This is CS50

Guvf vf PF50

cs50.ly/survey

Think.
Pair.
Share.

cs50.ly/questions

- When are **structs** useful?
- What goes into defining and using our own **functions**?
- What is **Big Oh notation**, and why do we care?
- How can we identify a **sorting** algorithm from nothing but its binary?

Structs

```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

```
candidate new_candidate;
candidate new_candidate;
new_candidate.name = "Alyssa";
new_candidate.votes = 10;
```

Struct Exercise

Create a struct to represent a candidate in an election that minimally includes:

- The candidate's name (as a string)
- The candidate's probability of winning (as a float)

Add attributes to a candidate and print those out to the user.

Structs and Functions Exercise

Create your own **get_candidate** function that prompts the user to input attributes for a candidate. You may rely on **get_string**, **get_float**, etc., and your function should return a candidate.

```
int count_votes(string candidate_name);
int count_votes(string candidate_name)
{
  // Code in our function
}
```

```
int count_votes(string candidate_name)
{
  int votes;
  // Code in our function
  return votes;
}
```

```
int total_votes;
total_votes = count_votes("Carter");
```

Arrays of Structs Exercise

Use your **get_candidate** function to create an array of three candidates, each of which should have attributes input by the user.

name	Alice	Bob	Charlie
votes	2	1	3

```
candidates[0];
```

name	Alice	Bob	Charlie
votes	2	1	3

```
candidates[0].name;
```

name	Alice	Bob	Charlie
votes	2	1	3

```
candidates[0].votes;
```

Searching

Searching an Array

Within your array of candidates, use **linear search** to find the first candidate that has a probability of winning that is greater than or equal to 0.51.

Print the candidate's name to the screen and stop looping.

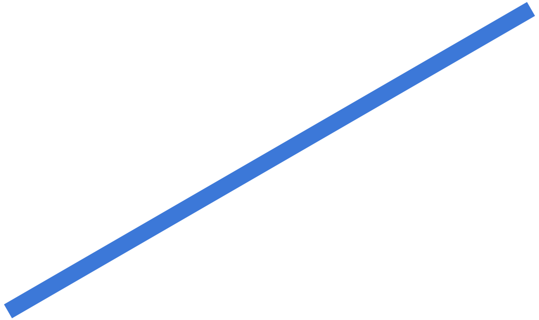
Runtime analysis

O(N) — "worst case" definition

In the worst case, I need to do approximately N steps for an input of size N.

O(N) — "scaling" definition

For every new item that gets added to my input, I need to do a new step. We say "our runtime scales **linearly** with the size of our input".

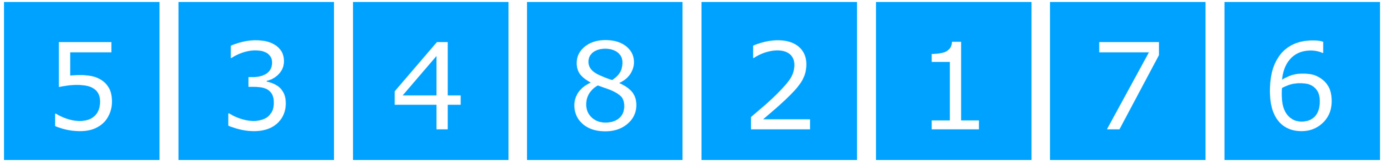


Ω(1) — "best case" definition

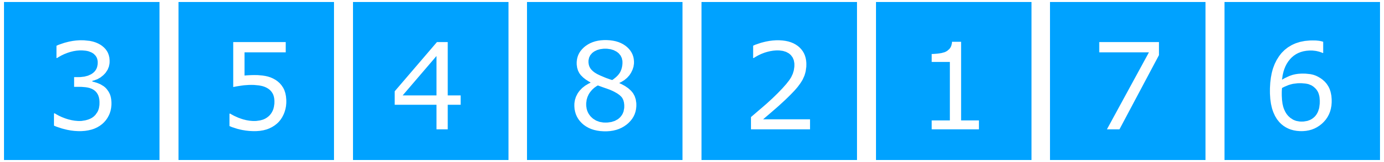
In the best case, I only need to do a constant number of steps to find my solution.

Sorting

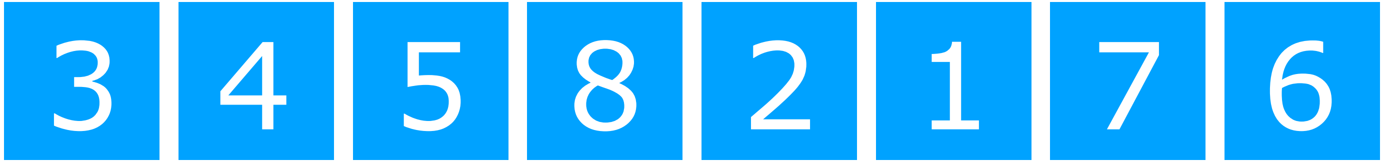
Bubble Sort



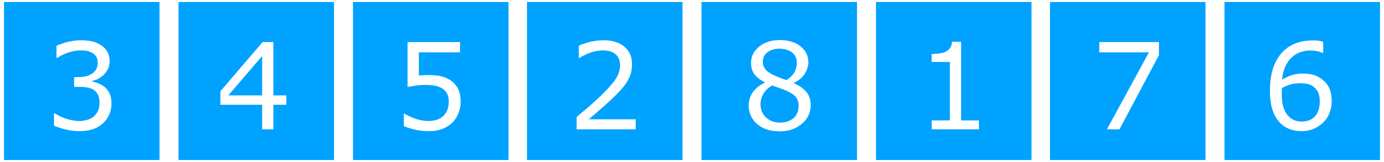
5	3	4	8	2	1	7	6



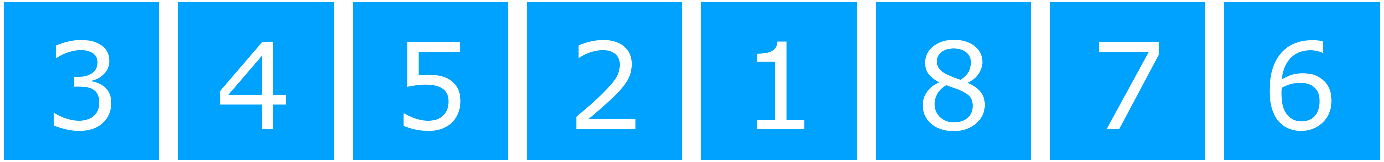
3	5	4	8	2	1	7	6



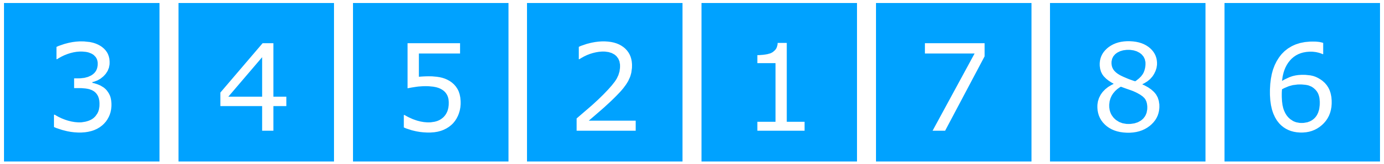
3 4 5 8 2 1 7 6



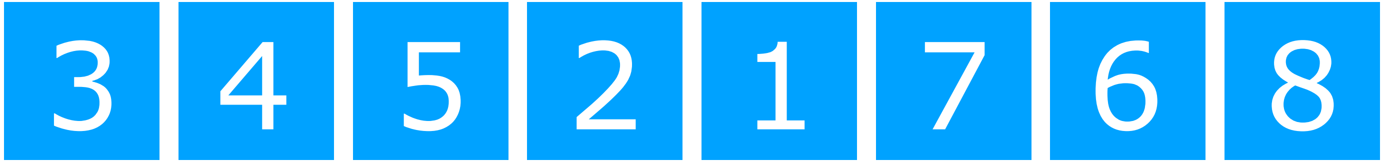
3 4 5 2 8 1 7 6



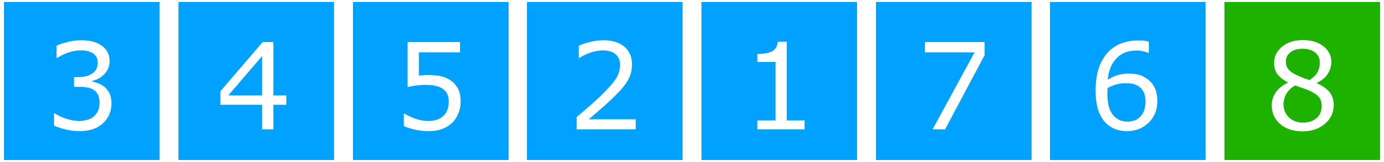
3 4 5 2 1 8 7 6



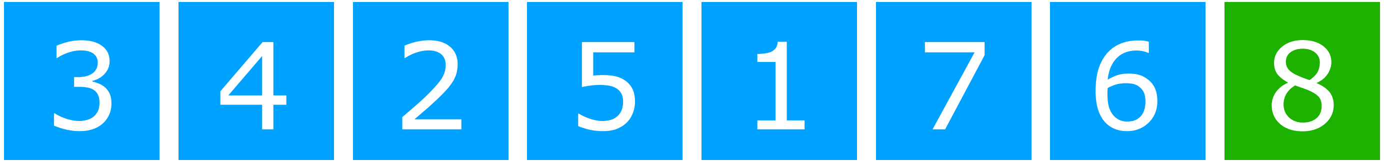
3	4	5	2	1	7	8	6



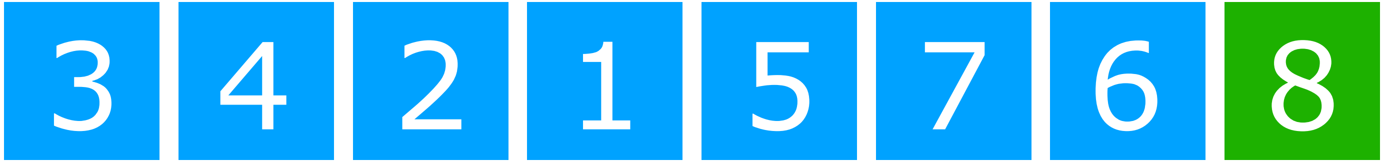
3	4	5	2	1	7	6	8



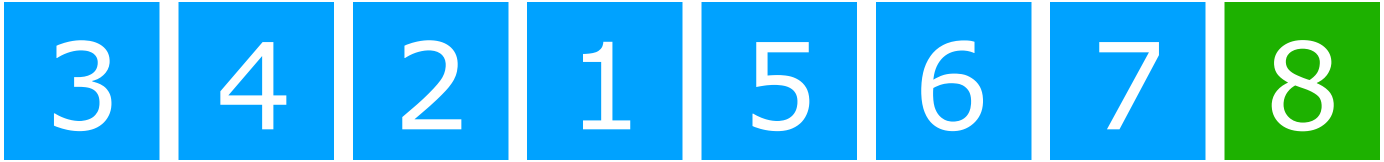
3	4	5	2	1	7	6	8



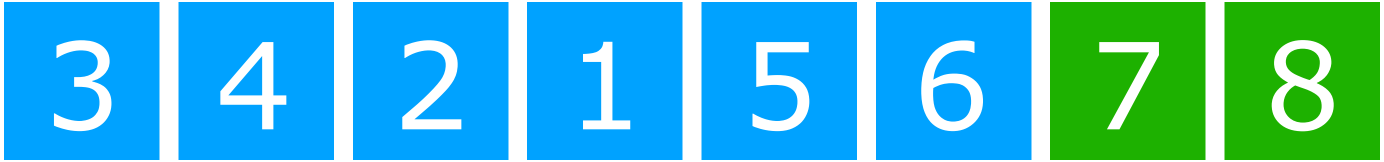
3	4	2	5	1	7	6	8



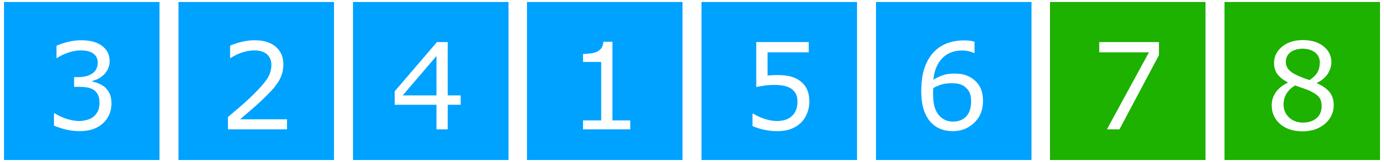
3	4	2	1	5	7	6	8



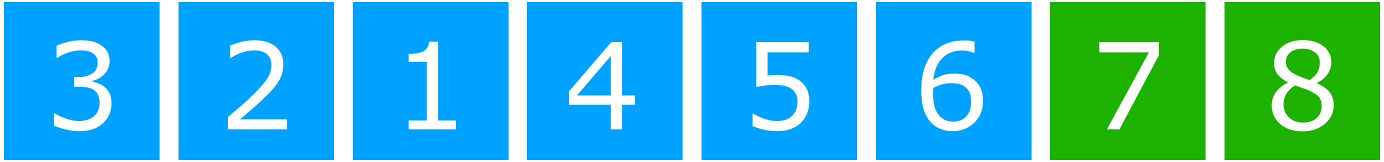
3 4 2 1 5 6 7 8



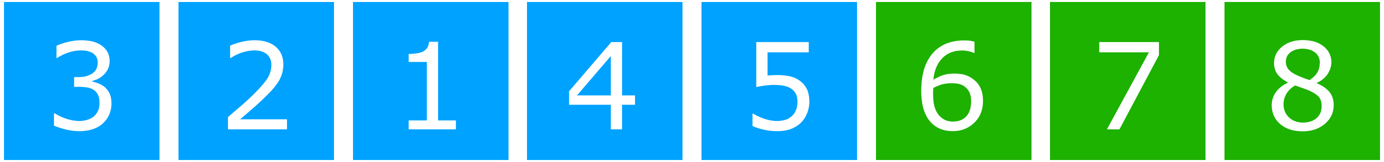
3 4 2 1 5 6 7 8

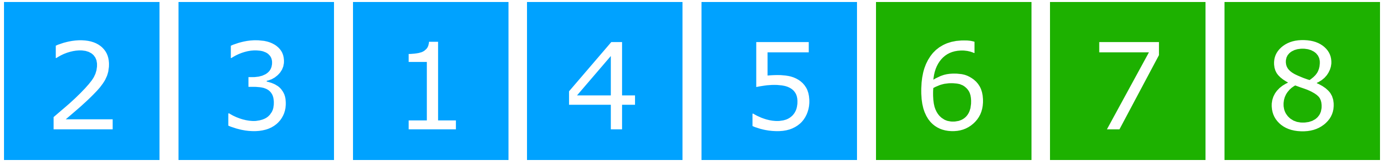


3 2 4 1 5 6 7 8

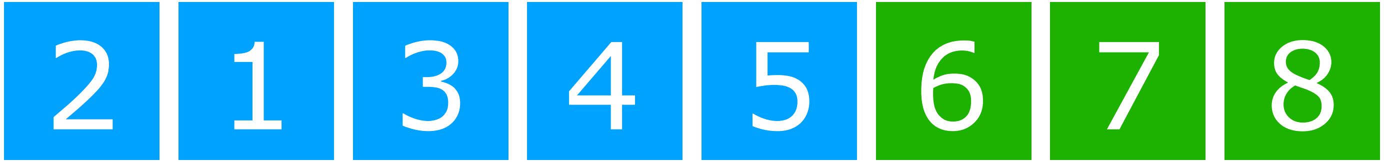


3 2 1 4 5 6 7 8


3
2
1
4
5
6
7
8

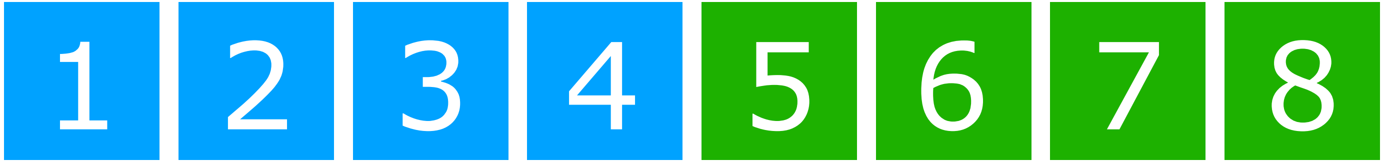


2 3 1 4 5 6 7 8

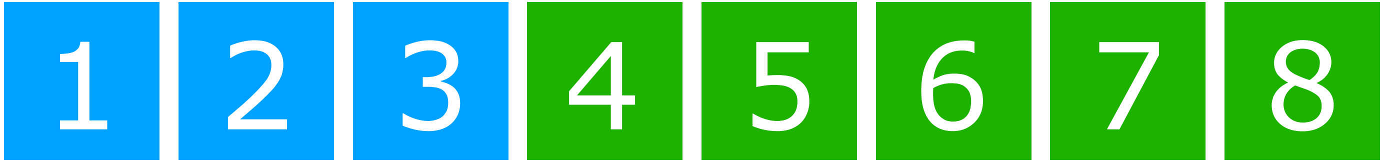


2	1	3	4	5	6	7	8

2	1	3	4	5	6	7	8



1 2 3 4 5 6 7 8



1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

```
Repeat for every element in our list, except last:
    Look at each element from first to second-to-last:
        If current and next elements out of order:
            Swap them
```

