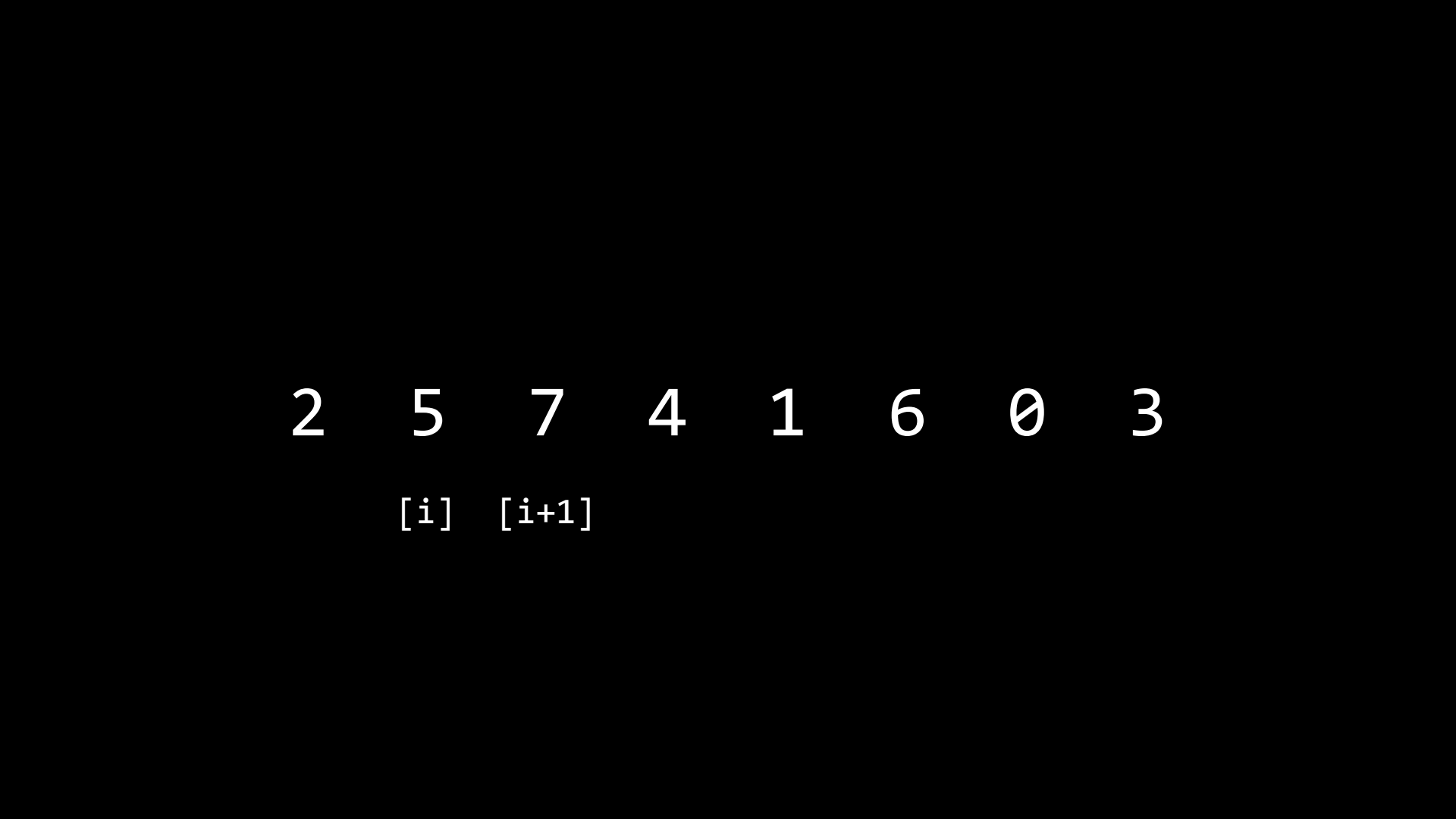
7 2 5 4 1 6 0 3

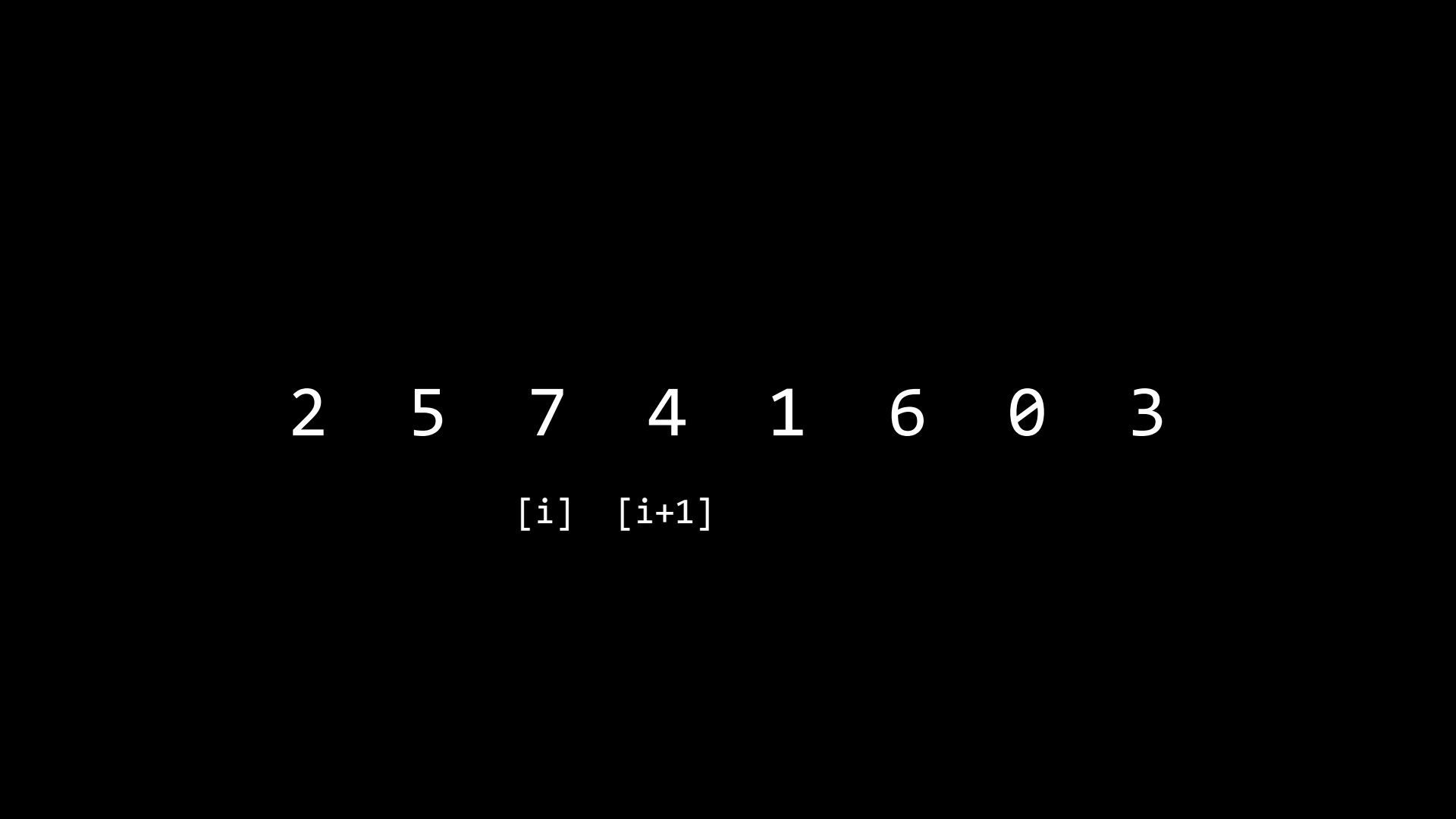
[i] [i+1]

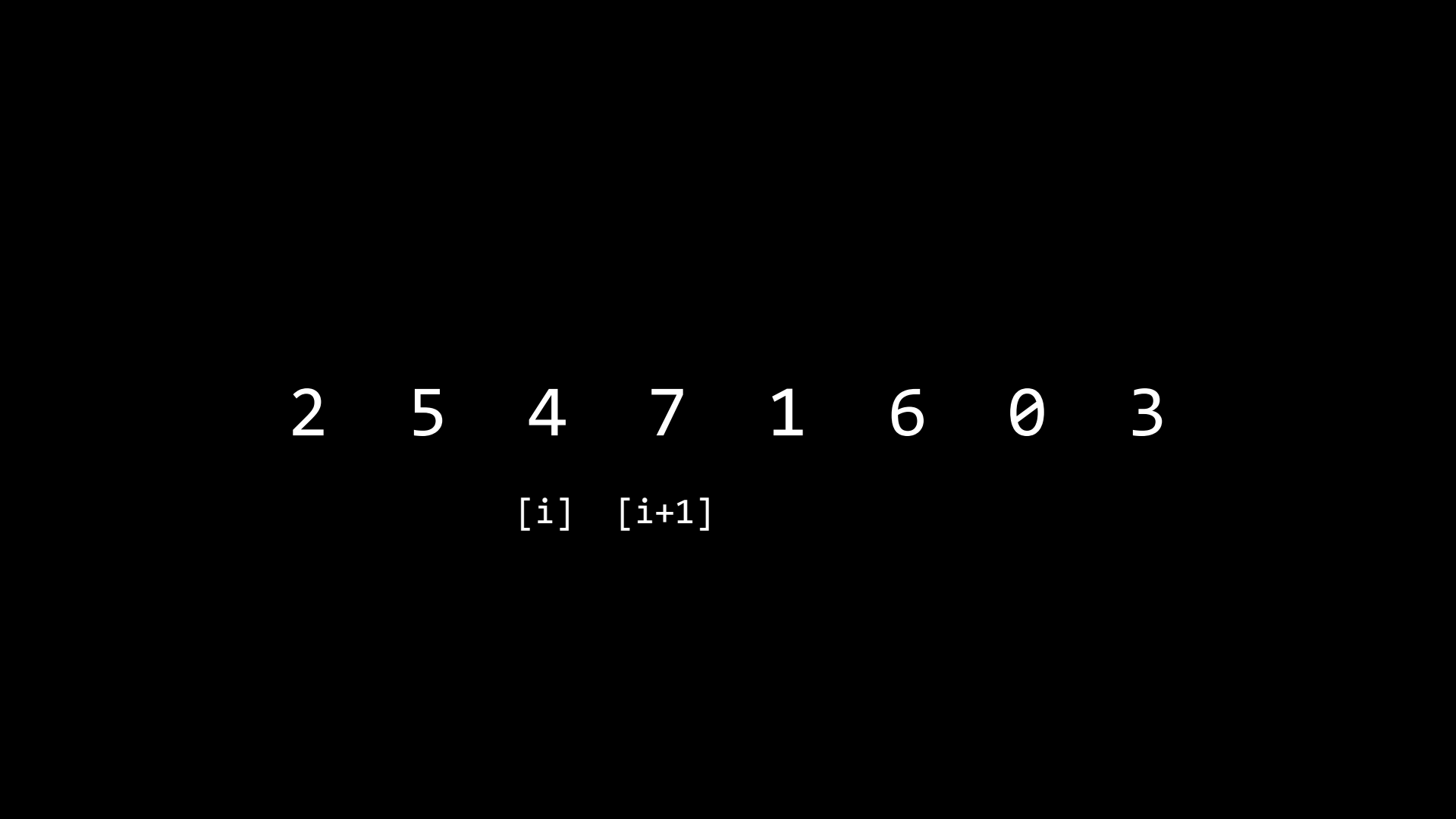

2 7 5 4 1 6 0 3
[i] [i+1]

2 7 5 4 1 6 0 3

[i] [i+1]


2 5 7 4 1 6 0 3
[i] [i+1]


2 5 7 4 1 6 0 3
[i] [i+1]


2 5 4 7 1 6 0 3
[i] [i+1]

2 5 4 7 1 6 0 3

[i] [i+1]

2 5 4 1 7 6 0 3

[i] [i+1]

2 5 4 1 7 6 0 3

[i] [i+1]

2 5 4 1 6 7 0 3

[i] [i+1]

2 5 4 1 6 7 0 3

[i] [i+1]

2 5 4 1 6 0 7 3

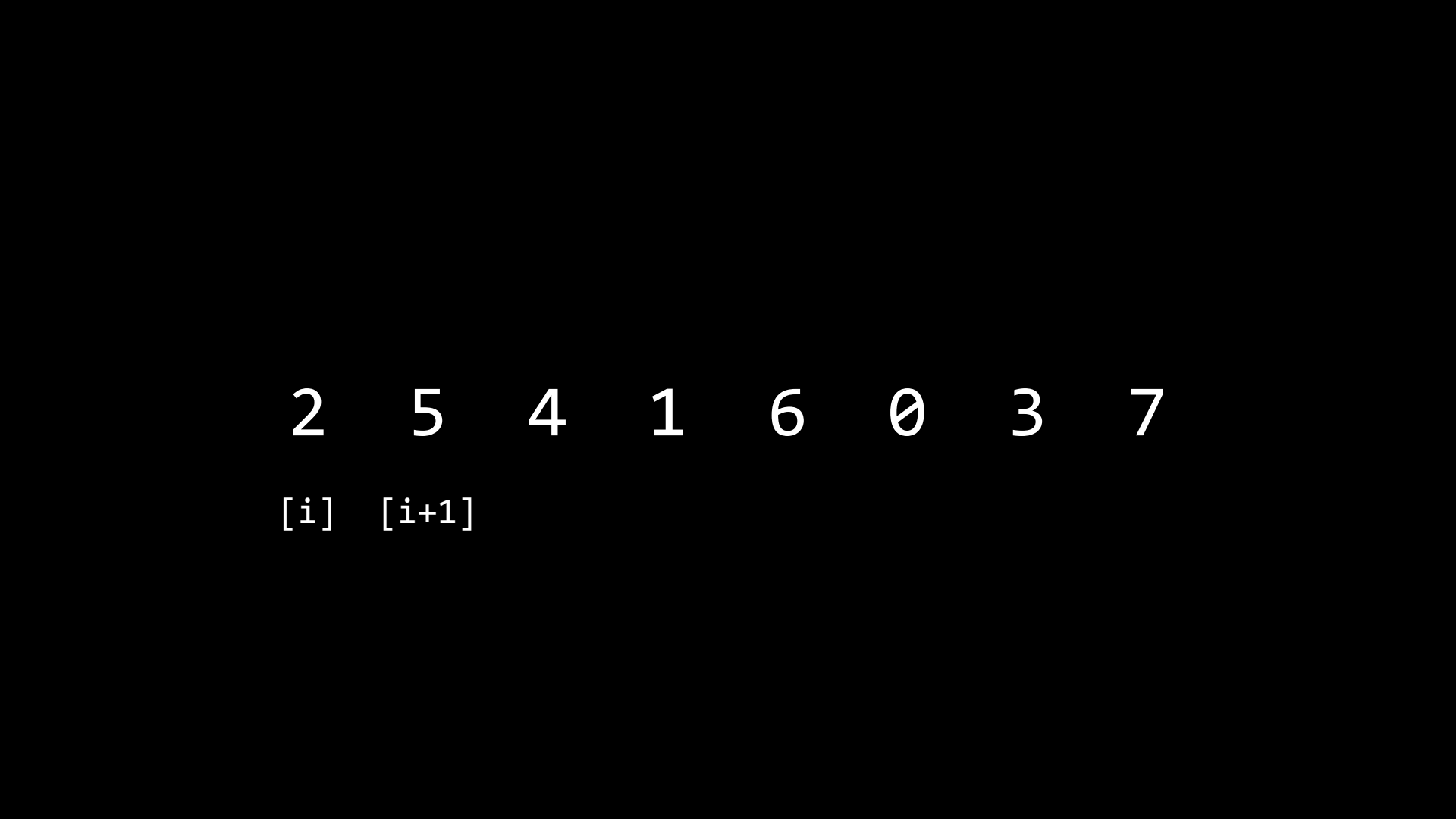
[i] [i+1]

