DIGI111
Algoritmer og programmering

David Grellscheid

input / print

True / False

betingelser: if

løkker: for, while

datastrukturer: list, ...

For å løse et problem:

finne mindre steg, der vi kan bruke strukturene

finne enklere spørsmål som allerede har en løsning

Standard-algoritmer for bla.
sortering, grafer, søk, ...

Ulike datastrukturer:
list, set, numpy-array, ...

```
xs = [2,7,5,1,4,3,6,8]
```

```
while not is_sorted(xs):
    random.shuffle(xs)
```

$O(N\,N!)$

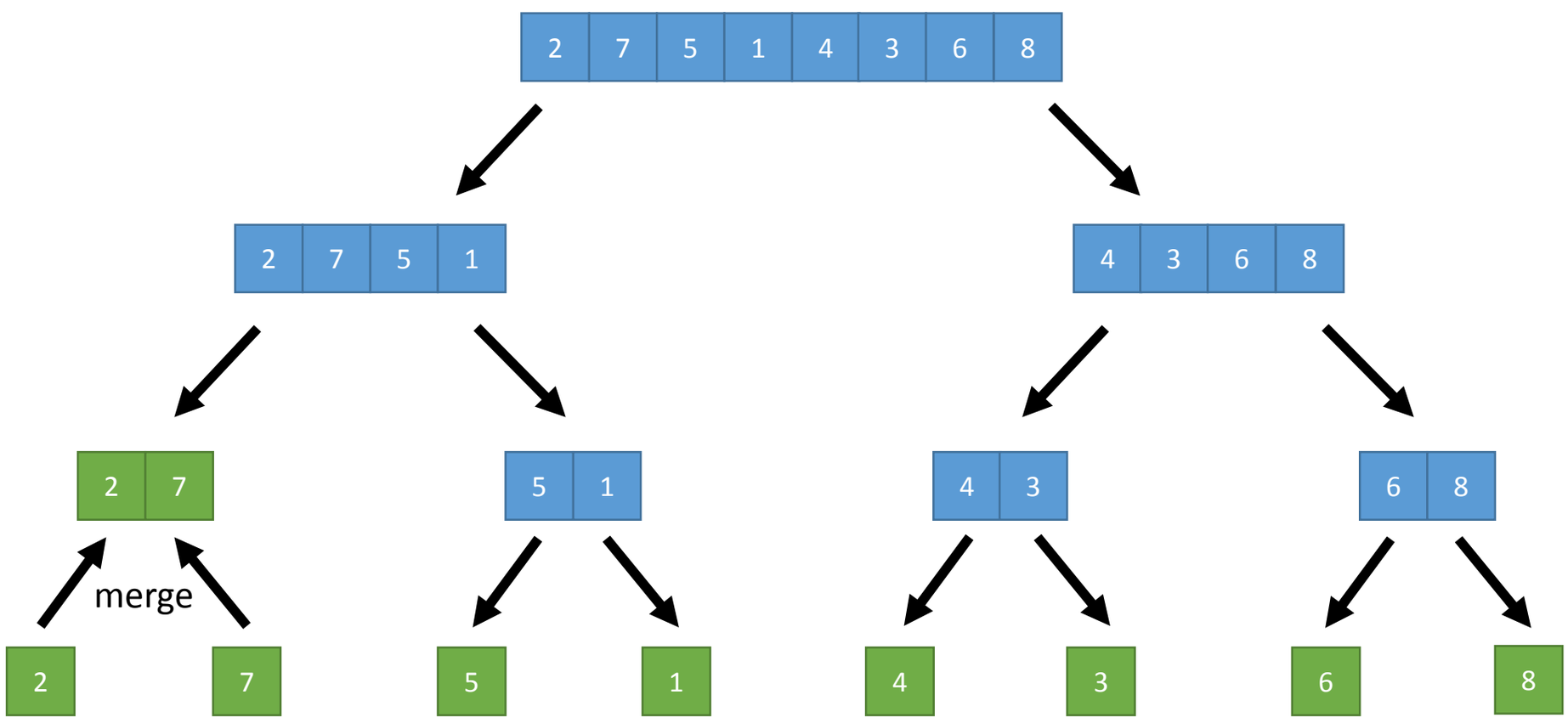
Scaling behaviour with size N of problem set:
$O(1)$ - constant time independent of N
$O(N)$ - linear with N
$O(N^2)$ - quadratic in N