```
Repeat n - 1 times
    For j from 0 to n - 2
        If j'th and j + 1'th elements out of order
            Swap them
```

Bubble Sort Analysis

Download **bubble_solved.c** from the Week 3 page, under "Section". Upload it to VS Code and open the file.

Read: What questions do you have about the code, as written? What seems confusing?

Bubble Sort Analysis

Download **bubble_solved.c** from the Week 3 page, under "Section". Upload it to VS Code and open the file.

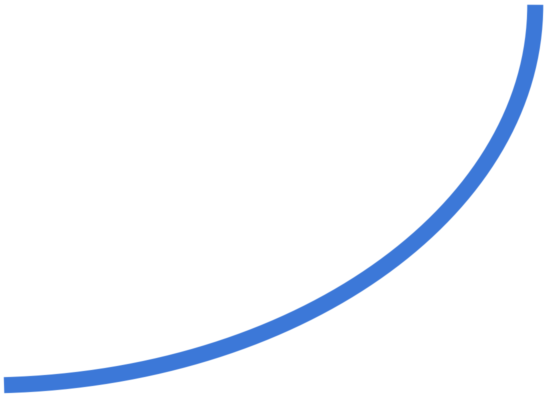
Discuss: Which pieces of code indicate that Bubble Sort runs in **$O(N^2)$** and **$\Omega(N)$**?

O(N^2) — "worst case" definition

In the worst case, I need to do approximately N^2 steps if my input size is N.

O(N^2) — "scaling" definition

For every new item that gets added to my input, I need to do approximately **N** new steps.

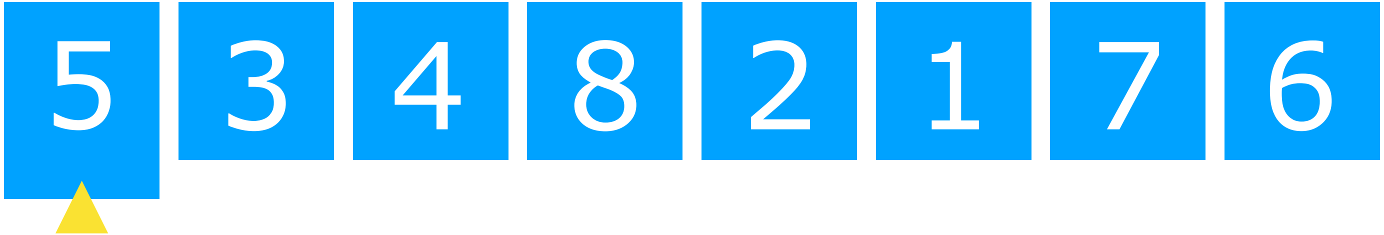


Ω(N) — "best case" definition

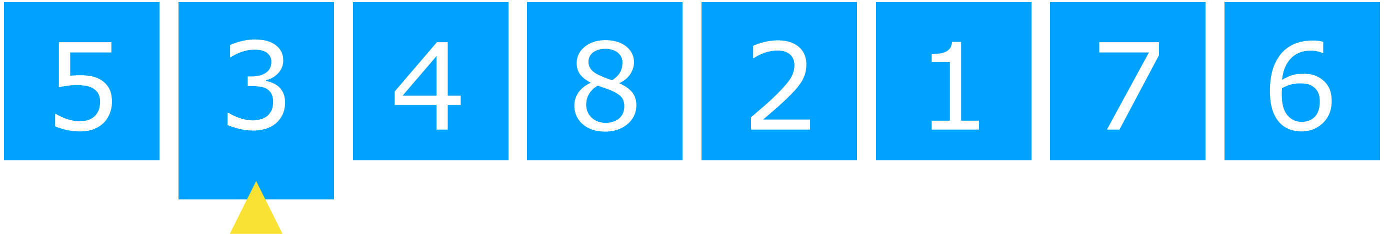
In the best case, I need to do approximately N steps if my input size is N.

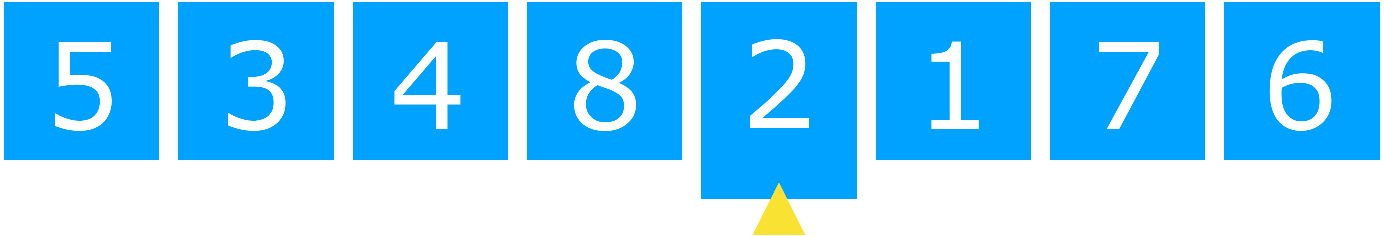
Selection Sort

5	3	4	8	2	1	7	6