2 5 4 1 6 0 7 3

[i] [i+1]

2 5 4 1 6 0 3 7

[i] [i+1]


2 5 4 1 6 0 3 7
[i] [i+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  n-1
    For i from 0 to n-2
        If numbers[i] and numbers[i+1] out of order
            Swap them
```

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

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

```
Repeat n-1 times  n-1
    For i from 0 to n-2  n-1
        If numbers[i] and numbers[i+1] out of order  4
            Swap them
```

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

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

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

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

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

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

$n^2 - 2n + 1$

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

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

$n^2 - 2n + 1$

$O(n^2)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$ 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
```

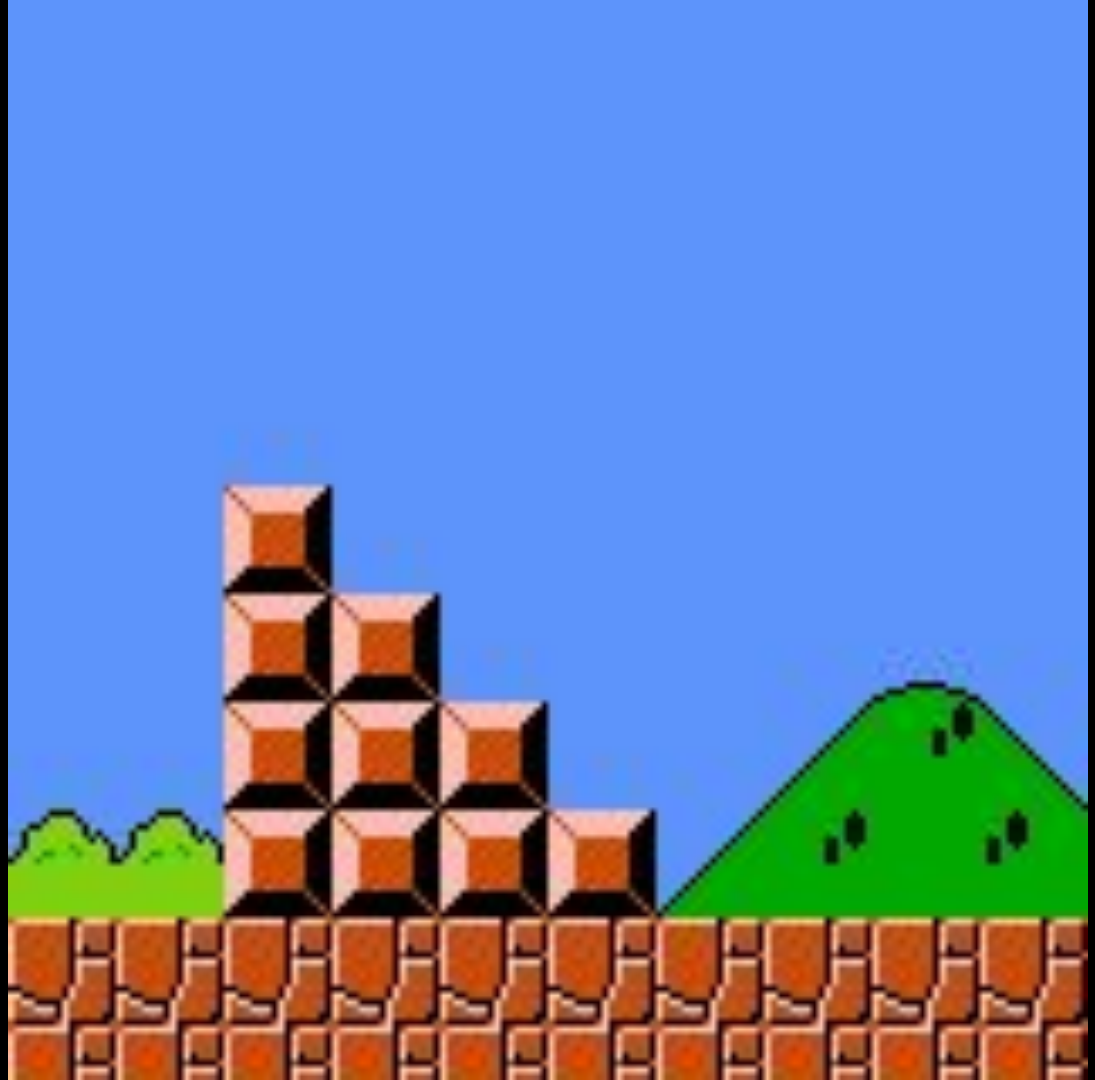
```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Open to middle of left half of book
8       Go back to line 3
9   Else if person is later in book
10      Open to middle of right half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