Insertion sort

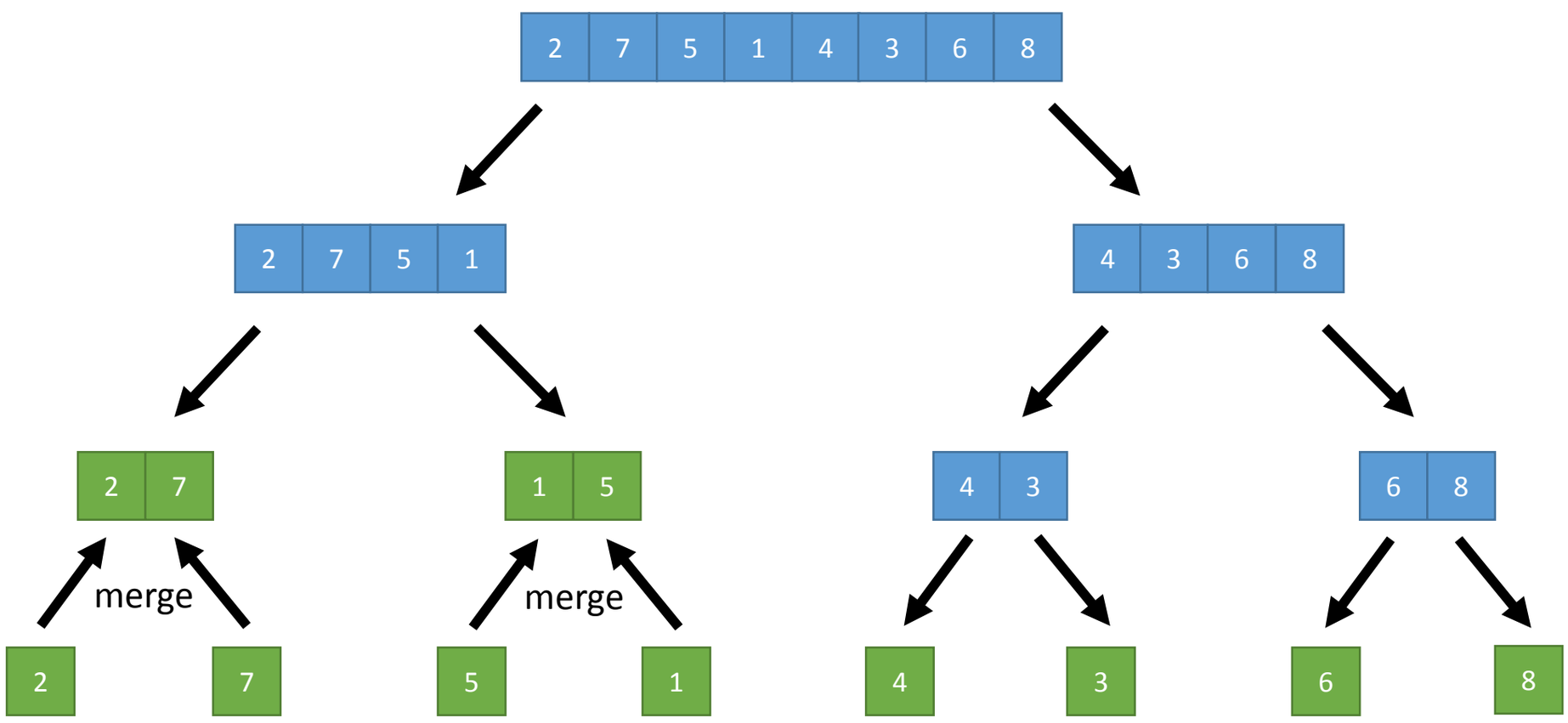
$O(N^2)$

2	7	5	1	4	3	6	8
2	5	7	1	4	3	6	8
1	2	5	7	4	3	6	8
1	2	4	5	7	3	6	8
1	2	3	4	5	7	6	8
1	2	3	4	5	6	7	8