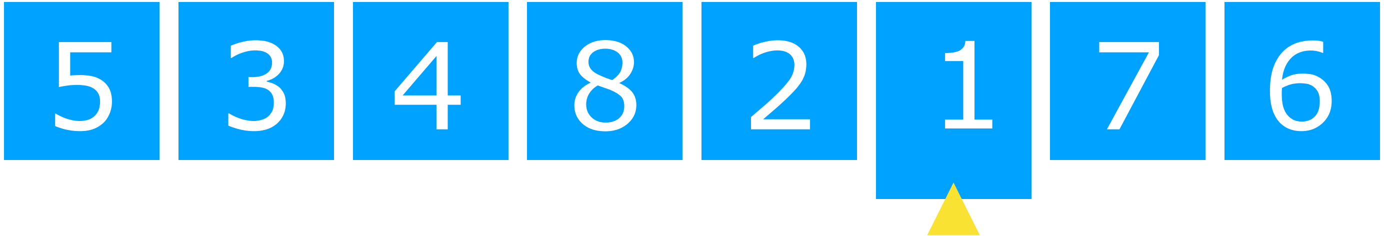


5	3	4	8	2	1	7	6

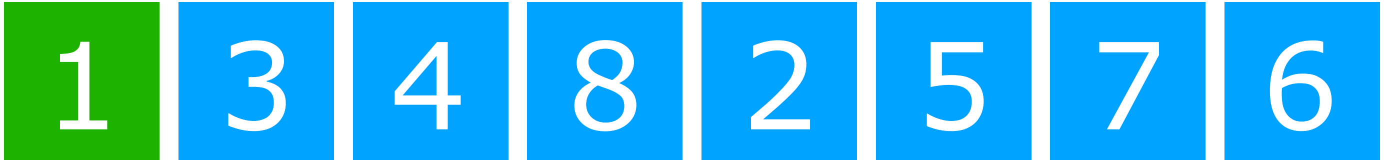

5
3
4
8
2
1
7
6



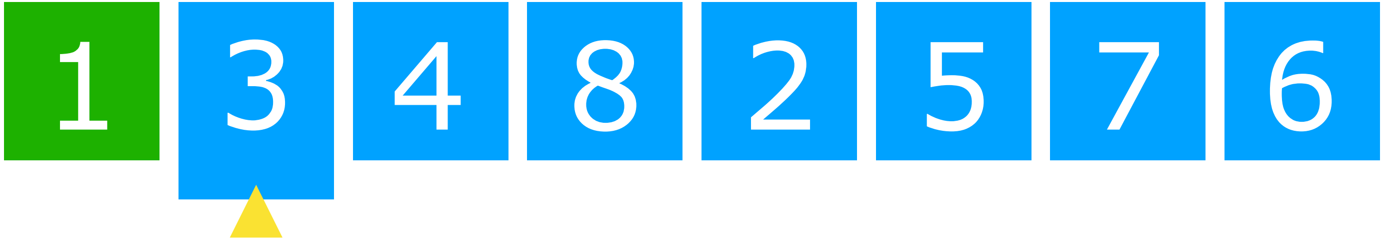
5	3	4	8	2	1	7	6

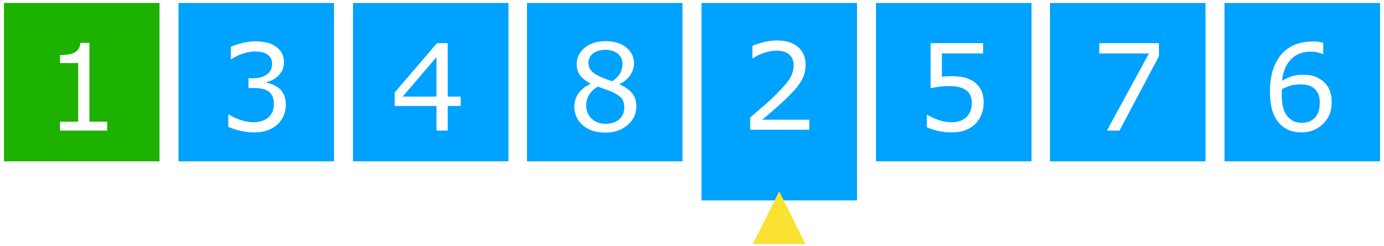


5 3 4 8 2 1 7 6

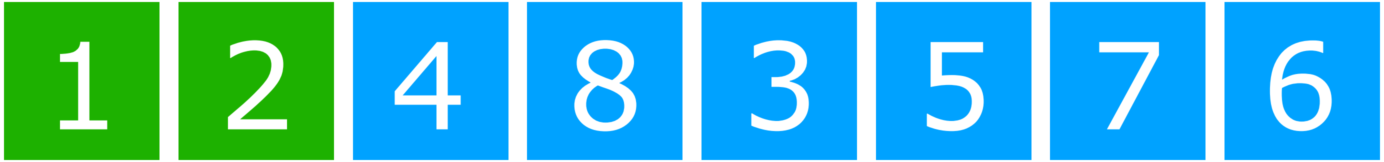


1	3	4	8	2	5	7	6

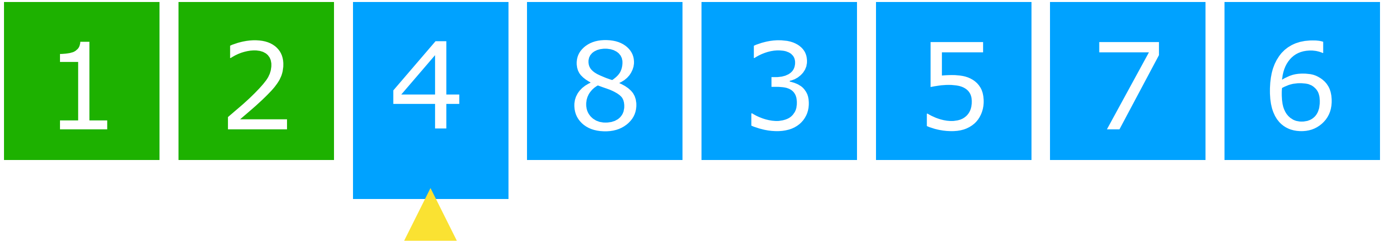

1
3
4
8
2
5
7
6



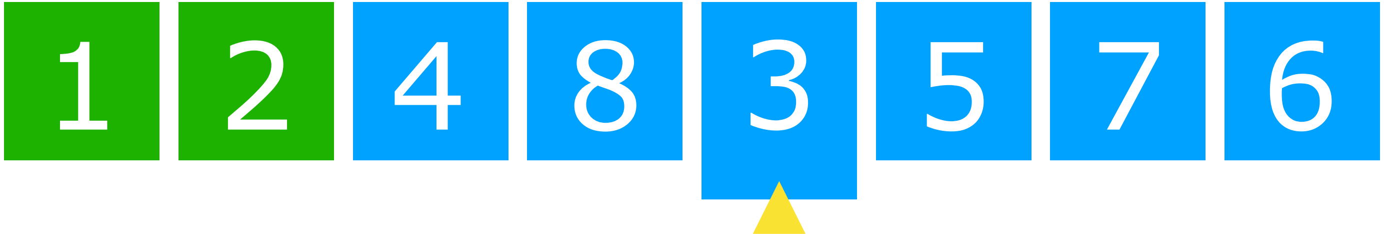
1 3 4 8 2 5 7 6



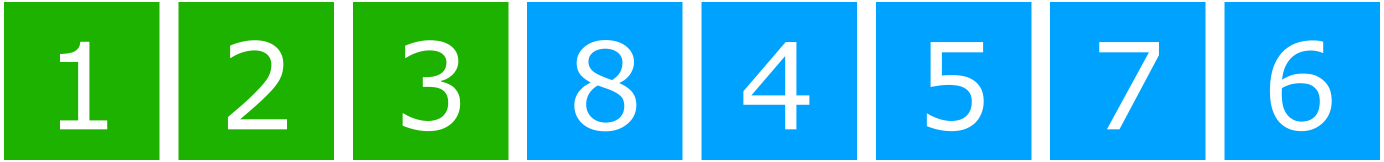
1 2 4 8 3 5 7 6



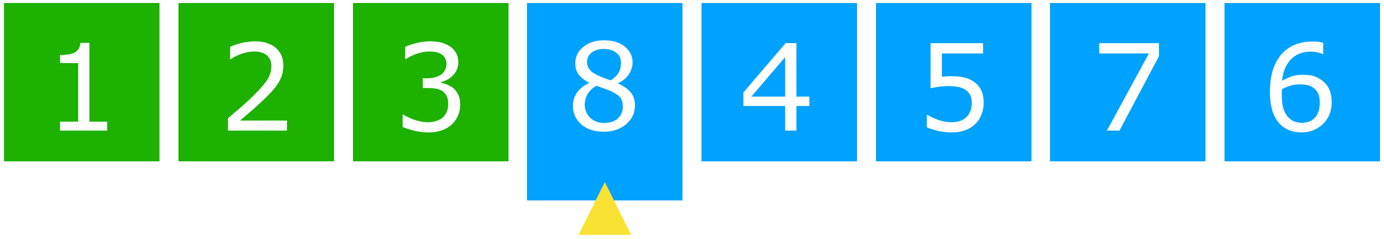
1 2 4 8 3 5 7 6



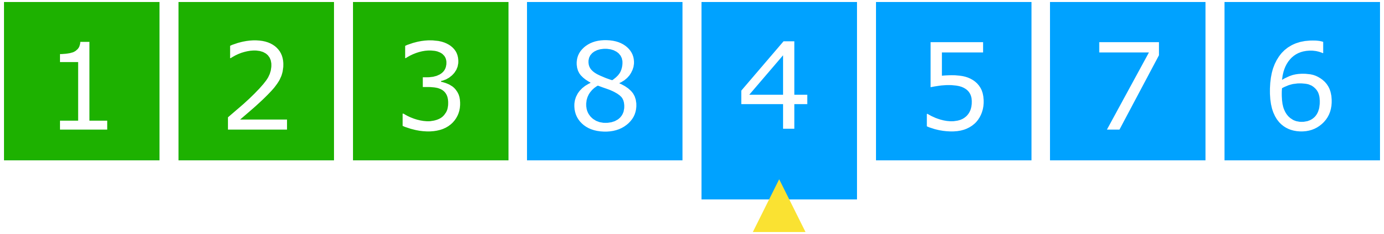
1	2	4	8	3	5	7	6



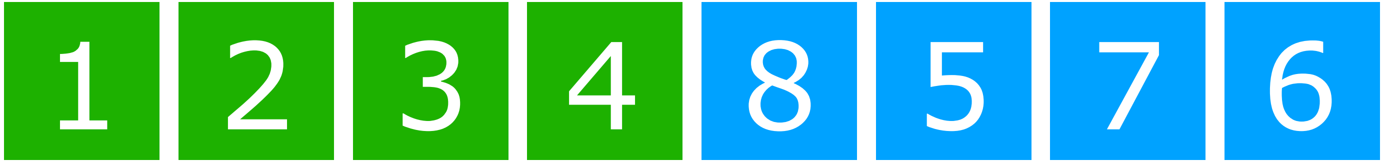
1	2	3	8	4	5	7	6



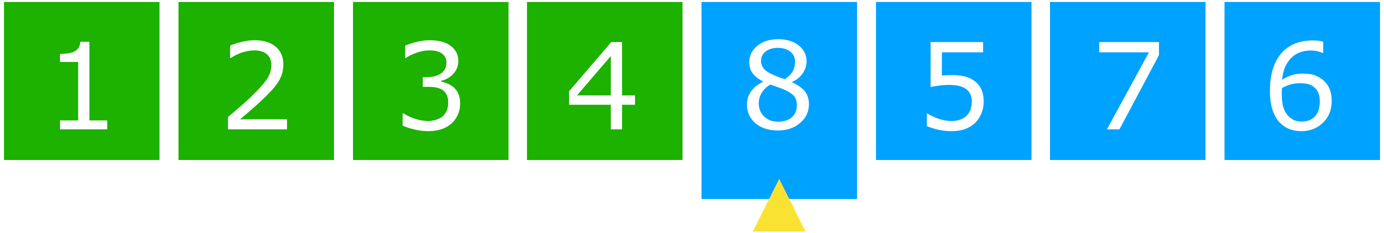
1	2	3	8	4	5	7	6

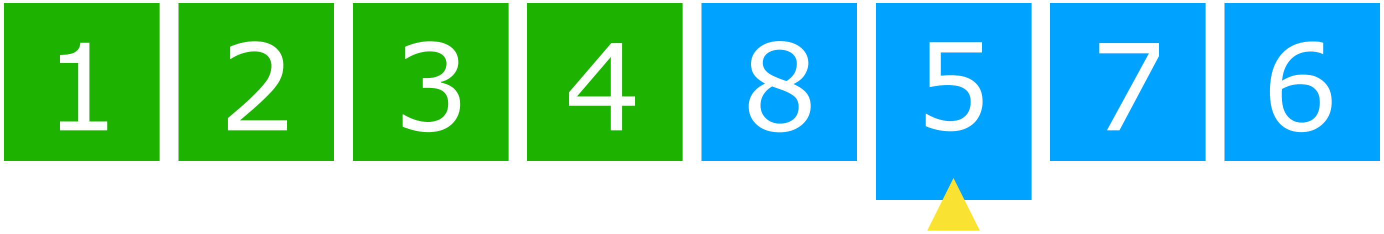


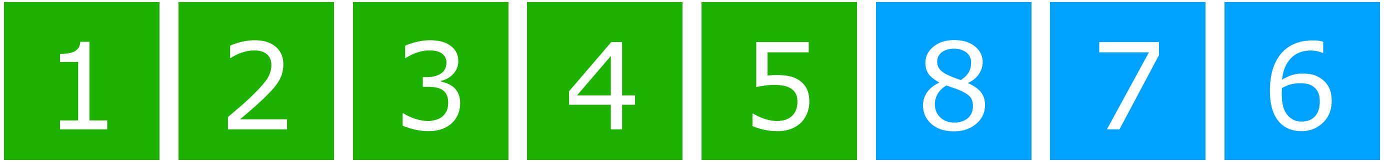
1 2 3 8 4 5 7 6



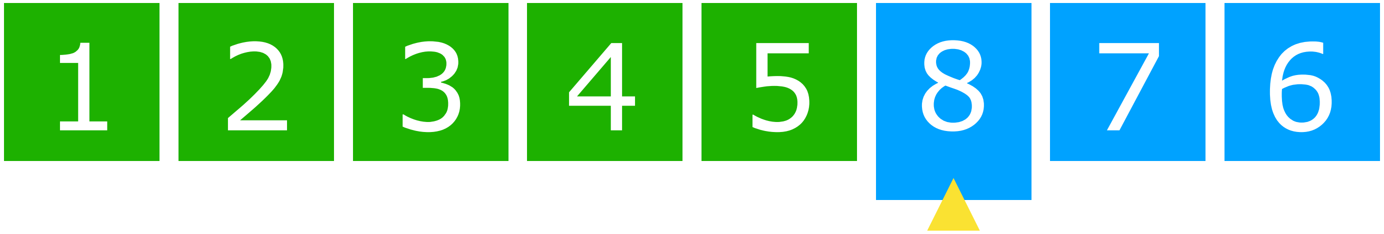
1 2 3 4 8 5 7 6


1
2
3
4
8
5
7
6

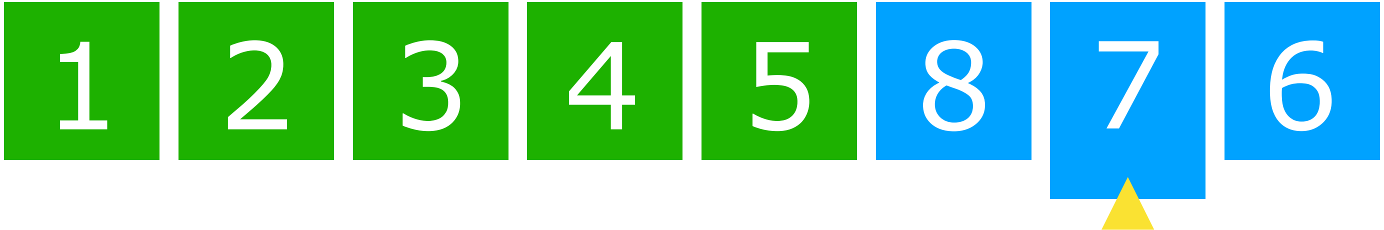

1
2
3
4
8
5
7
6



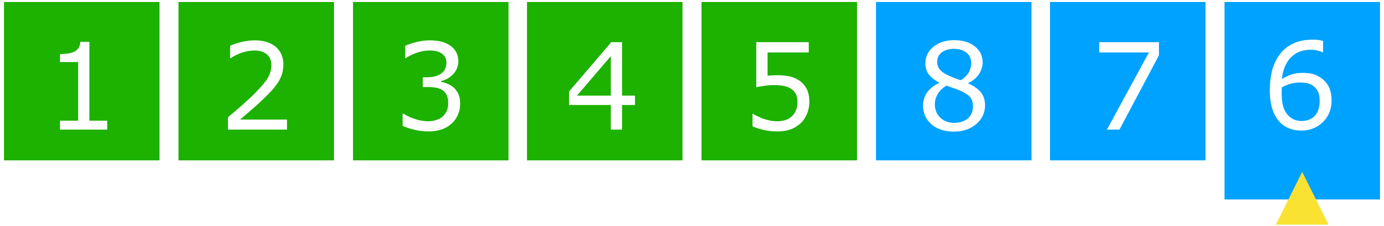
1	2	3	4	5	8	7	6



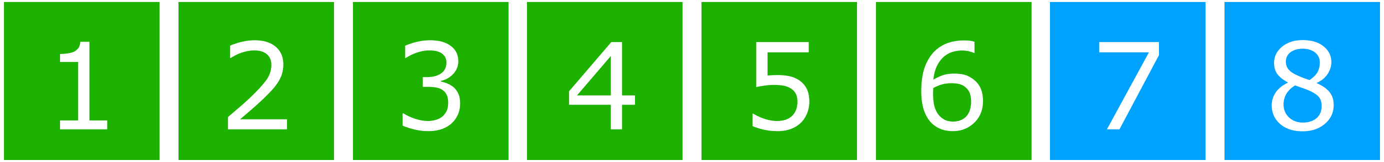
1 2 3 4 5 8 7 6



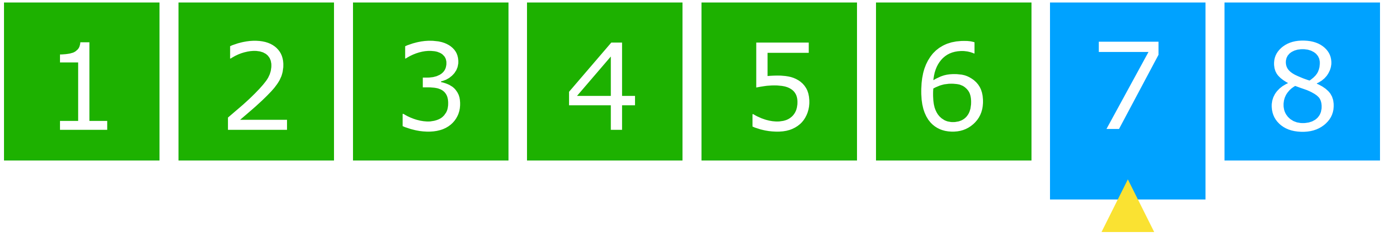