```
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       Search left half of book
8
9   Else if person is later in book
10      Search right half of book
11
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       Search left half of book
8   Else if person is later in book
9       Search right half of book
10  Else
11      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
```

1 3 4 6 0 2 5 7

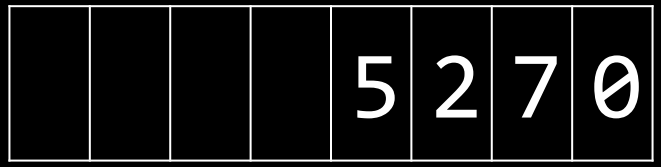
```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge sorted halves
```

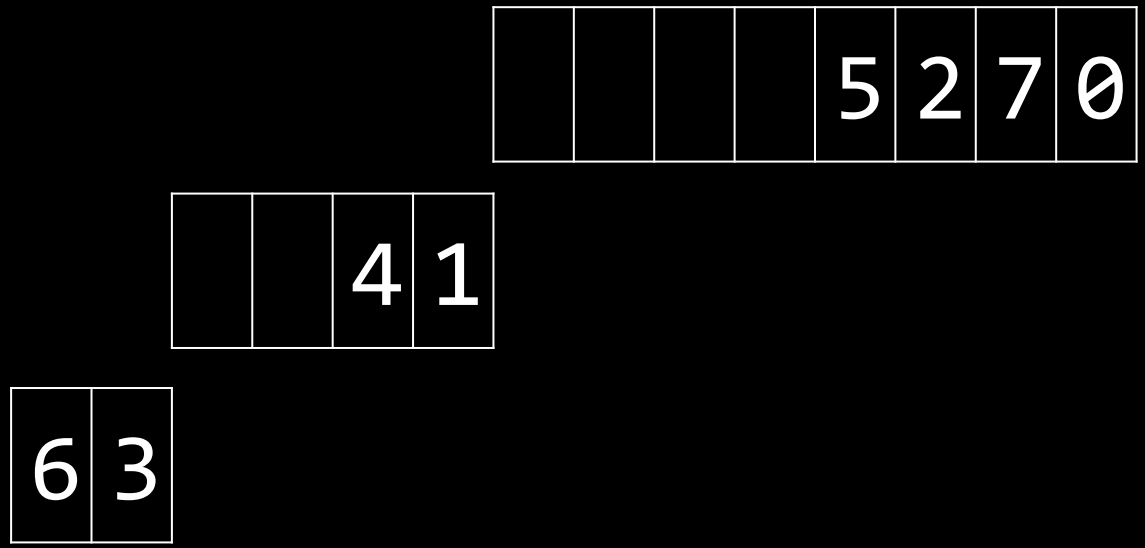