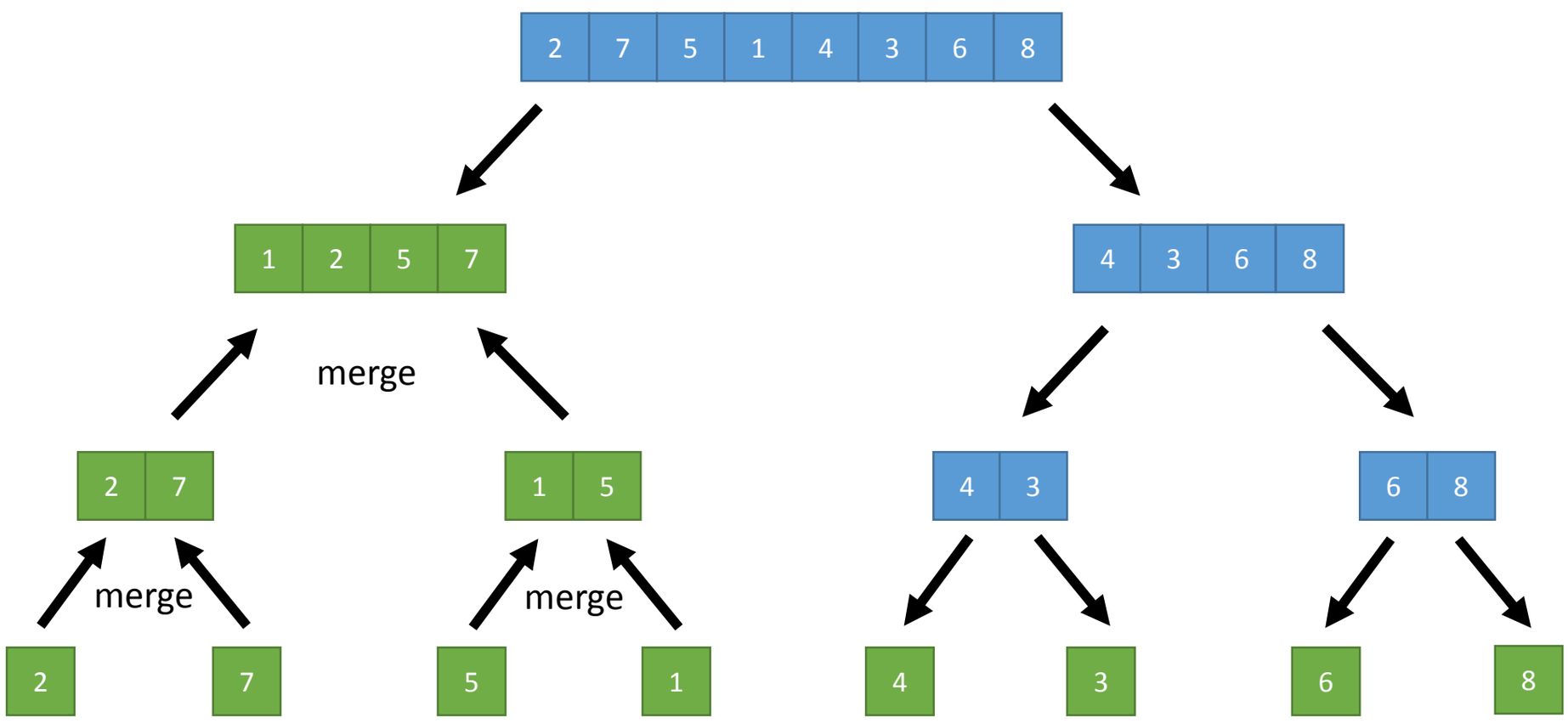
Merge Sort



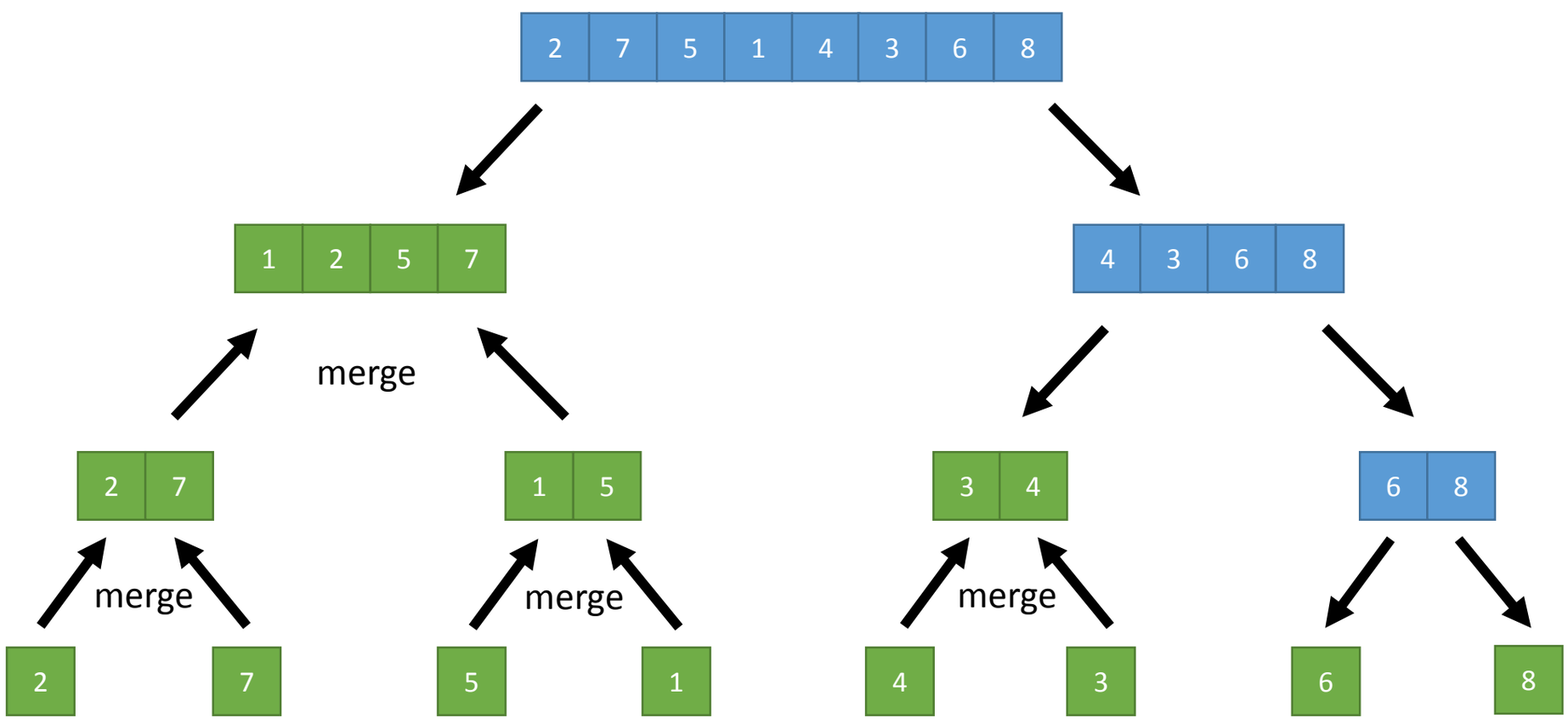