1 2 3 4 5 8 7 6



1 2 3 4 5 8 7 6



1 2 3 4 5 6 7 8



1	2	3	4	5	6	7	8

1 2 3 4 5 6 7 8

1	2	3	4	5	6	7	8

Selection Sort Analysis

Download **selection_solved.c** from the Week 3 page, under "Section". Upload it to VS Code and open the file.

Read: What questions do you have about the code, as written? What seems confusing?

Selection Sort Analysis

Download **selection_solved.c** from the Week 3 page, under "Section". Upload it to VS Code and open the file.

Discuss: How do you know Selection Sort runs in **$O(N^2)$** and **$\Omega(N^2)$**?

$O(N^2)$ — "worst case" definition

In the worst case, I need to do approximately N^2 steps if my input size is N.

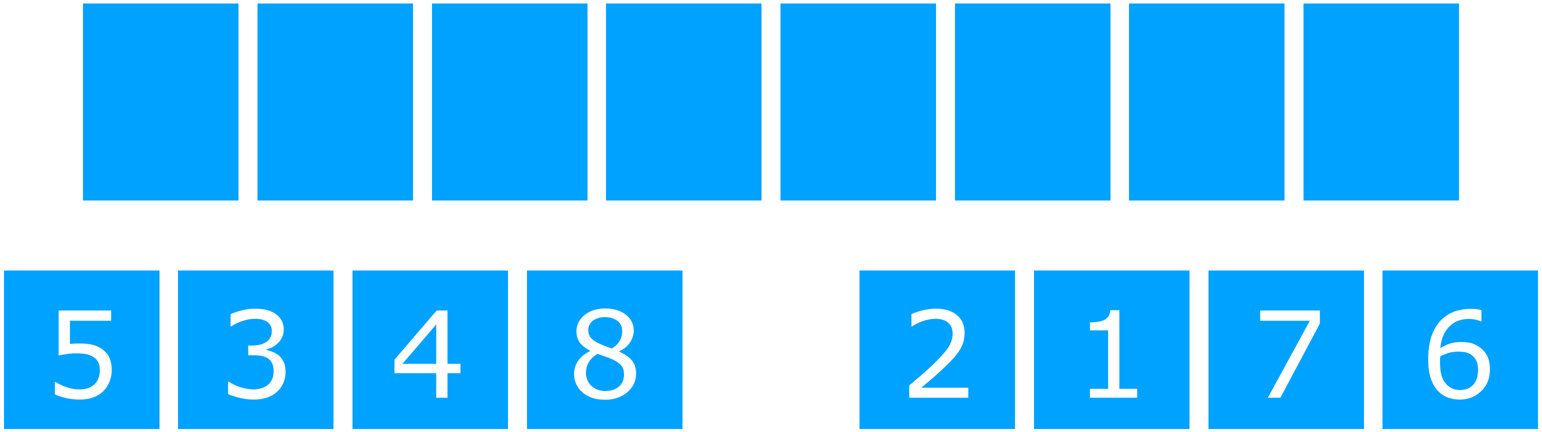
$\Omega(N^2)$ — "best case" definition

In the worst case, I need to do approximately N^2 steps if my input size is N.

Merge Sort

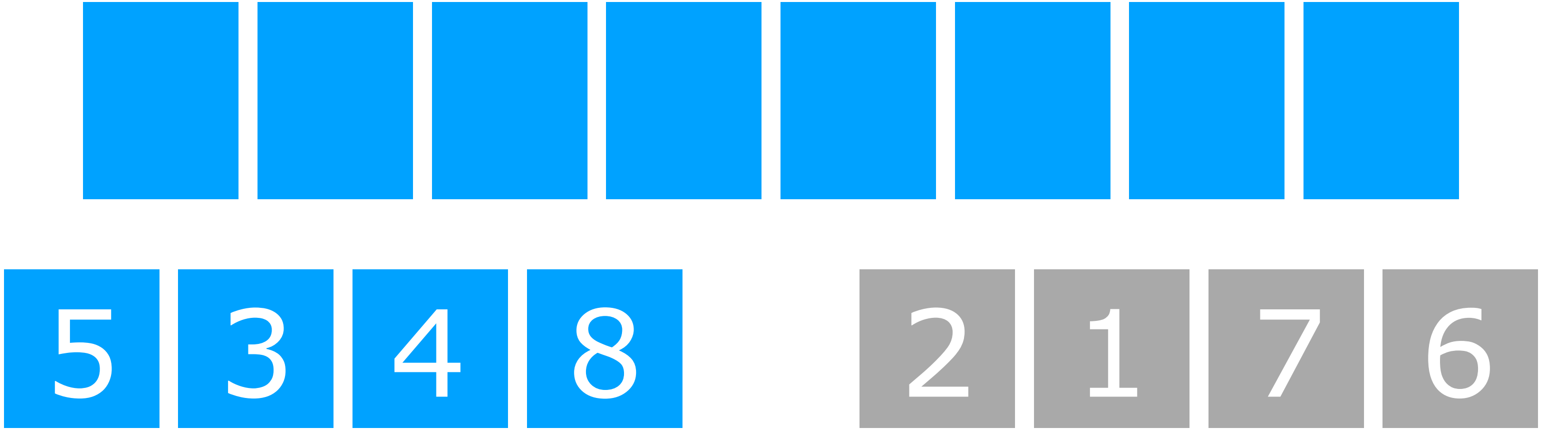
5	3	4	8	2	1	7	6

5	3	4	8	2	1	7	6



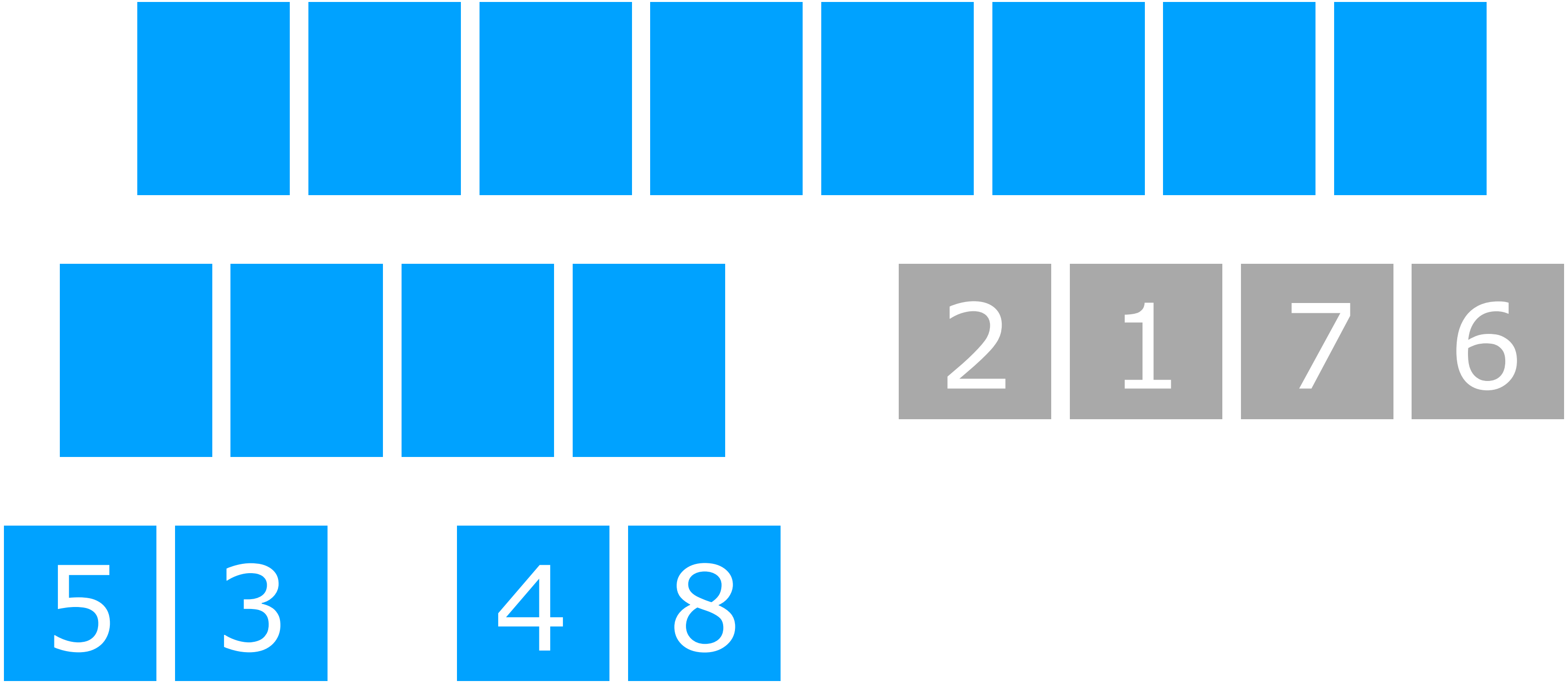
5 3 4 8

2 1 7 6

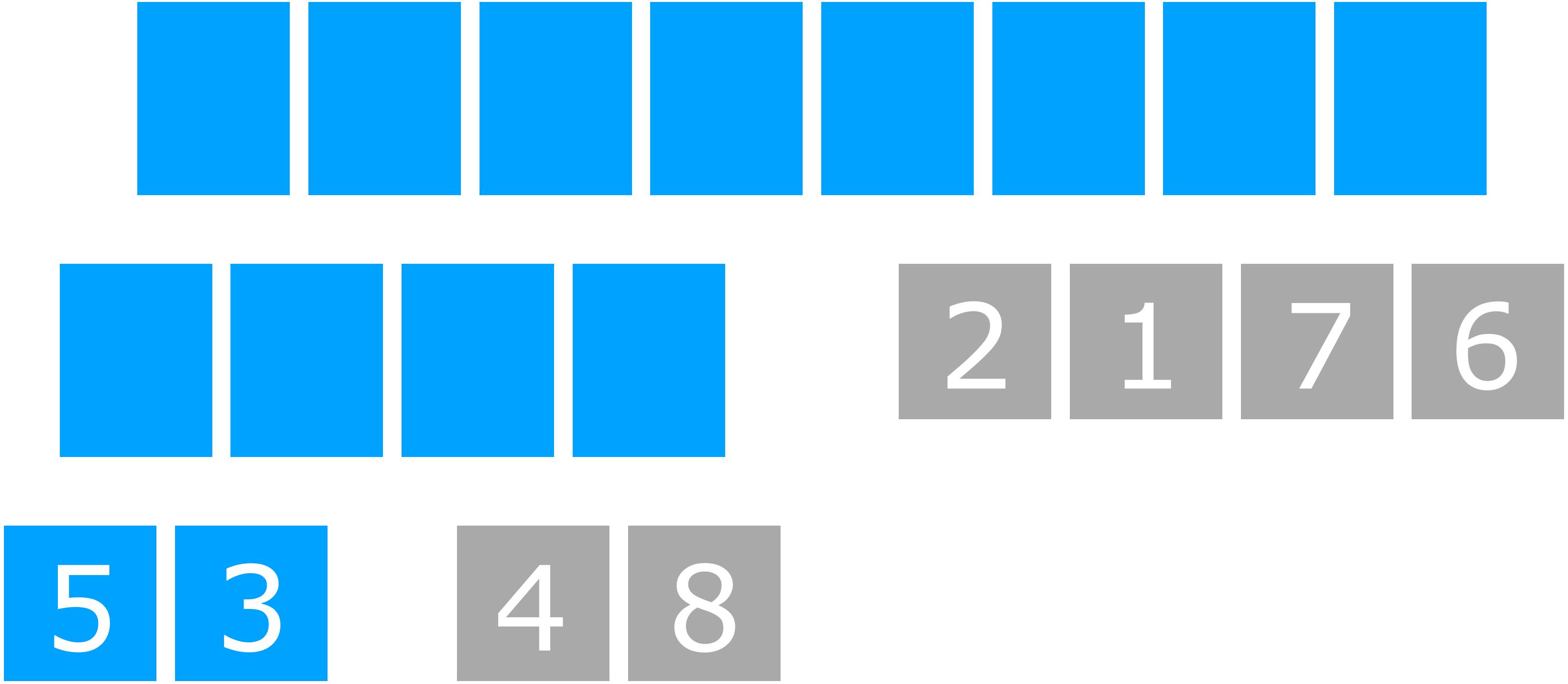


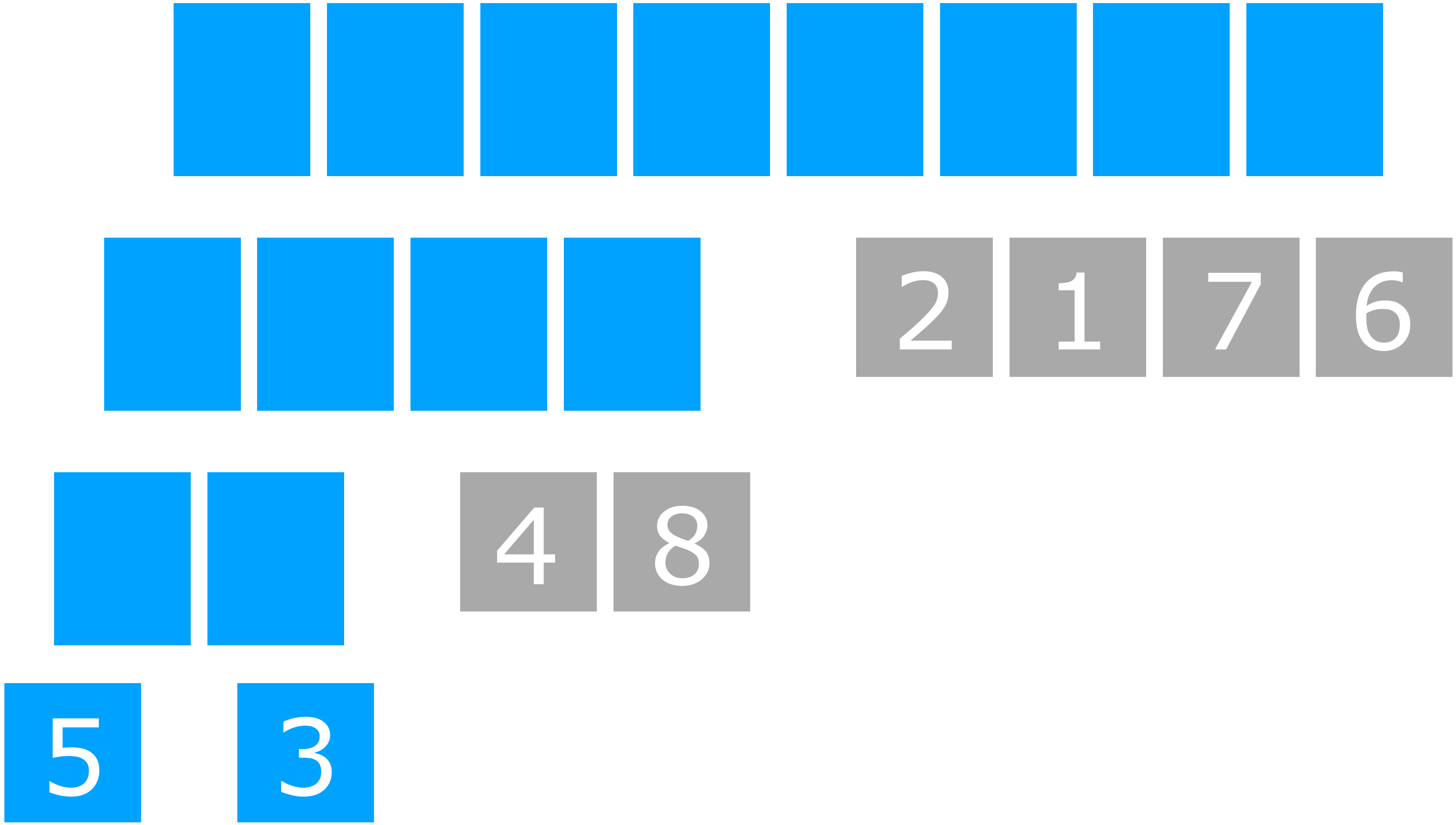
5 3 4 8

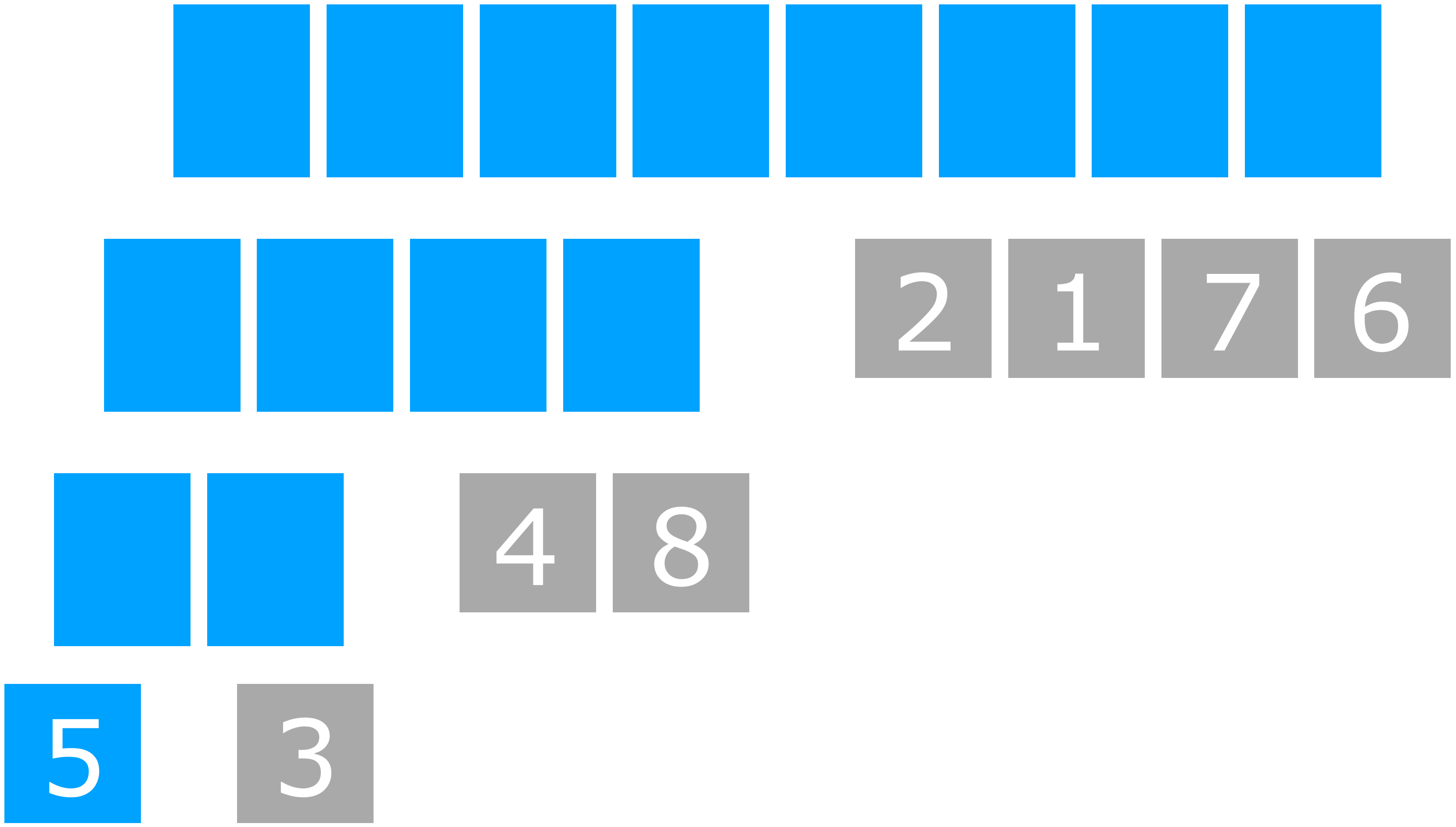