6 3 4 1 5 2 7 0

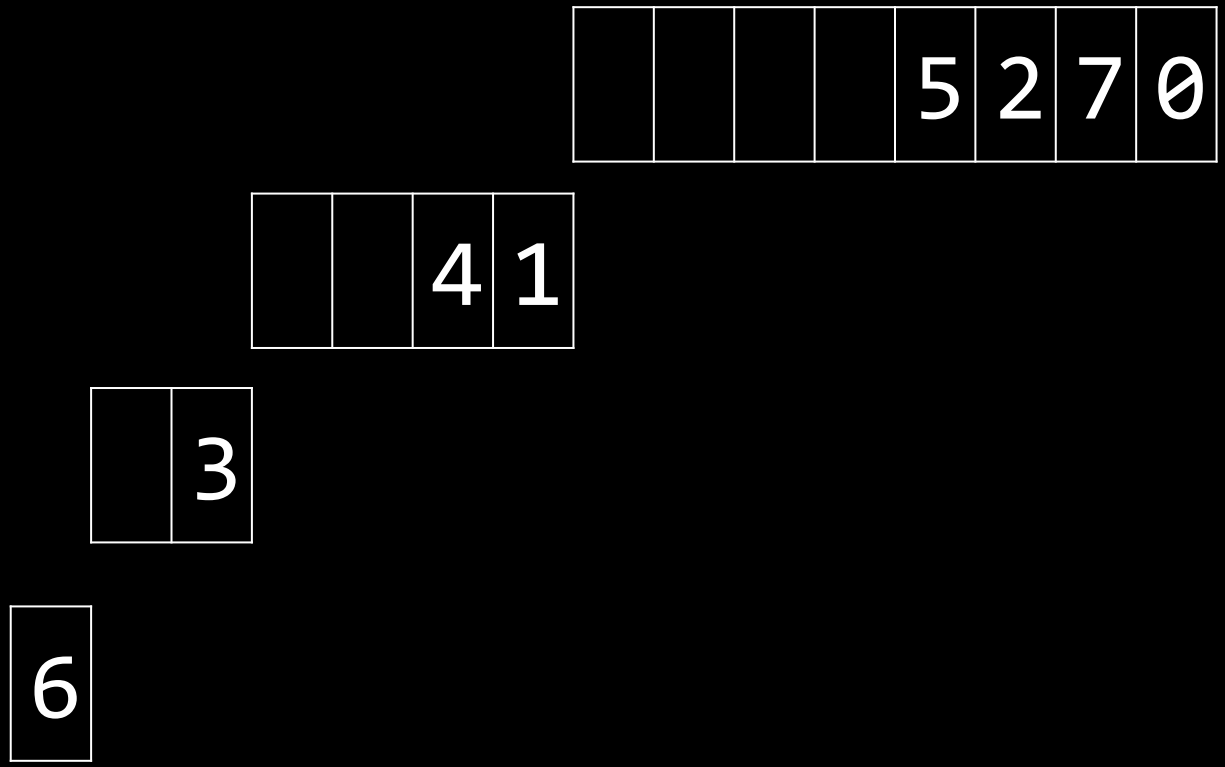
6 3 4 1 5 2 7 0

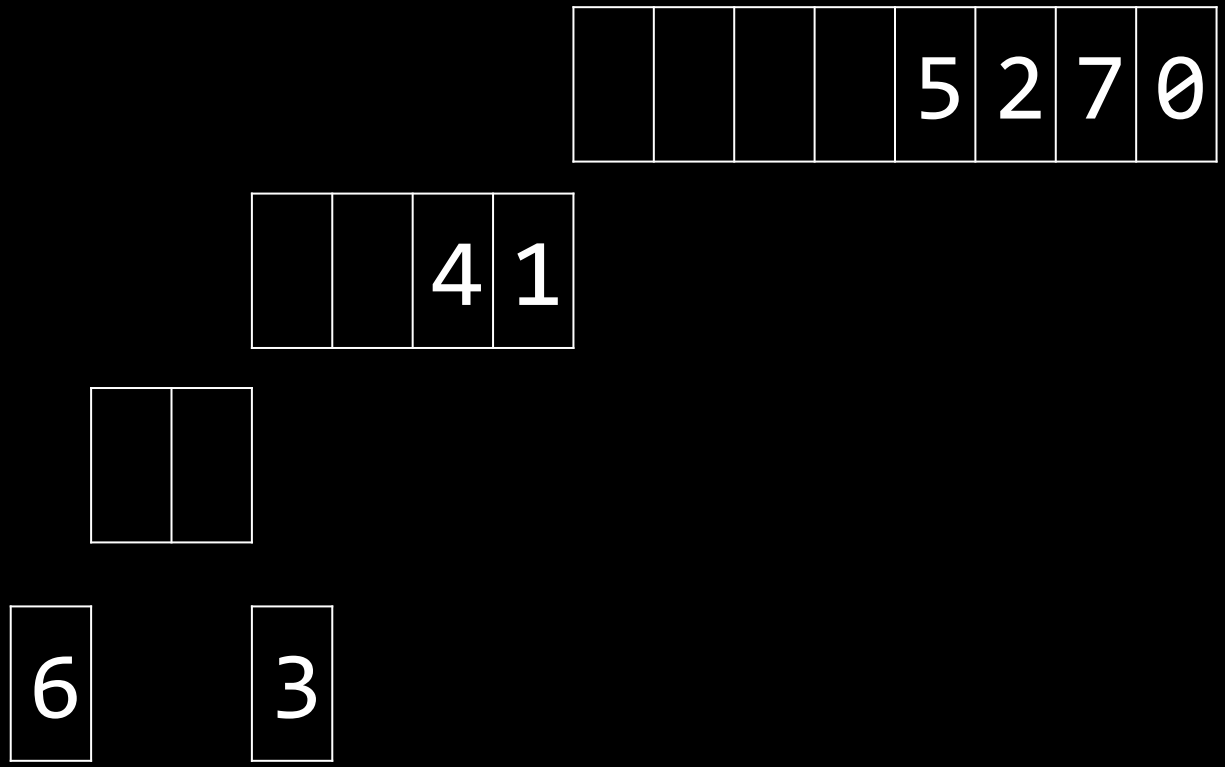

6 3 4 1 5 2 7 0

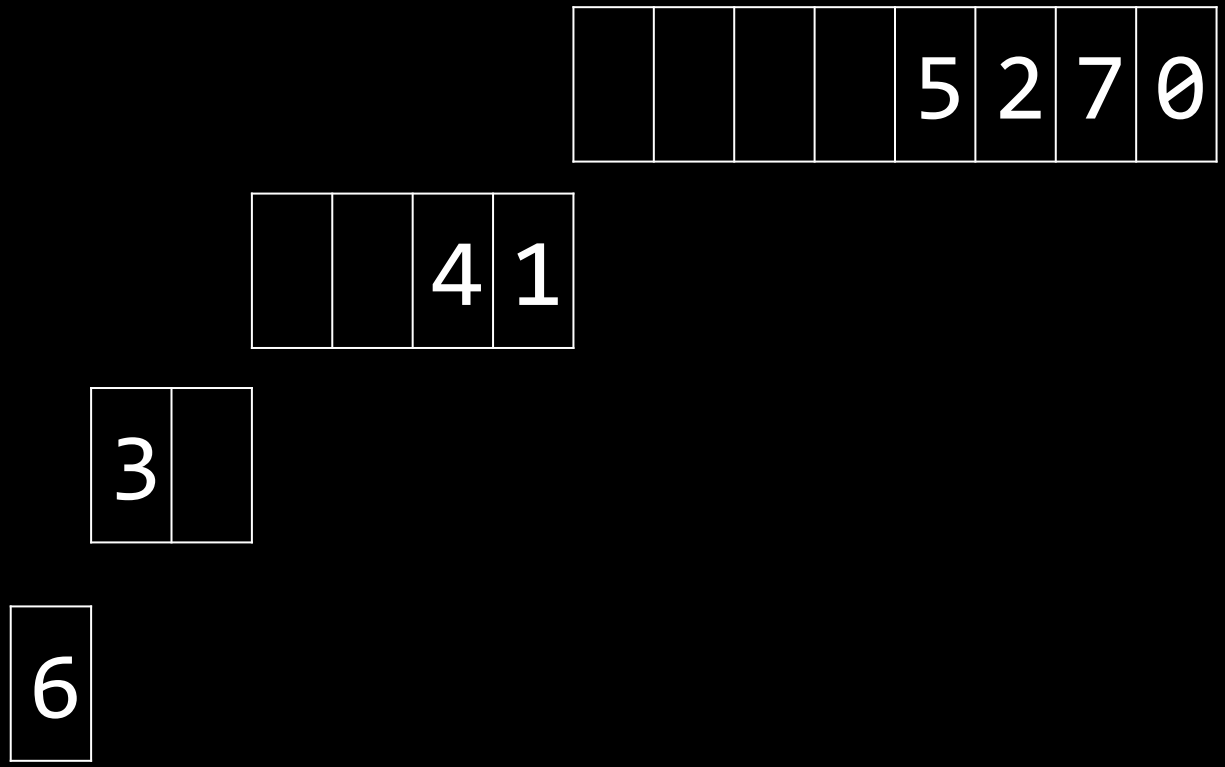

6341

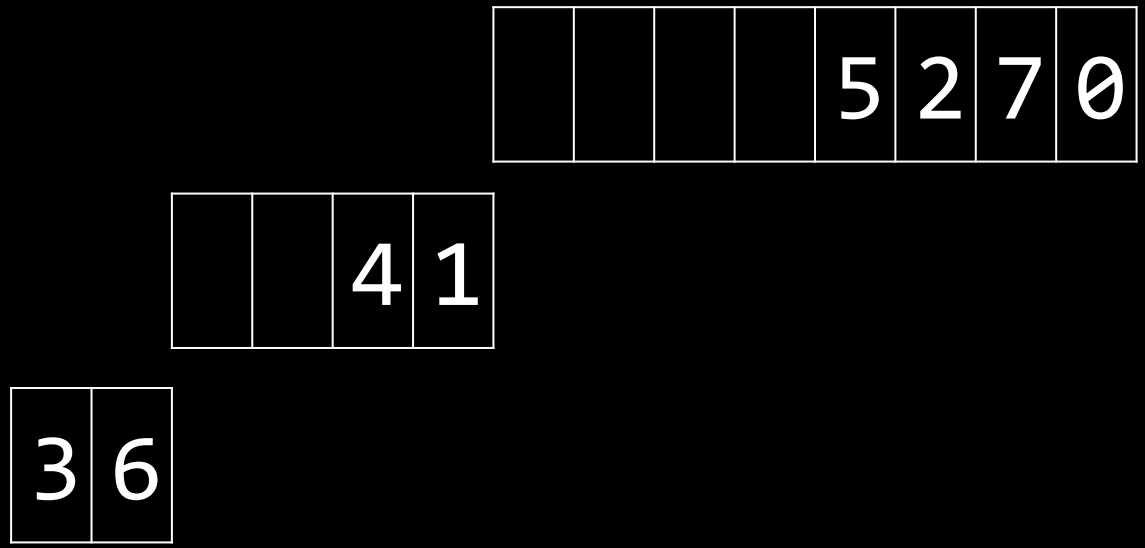

5270

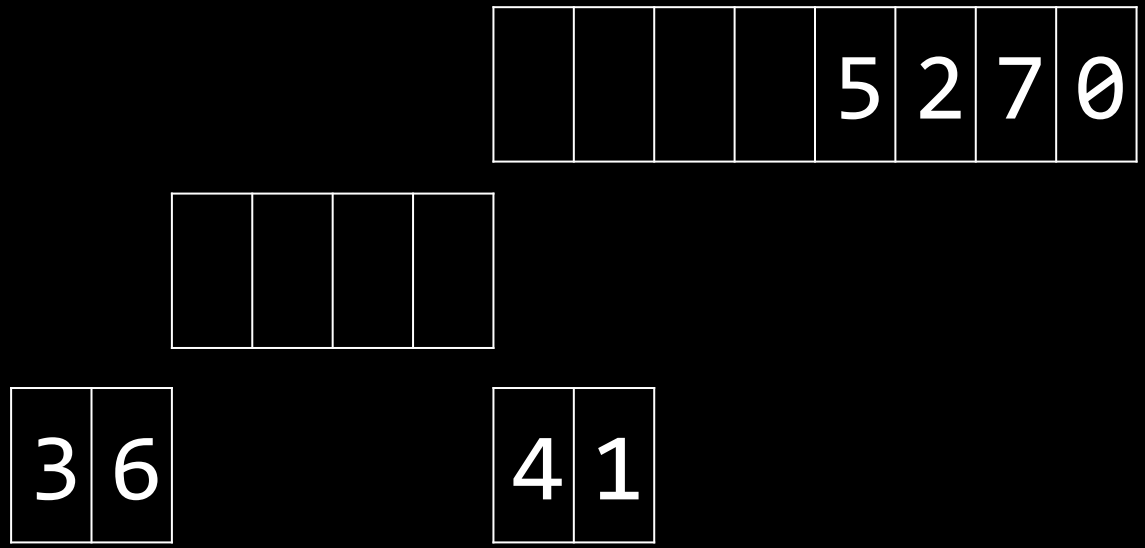

5270
41
63

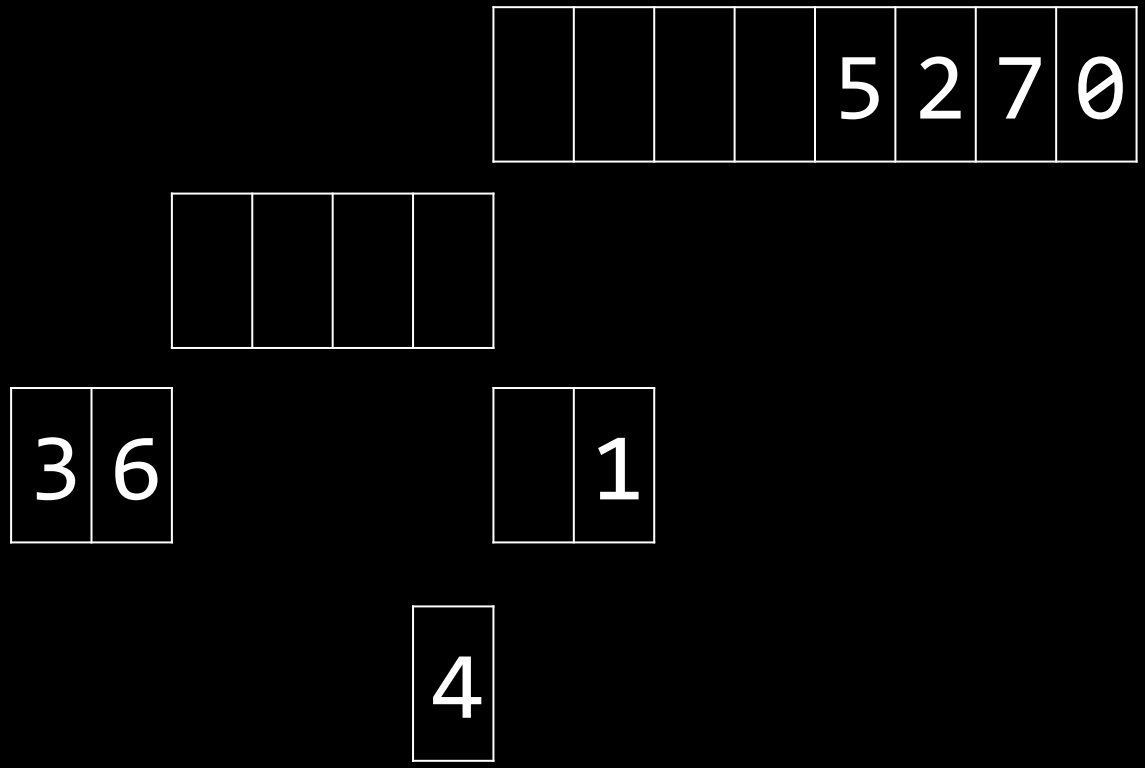

5270
41
3
6

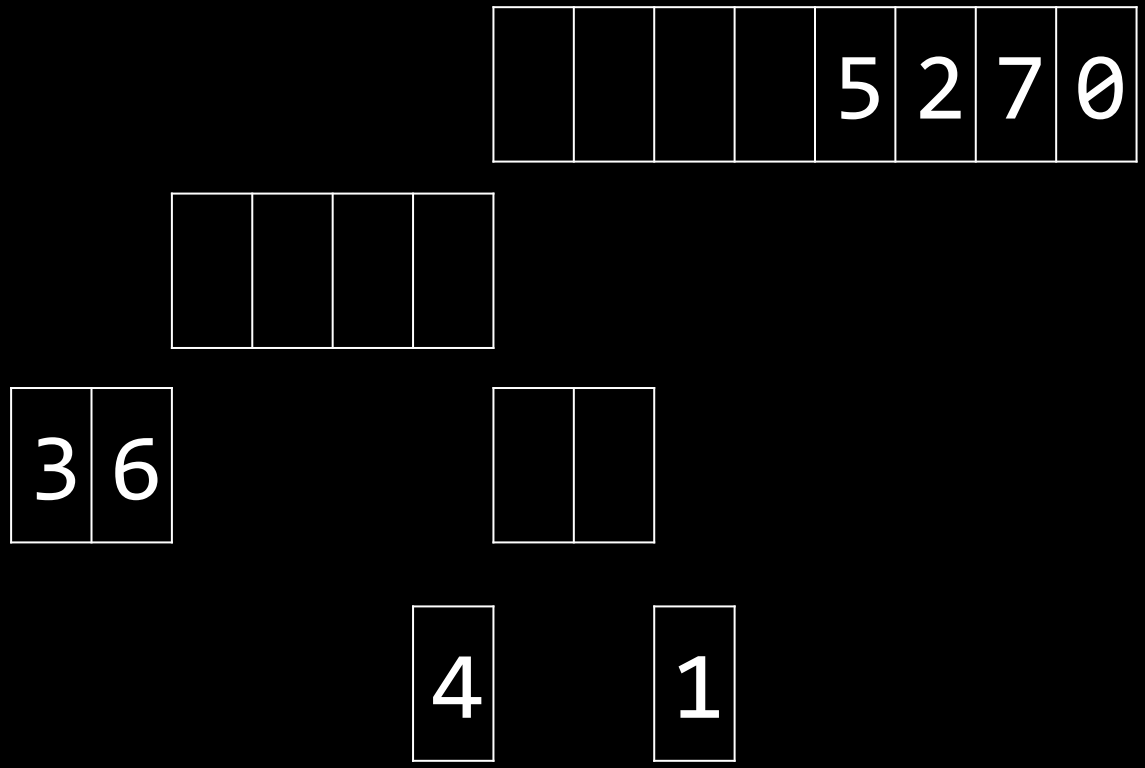

5270
41
6
3

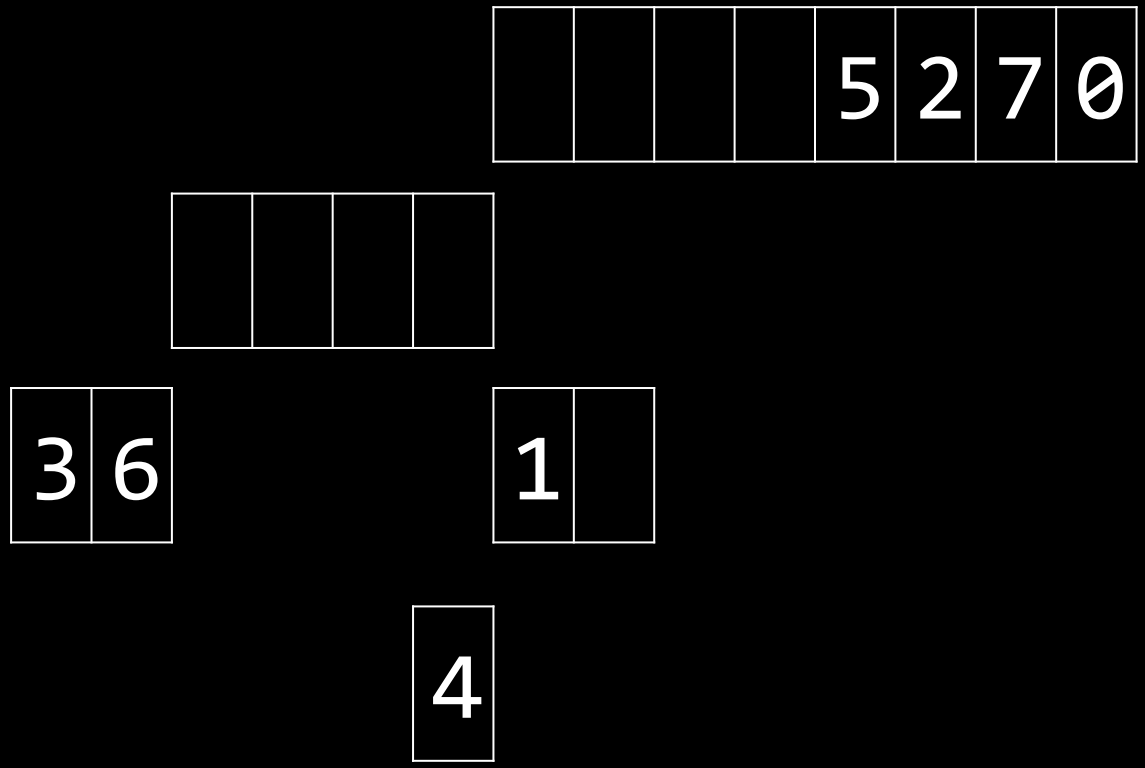

5270
41
3
6

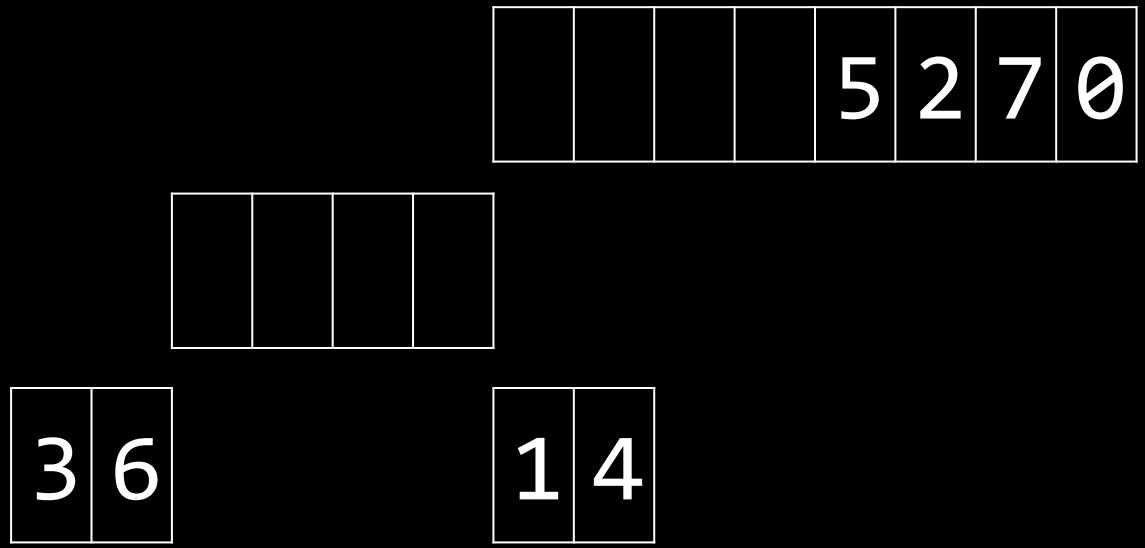

5270
41
36

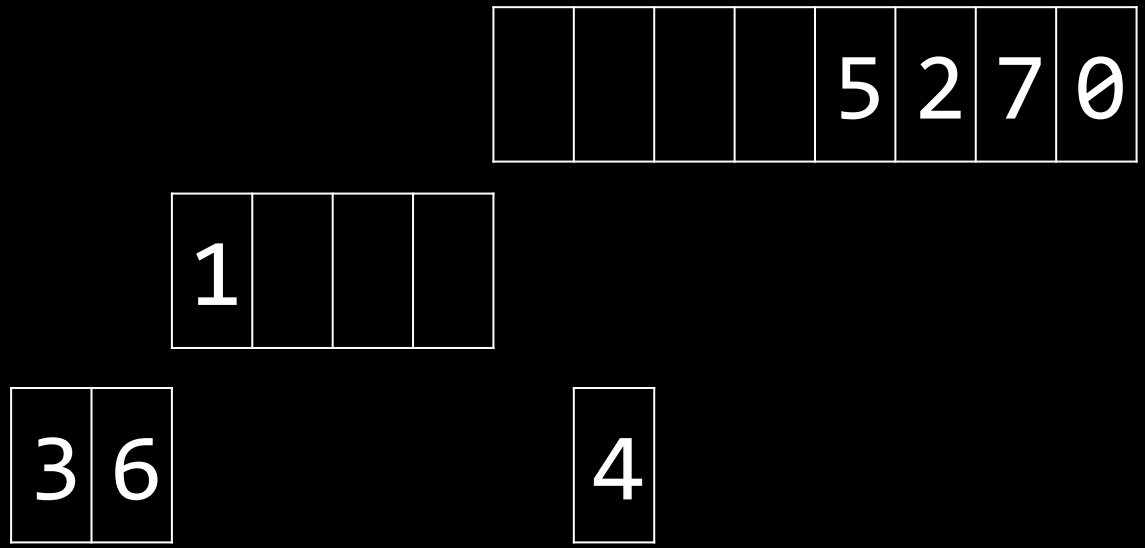

5270
36
41