Merge Sort



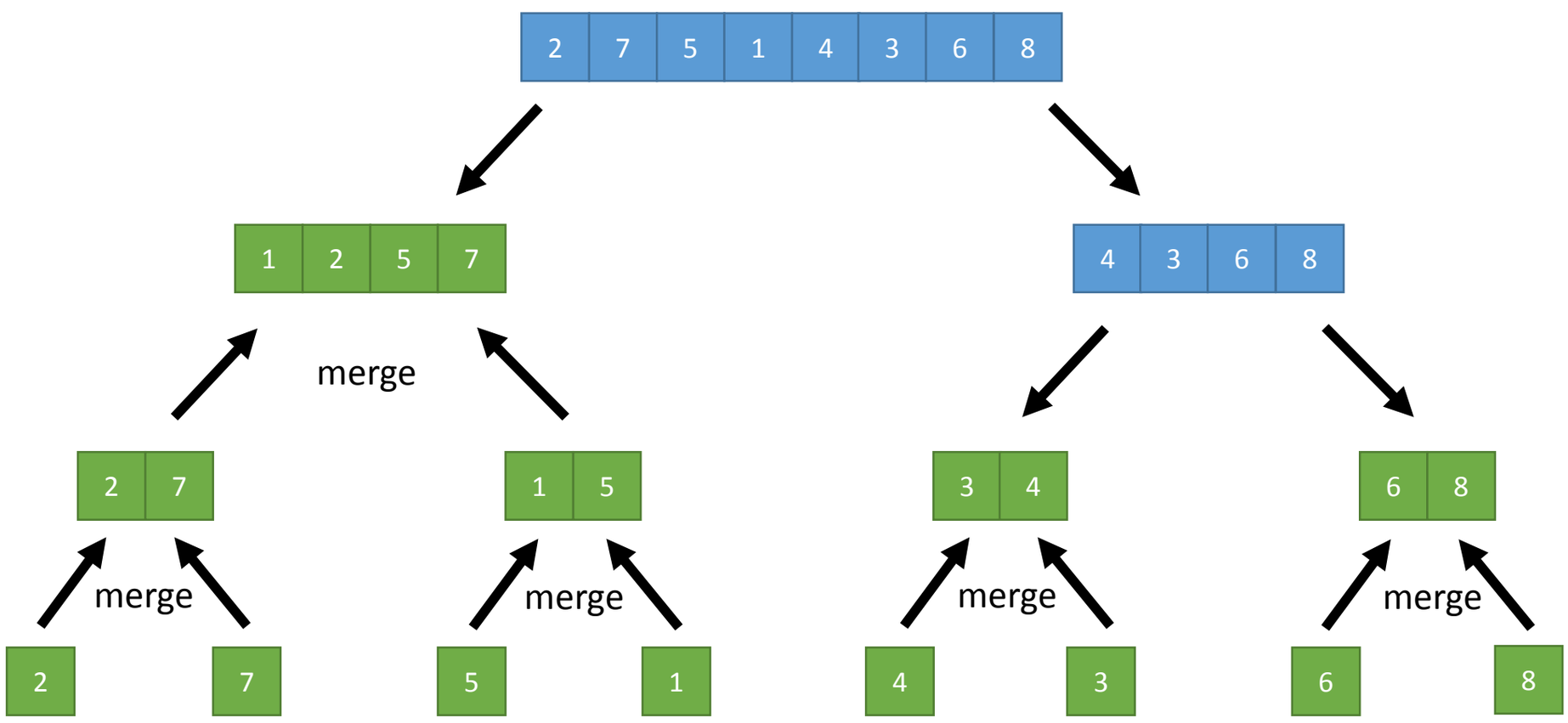