2 1 7 6


2 1 7 6

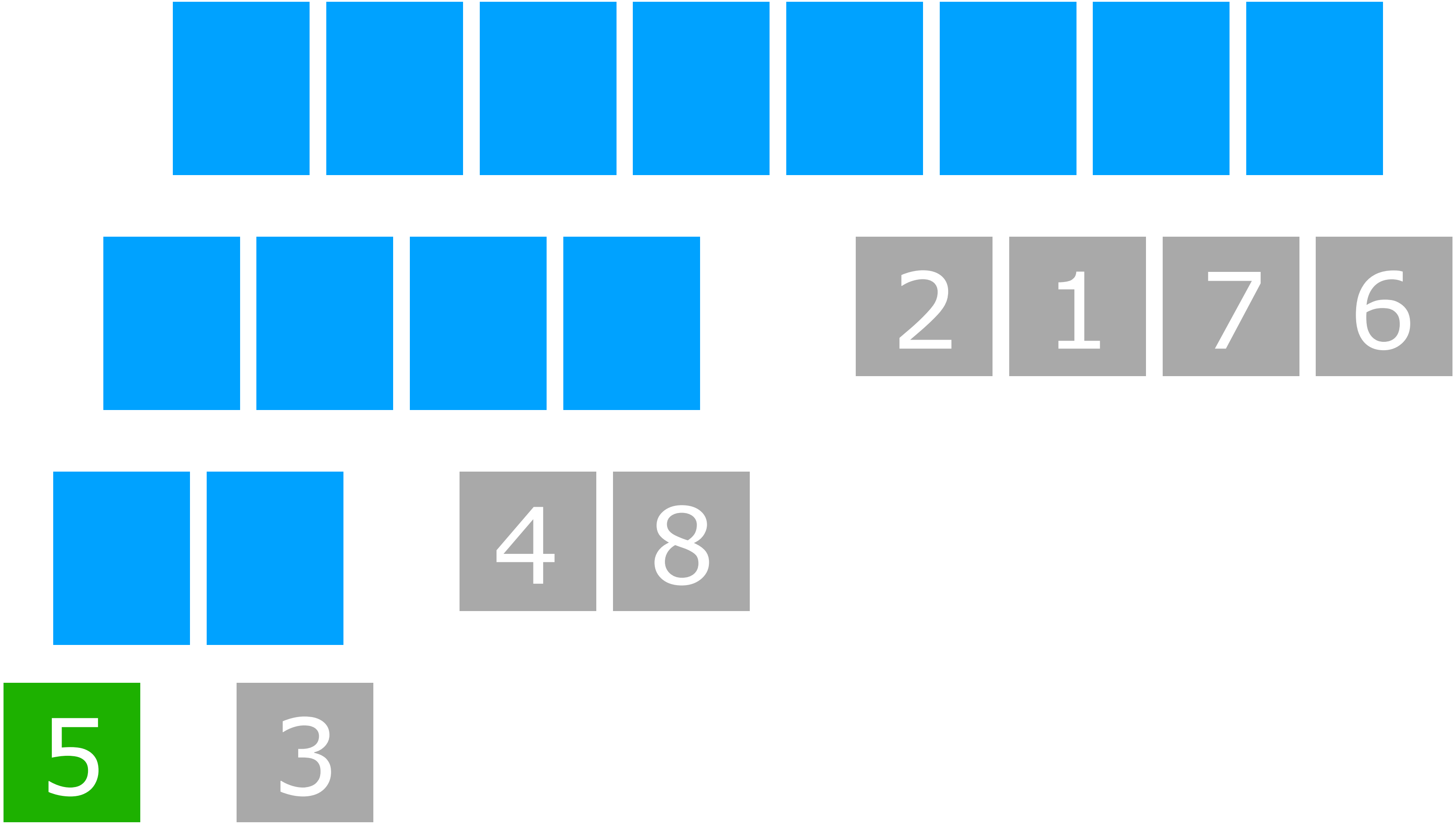
5 3 4 8

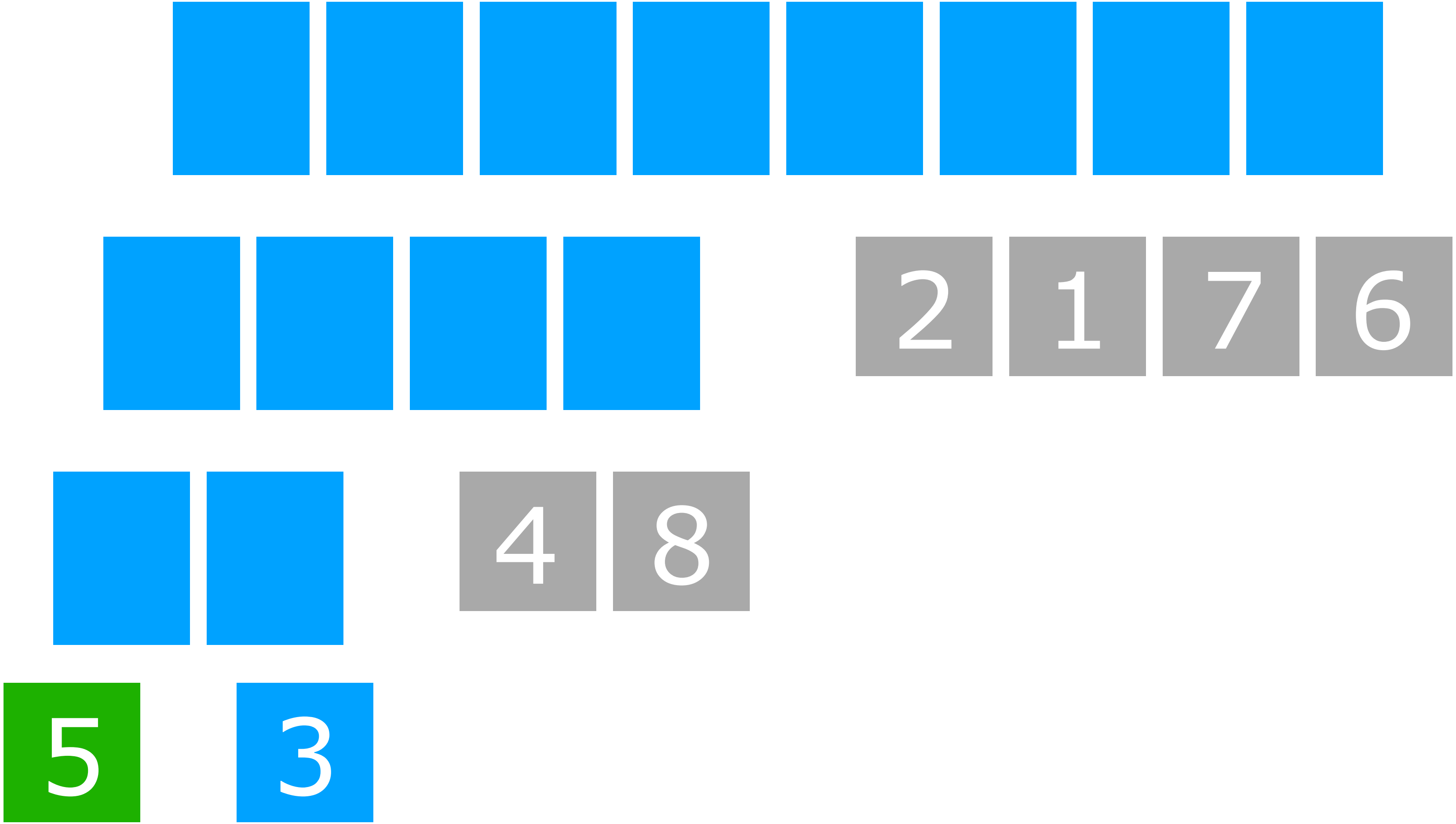

2
1
7
6
5
3
4
8

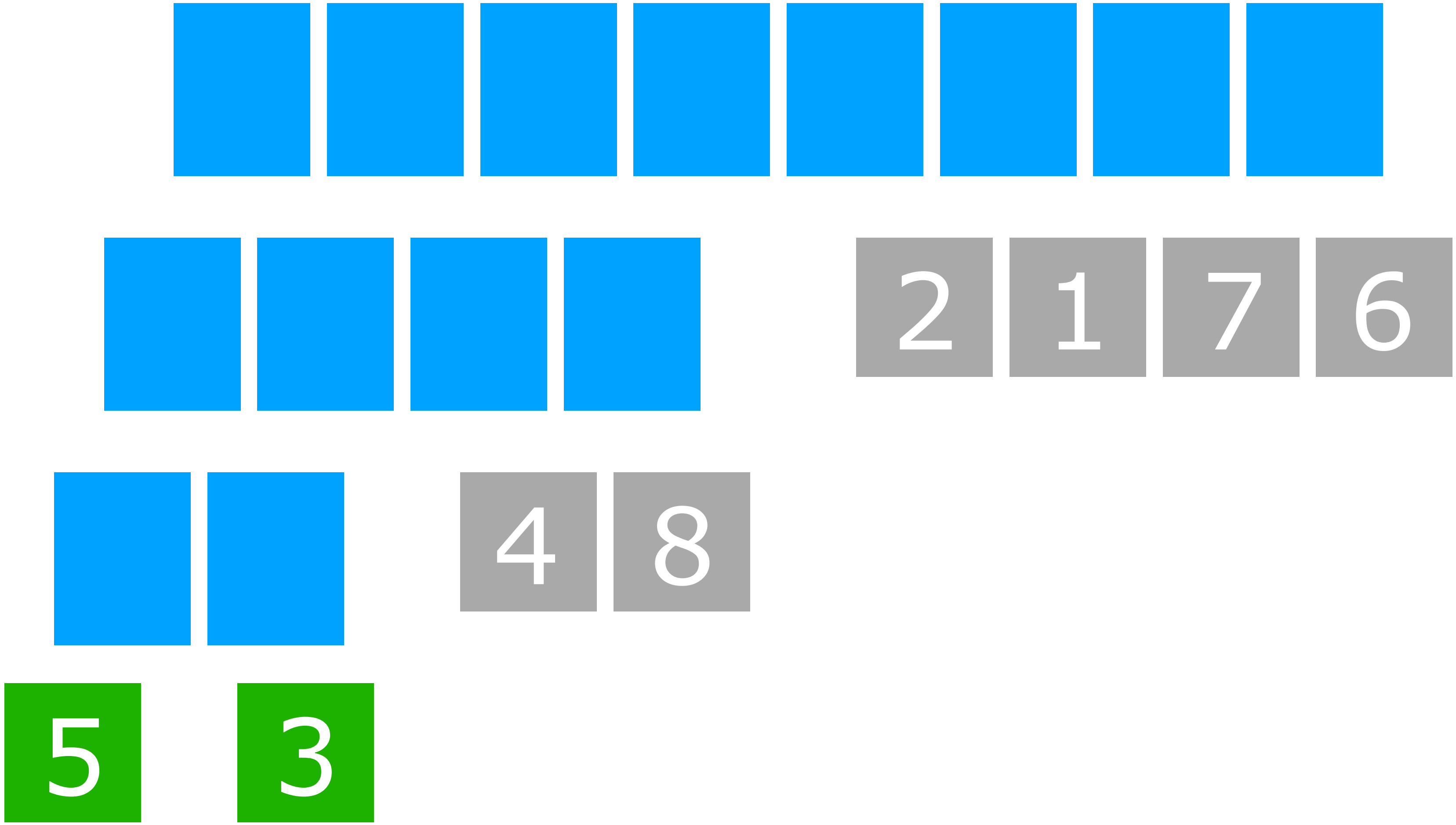

2
1
7
6
4
8
5
3

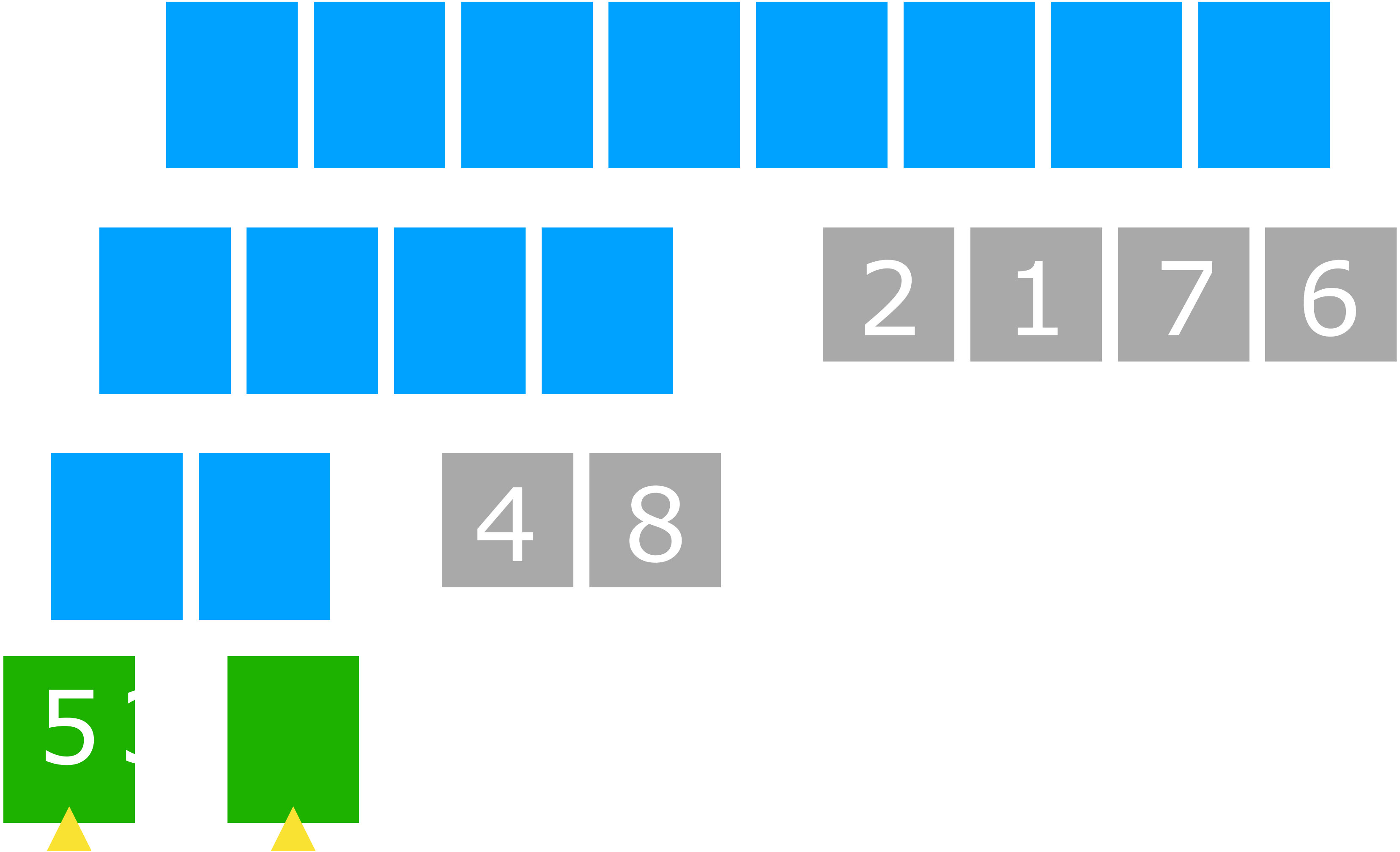


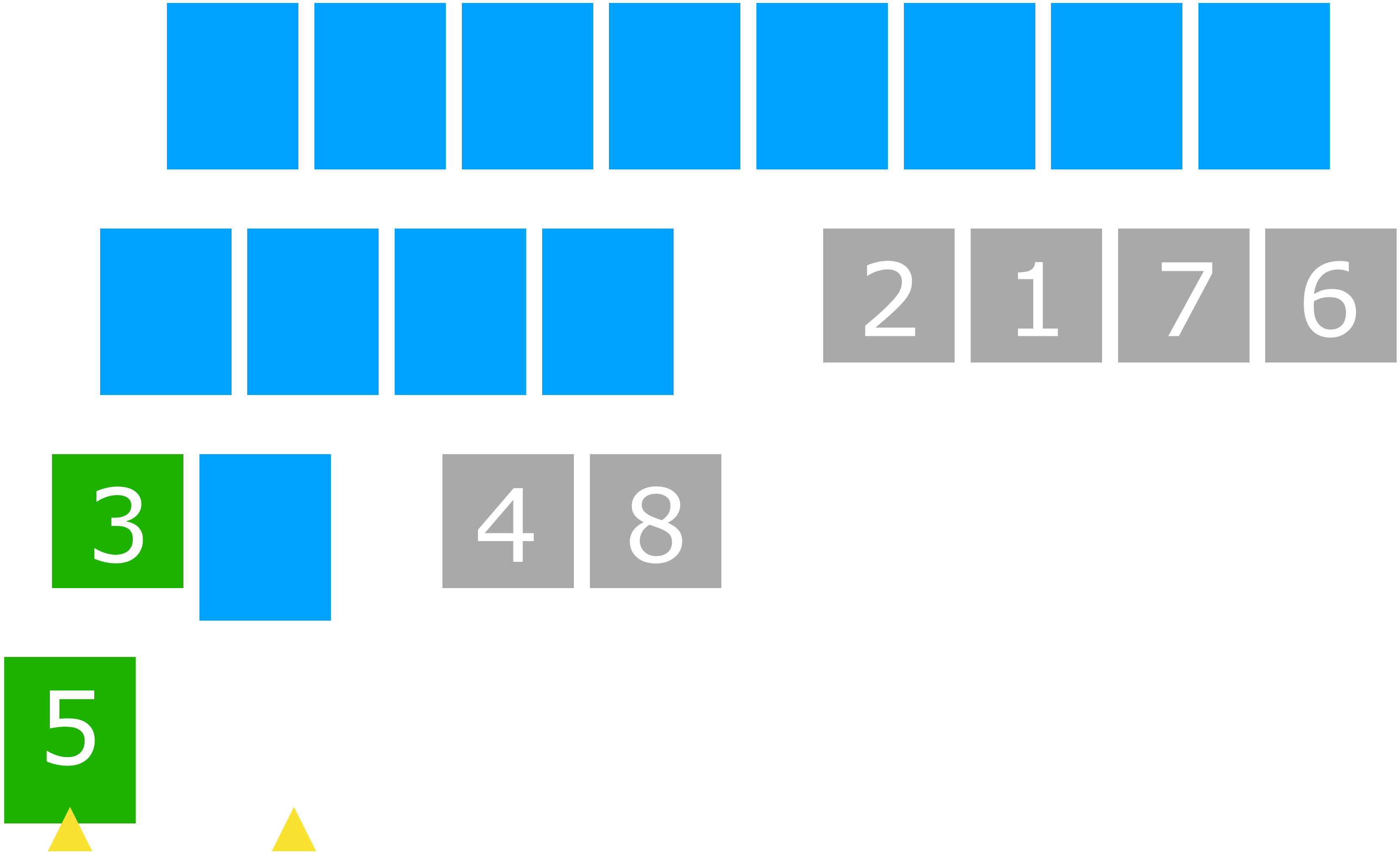
2
1
7
6
4
8
5
3

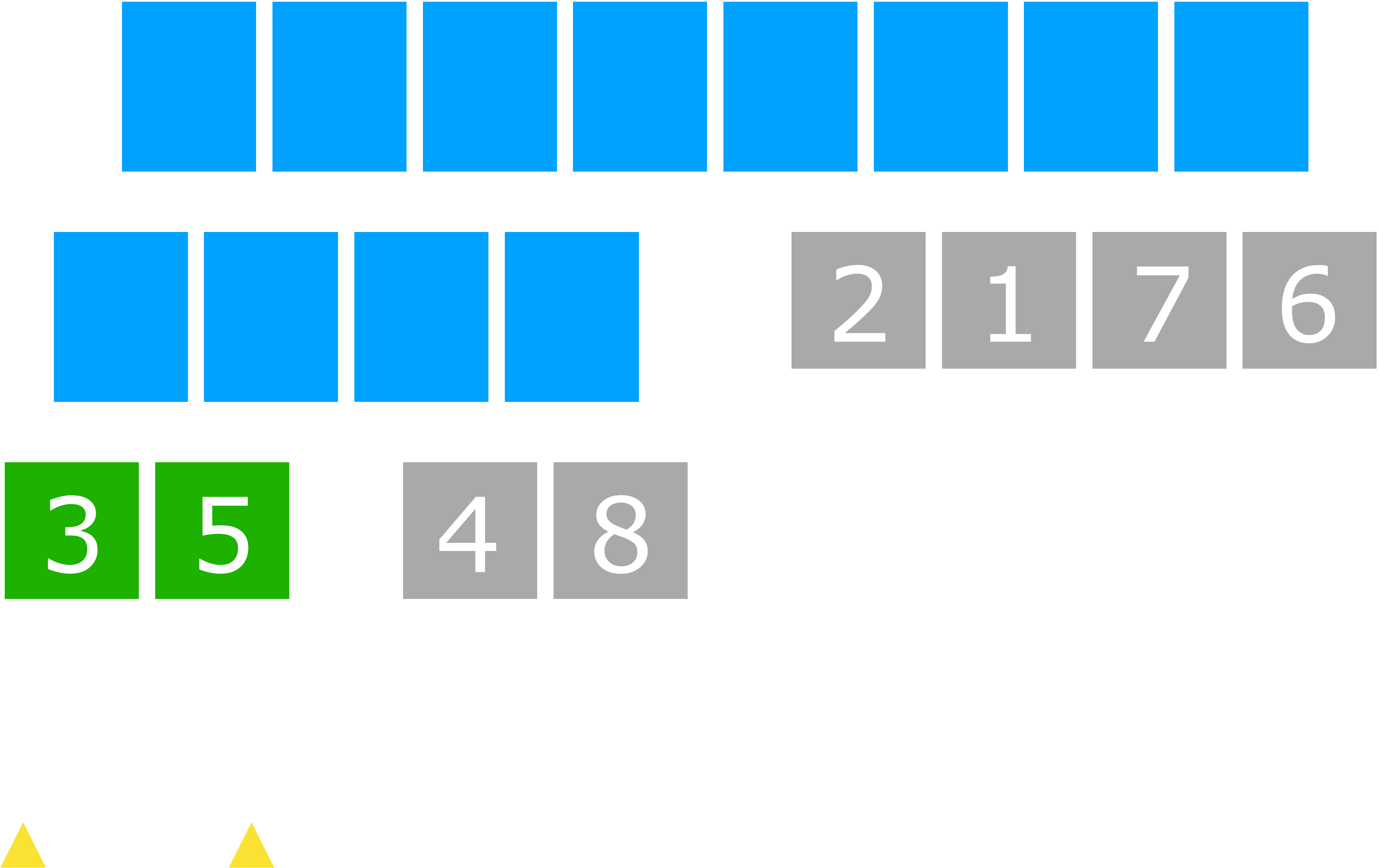

2
1
7
6
4
8
5
3

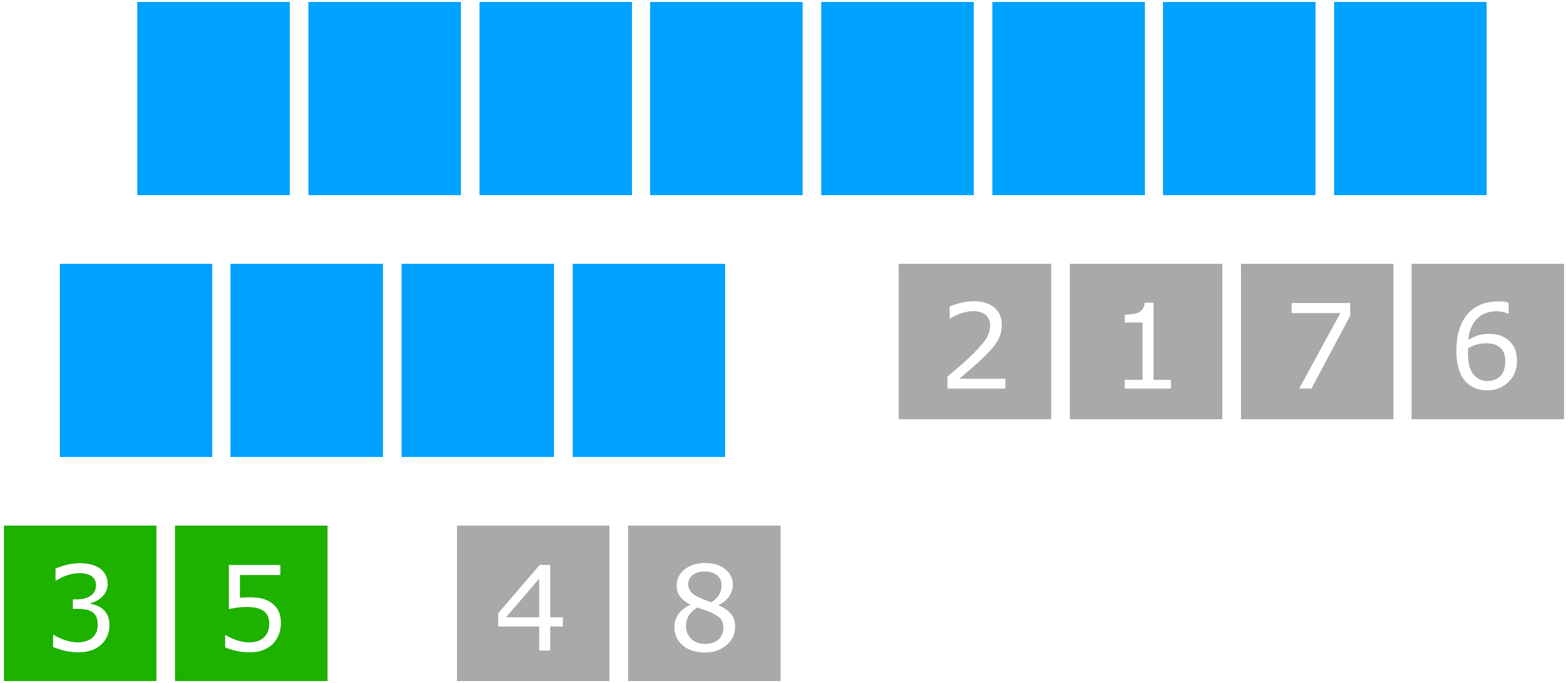

2
1
7
6
4
8
5
3


2 1 7 6
4 8
5 3

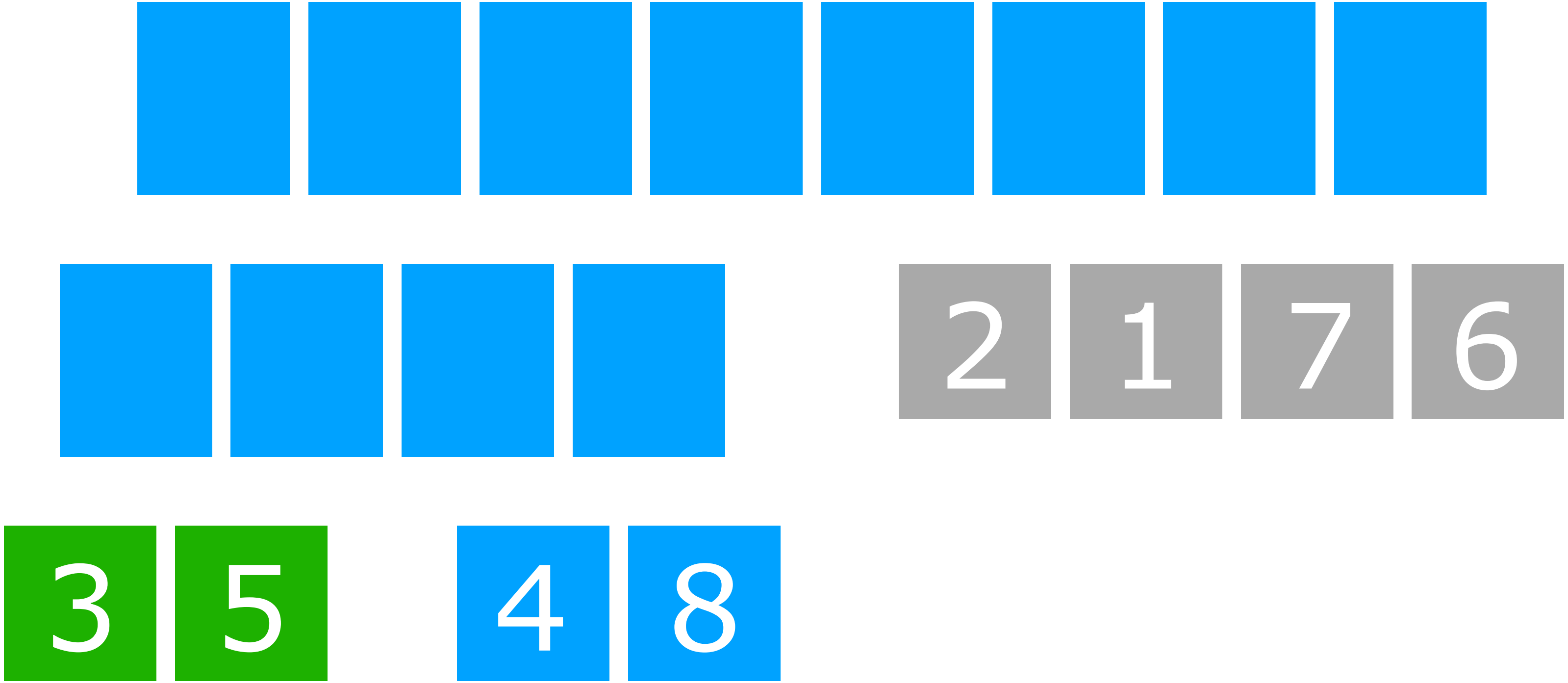

2
1
7
6
4
8
5

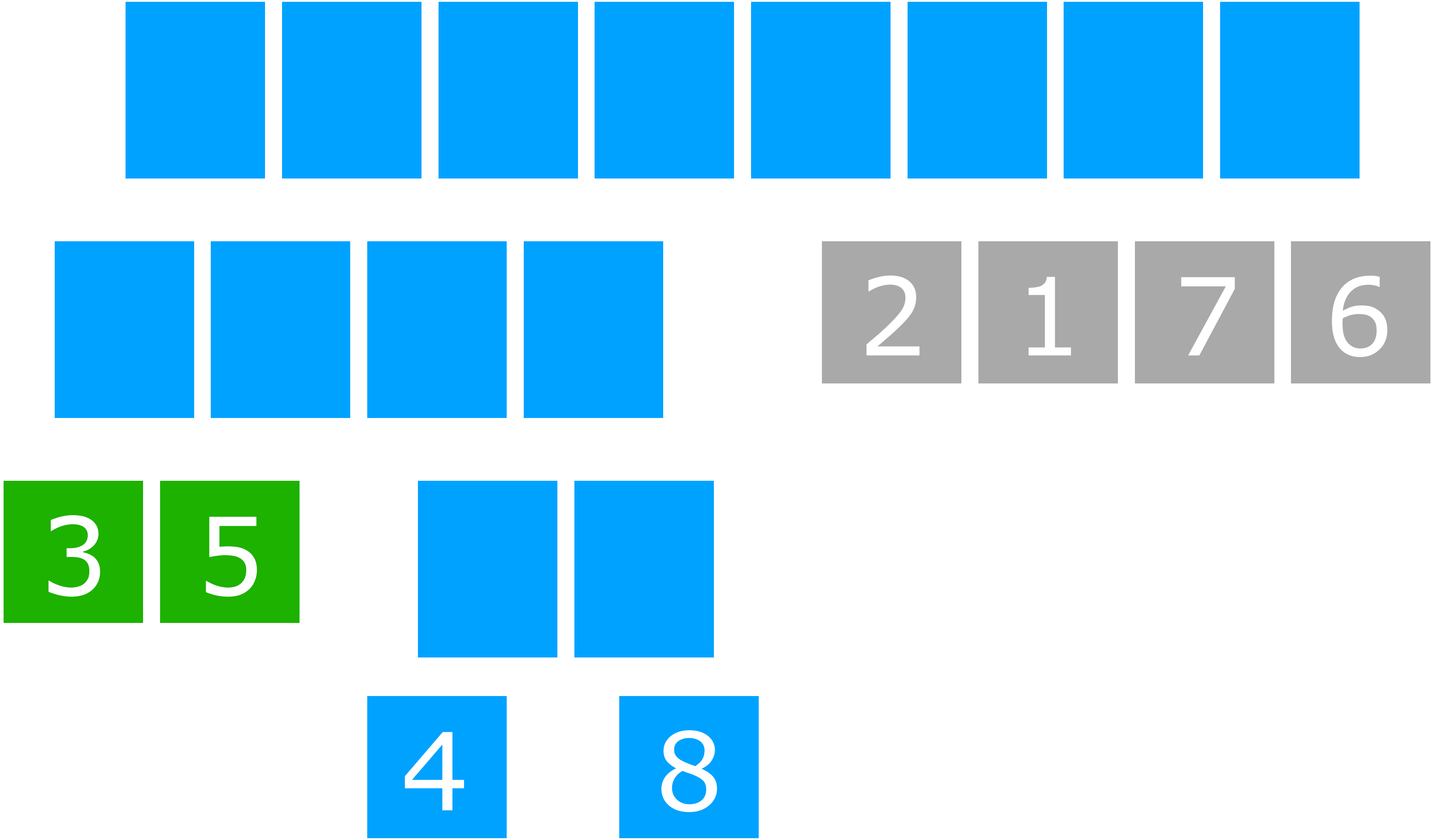

2
1
7
6
3
4
8
5

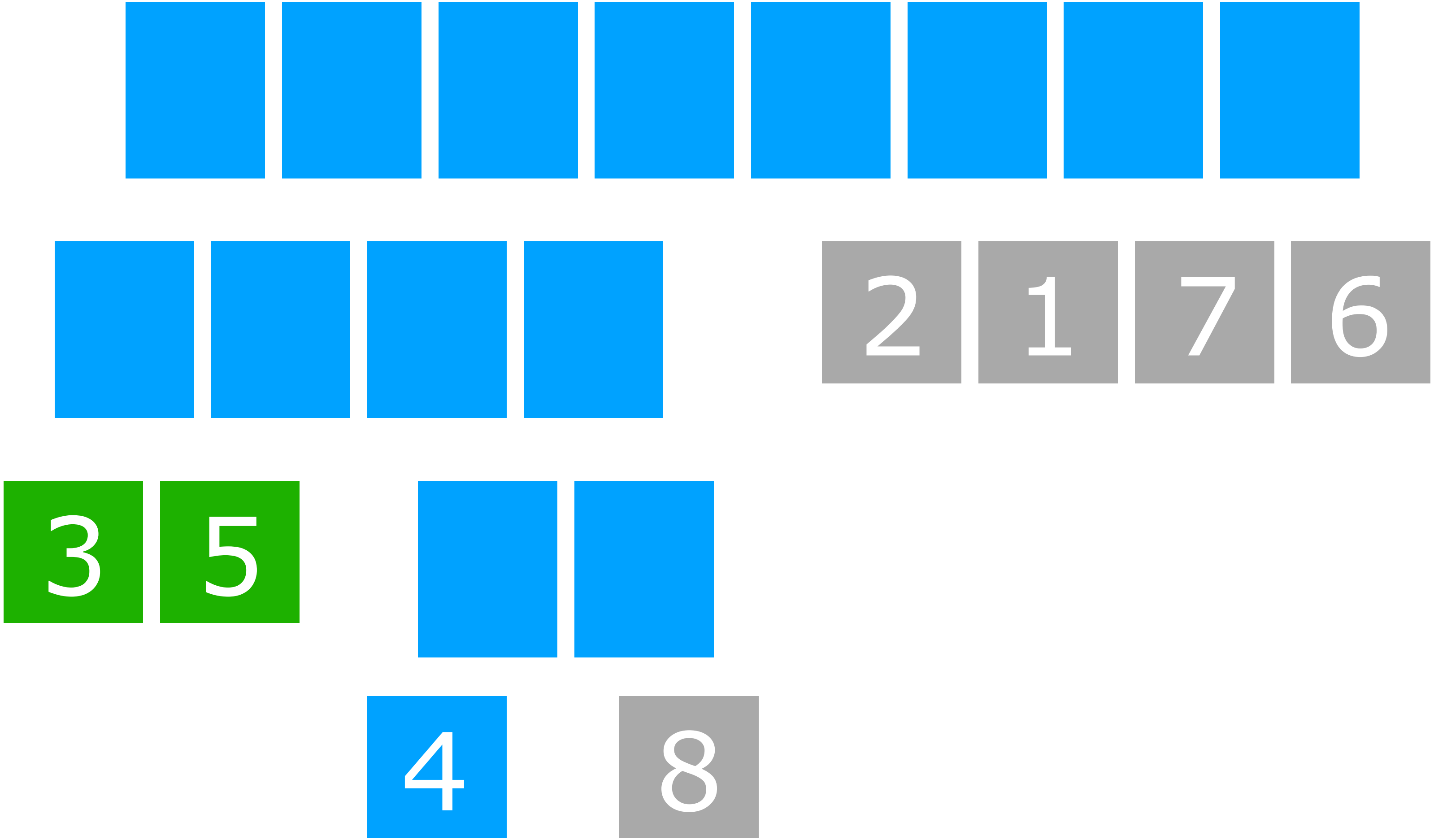

2
1
7
6
3
5
4
8

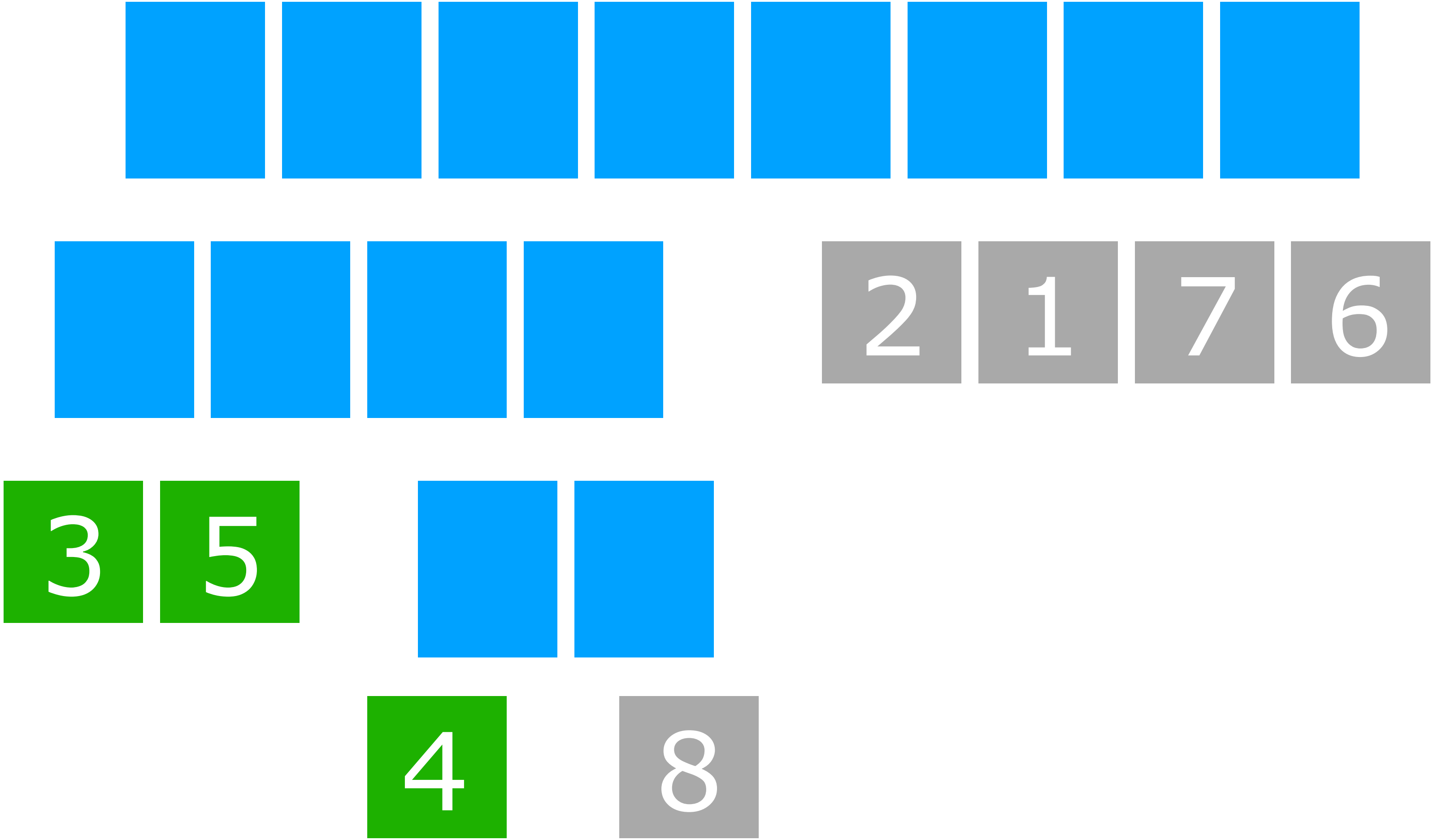


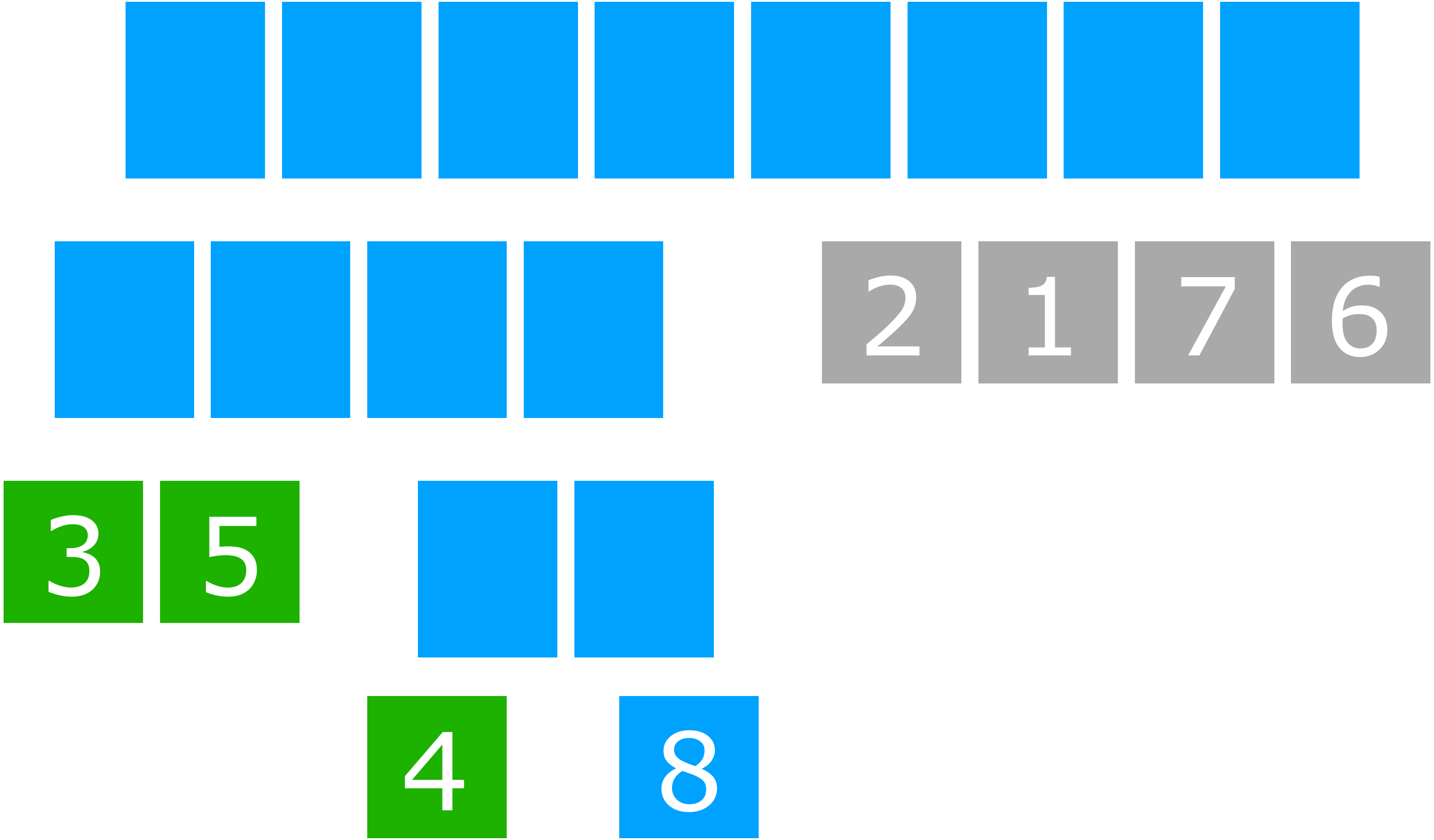
2 1 7 6
3 5
4 8


2
1
7
6
3
5
4
8


2
1
7
6
3
5
4
8


2
1
7
6
3
5
4
8


2
1
7
6
3
5
4
8

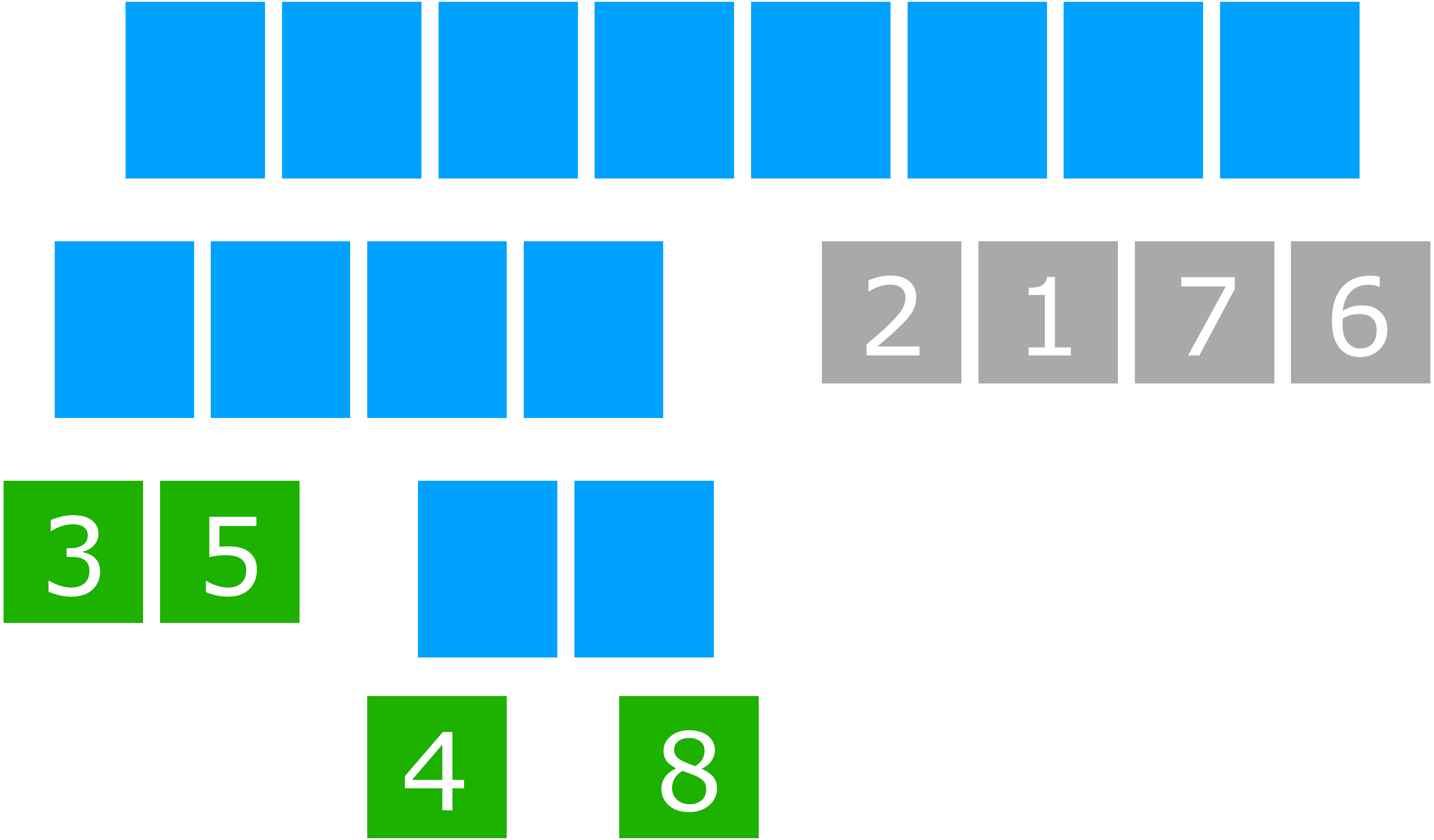


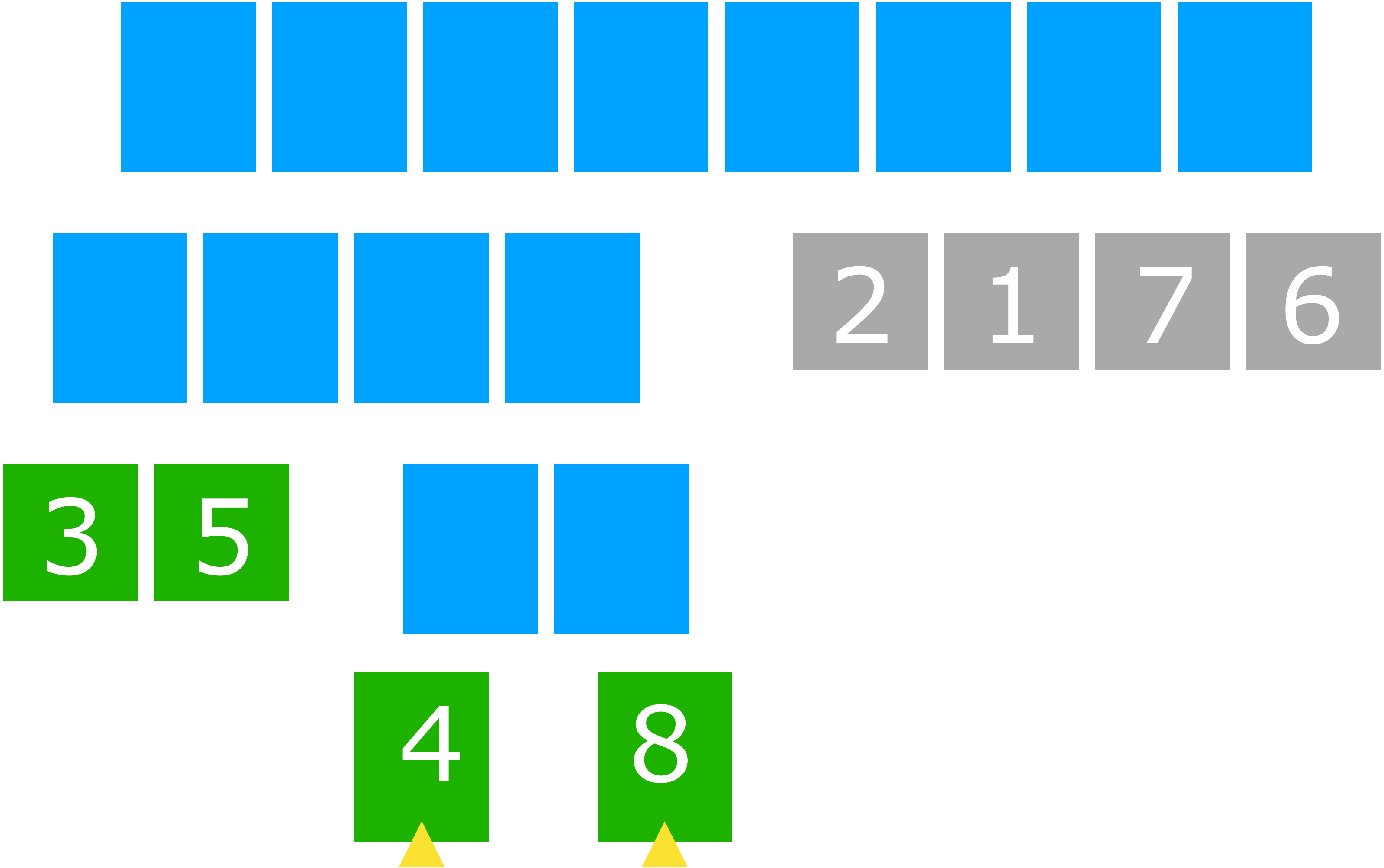
2 1 7 6

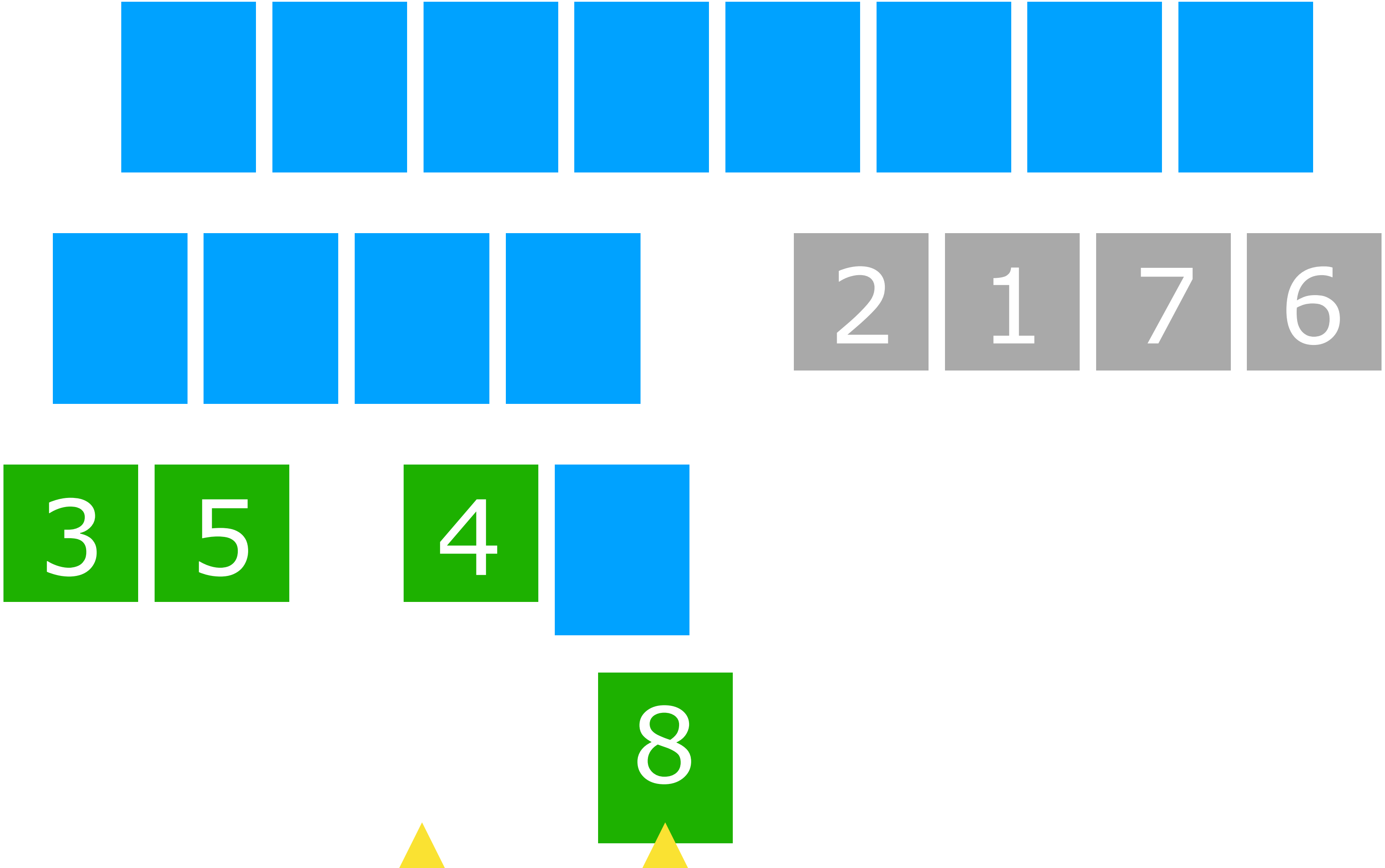
3 5

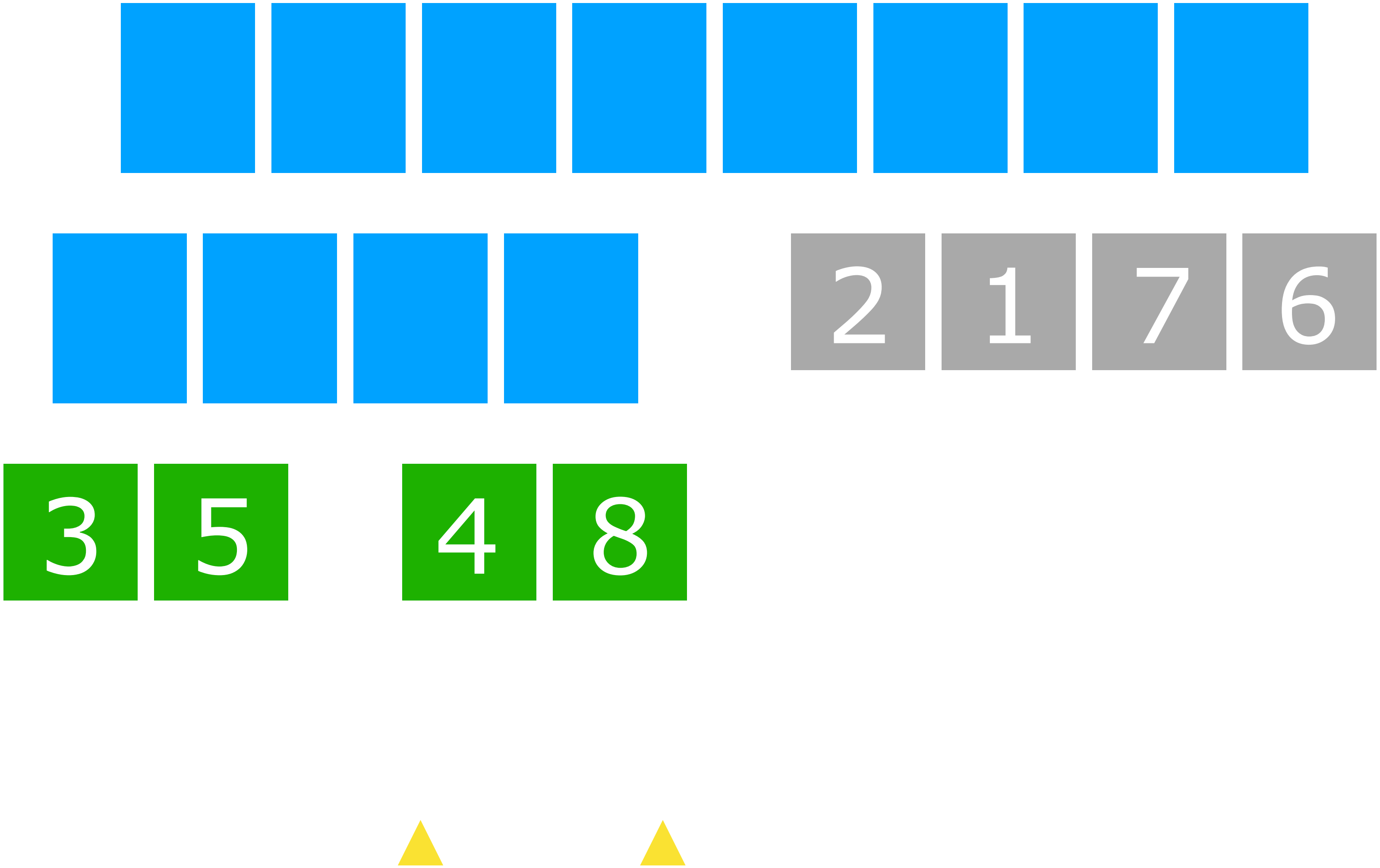
4

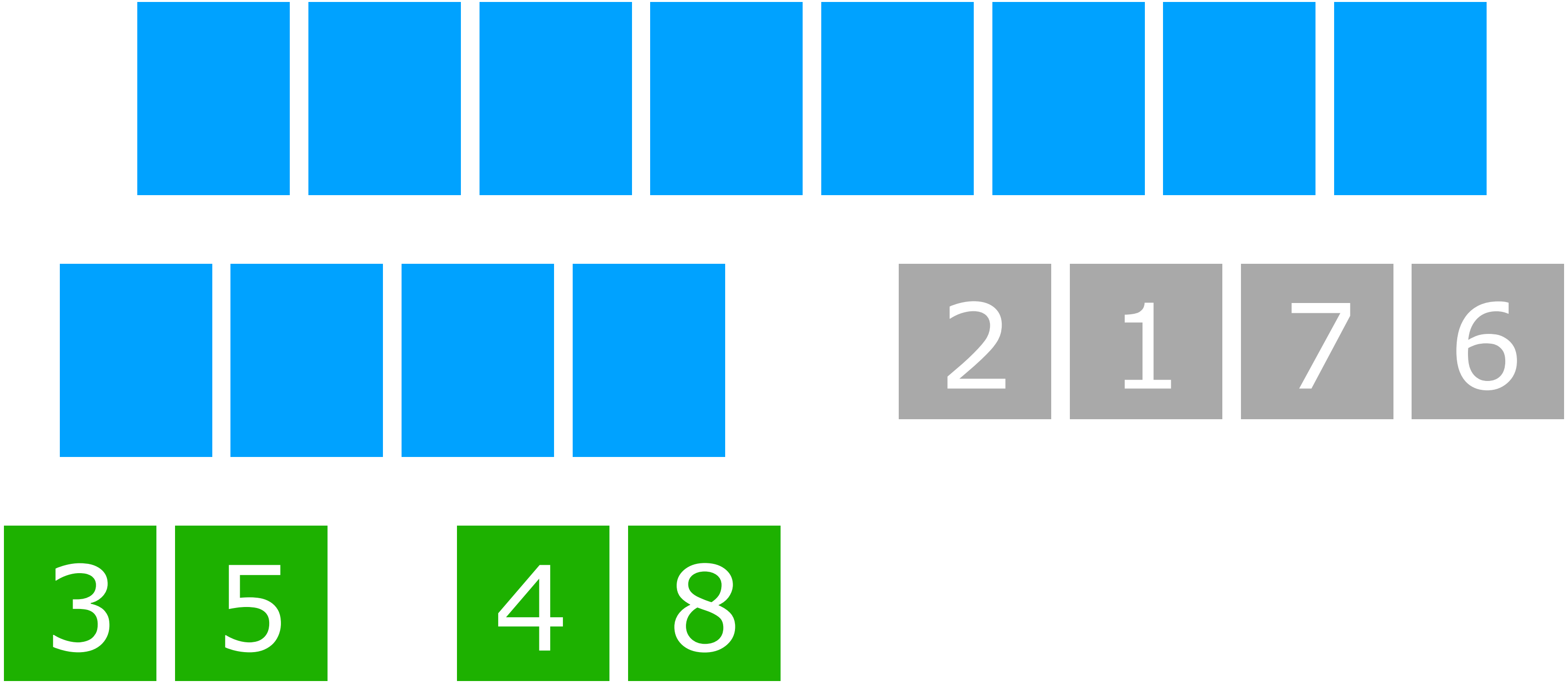
8


2
1
7
6
3
5
4
8


2
1
7
6
3
5
4
8

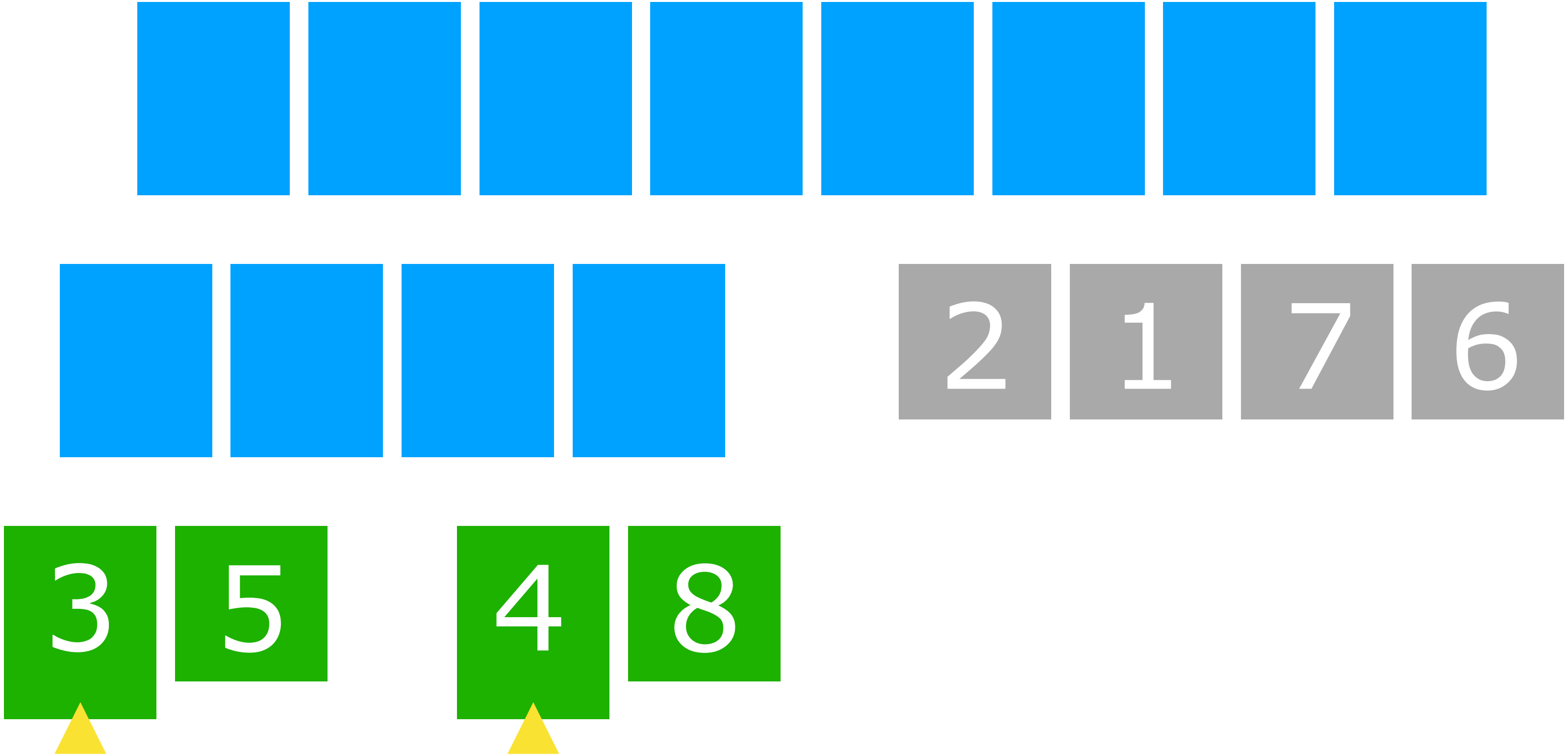

2
1
7
6
3
5
4
8