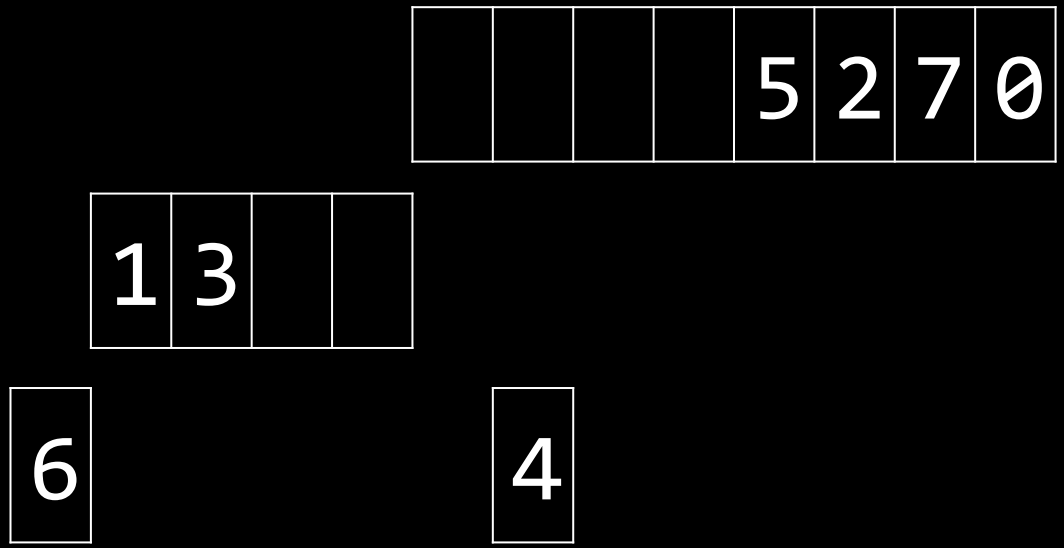

5270
36
1
4

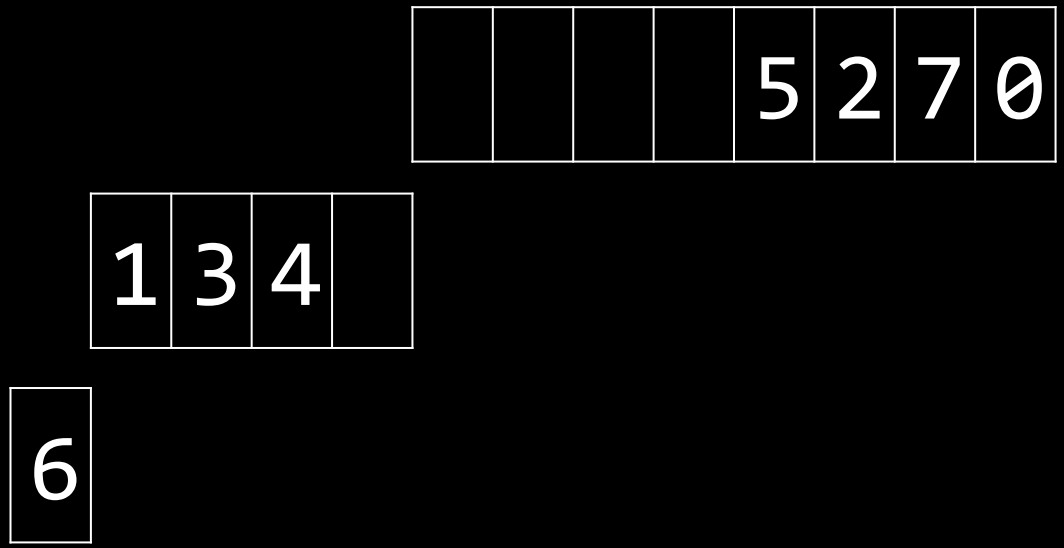

5270
36
4
1


5270
36
1
4

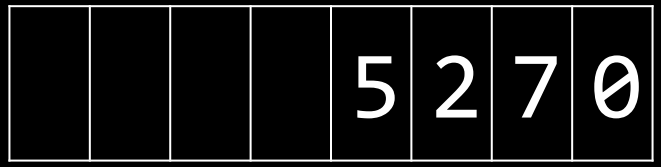

5270
36
14

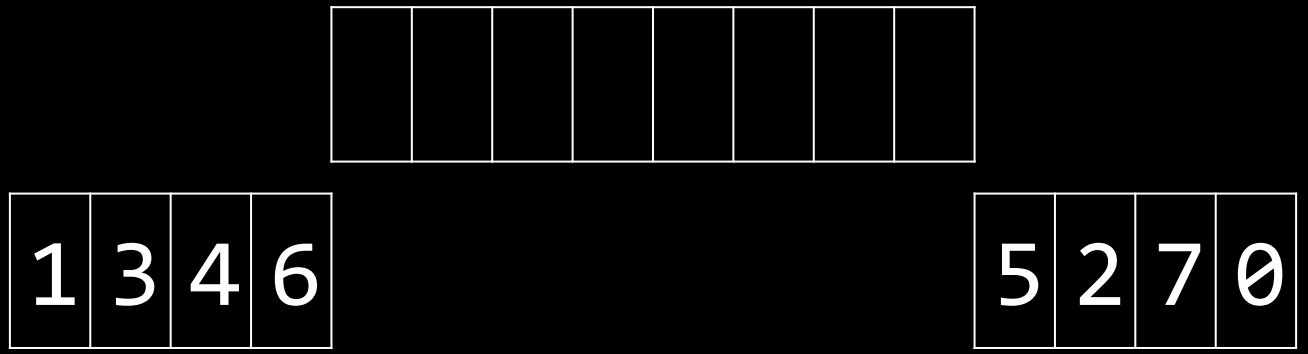

5270
1
36
4

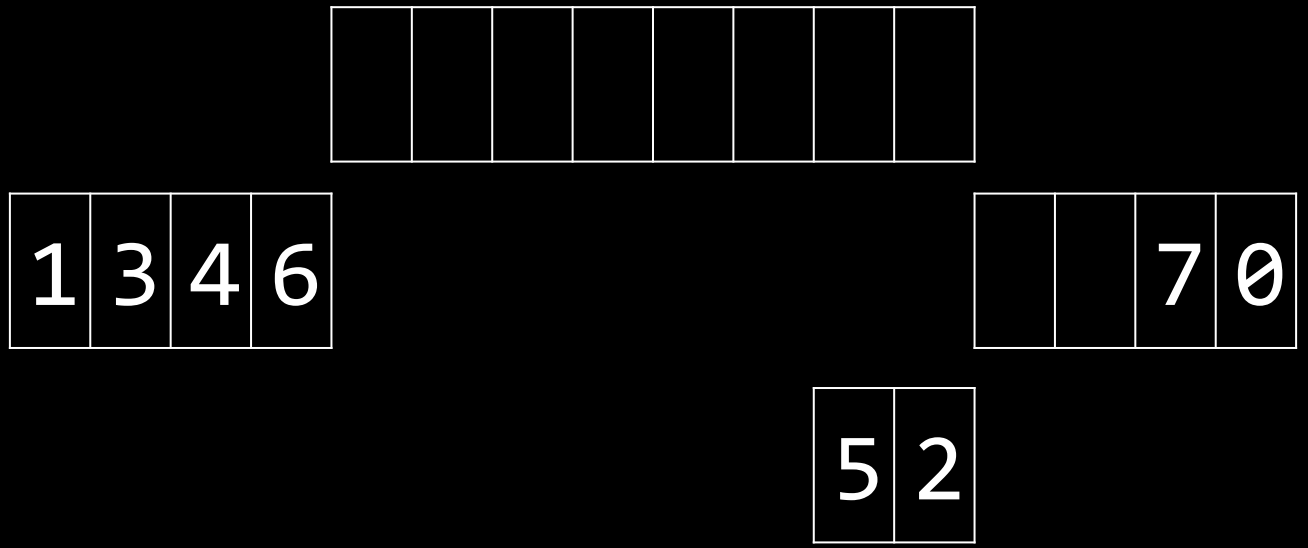

5270
13
6
4

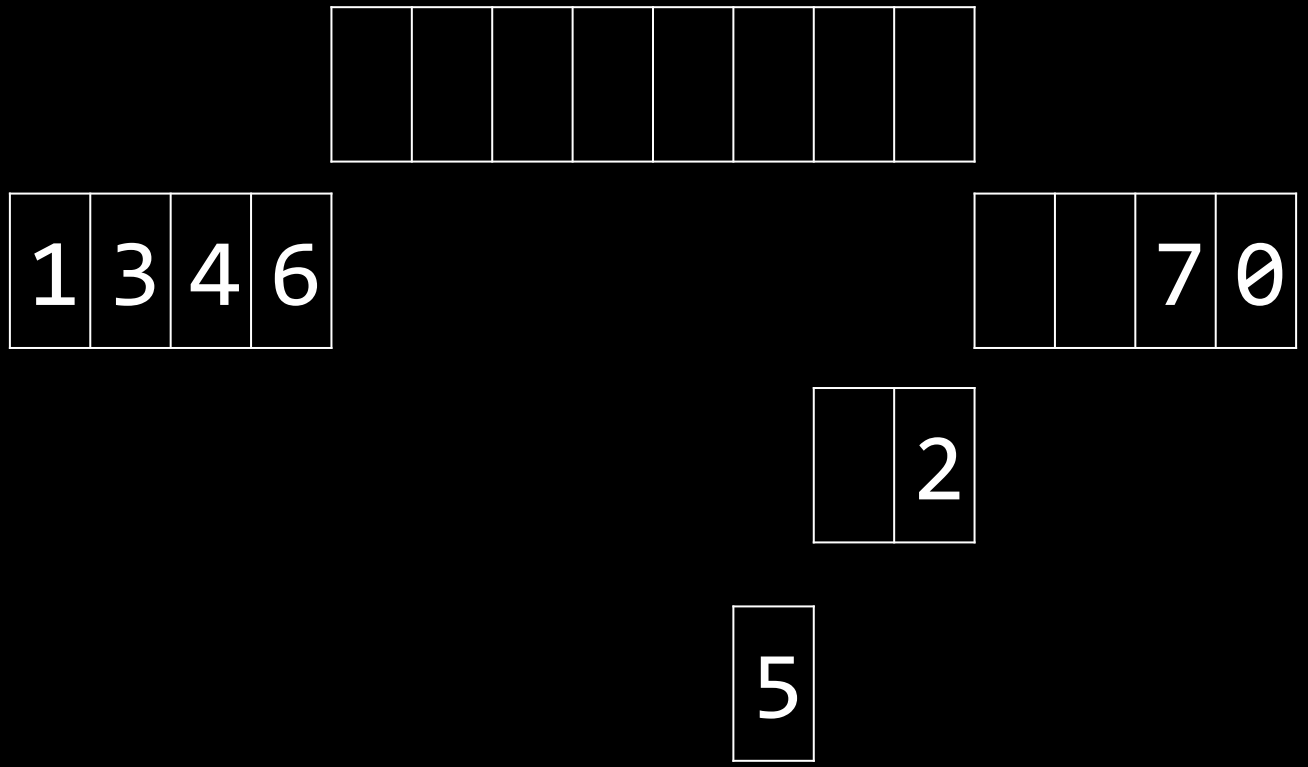

5270
134
6

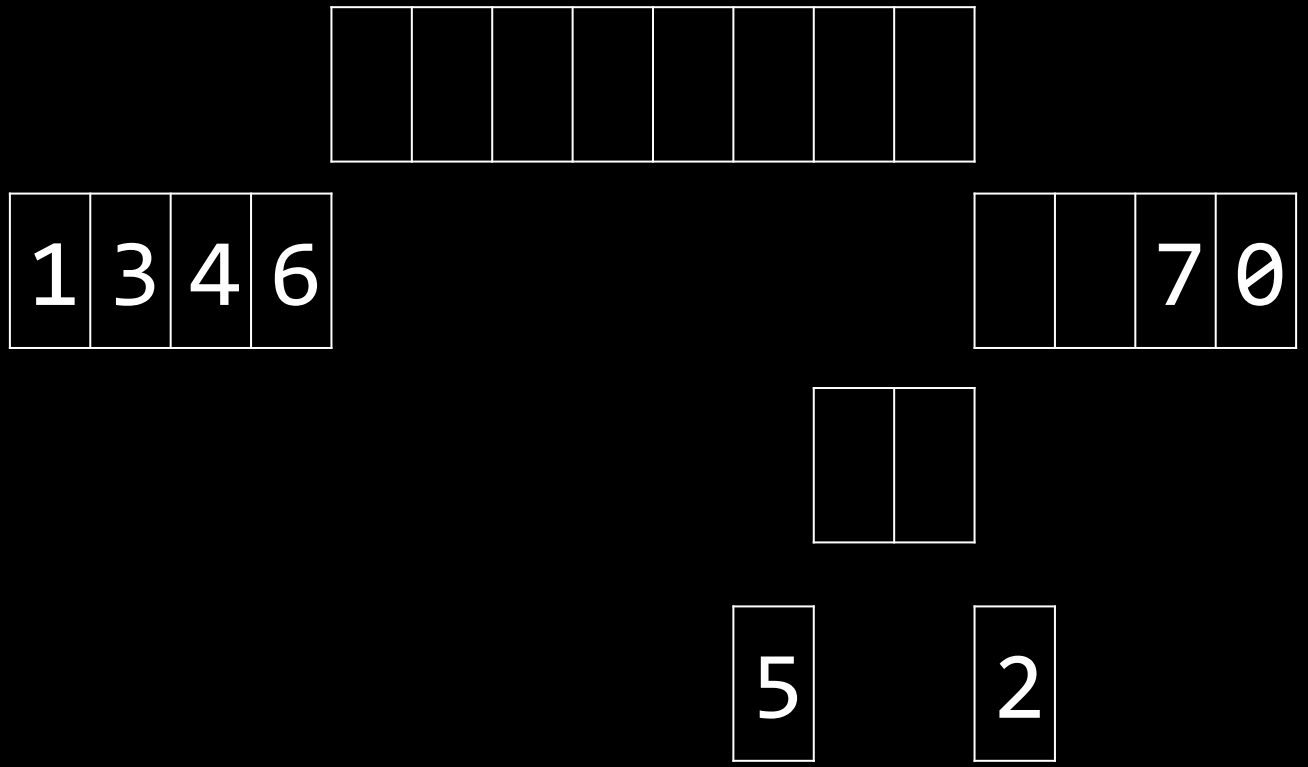

1346

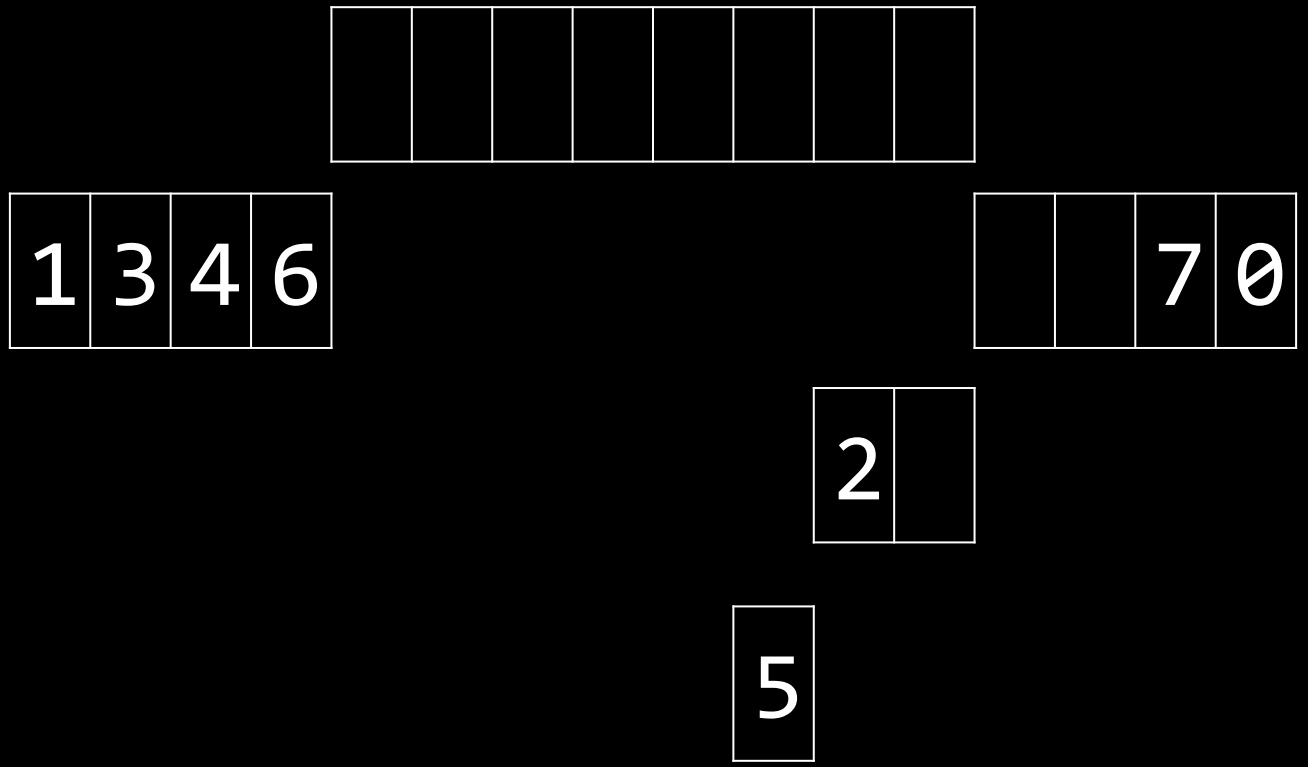

5270

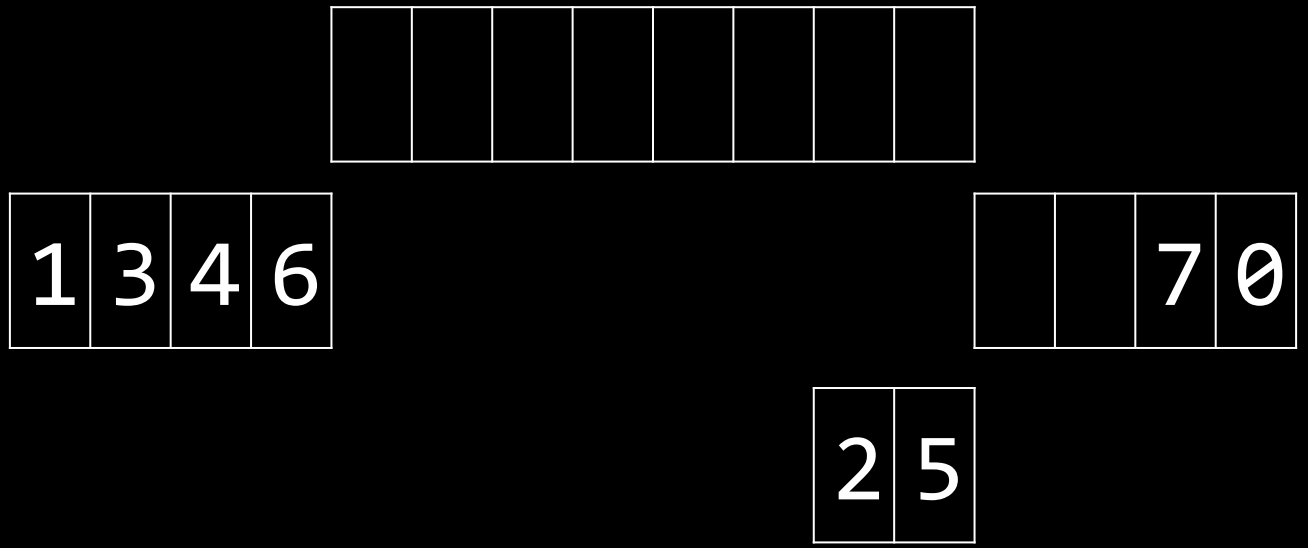

1 3 4 6
5 2 7 0

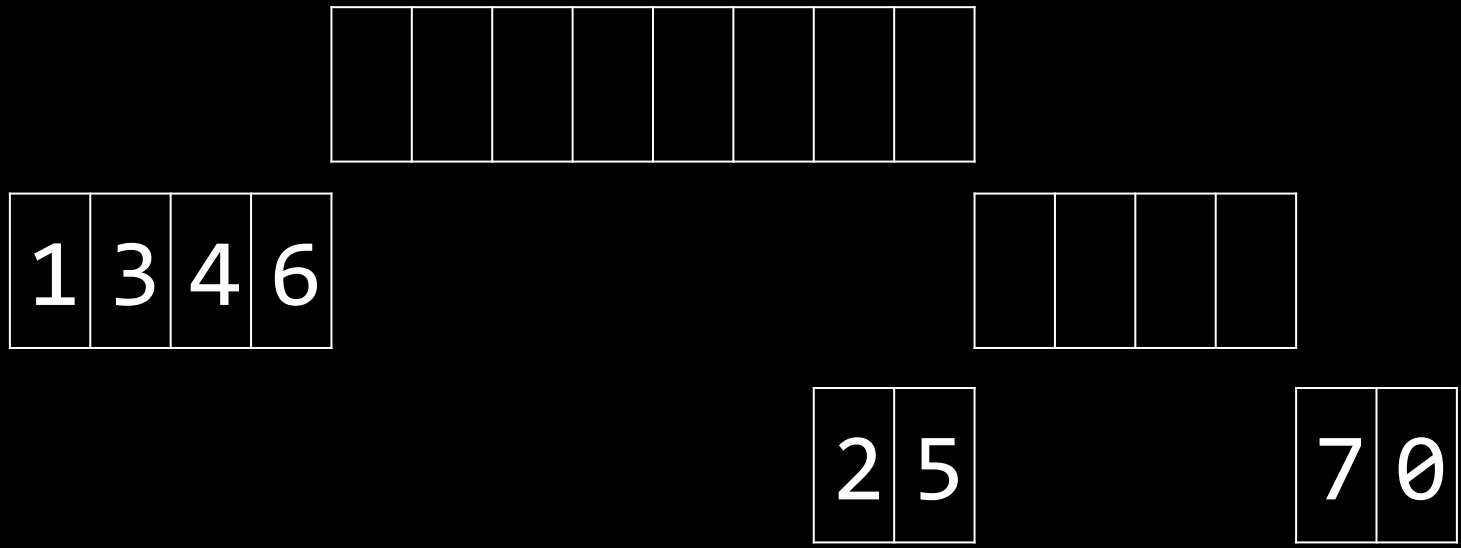

1 3 4 6
7 0
5 2

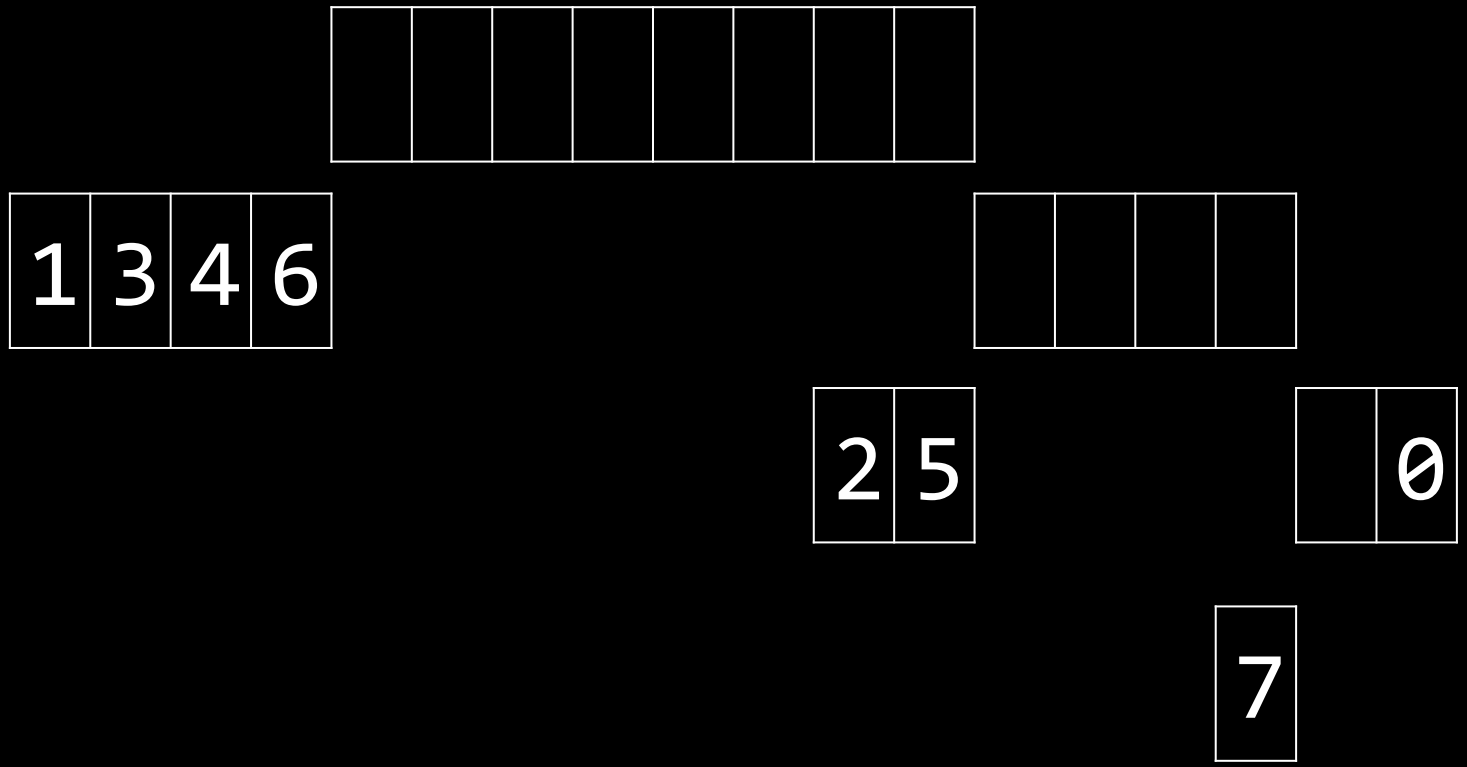