Merge Sort



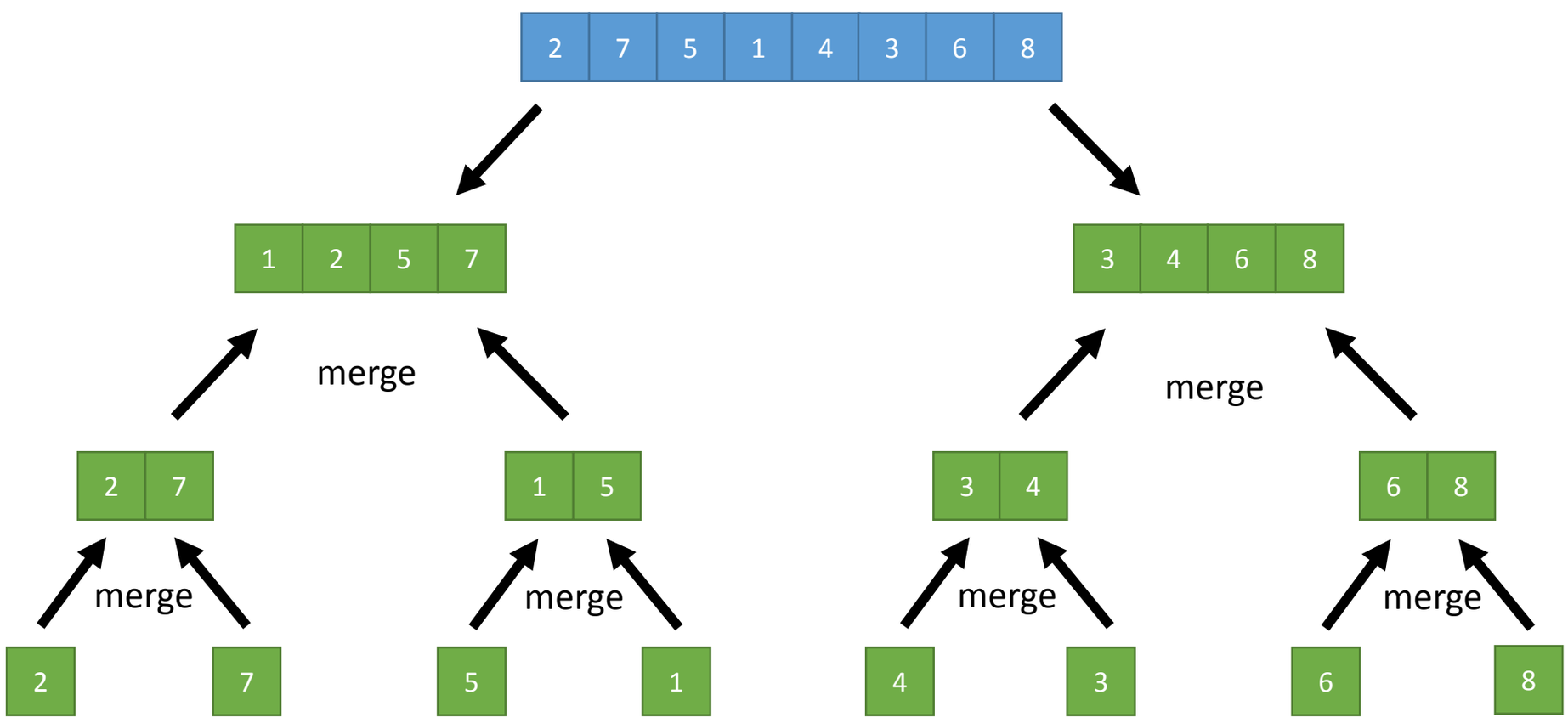
Merge Sort


2 7 5 1 4 3 6 8
1 2 5 7
merge
2 7
1 5
4 3 6 8
3 4
6 8
merge
merge
merge
2
7
5
1
4
3
6
8