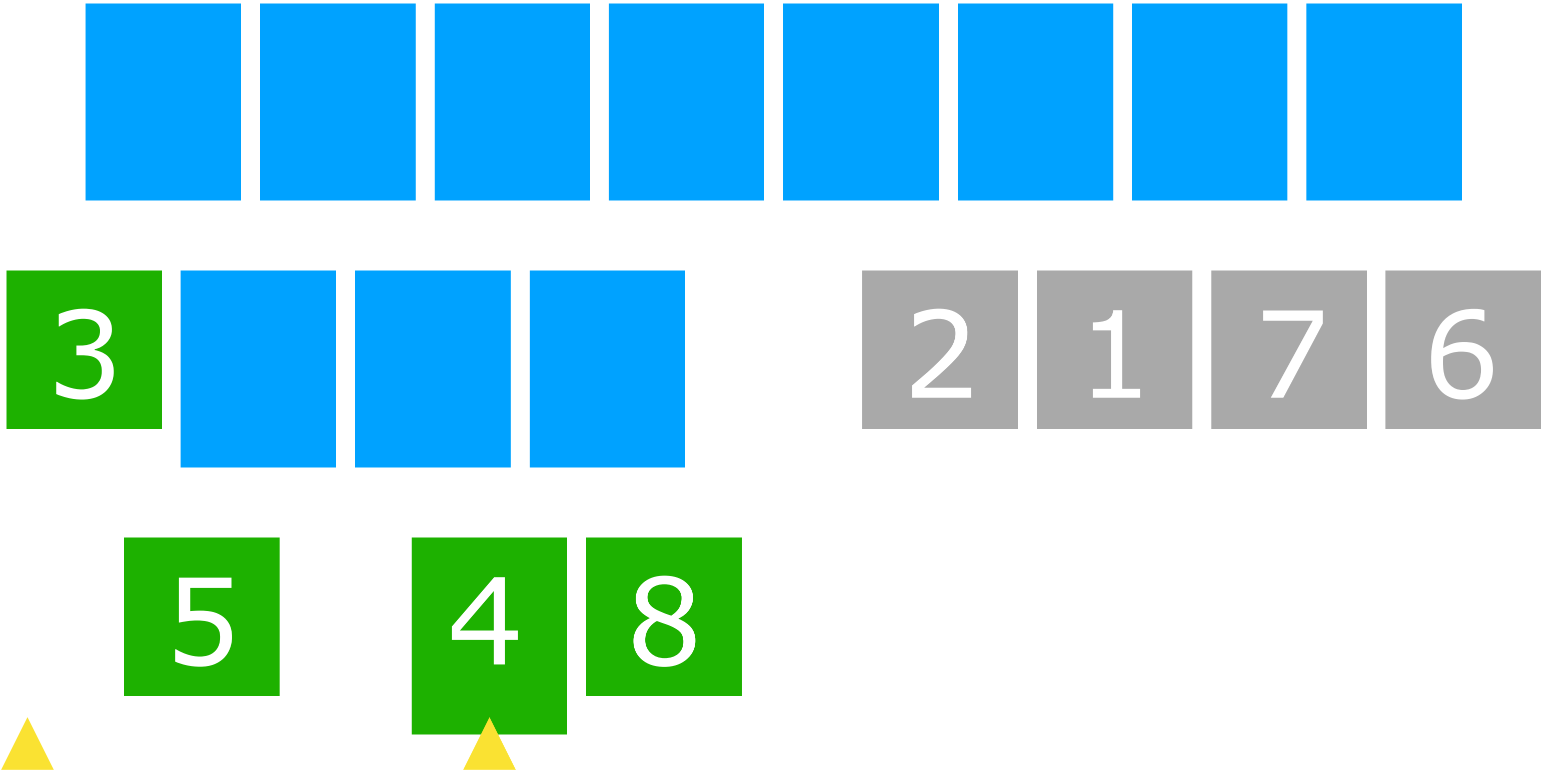

2
1
7
6
3
5
4
8

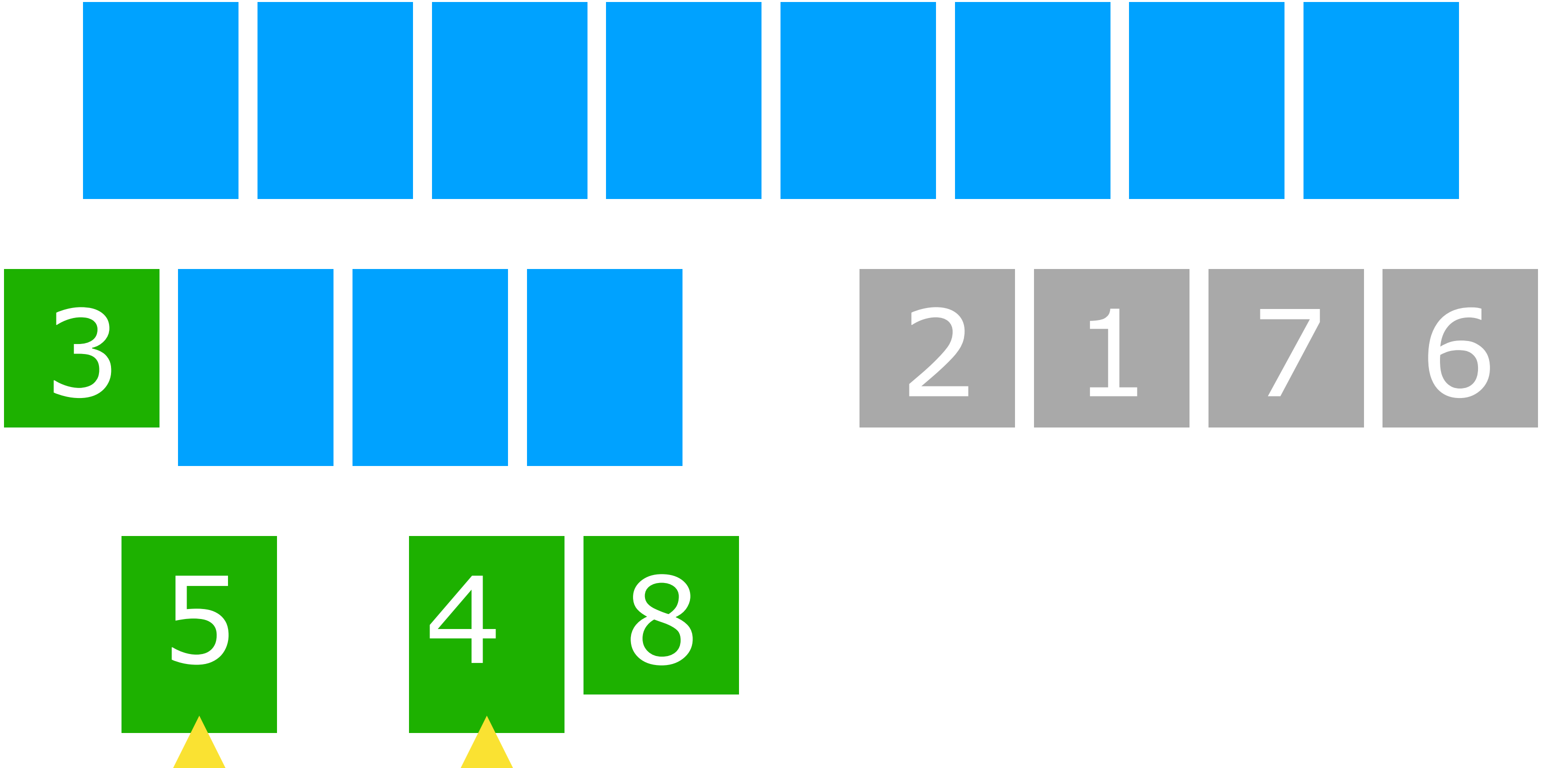


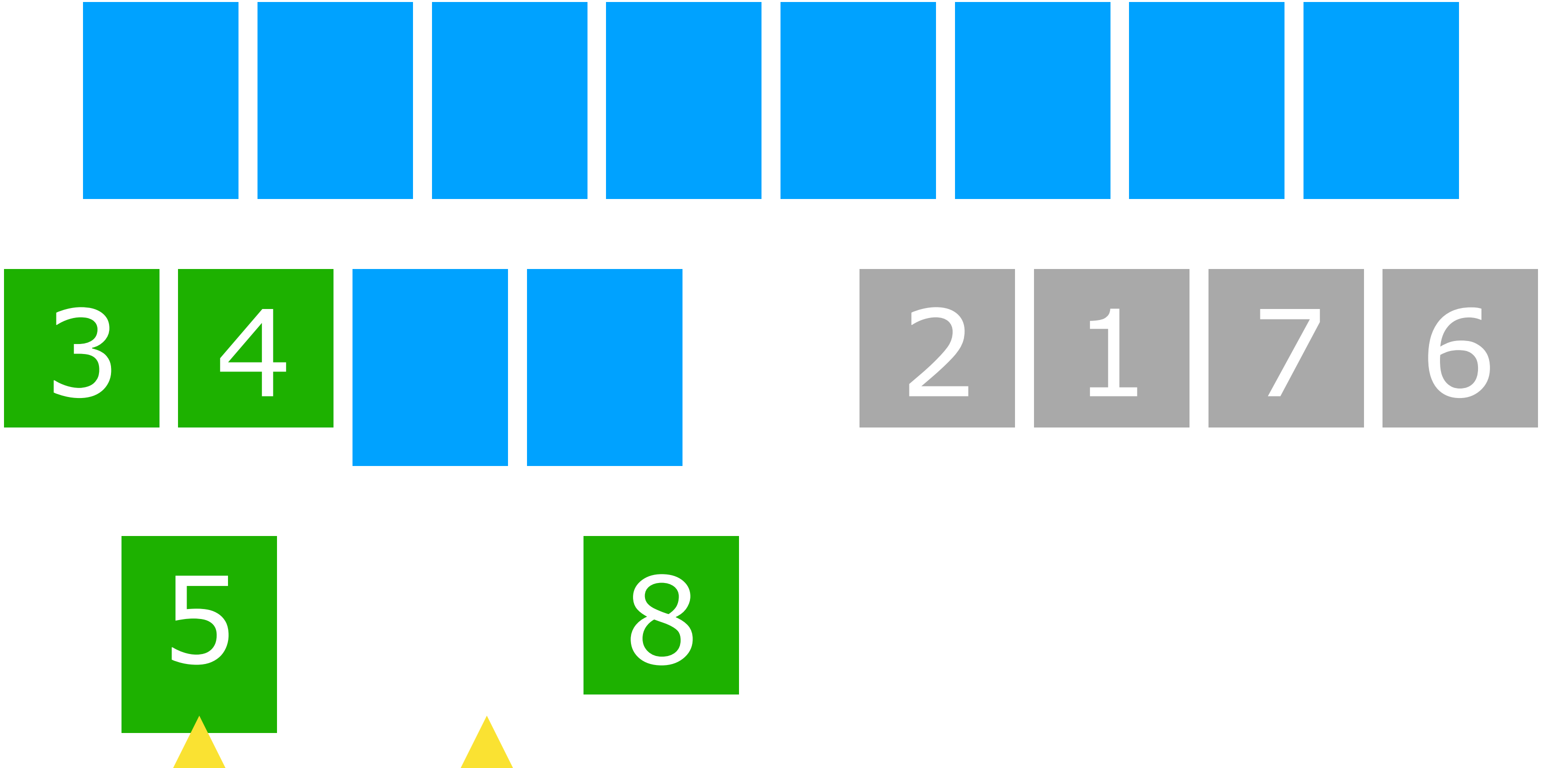
2 1 7 6

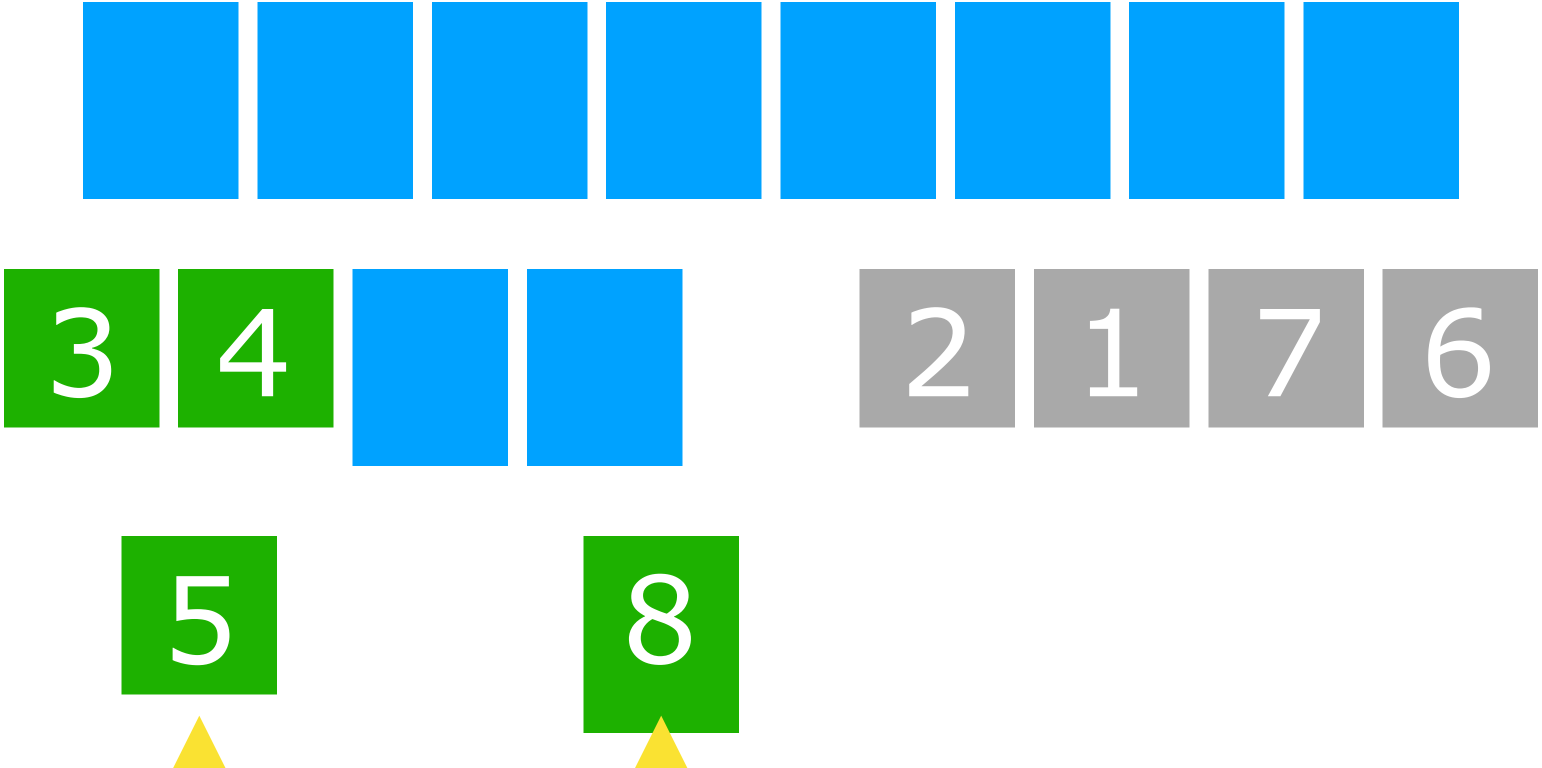
3 5 4 8

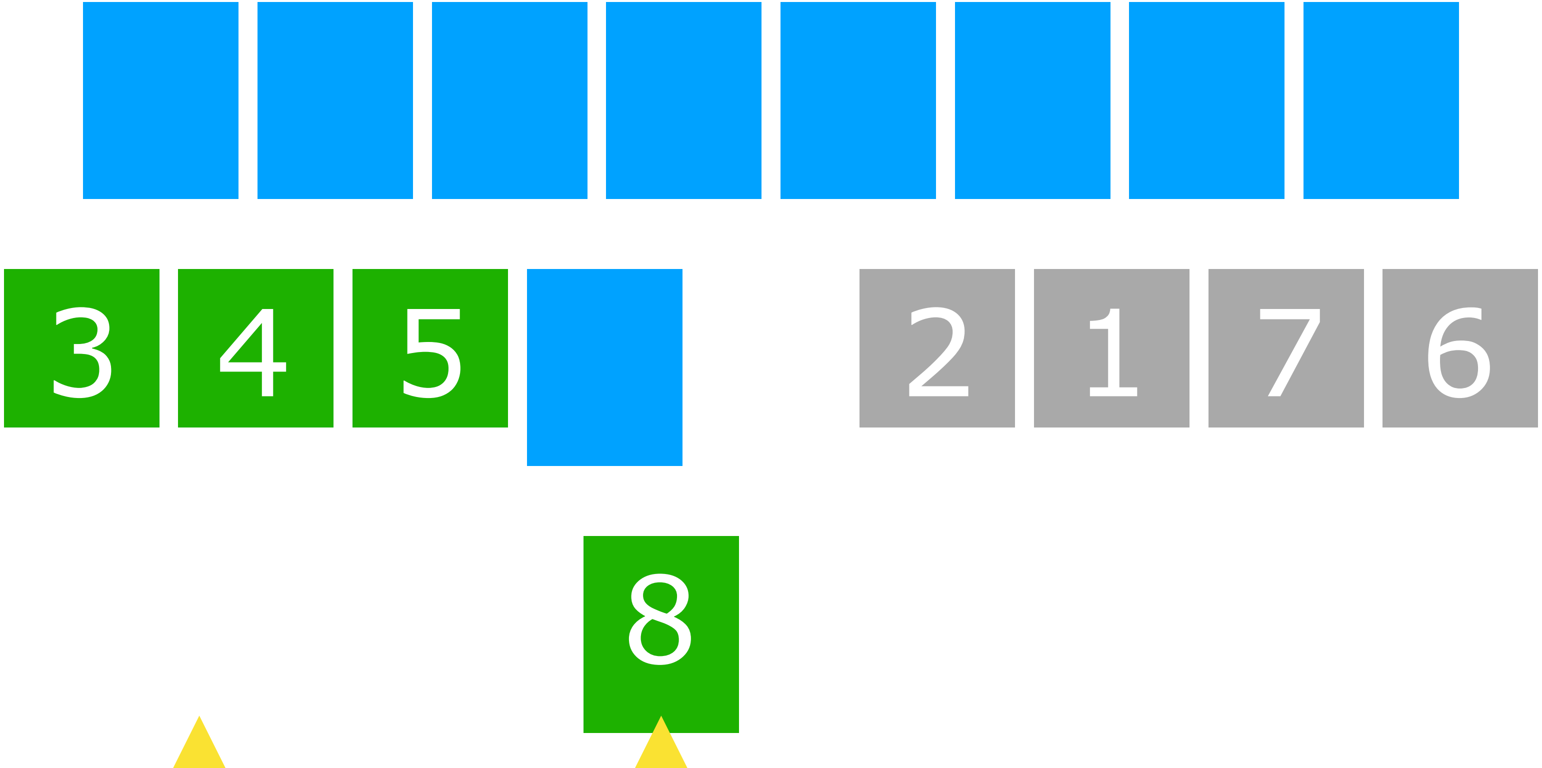

2
1
7
6
3
5
4
8


3
2
1
7
6
5
4
8

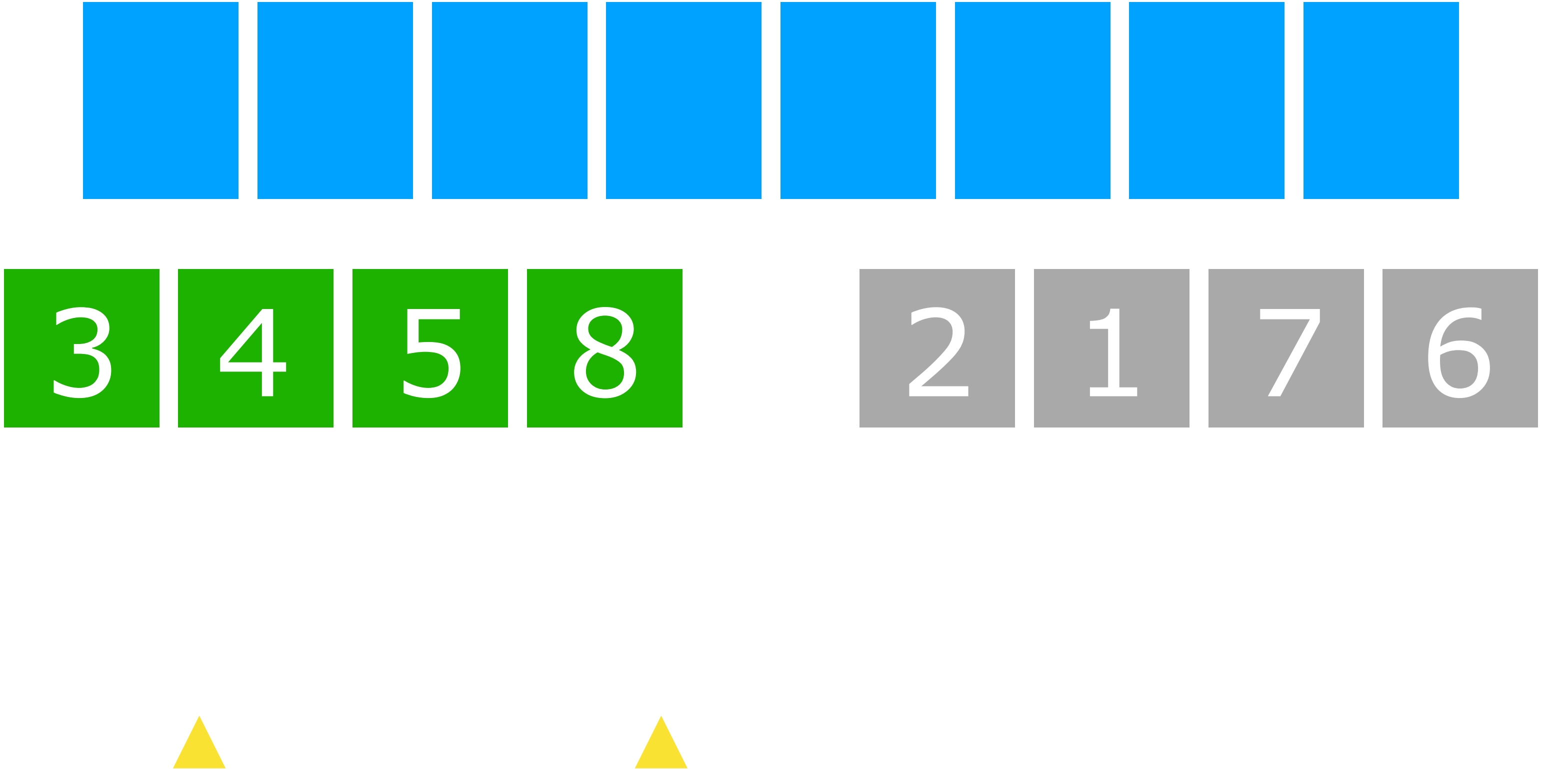

3
2
1
7
6
5
4
8


3
4
2
1
7
6
5
8


3
4
2
1
7
6
5
8

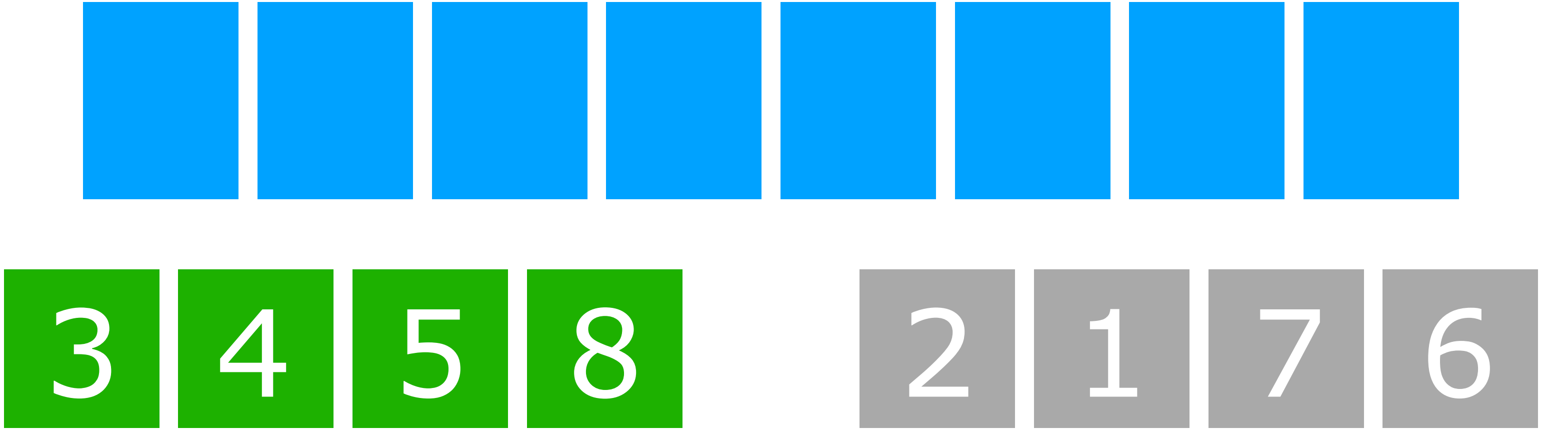


3
4
5
2
1
7
6
8



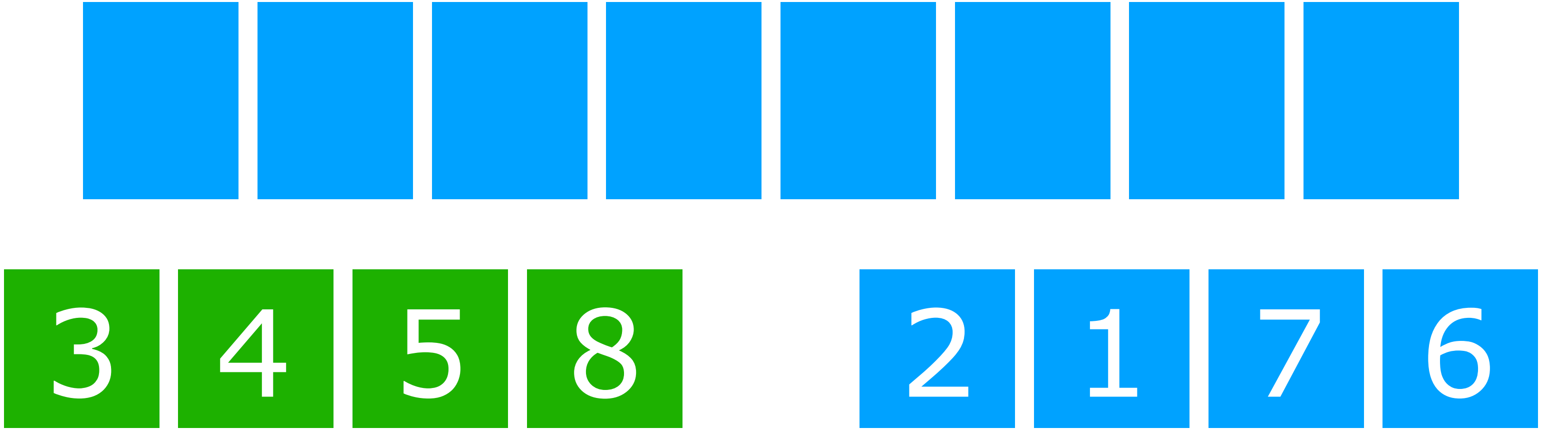
3 4 5 8

2 1 7 6



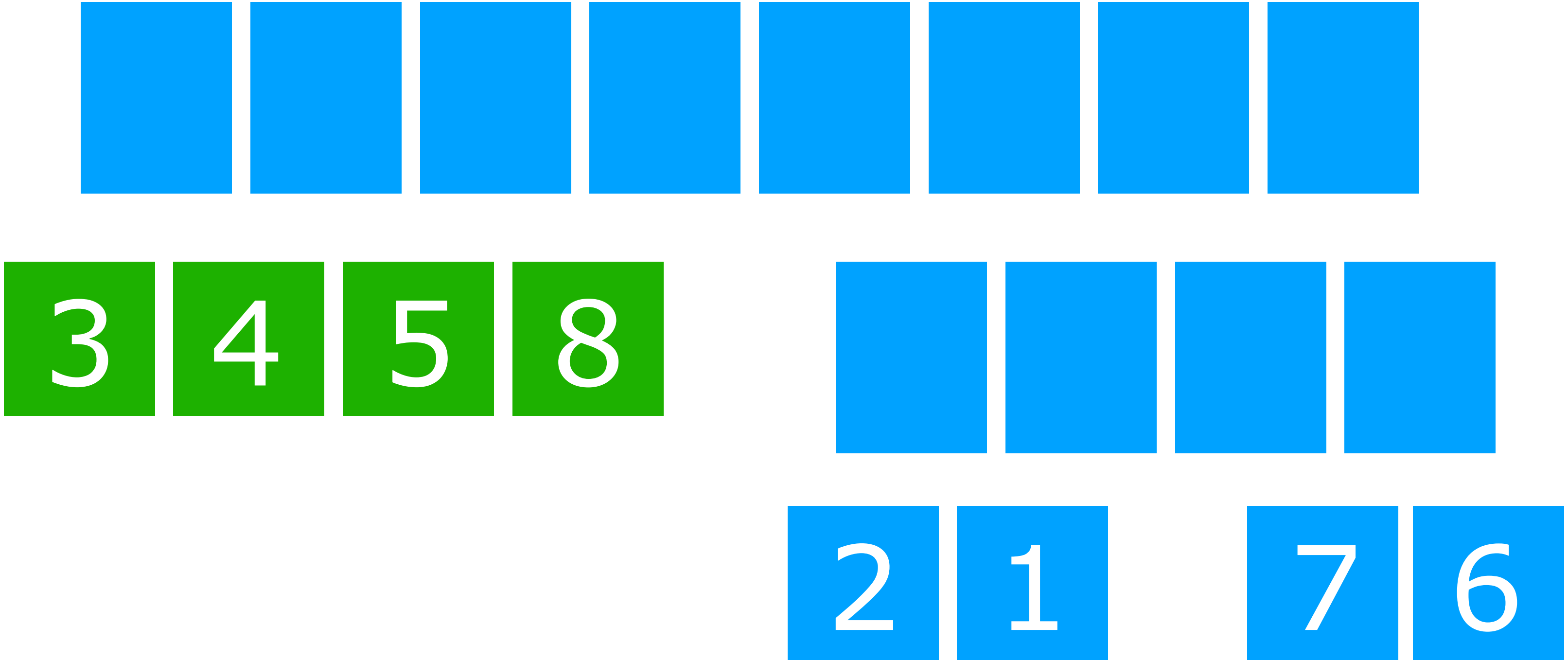
3 4 5 8

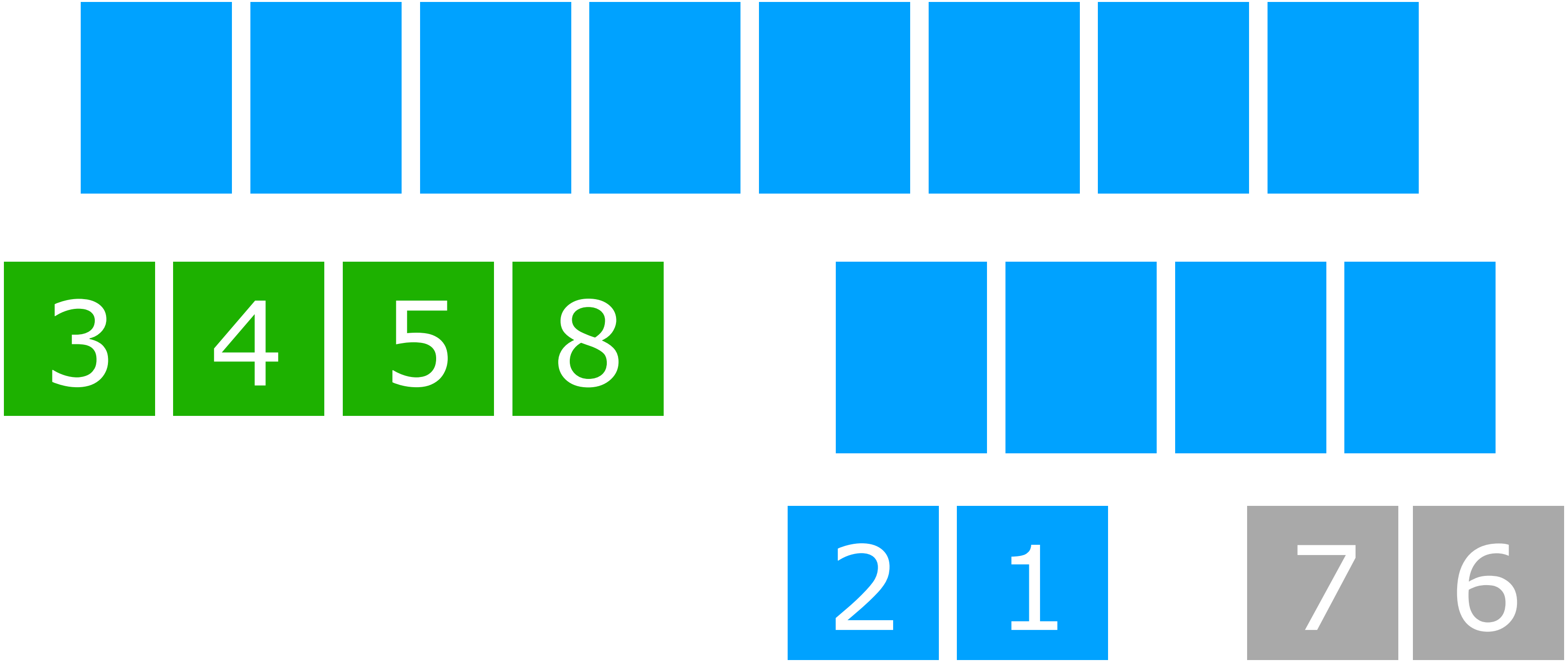
2 1 7 6

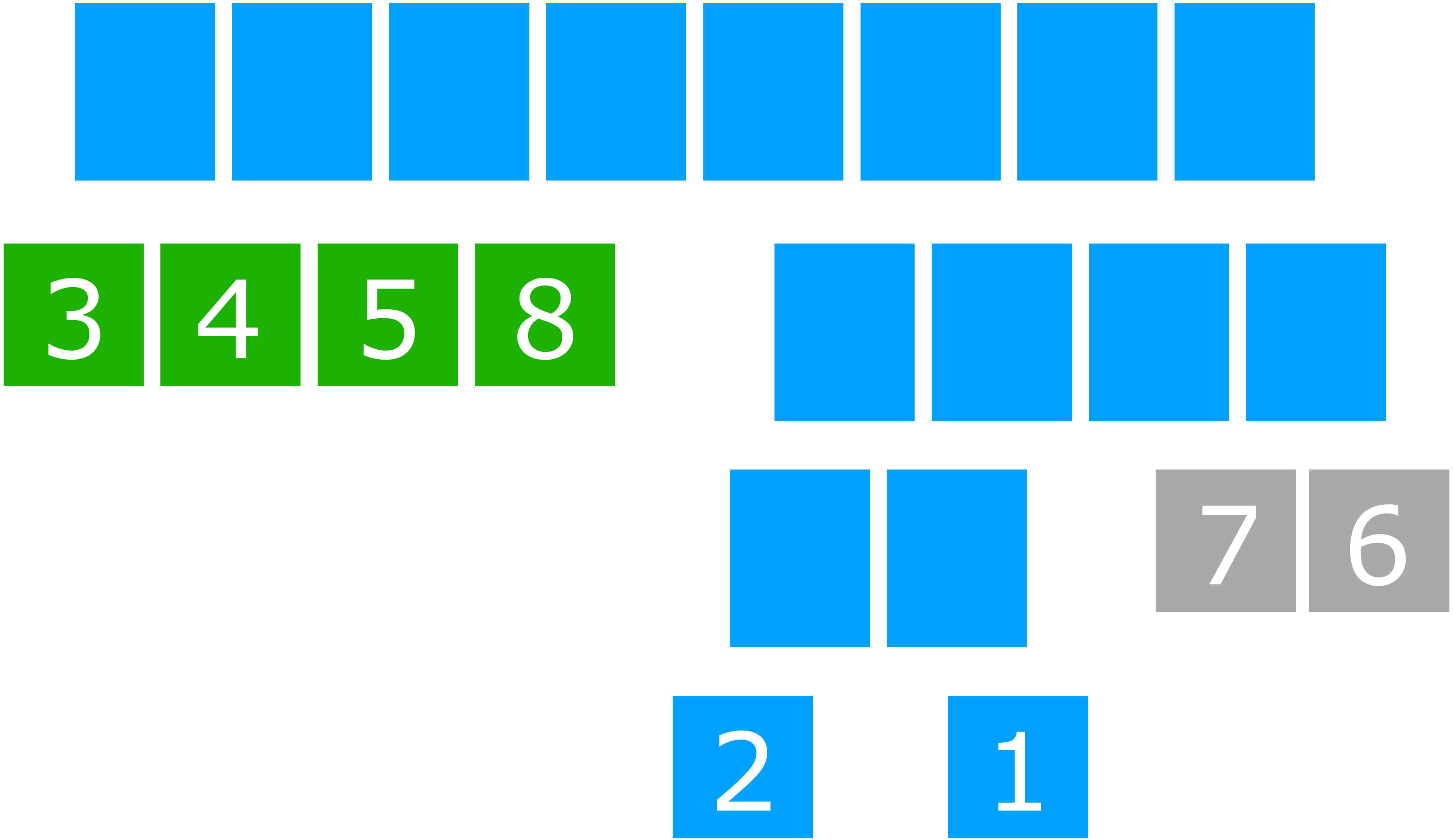


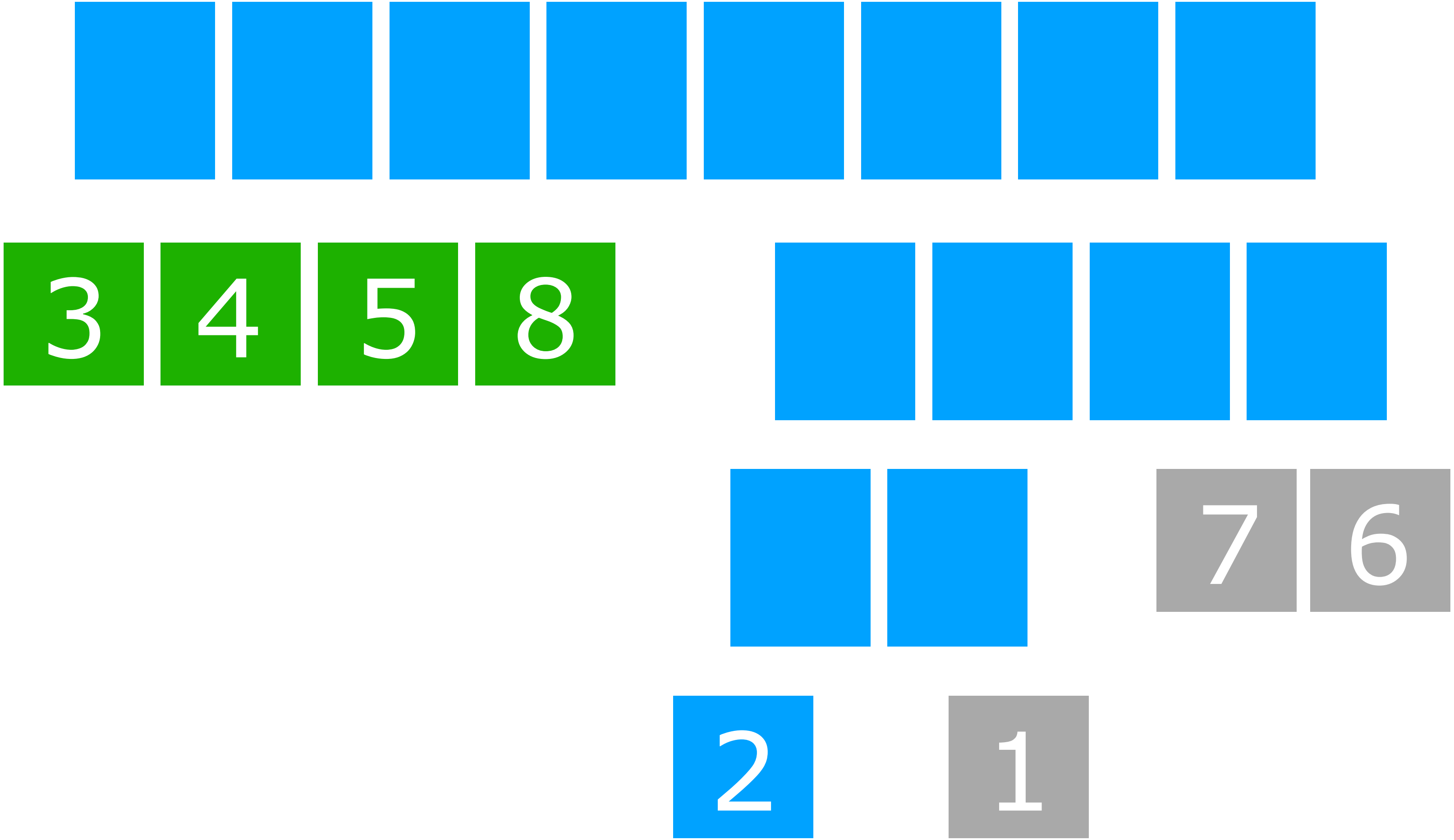
3 4 5 8

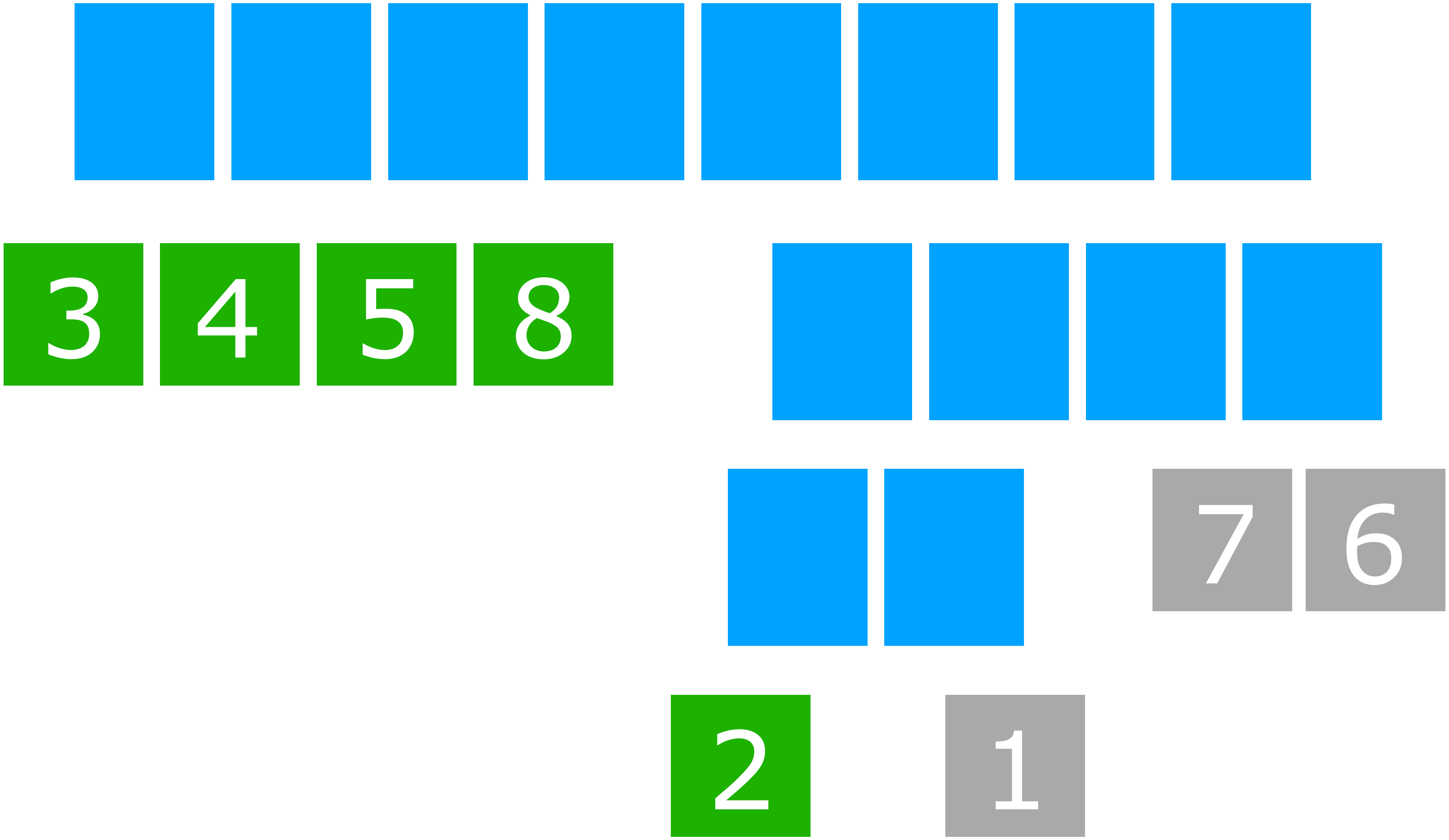
2 1 7 6

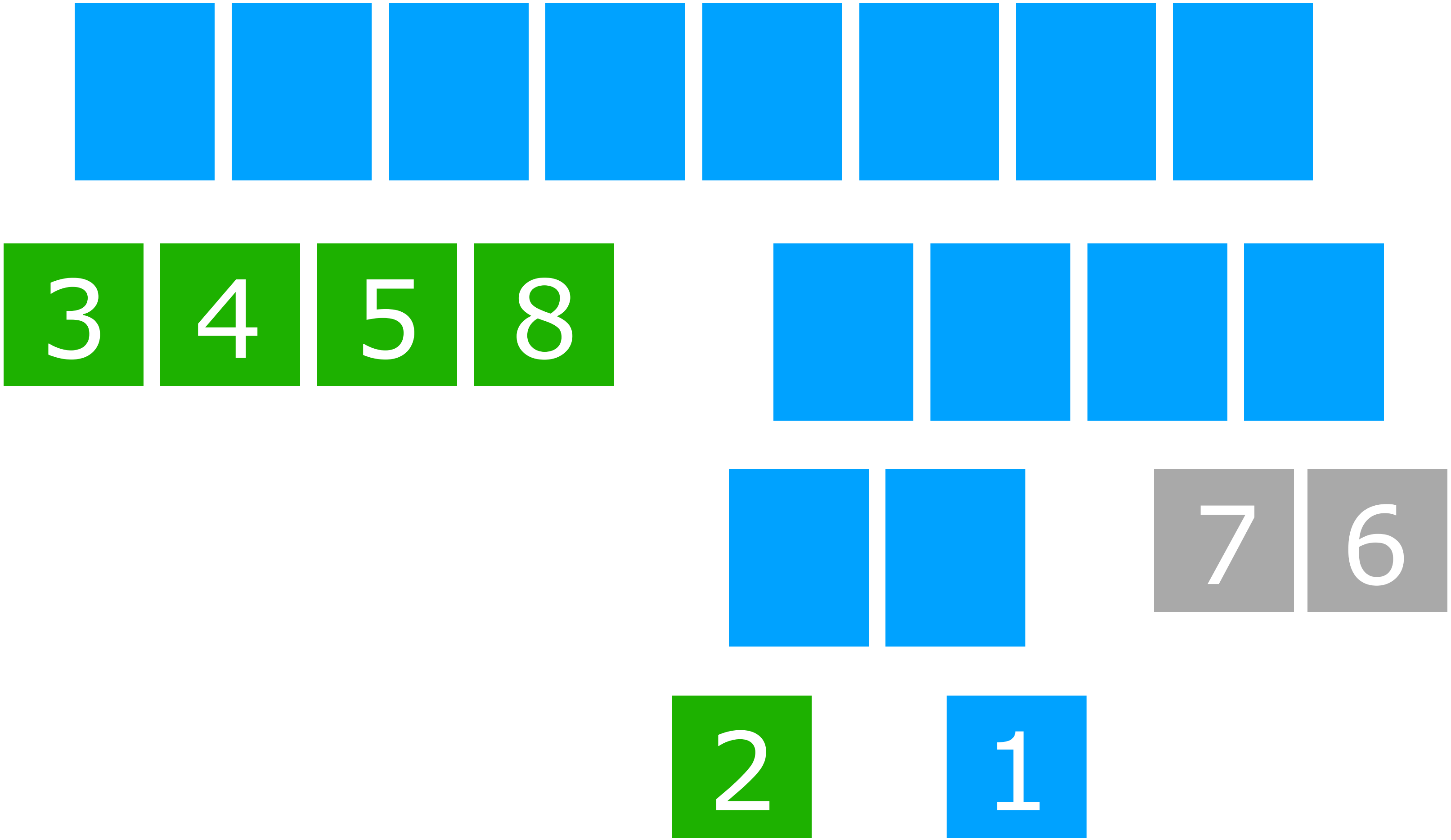

3
4
5
8
2
1
7
6

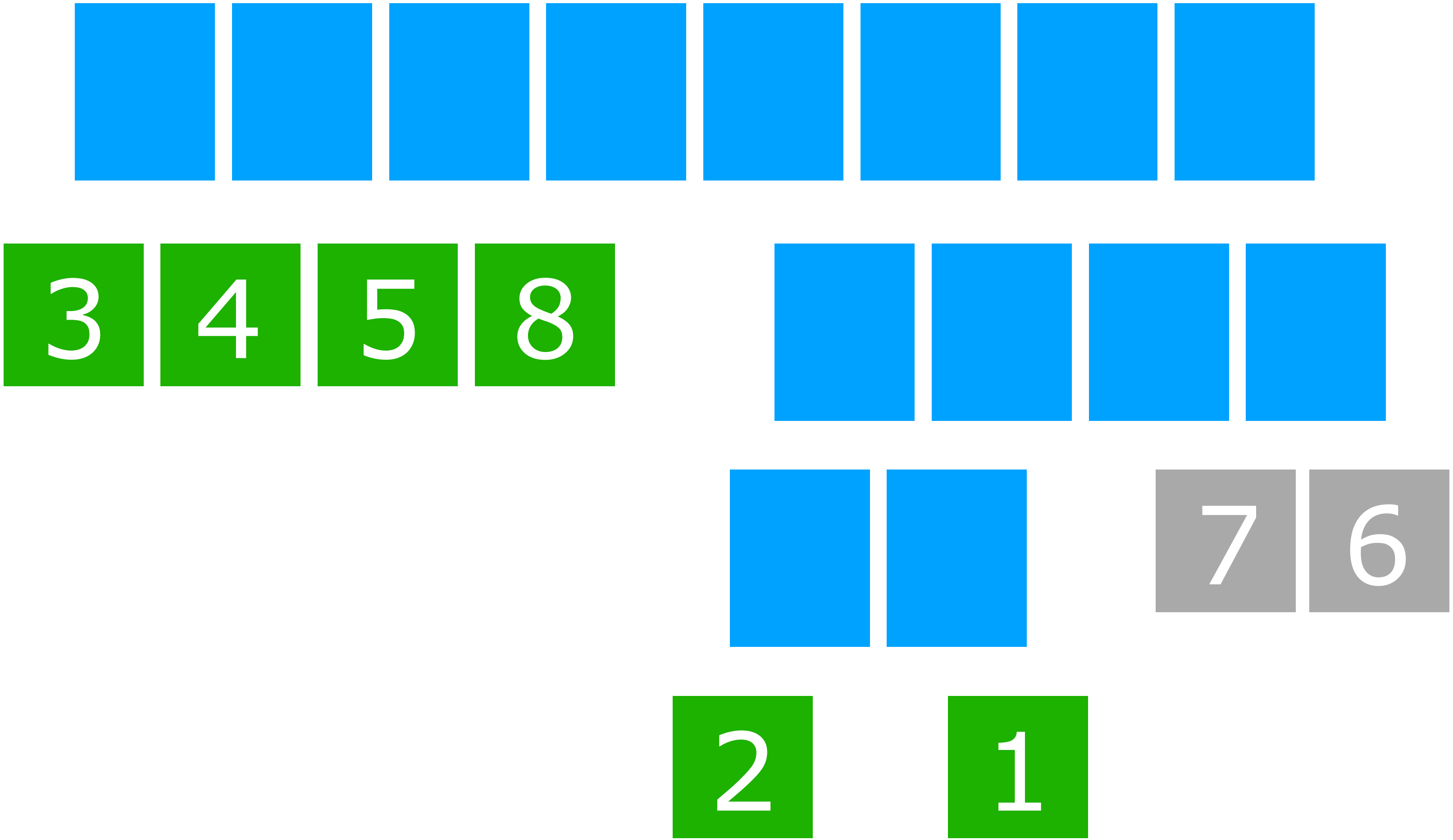

3
4
5
8
2
1
7
6

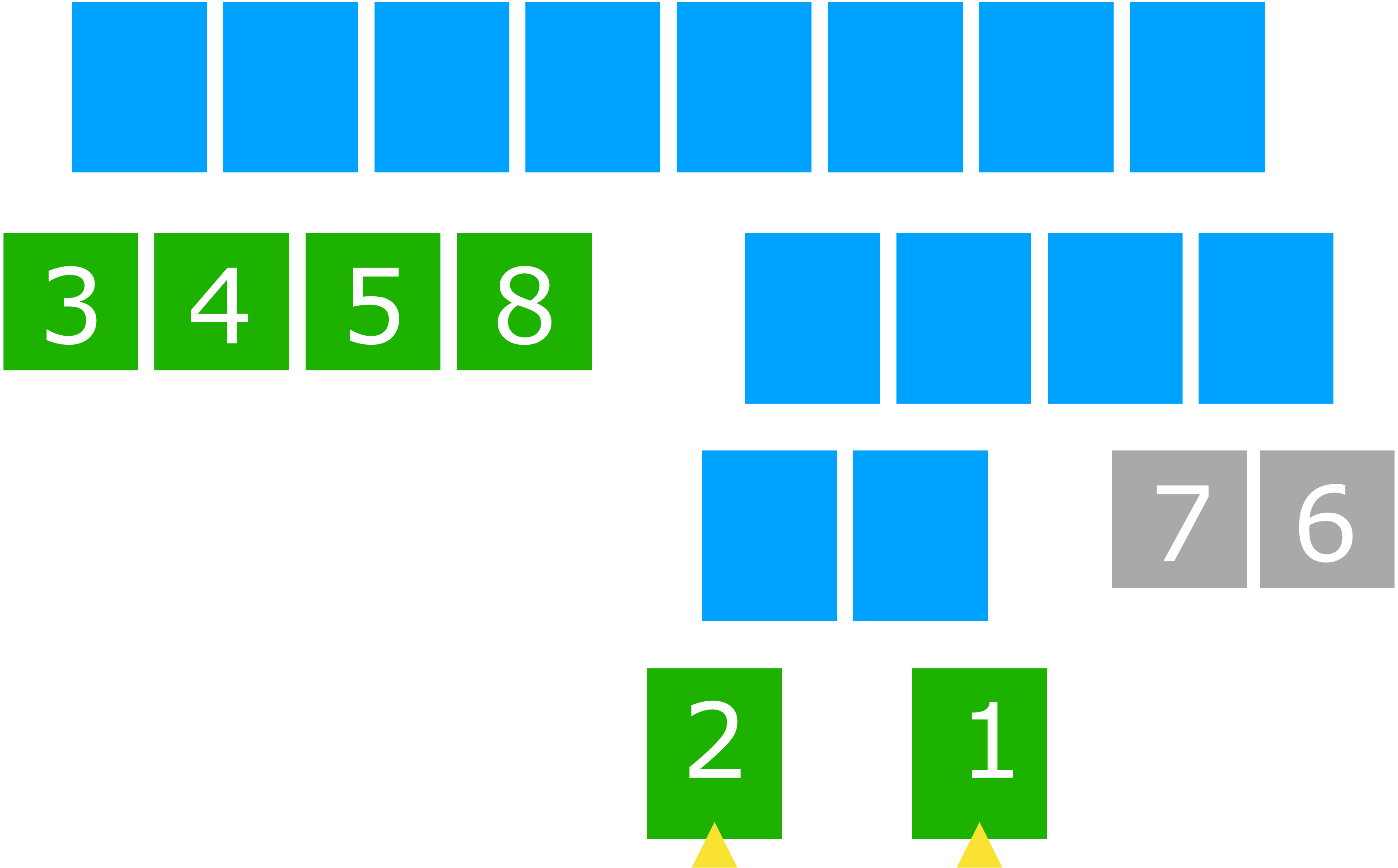

3
4
5
8
7
6
2
1

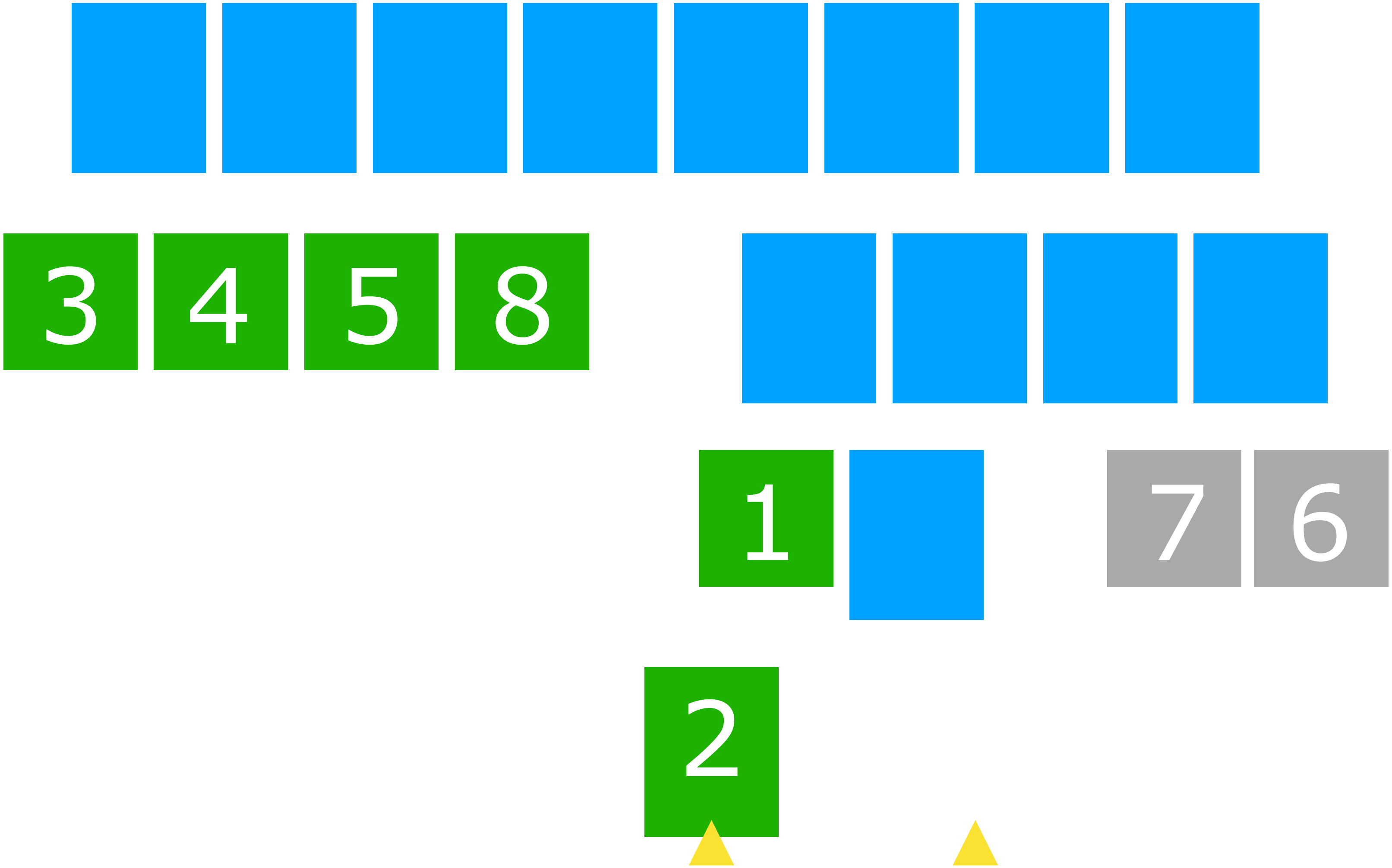

3
4
5
8
7
6
2
1

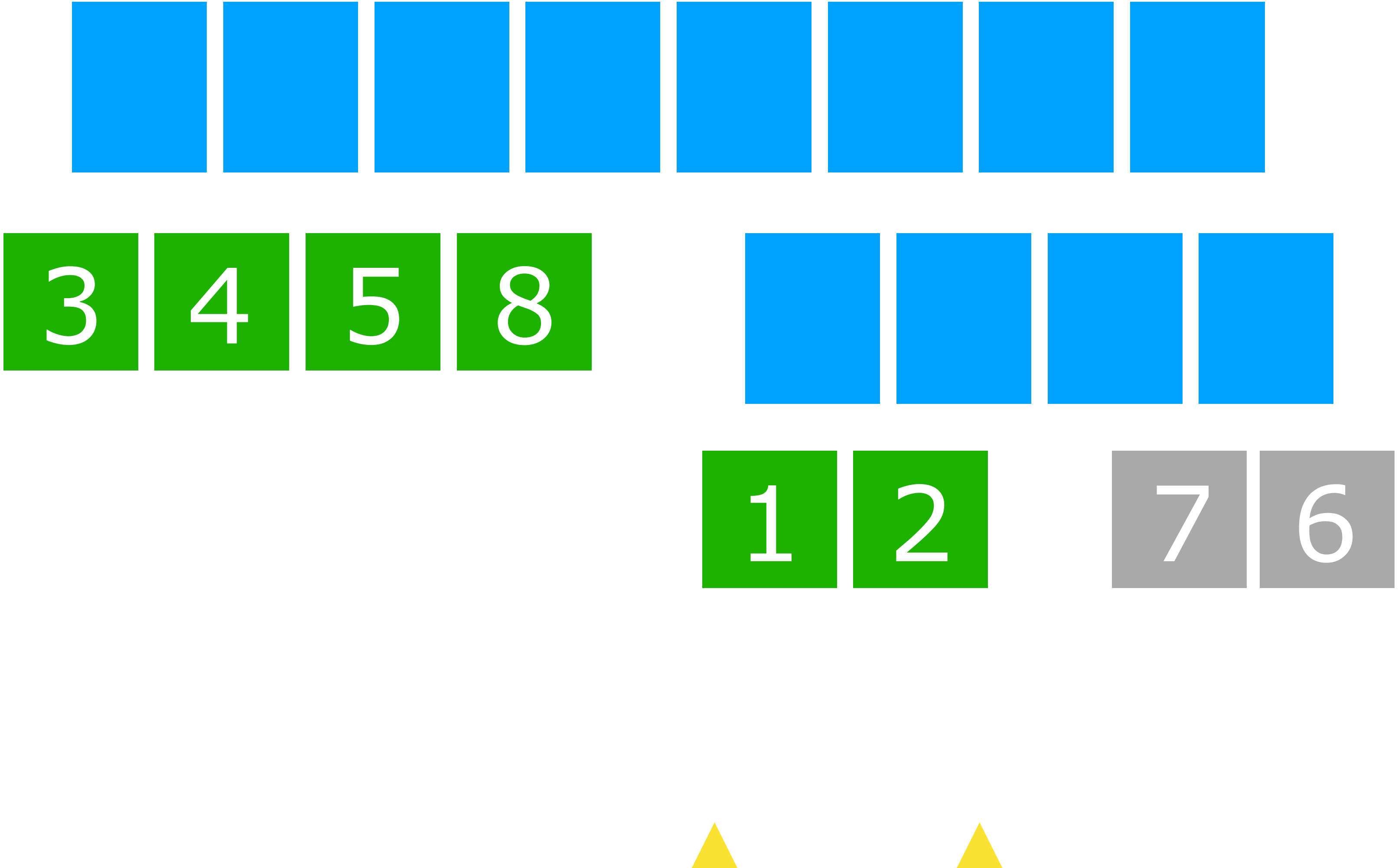

3
4
5
8
7
6
2
1

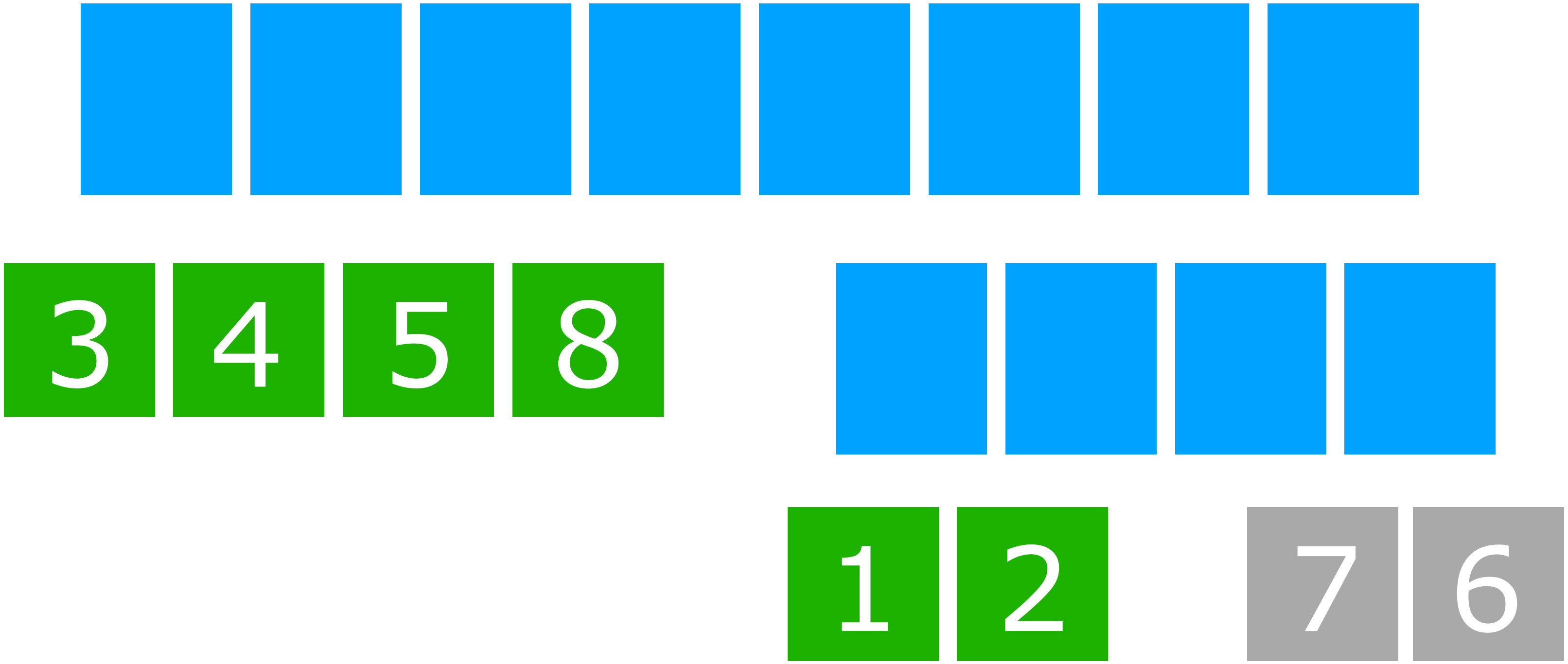

3
4
5
8
7
6
2
1