1 3 4 6
7 0
2
5

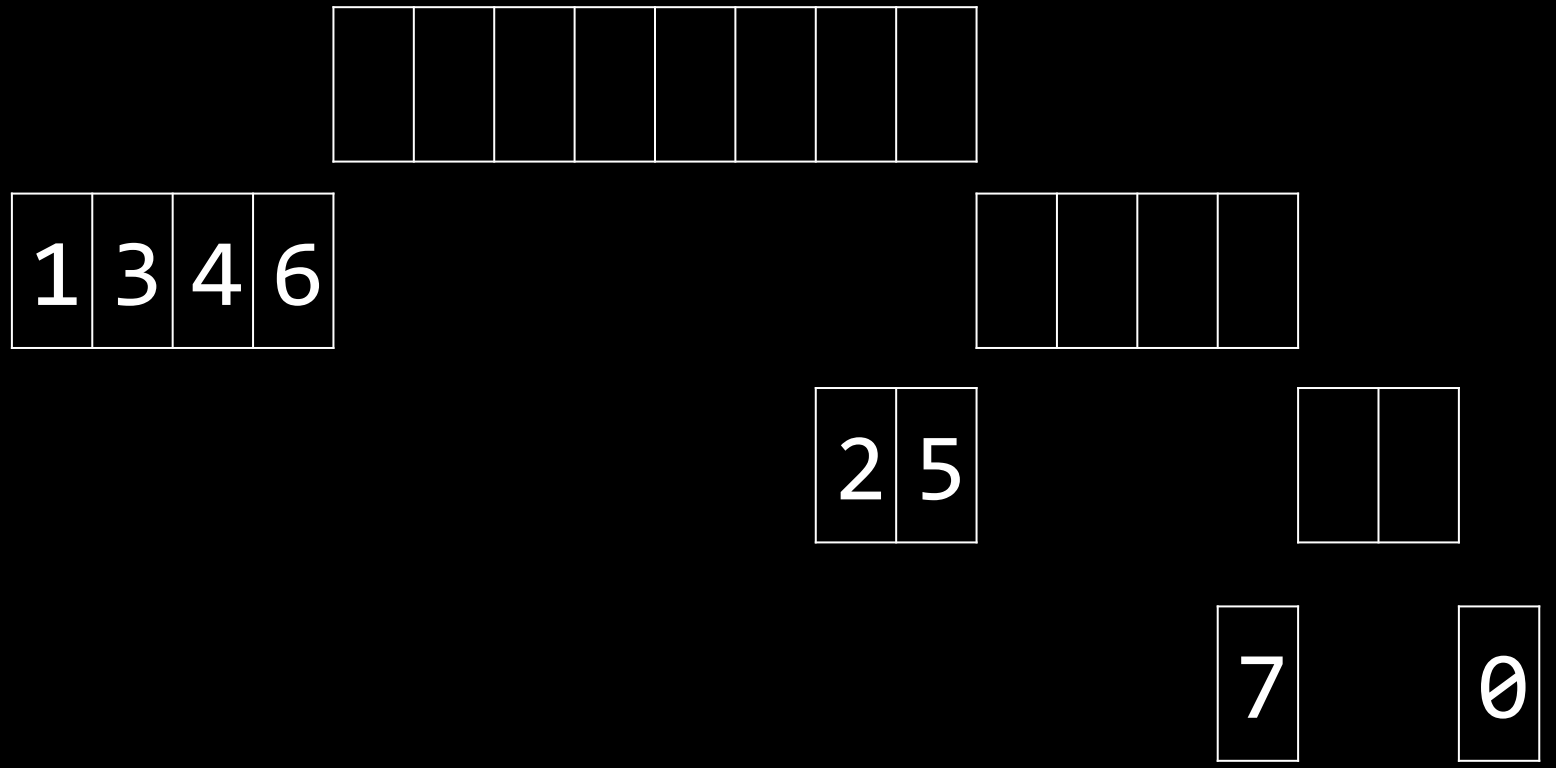

1 3 4 6
7 0
5
2

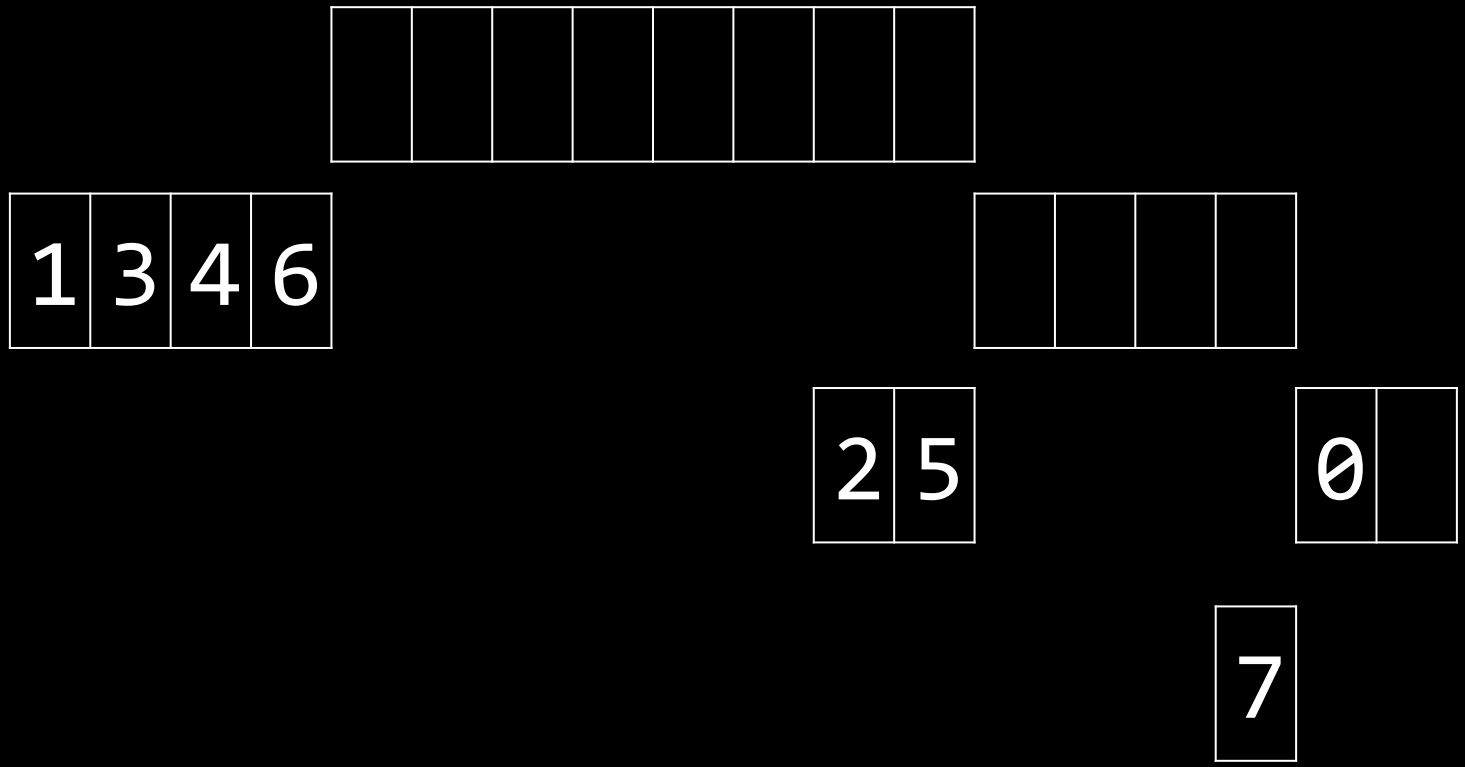

1 3 4 6
7 0
2
5

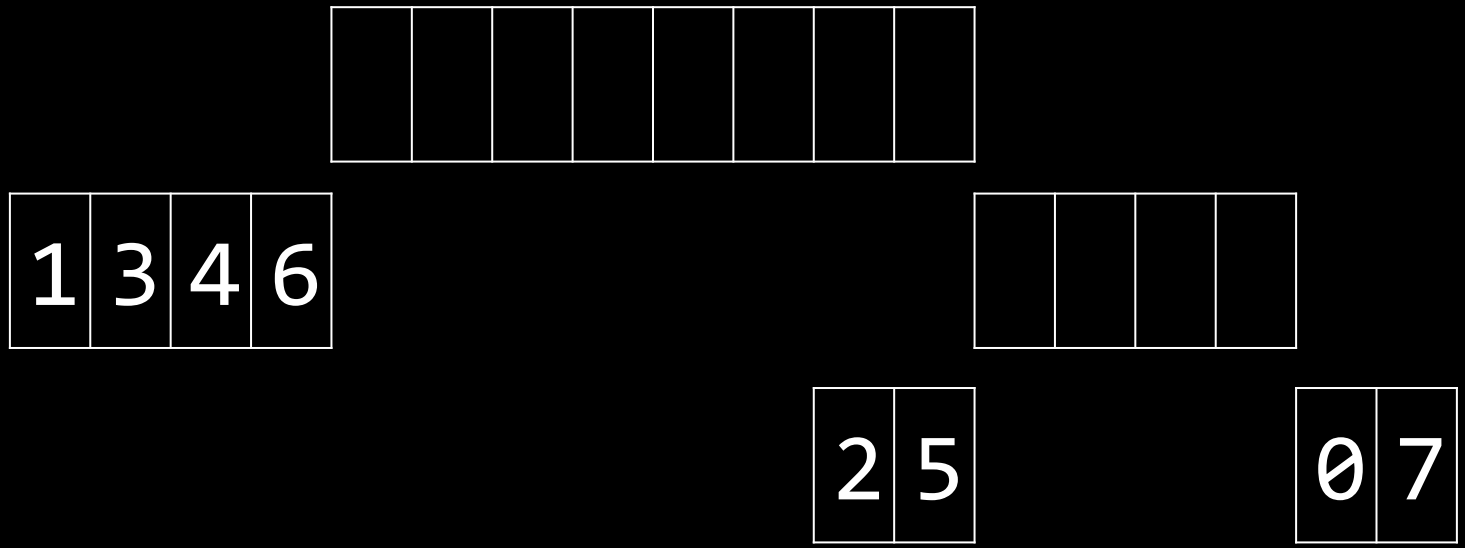

1 3 4 6
7 0
2 5

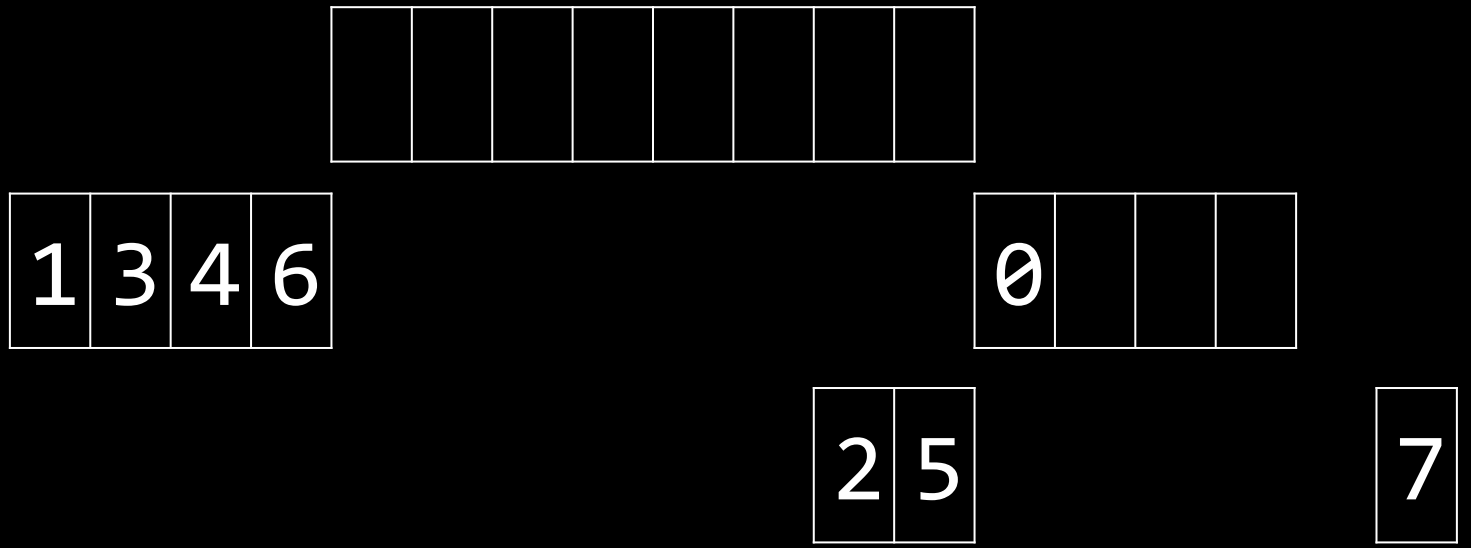

1 3 4 6
2 5
7 0

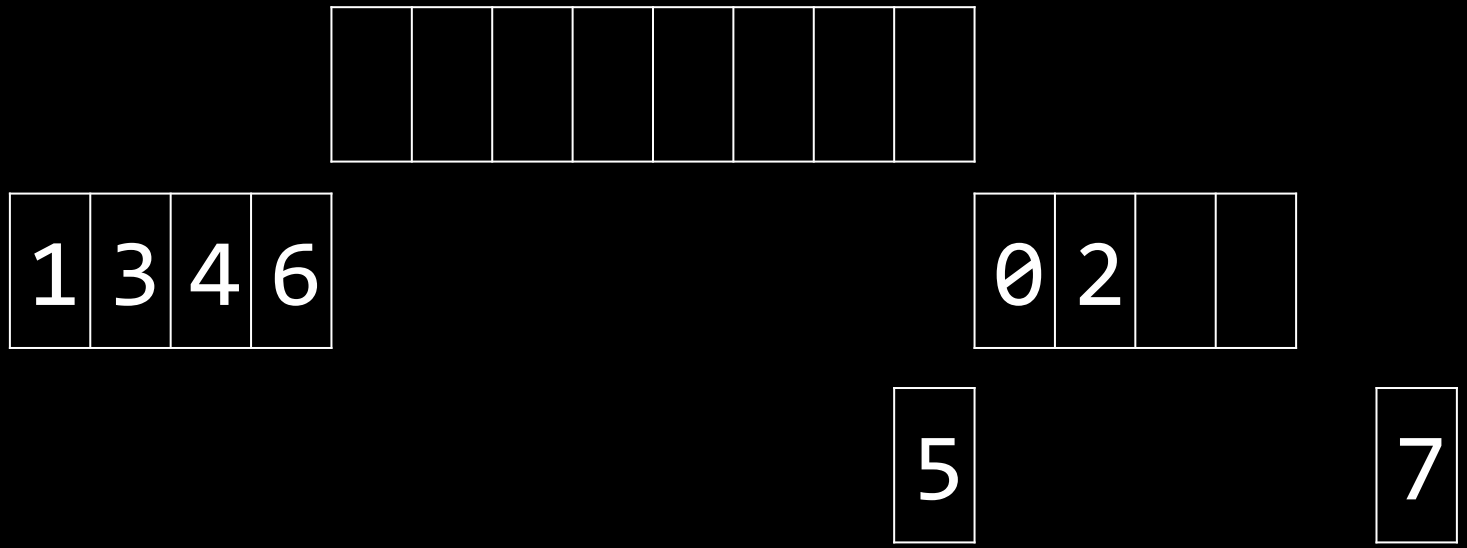

1 3 4 6
2 5
0
7

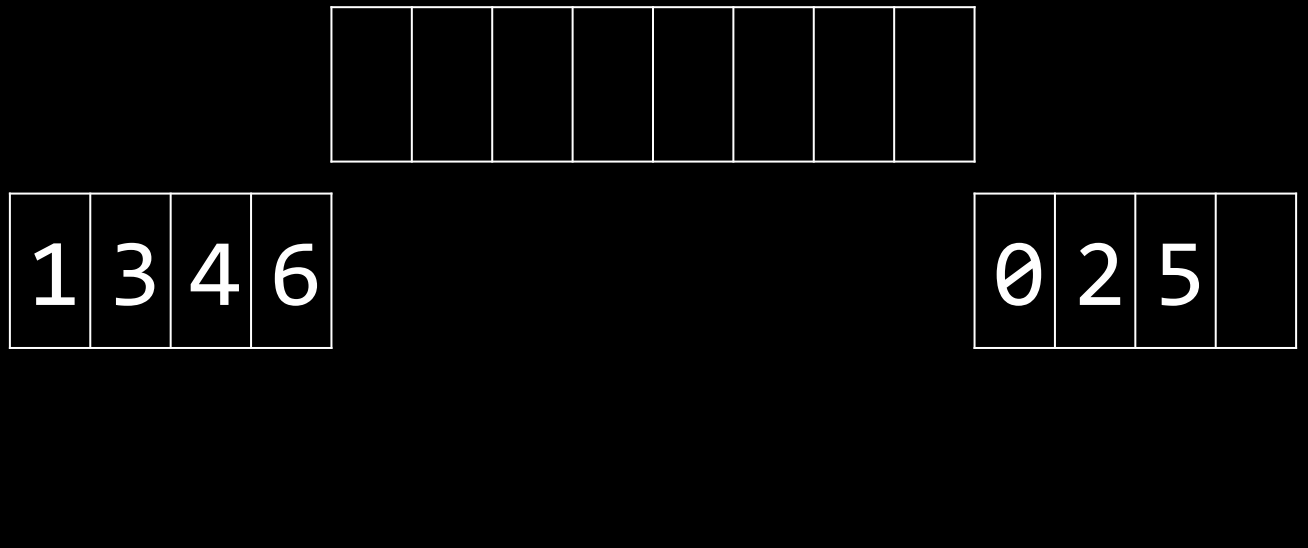

1 3 4 6
2 5
7
0

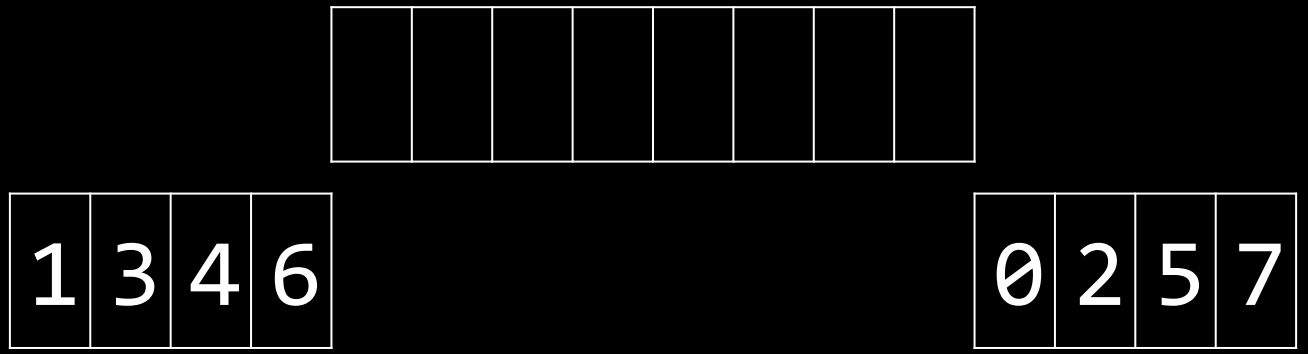

1 3 4 6
2 5
0
7

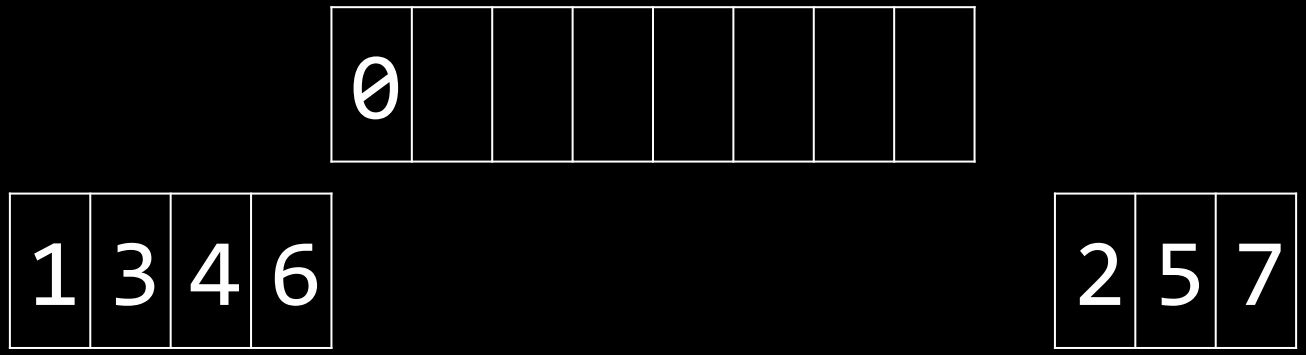

1 3 4 6
2 5
0 7

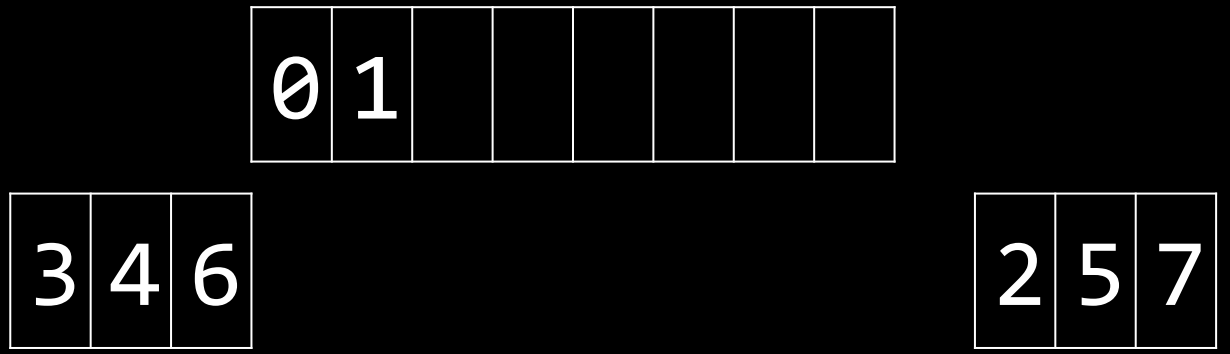

1 3 4 6
0
2 5
7

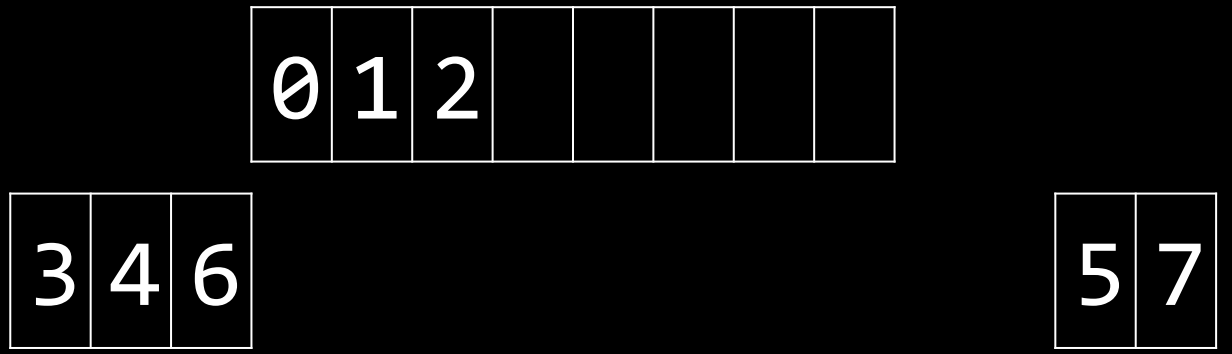