Merge Sort

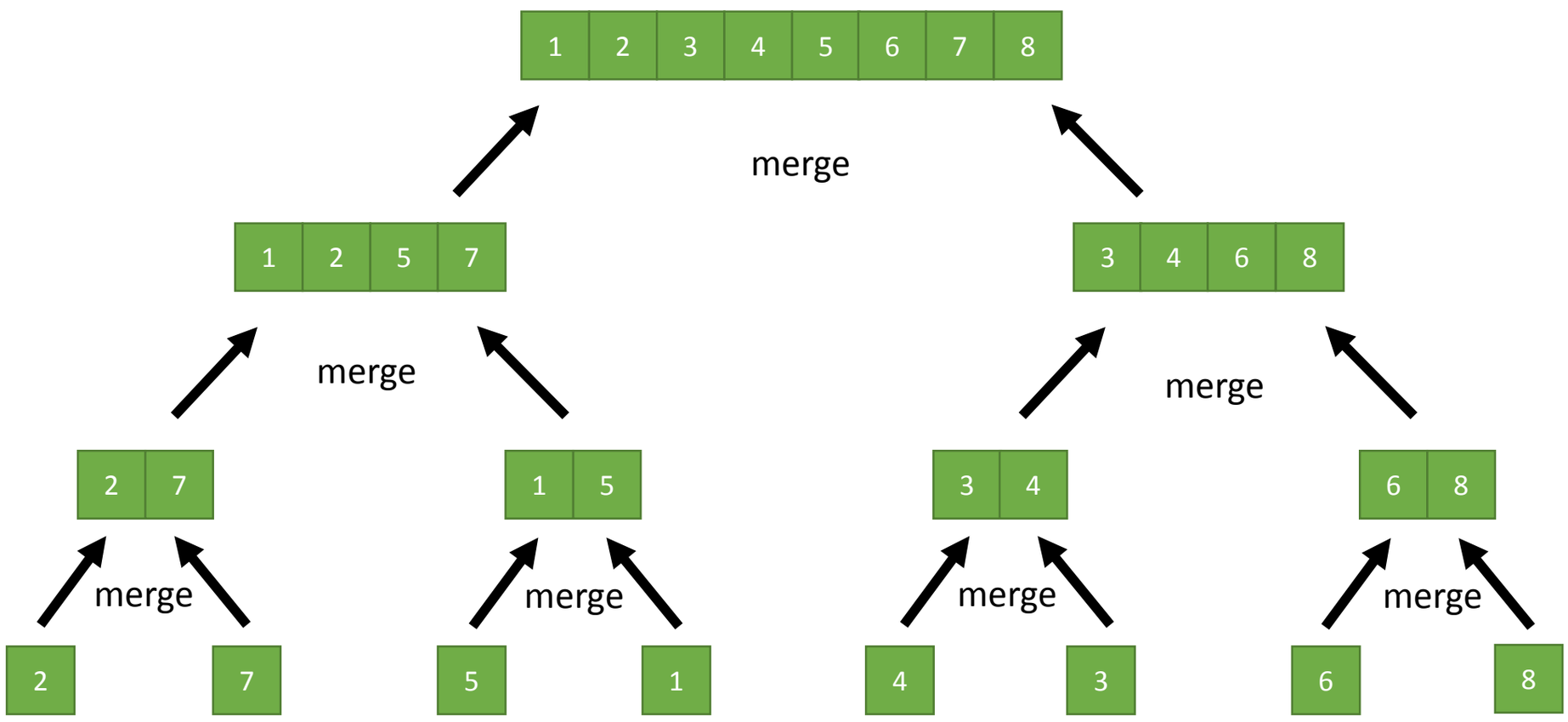

2 7 5 1 4 3 6 8
1 2 5 7
merge
2 7
1 5
4 3 6 8
3 4
6 8
merge
merge
merge
merge
2
7
5
1
4
3
6
8

Merge Sort


2 7 5 1 4 3 6 8
1 2 5 7
3 4 6 8
merge
merge
2 7
1 5
3 4
6 8
merge
merge
merge
merge
2
7
5
1
4
3
6
8

Merge Sort

$$O(N \log N)$$



15 Sorting Algorithms in 6 Minutes
http://youtu.be/kPRA0W1kECg

What are we optimizing?

Time 

Memory 

Disk

Electricity

Compile time

Ease of use

Ease of deployment

Ease of development

L1 cache reference ..... 0.5 ns

Branch mispredict ..... 5 ns

L2 cache reference ..... 7 ns

Mutex lock/unlock ..... 25 ns

Main memory reference ..... 100 ns

SSD random read ..... 150,000 ns = 150 µs

Read 1 MB sequentially from memory ..... 250,000 ns = 250 µs

Read 1 MB sequentially from SSD ..... 1,000,000 ns = 1 ms

Disk seek ..... 10,000,000 ns = 10 ms

Read 1 MB sequentially from disk .... 20,000,000 ns = 20 ms

Send packet EU->US->EU .... 150,000,000 ns = 150 ms

L1 cache reference	0.5 s
Branch mispredict	5 s
L2 cache reference	7 s
Mutex lock/unlock	25 s
Main memory reference	100 s
SSD random read	1.7 days
Read 1 MB sequentially from memory	2.9 days
Read 1 MB sequentially from SSD	11.6 days
Disk seek	16.5 weeks
Read 1 MB sequentially from disk	7.8 months
Send packet EU->US->EU	4.8 years

Python profiling options

time

```
from time import time
start = time()
somefunc(27)
end = time()
```

timeit

```
python -m timeit -s ‘import myfile as m;
x=27’ ‘m.somefunc(x)’
```

cProfile

```
import cProfile
cProfile.run(somefunc(27))
```

pyprof2calltree
qcachegrind

All are in the standard library