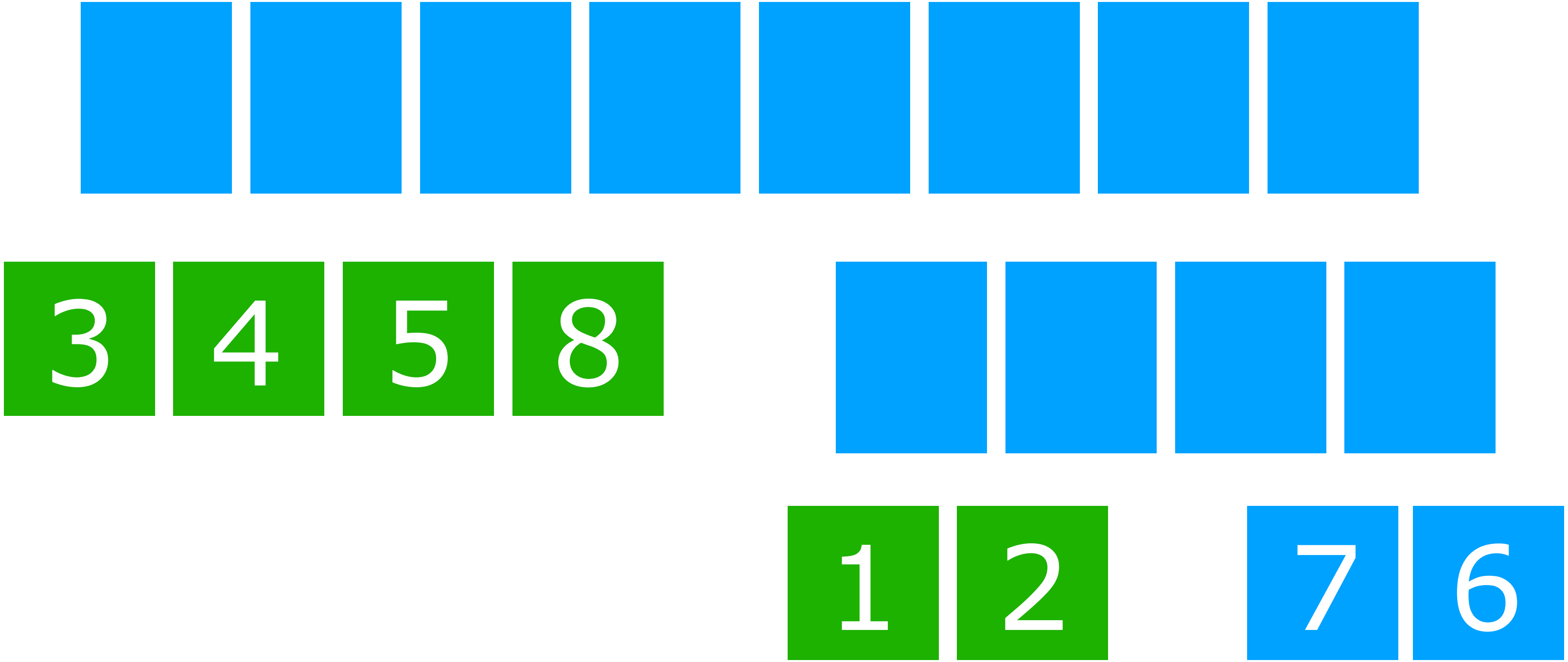

3
4
5
8
7
6
2
1

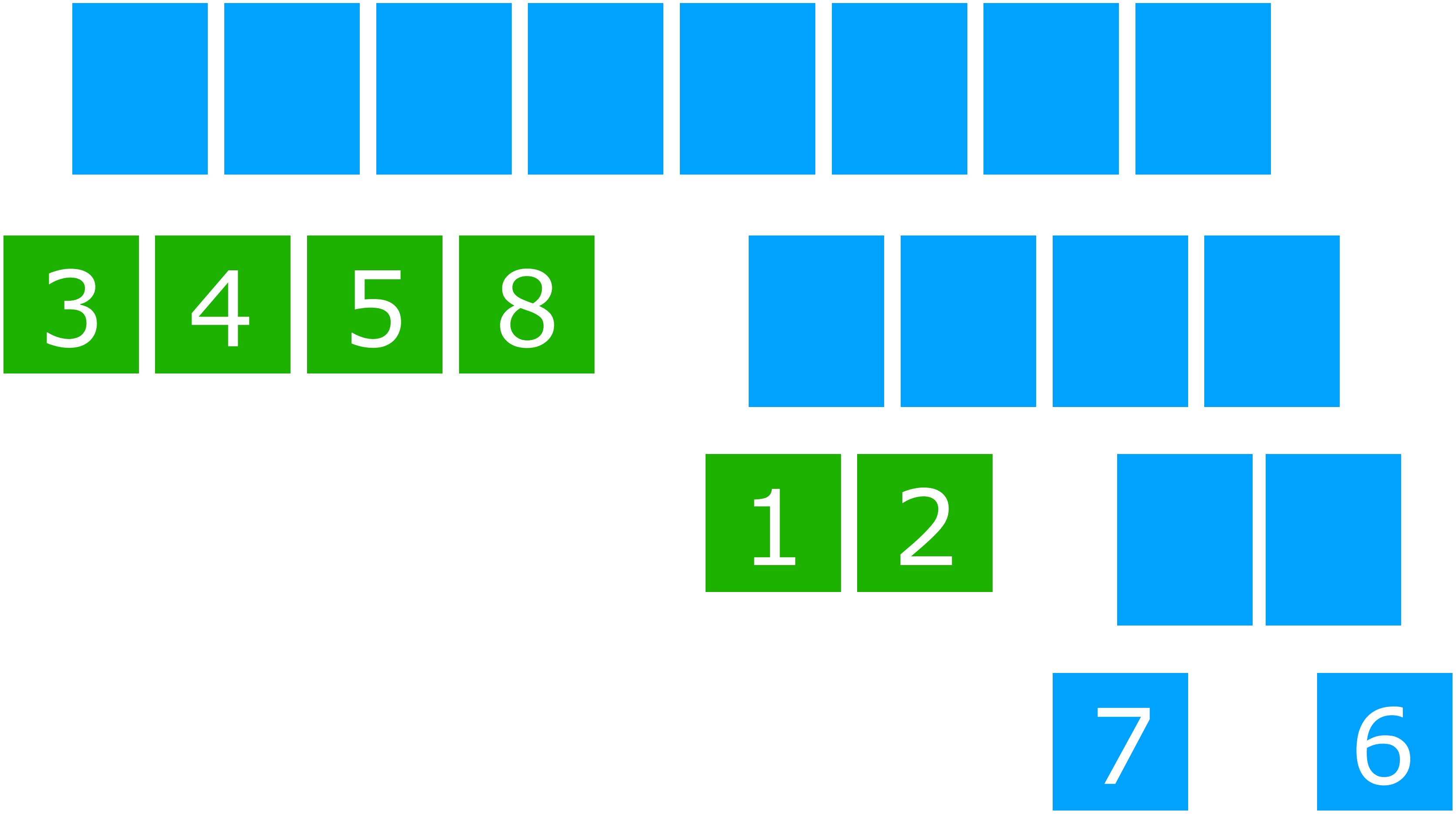

3
4
5
8
7
6
2
1

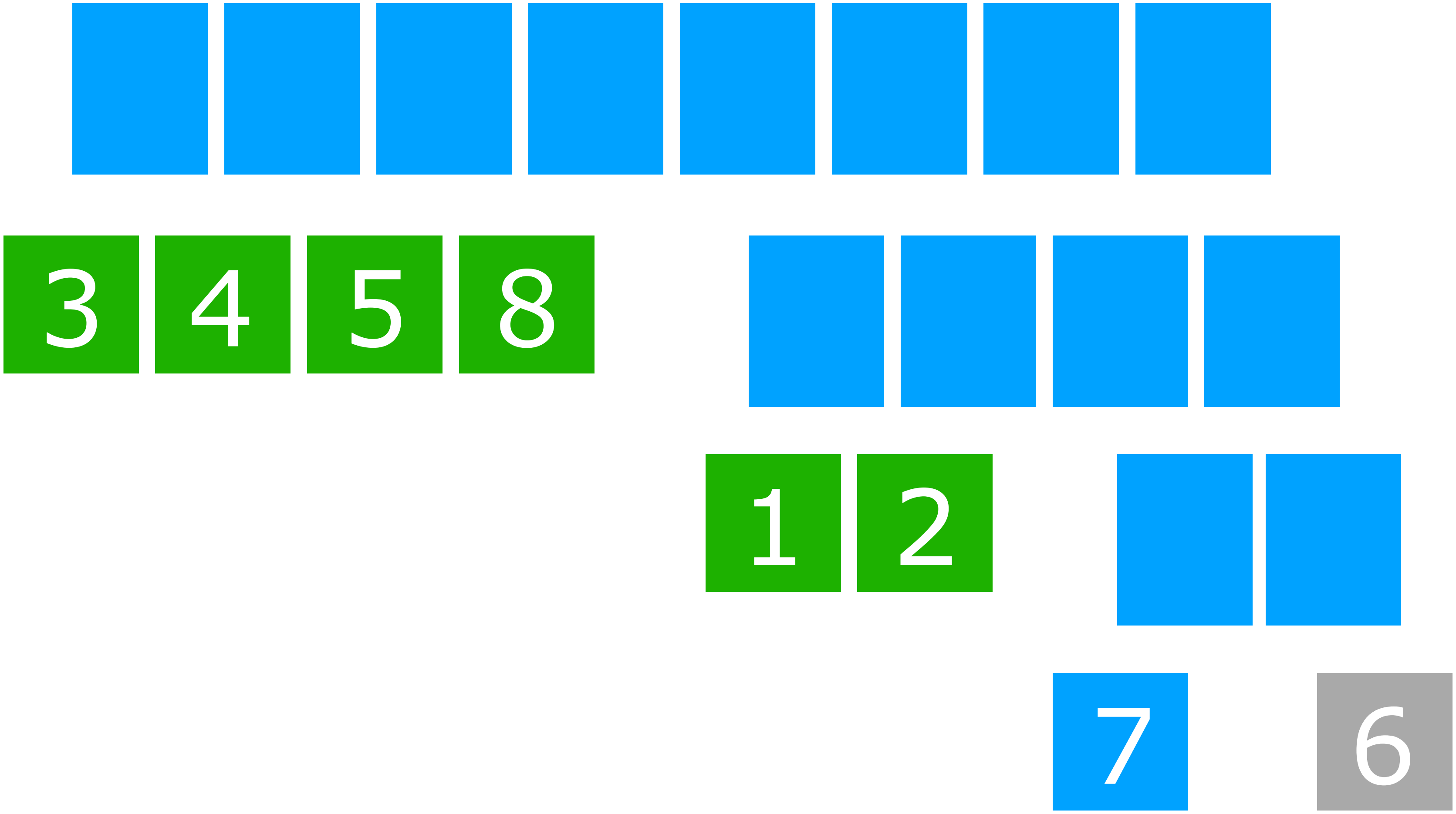

3
4
5
8
1
7
6
2

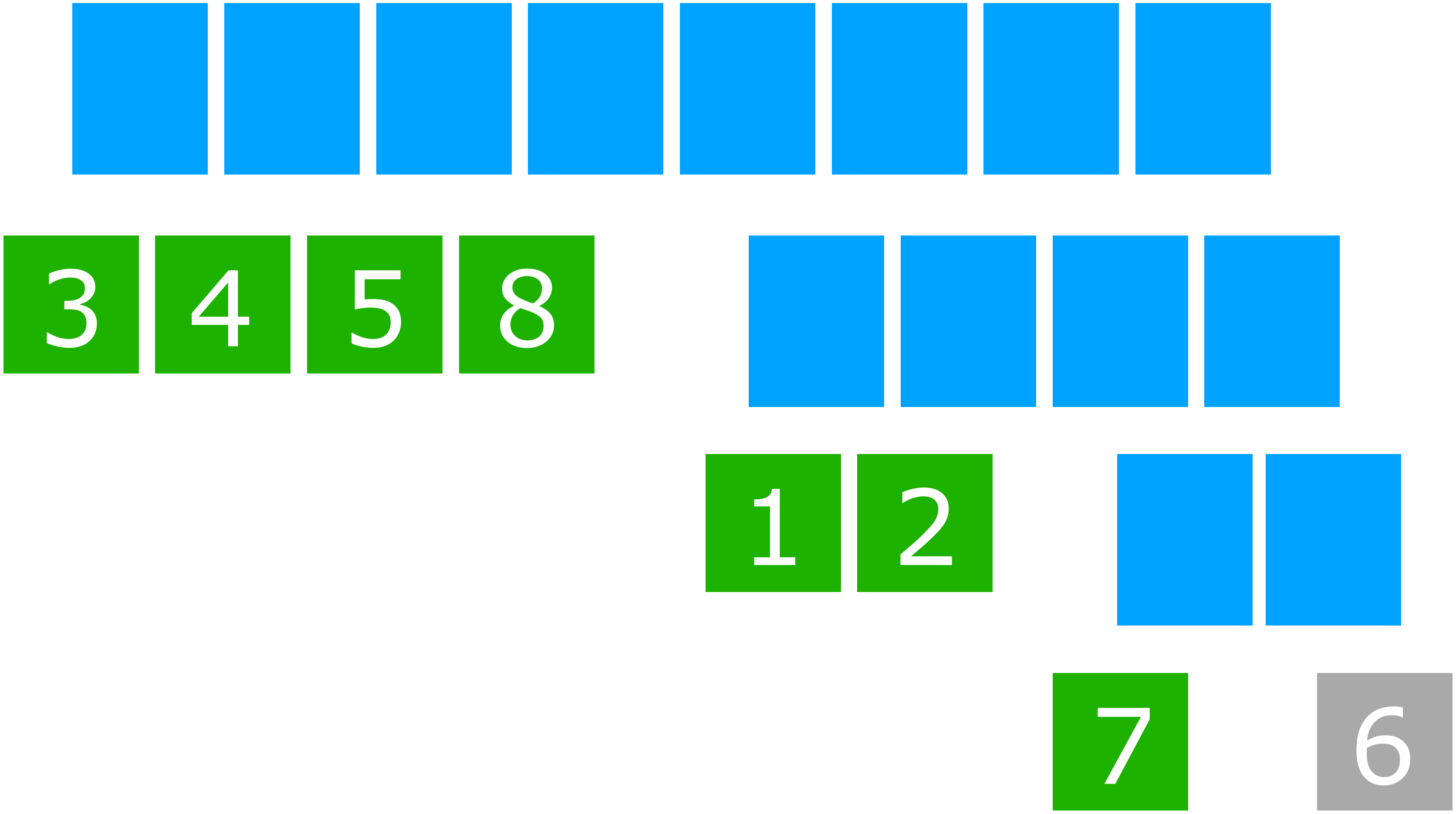

3 4 5 8
1 2
7 6

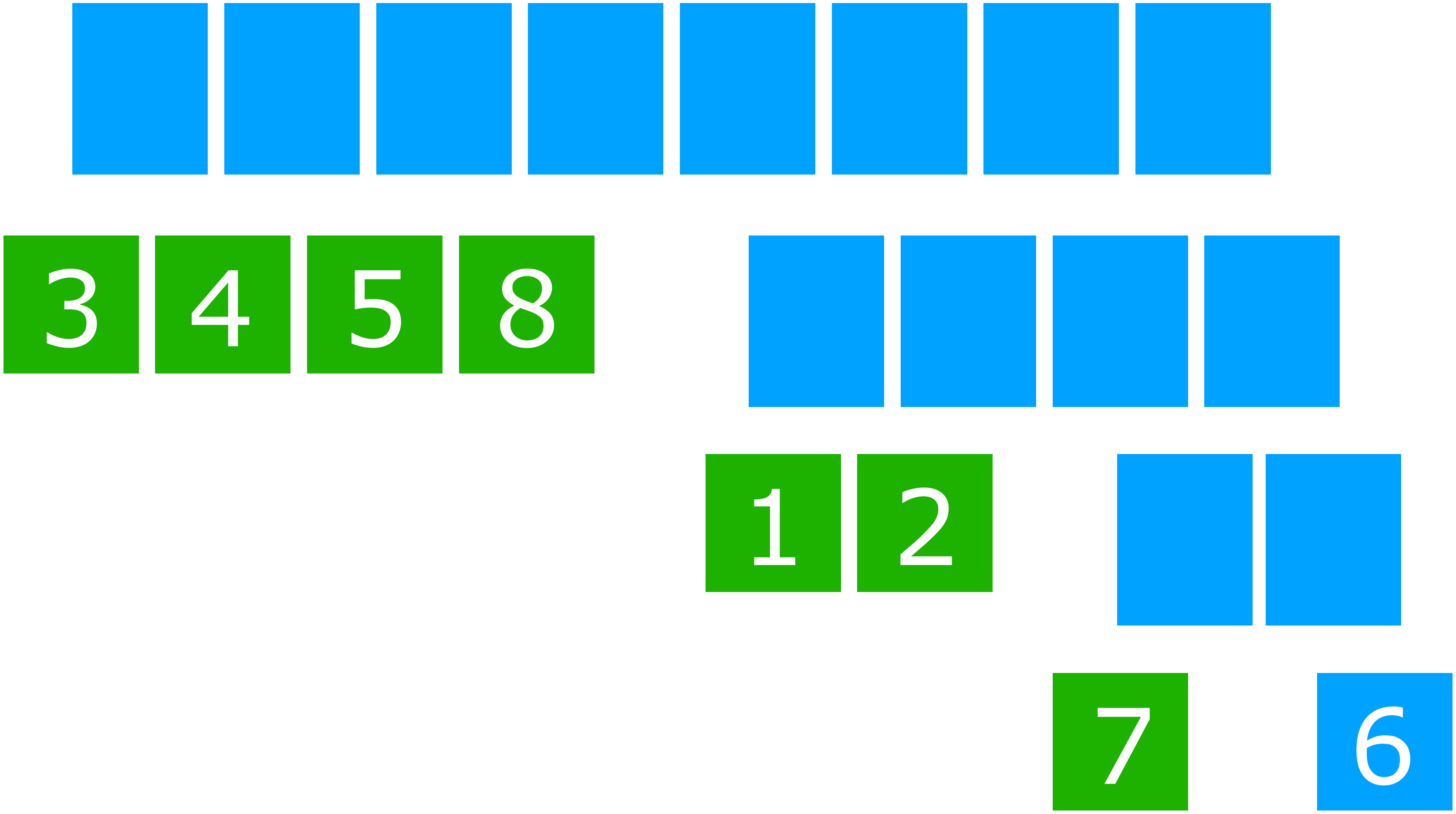

3
4
5
8
1
2
7
6


3
4
5
8
1
2
7
6


3
4
5
8
1
2
7
6


3
4
5
8
1
2
7
6

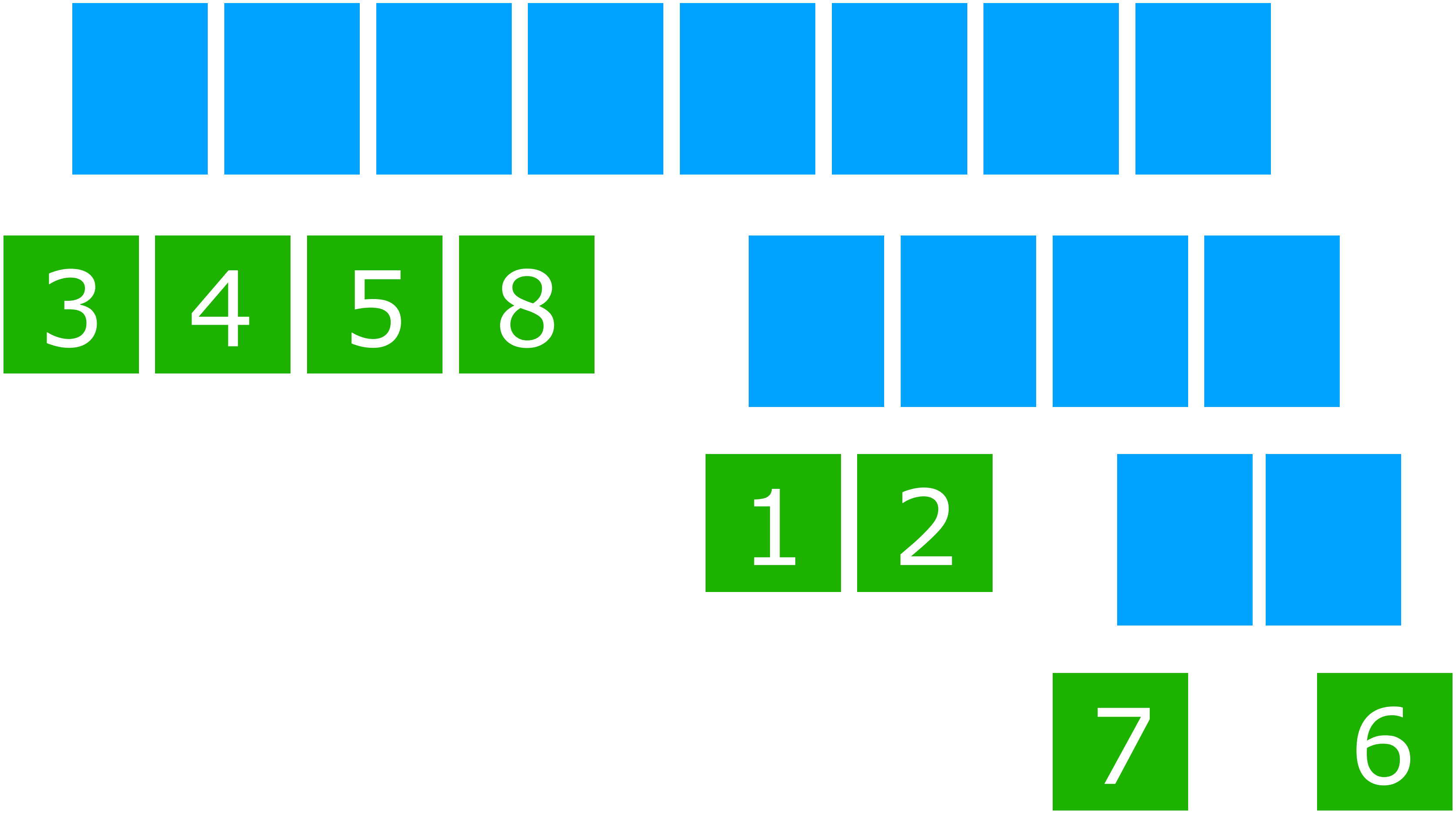

3
4
5
8
1
2
7
6

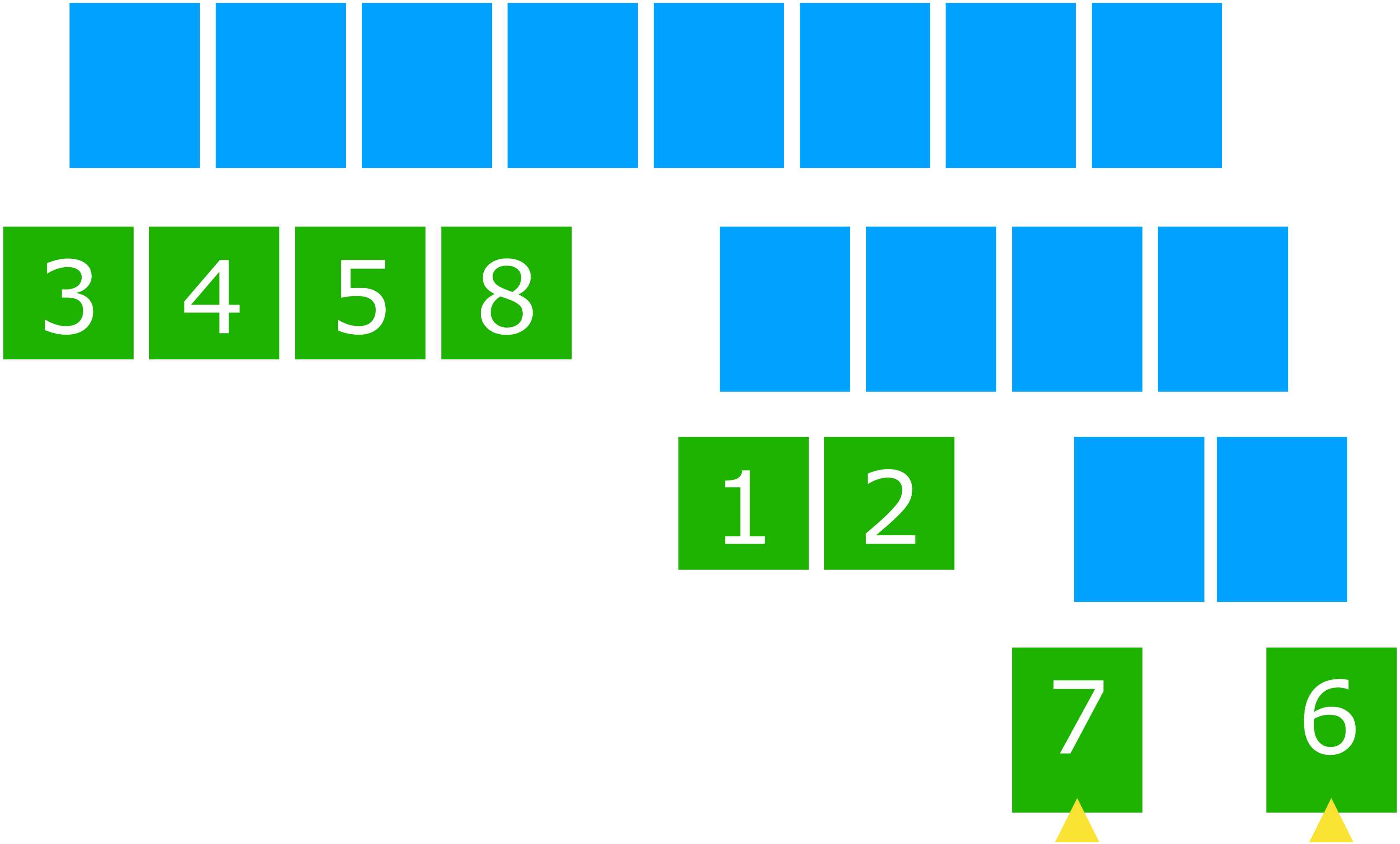


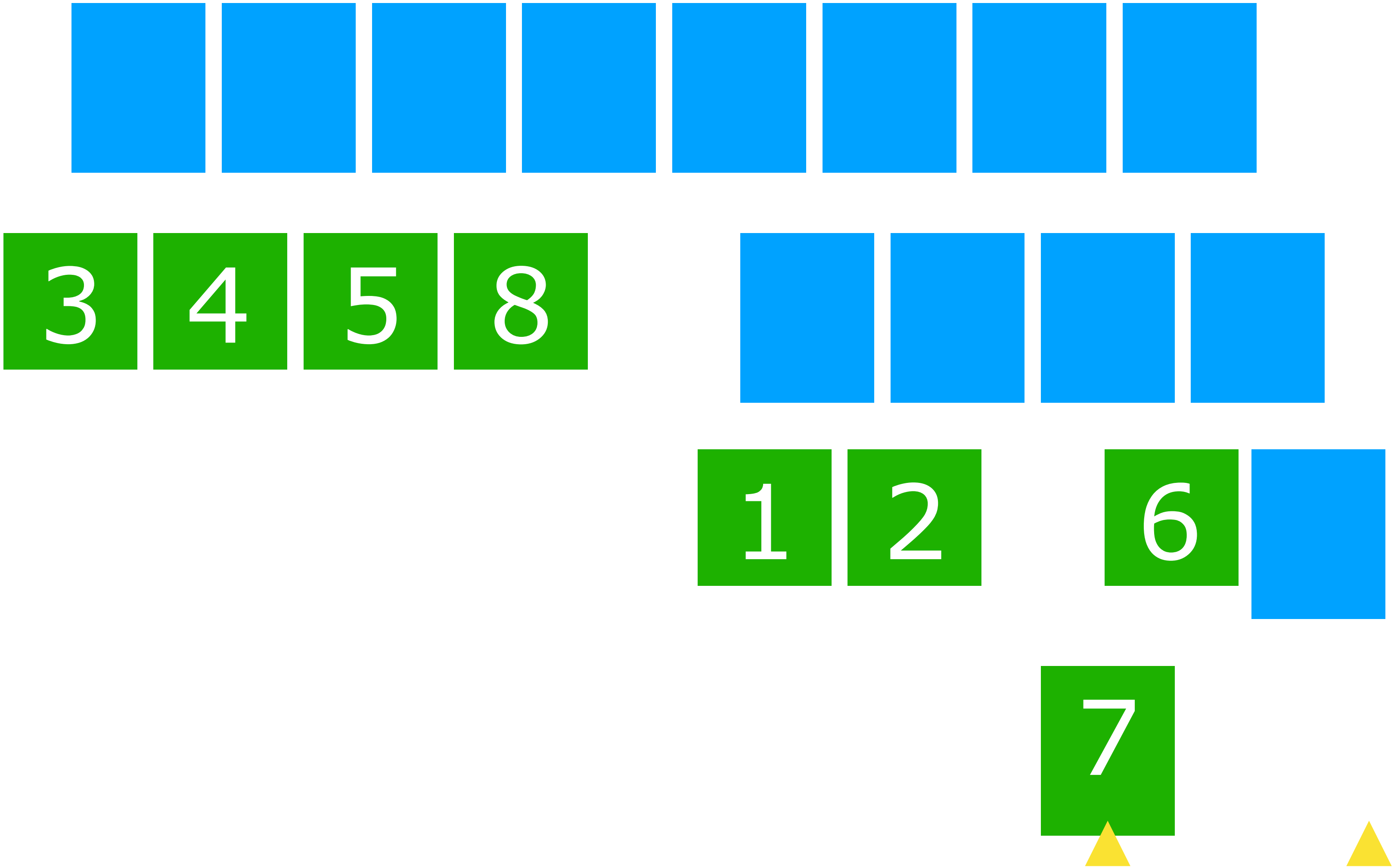
3 4 5 8

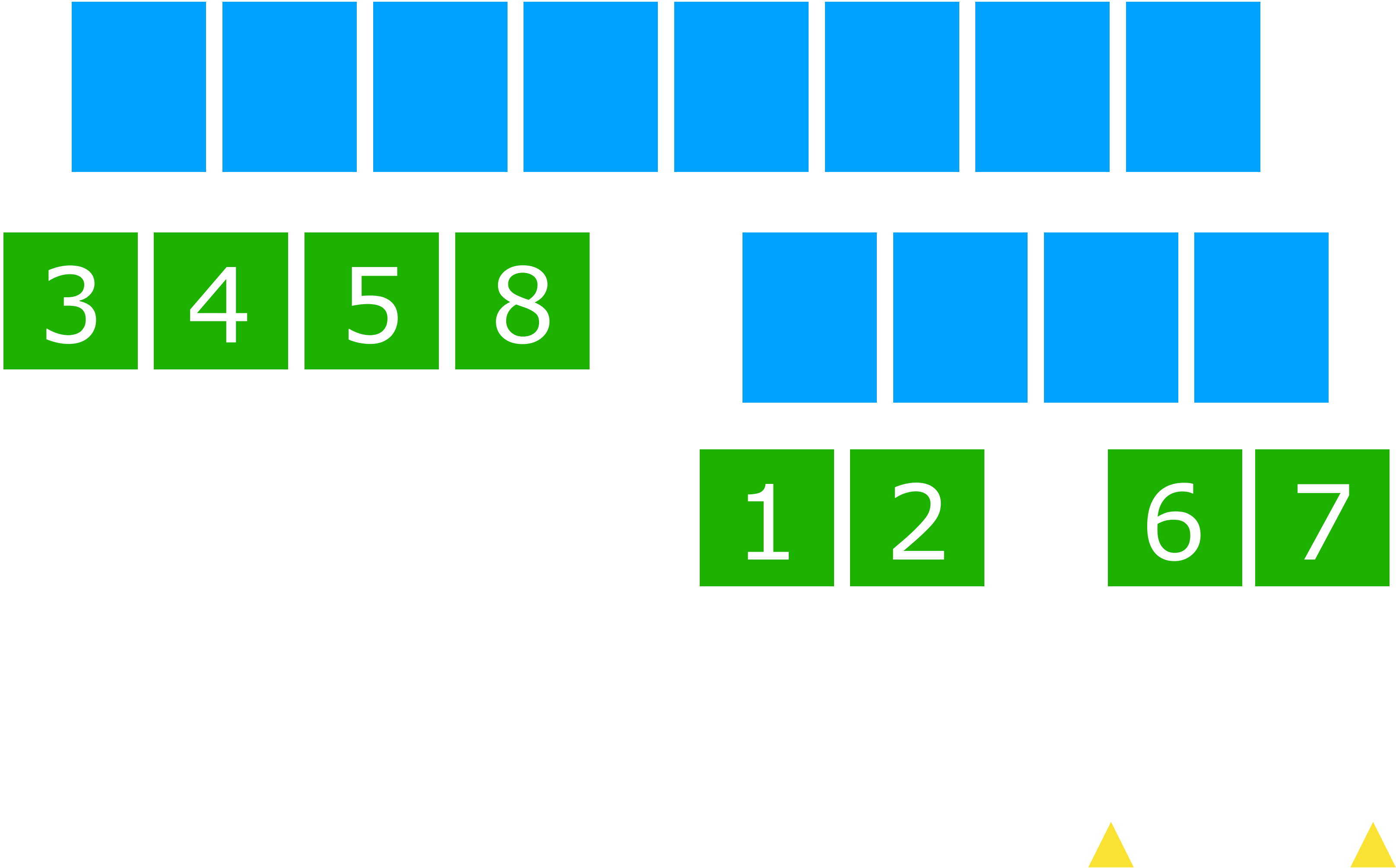
1 2

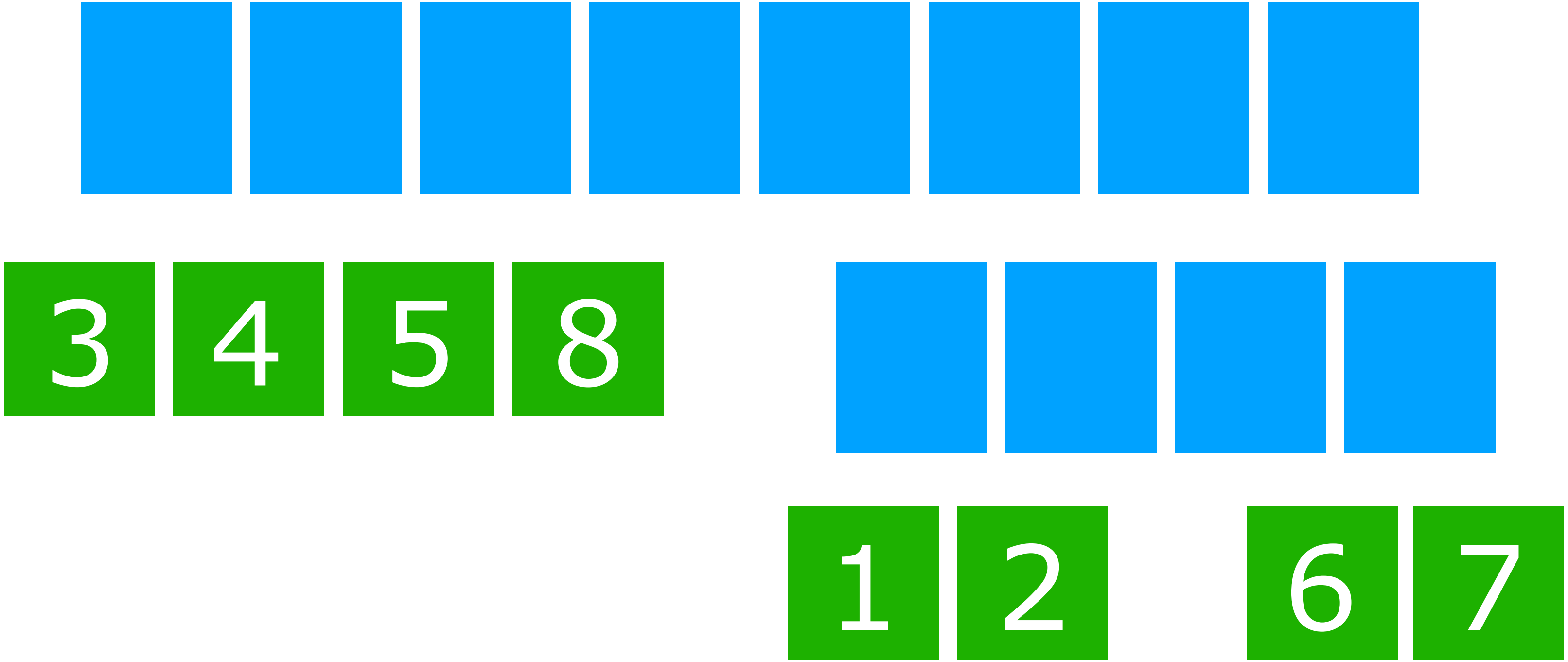
7 6


3
4
5
8
1
2
7
6


3
4
5
8
1
2
7
6


3
4
5
8
1
2
6
7

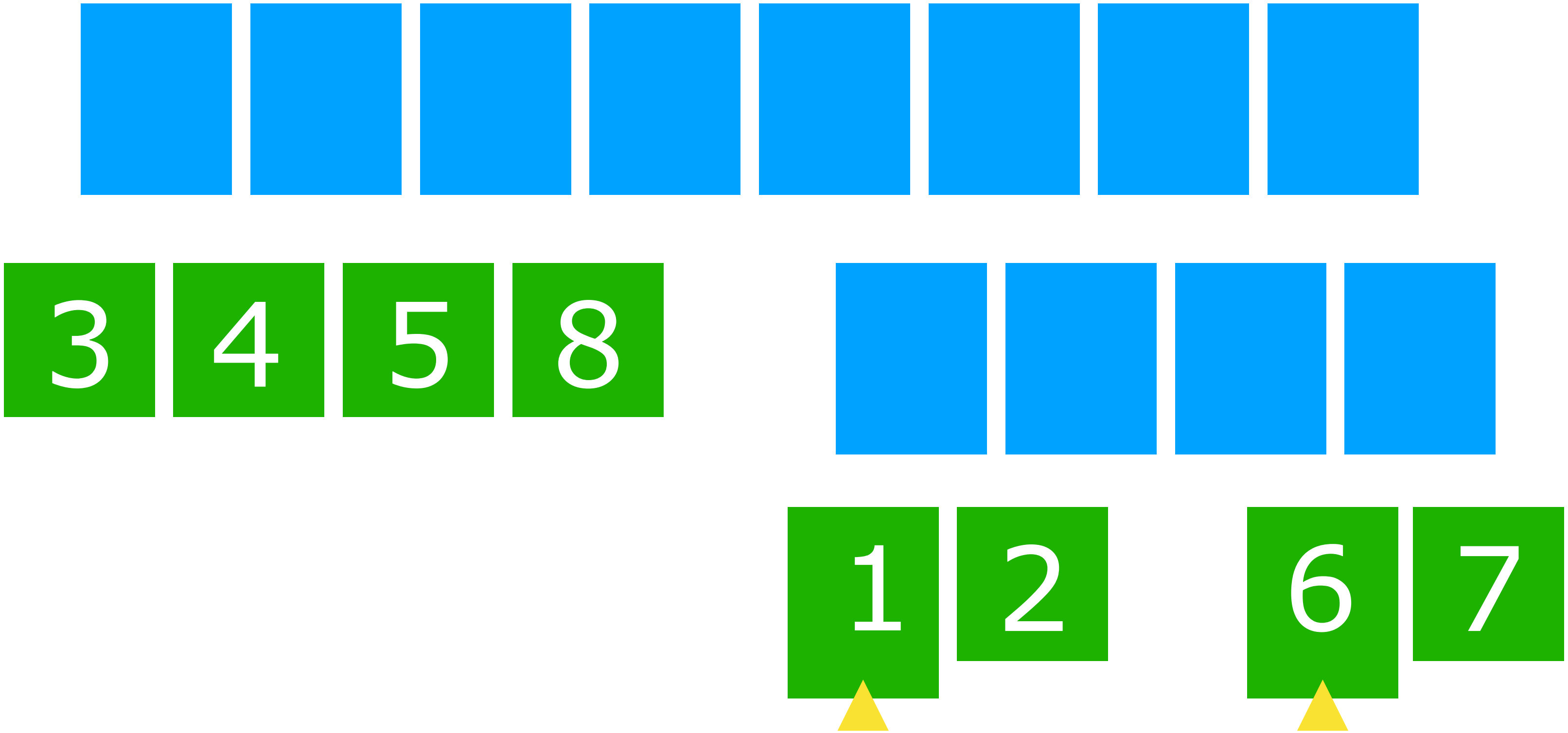

3
4
5
8
1
2
6
7

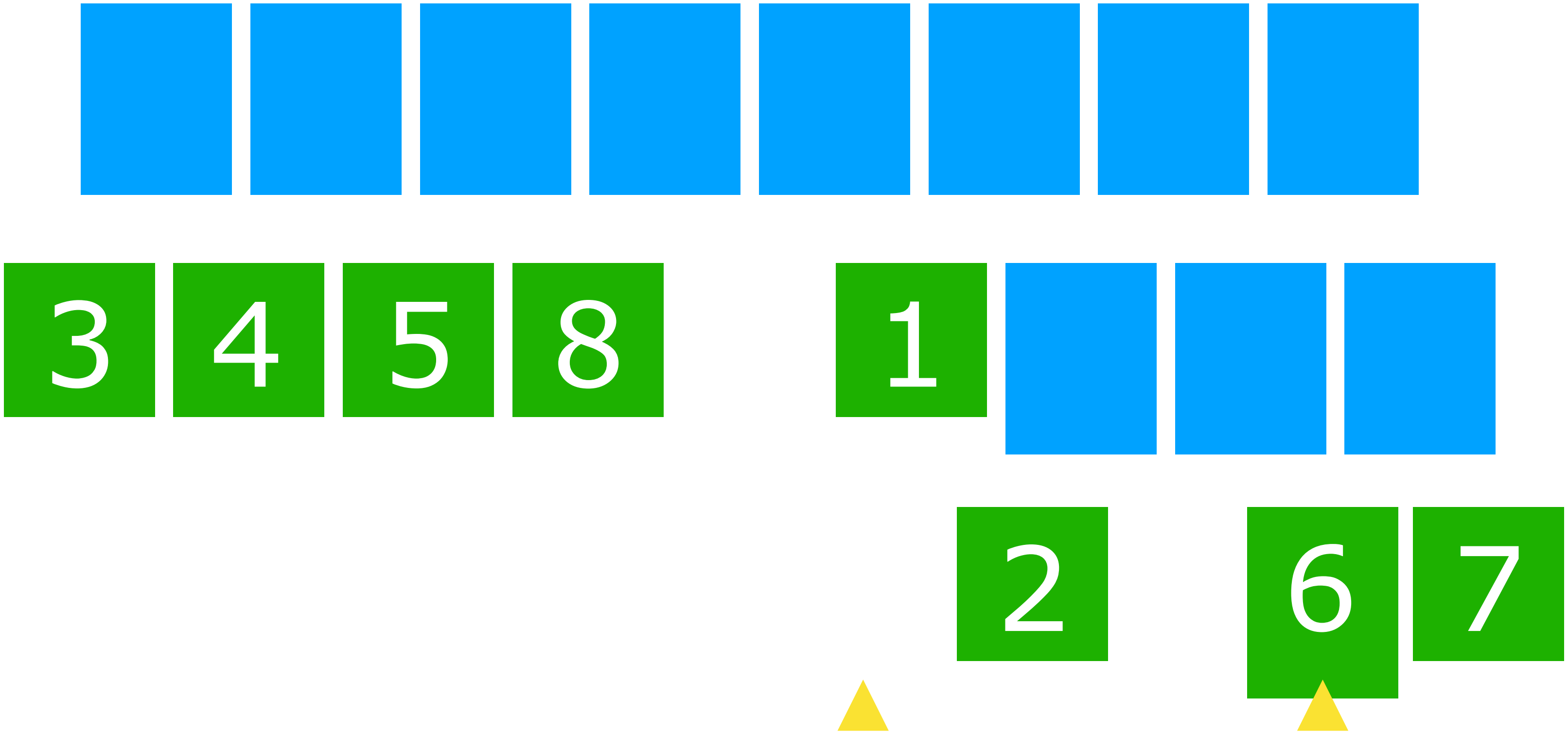


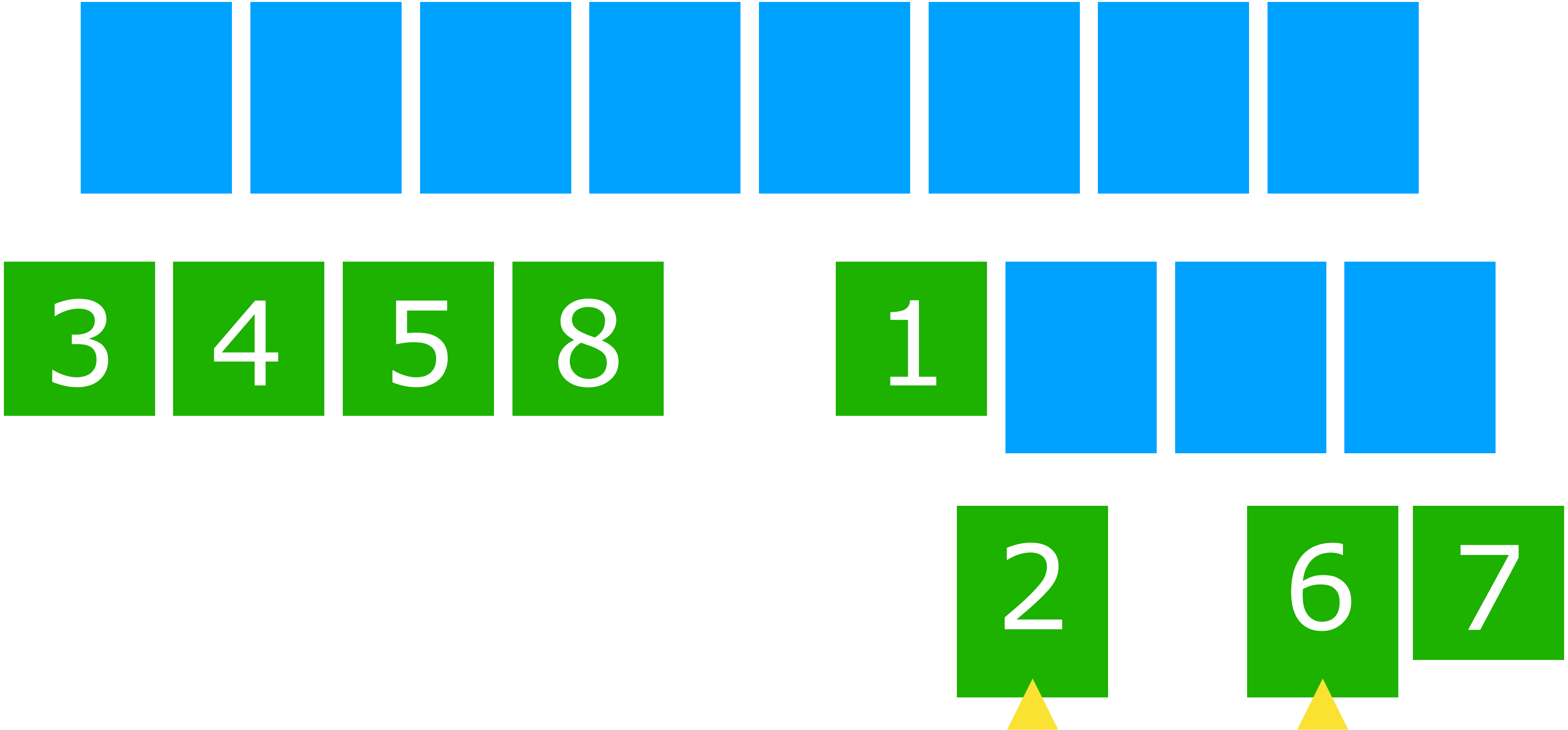
3 4 5 8

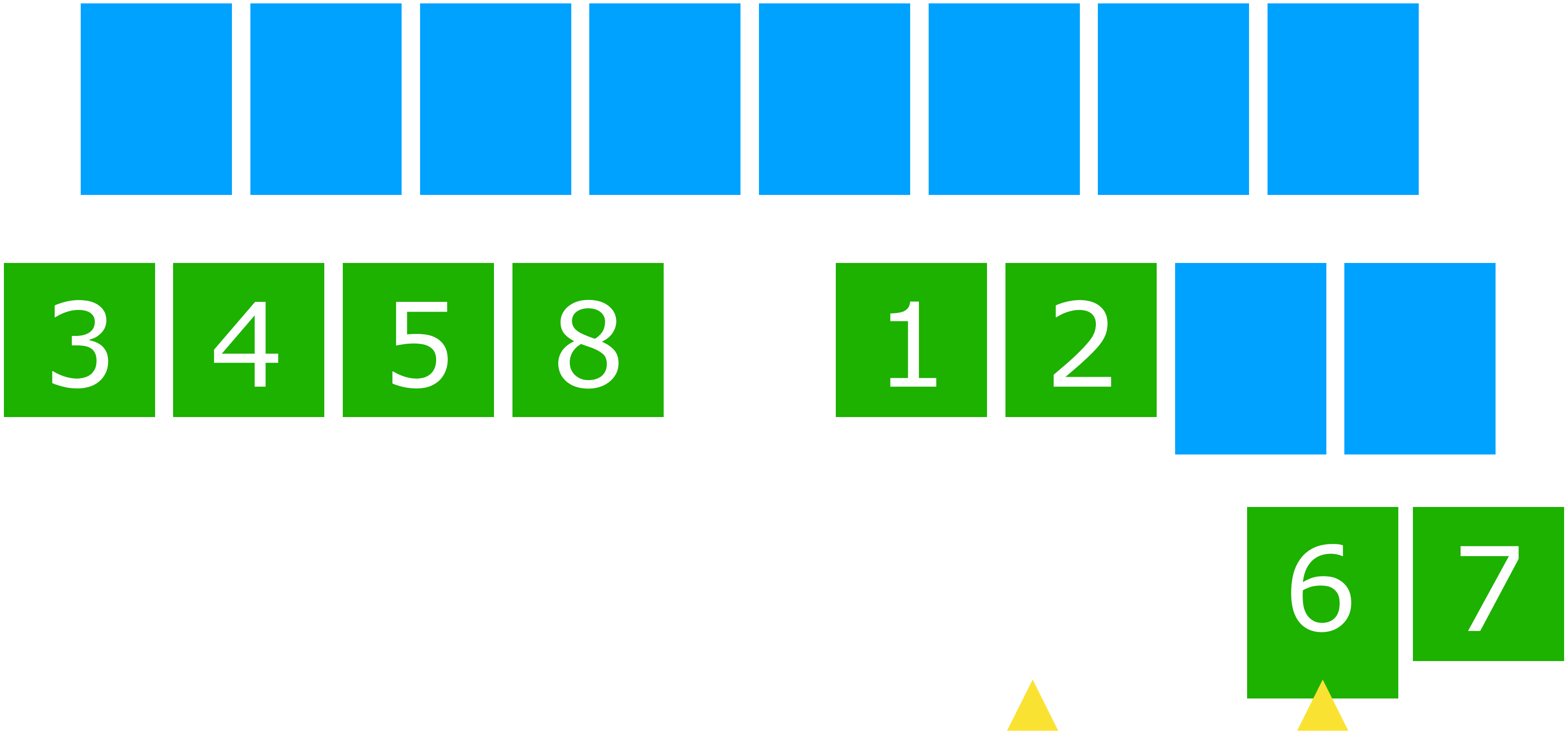
1 2

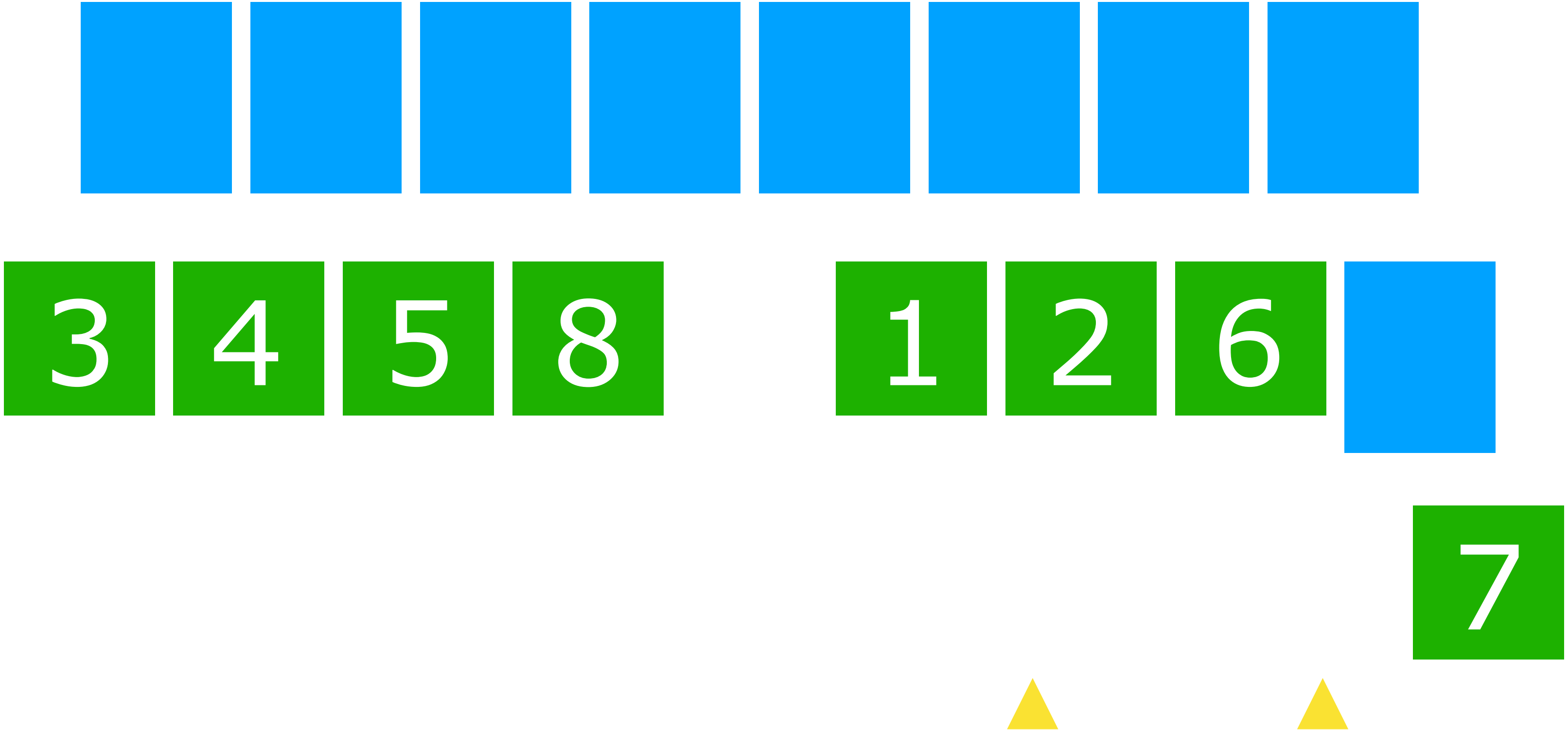
6 7

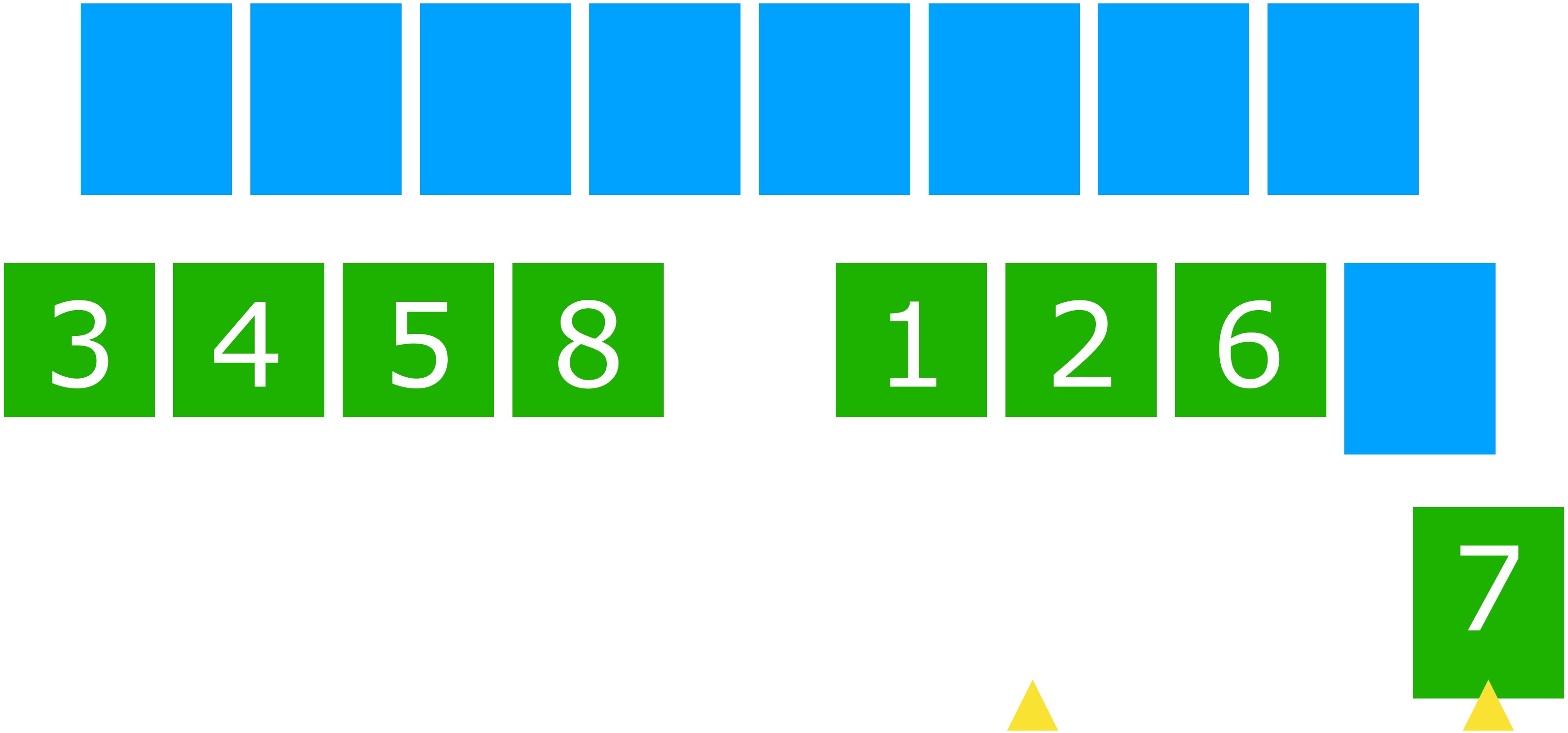

3
4
5
8
1
2
6
7

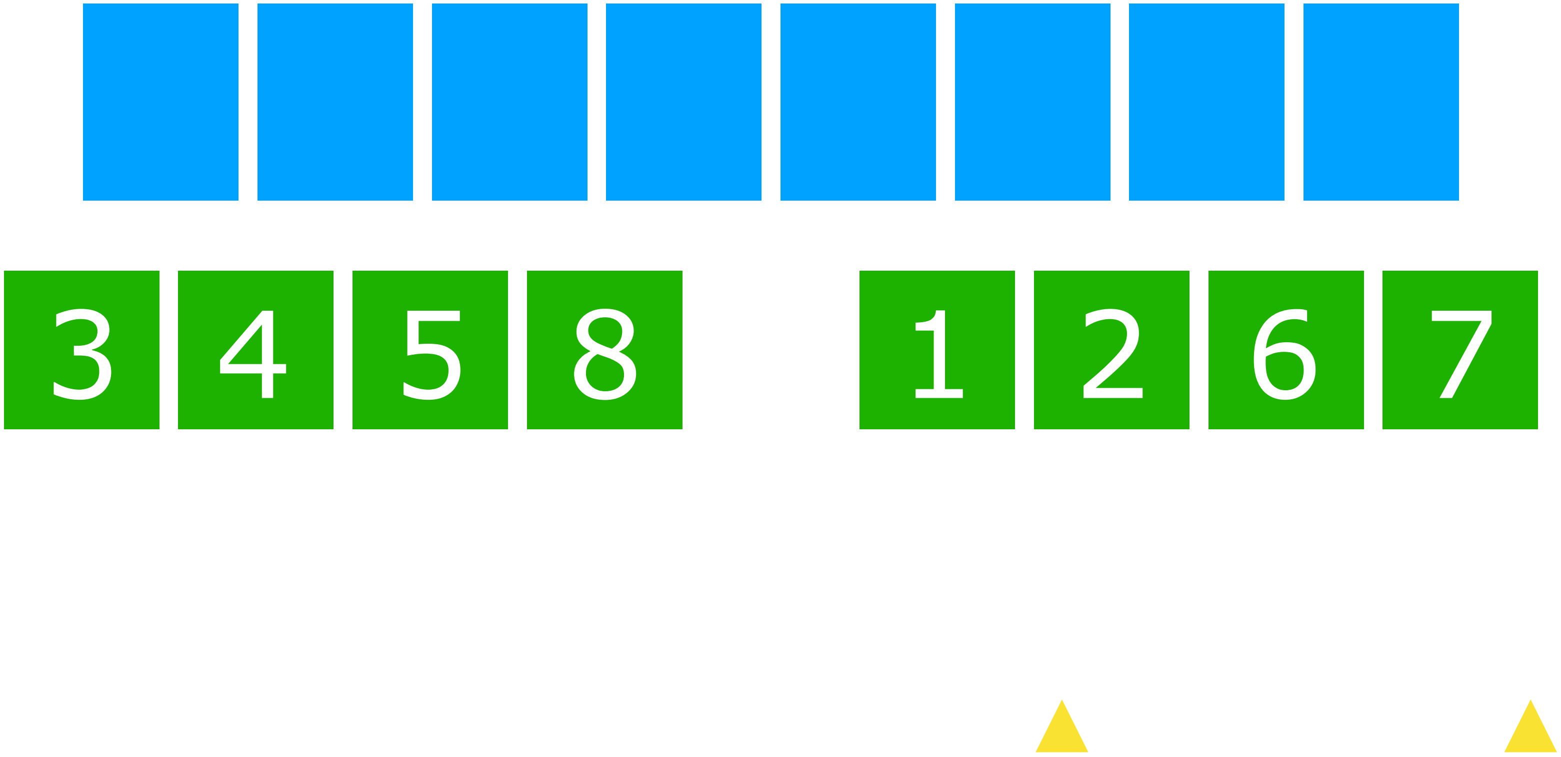

3
4
5
8
1
2
6
7

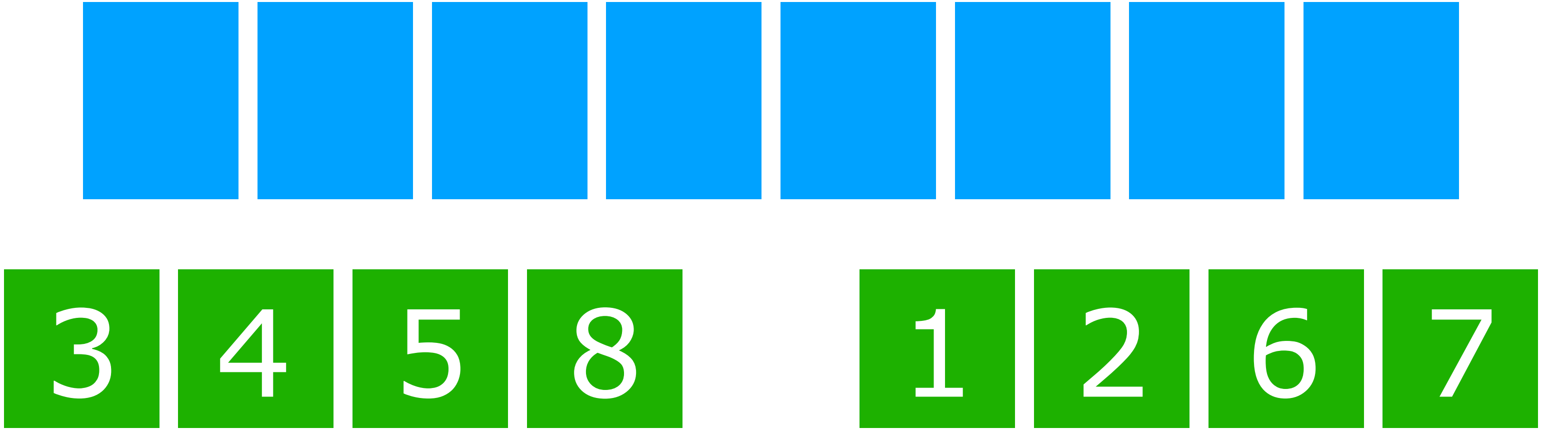

3
4
5
8
1
2
6
7

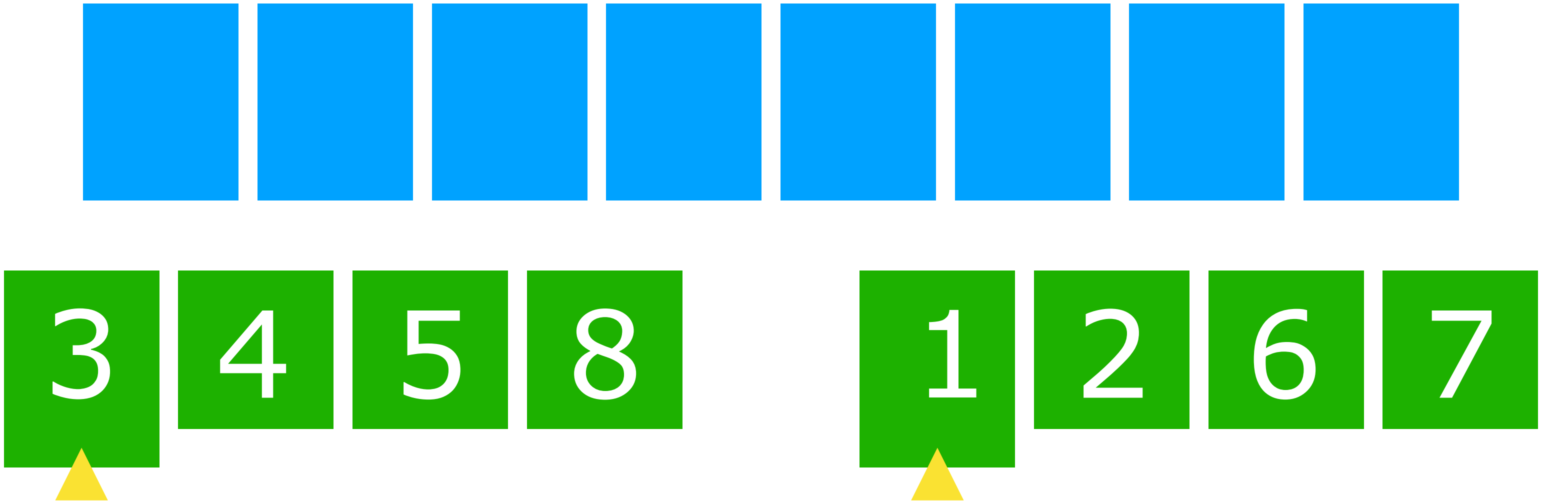

3
4
5
8
1
2
6
7