1 3 4 6
0 2
5
7

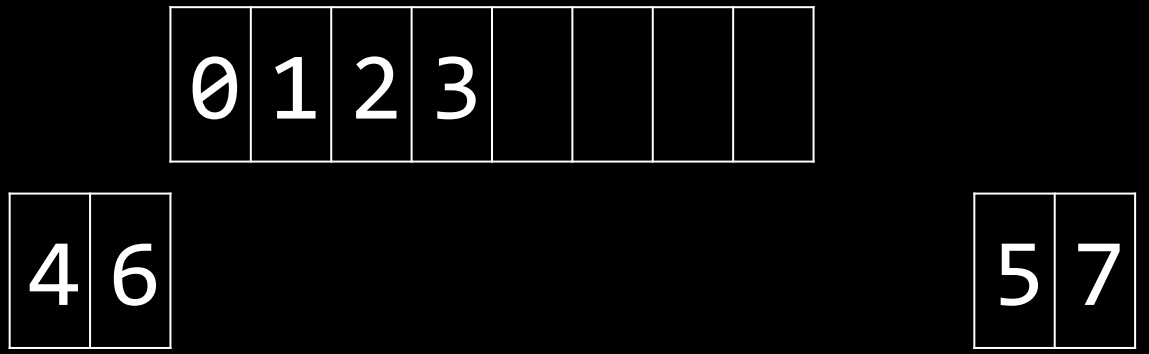

1 3 4 6
0 2 5
7

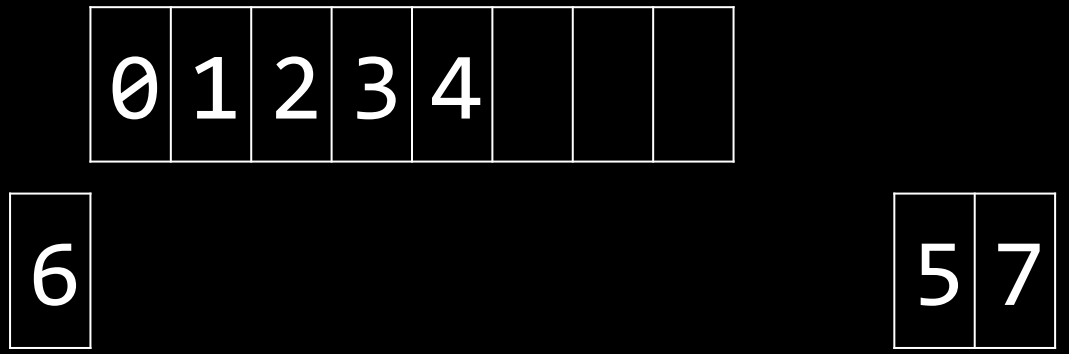

1 3 4 6
0 2 5 7

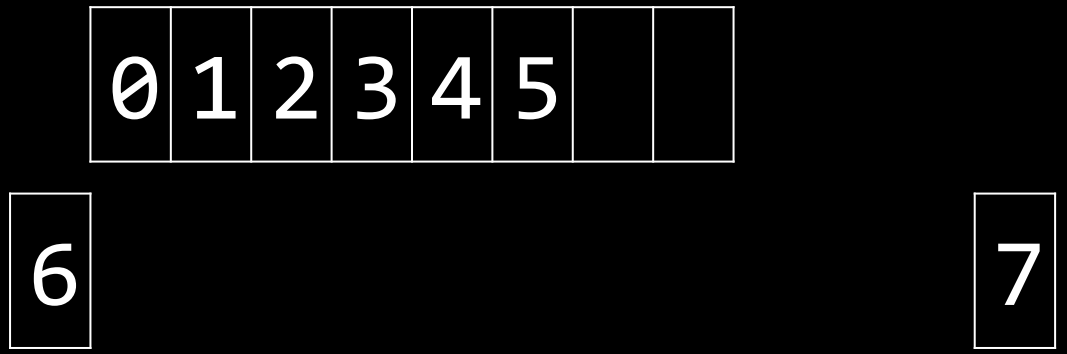

0
1 3 4 6
2 5 7


0 1
3 4 6
2 5 7


0 1 2
3 4 6
5 7


0 1 2 3
4 6
5 7


0 1 2 3 4
6
5 7


0 1 2 3 4 5
6
7

7


0 1 2 3 4 5 6 7

0 1 2 3 4 5 6 7

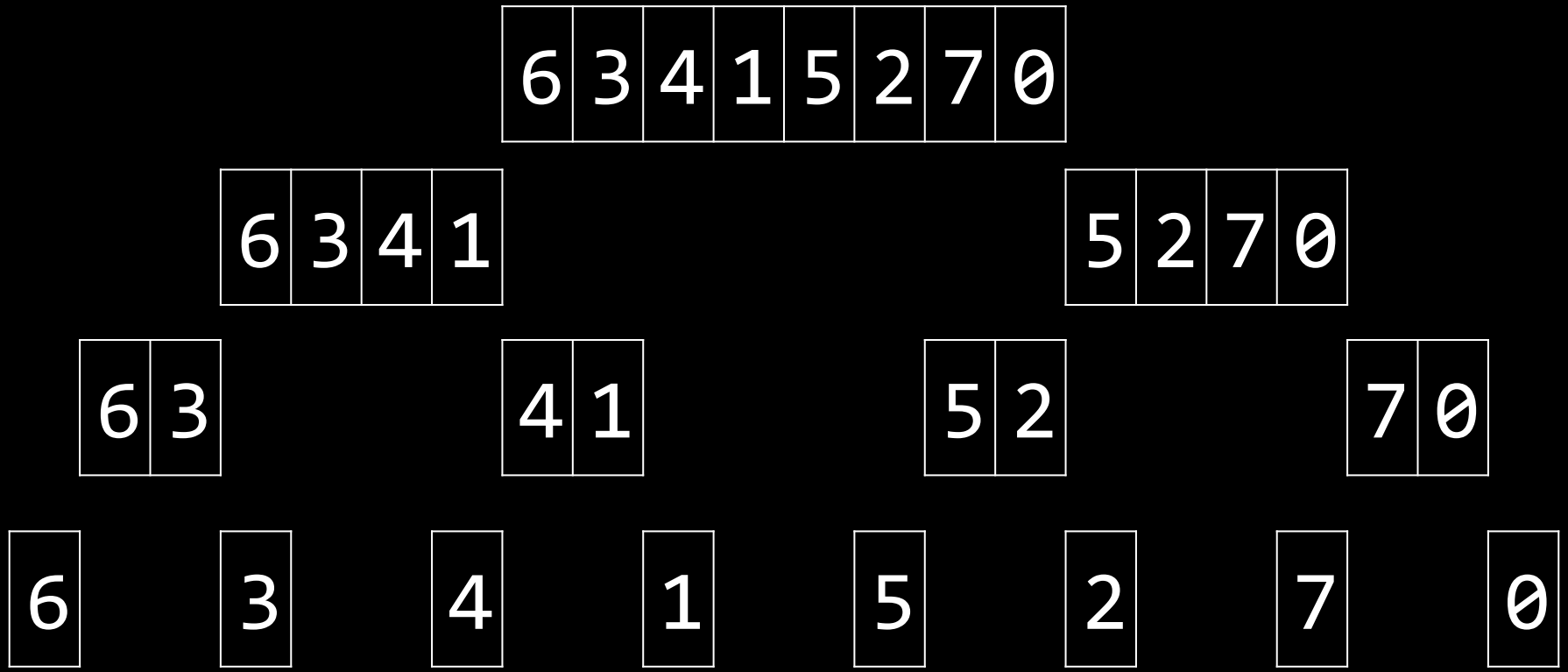
$O(n^2)$

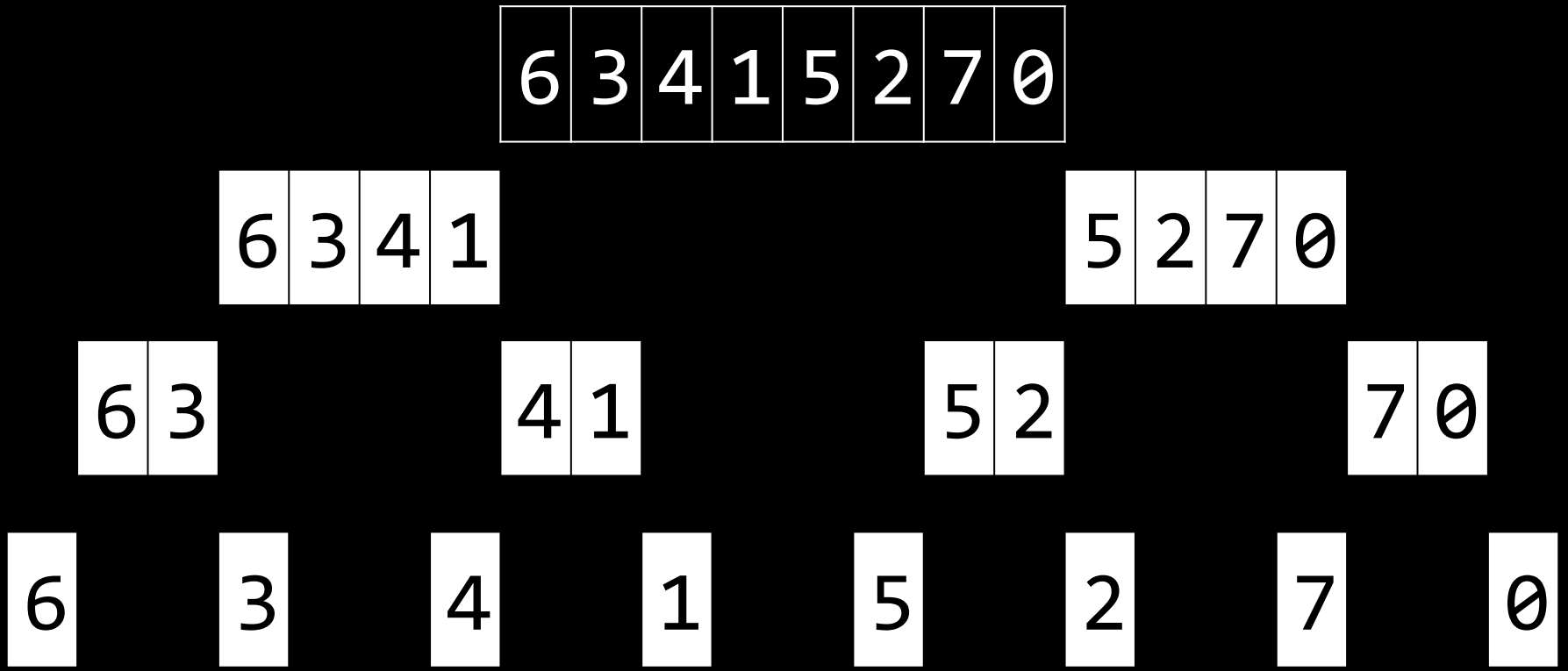
$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$


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

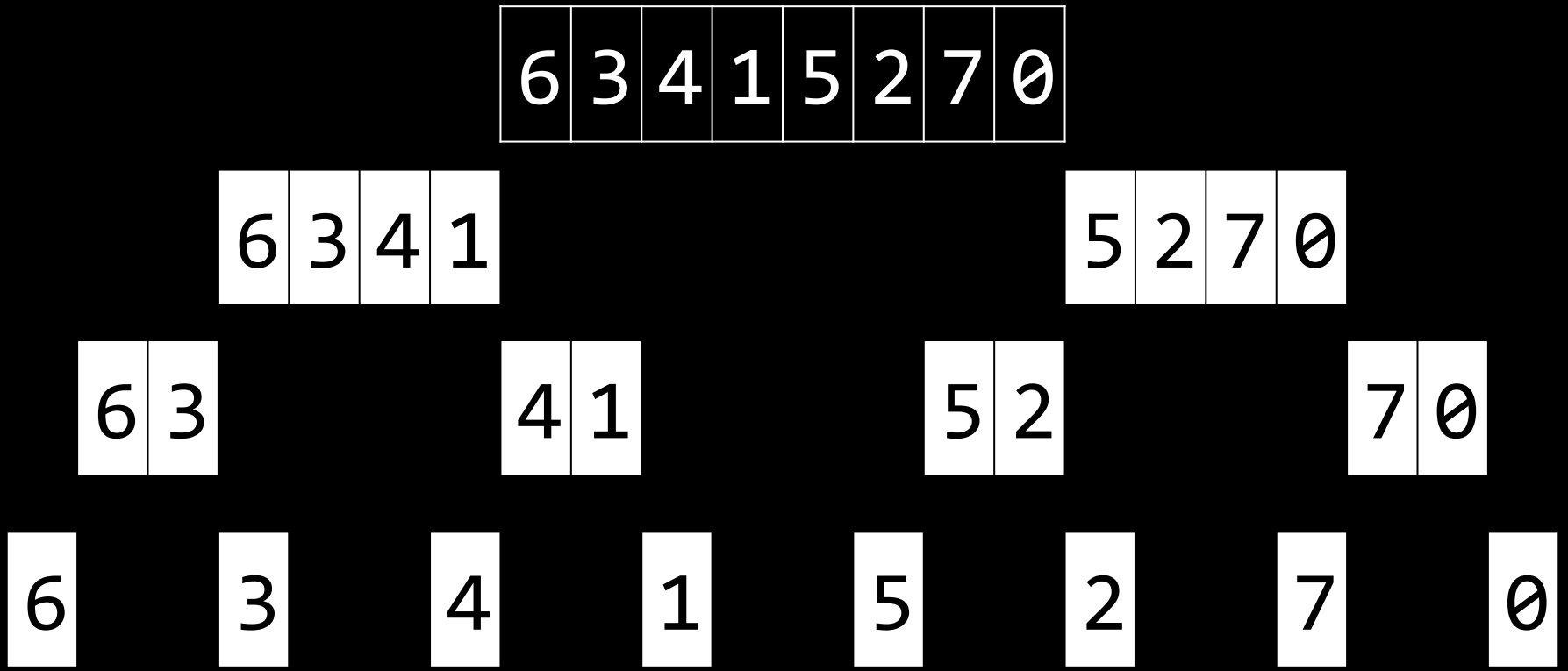