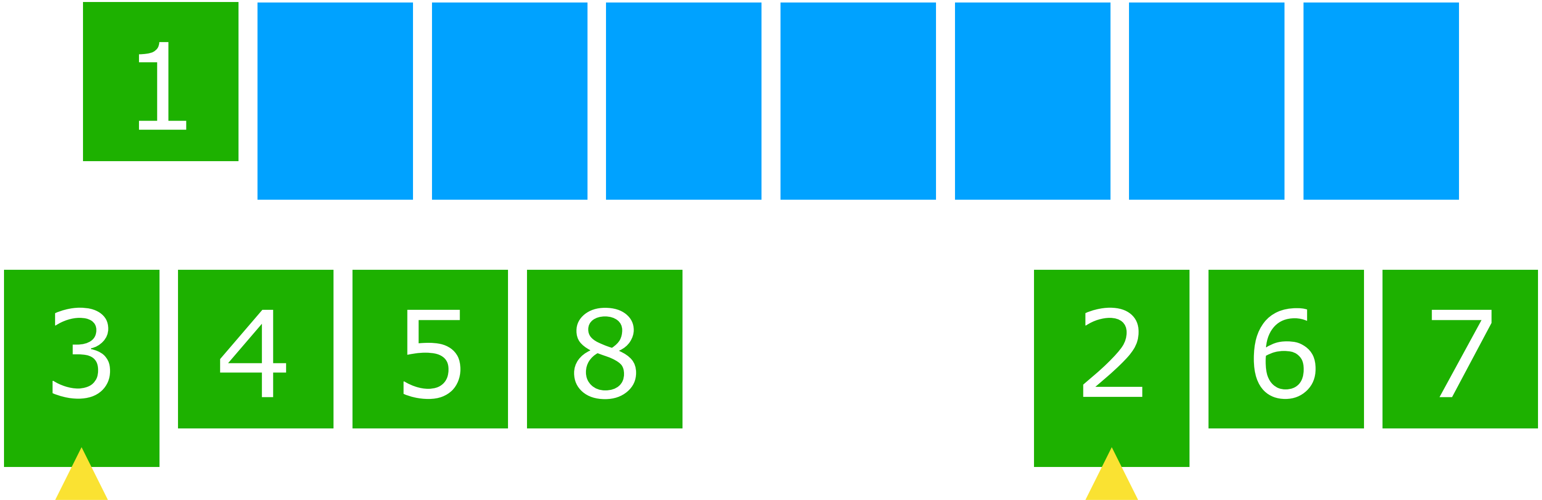

3
4
5
8
1
2
6
7

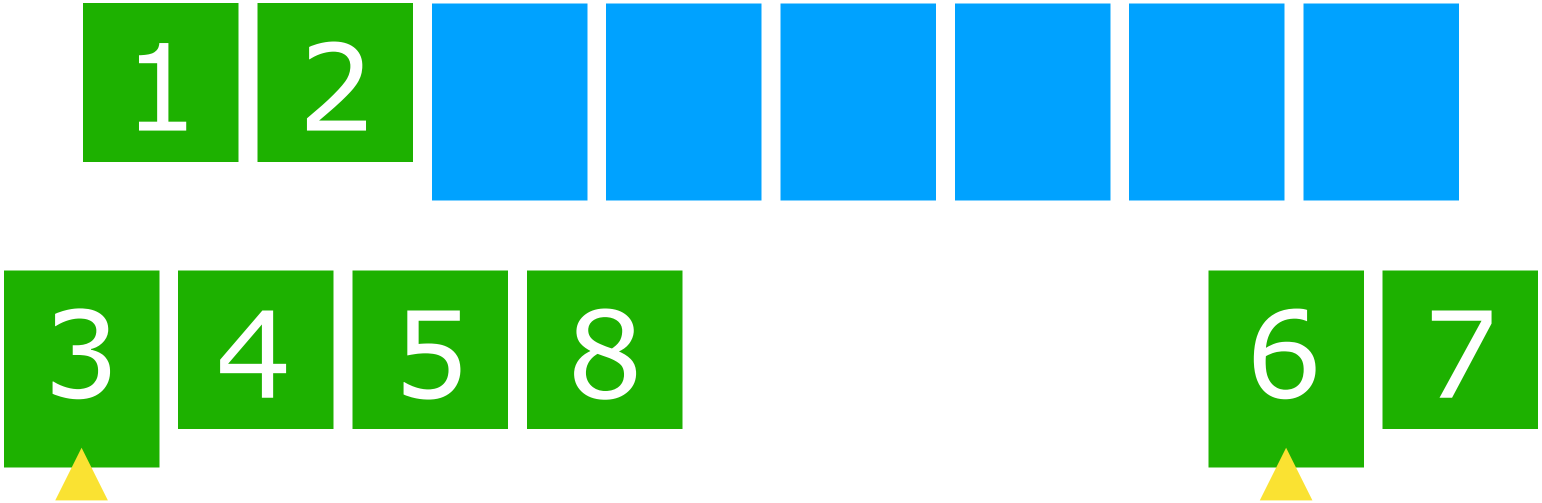

3
4
5
8
1
2
6
7

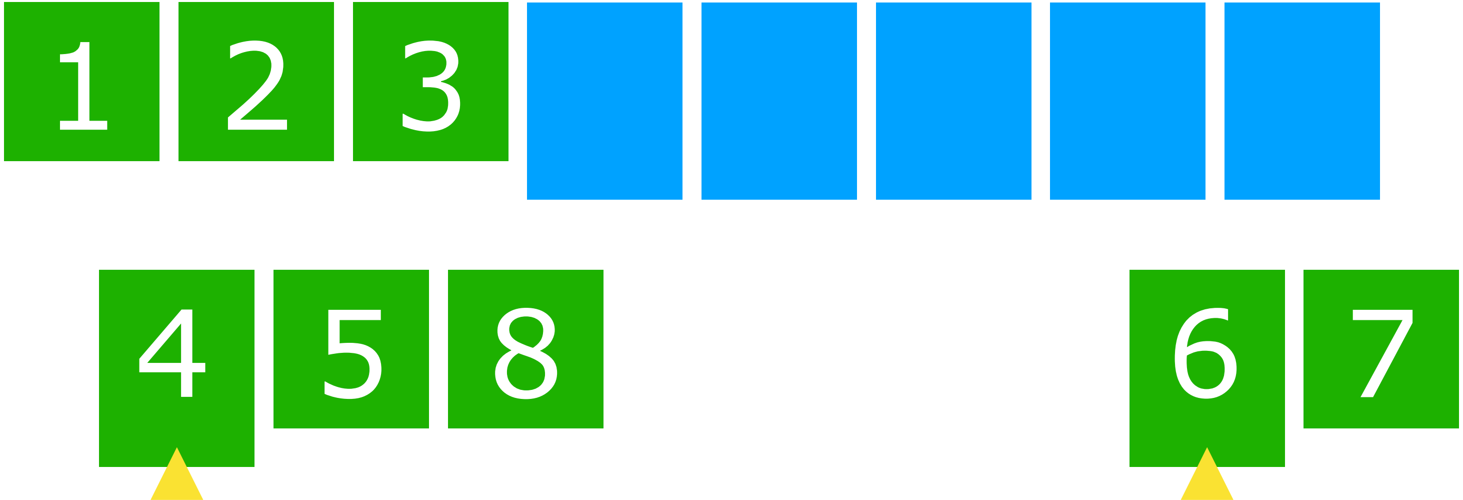

3
4
5
8
1
2
6
7

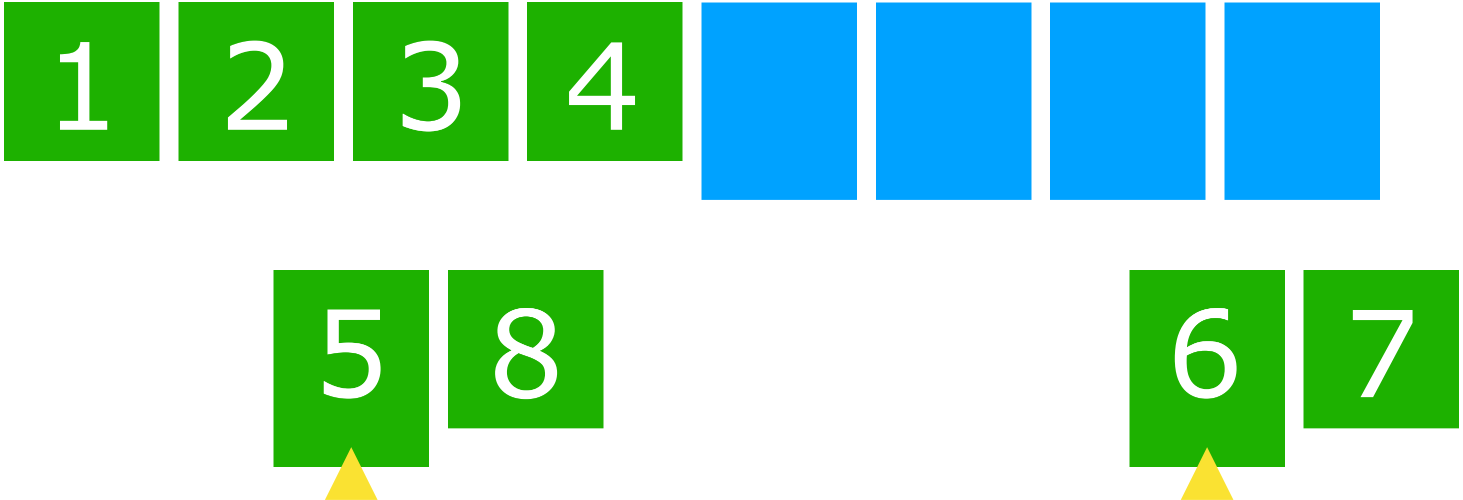

3
4
5
8
1
2
6
7

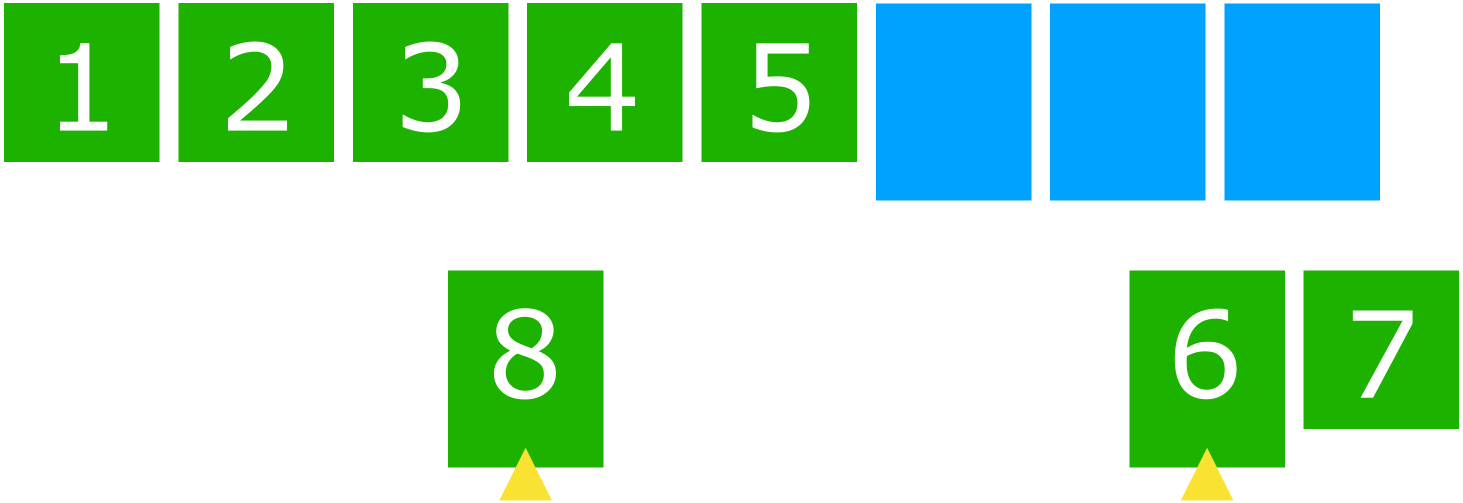

3
4
5
8
1
2
6
7

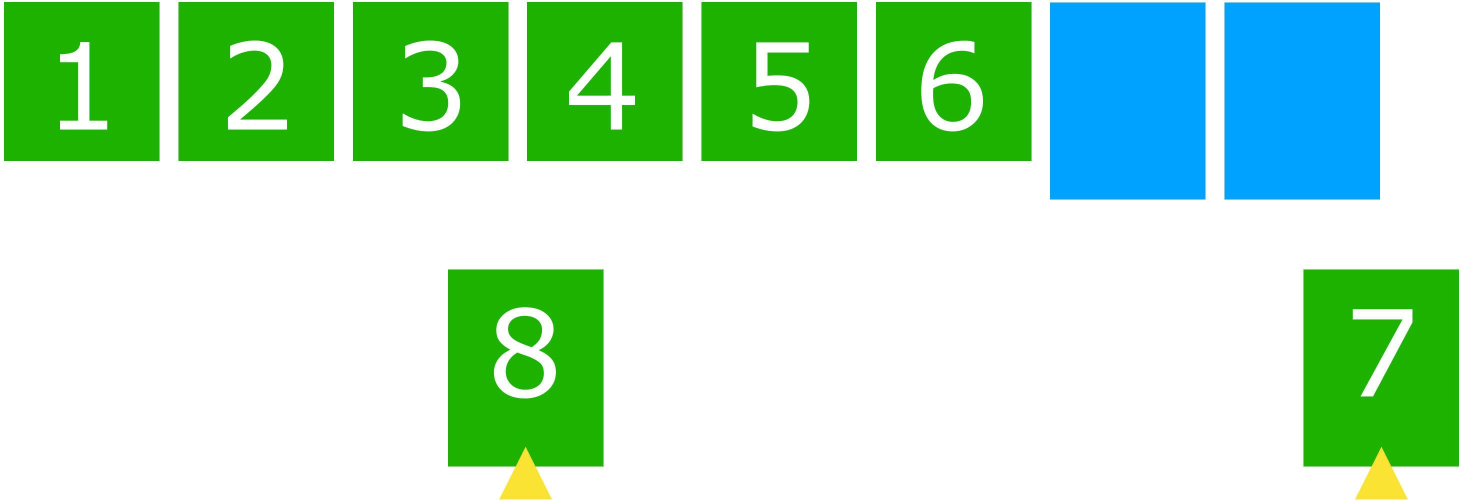

1
3
4
5
8
2
6
7

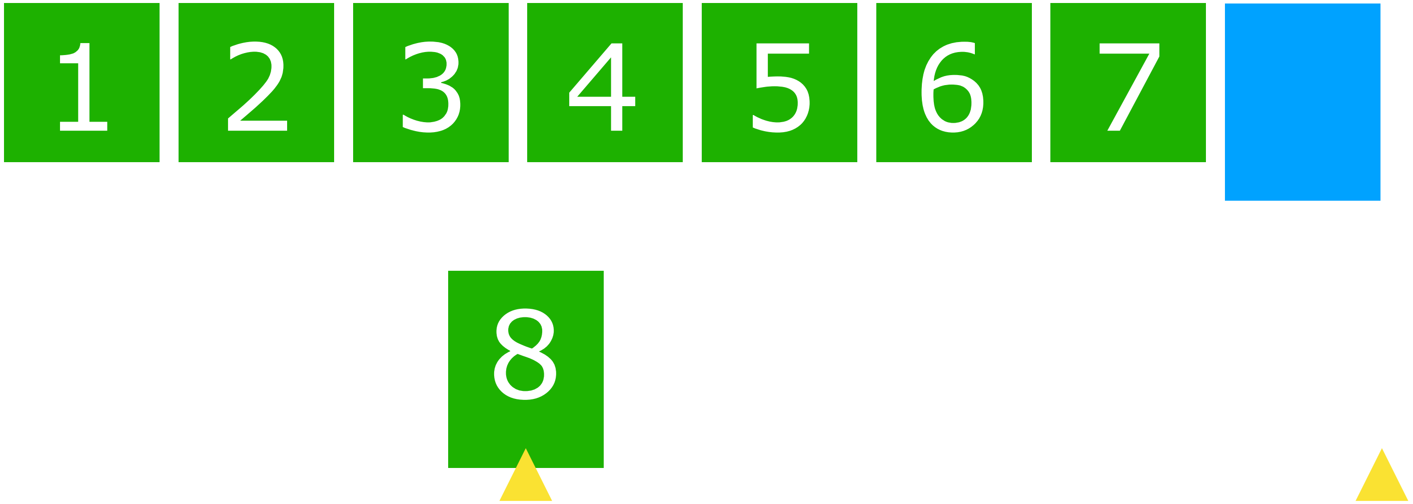

1
2
3
4
5
8
6
7

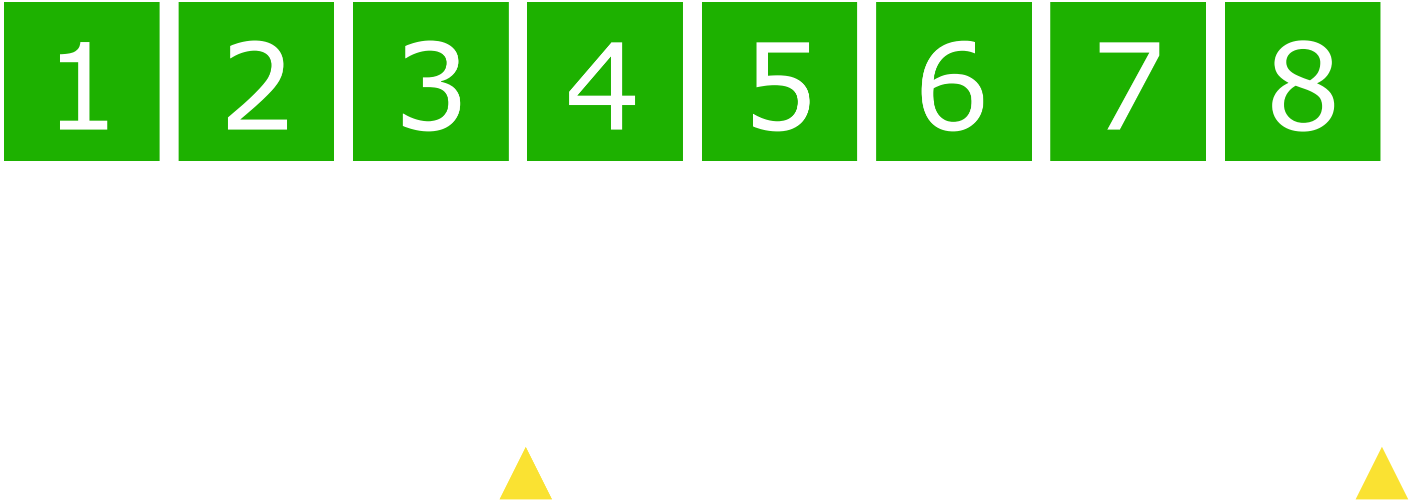

1
2
3
4
5
8
6
7


1
2
3
4
5
8
6
7


1
2
3
4
5
8
6
7


1
2
3
4
5
6
8
7


1
2
3
4
5
6
7
8



1	2	3	4	5	6	7	8

1	2	3	4	5	6	7	8

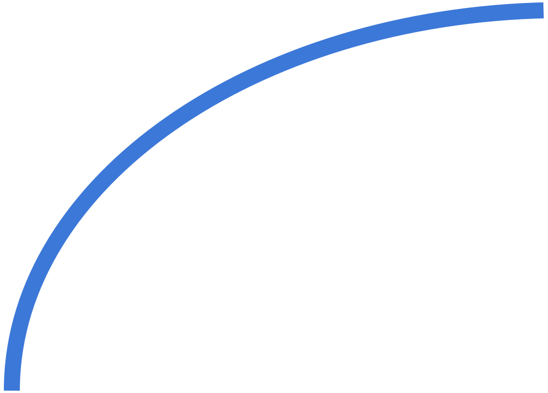
1 2 3 4 5 6 7 8

$O(\log_2(n))$ — "worst case" definition

In the worst case, I need to do $\log_2(n)$ steps to find my solution.

$O(\log_2(n))$ — "scaling" definition

I don't need to take another step in my algorithm until I double my input.



Lab

"Real" time (s)	**Bubble 1**	**Merge 2**	**Selection 3**
`Sorted 50,000`	.354s	.432s	3.599s
`Random 50,000`	7.558s	.495s	3.747s
`Reversed 50,000`	5.634s	.480s	3.838s

Tutorials

Office Hours

cs50.ly/attend

This was CS50

Guvf jnf PF50