6 3 4 1 5 2 7 0
6 3 4 1
5 2 7 0
6 3
4 1
5 2
7 0
6
3
4
1
5
2
7
0

$\log_2 n$

$\log_2 8$

$$\log_2 2^3$$

3


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

$n \log_2 n$

$n \log n$

$O(n^2)$

$O(n \log n)$ merge sort

$O(n)$

$O(\log n)$

$O(1)$

$\Omega(n^2)$

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

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

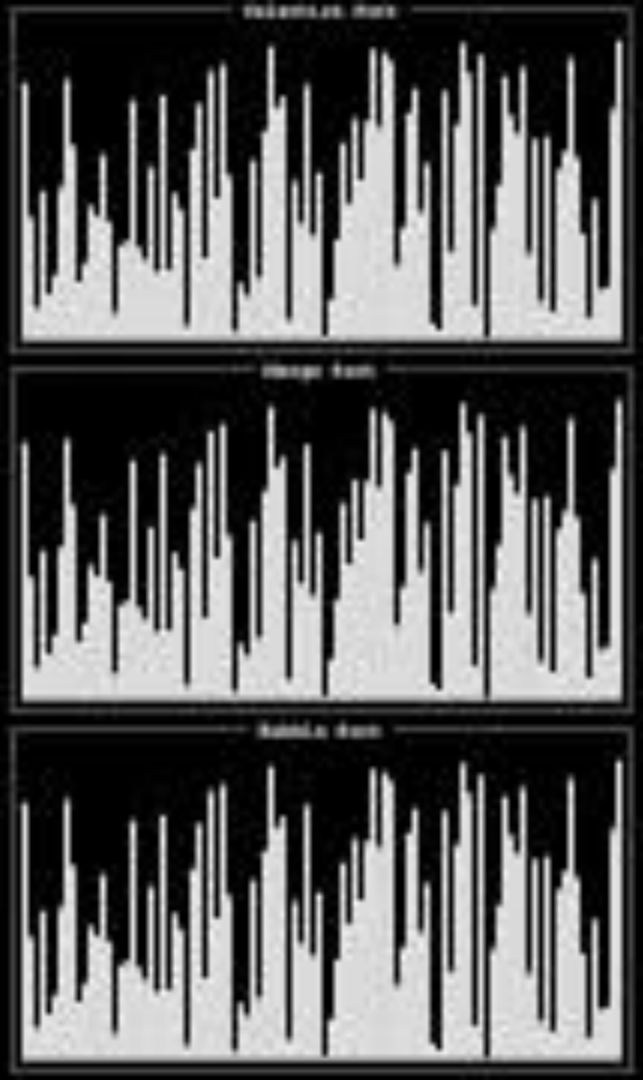
$\Theta(n^2)$

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

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$



This is CS50