Introduction to Artificial Intelligence with Python

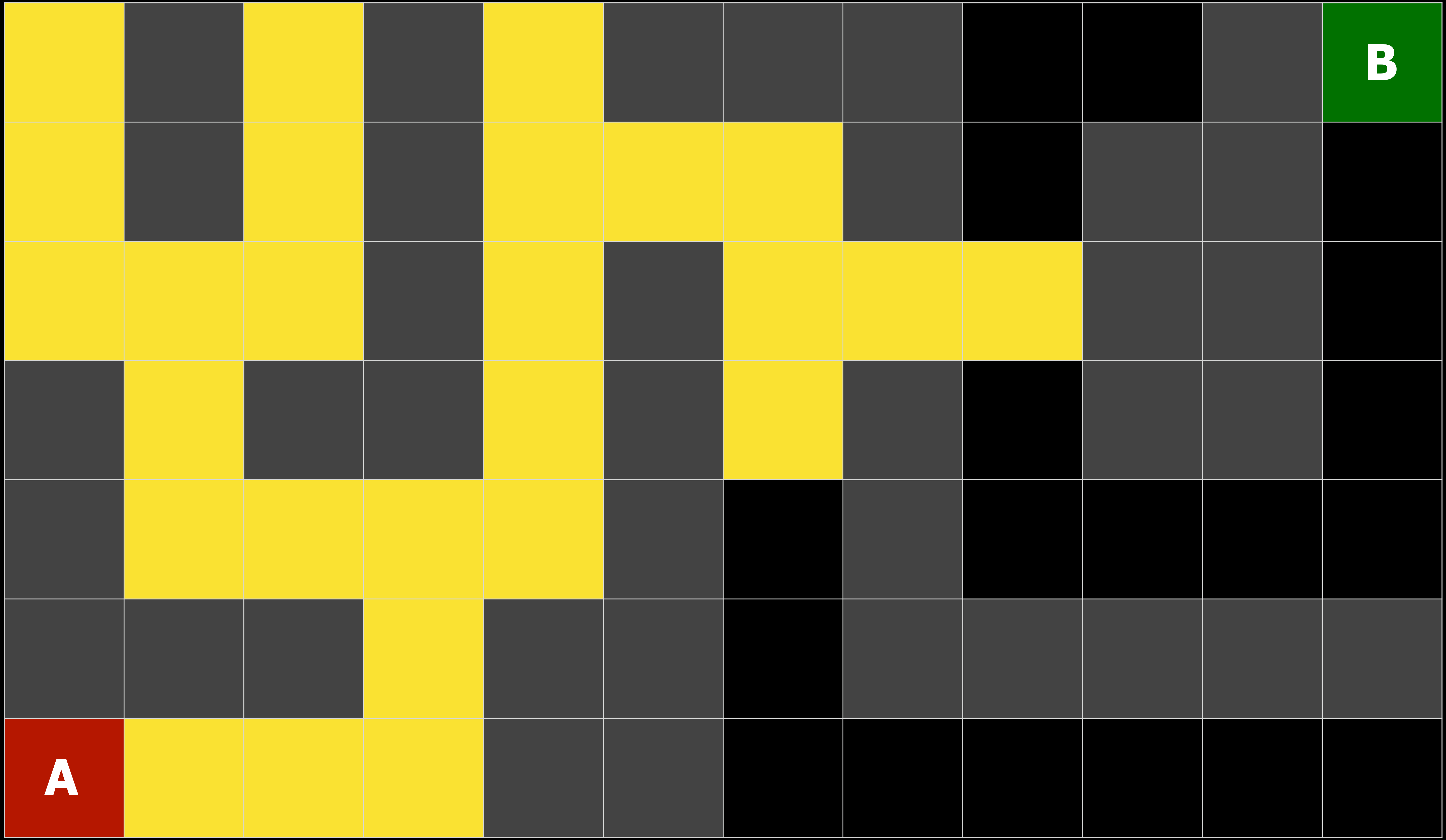
Optimization

optimization

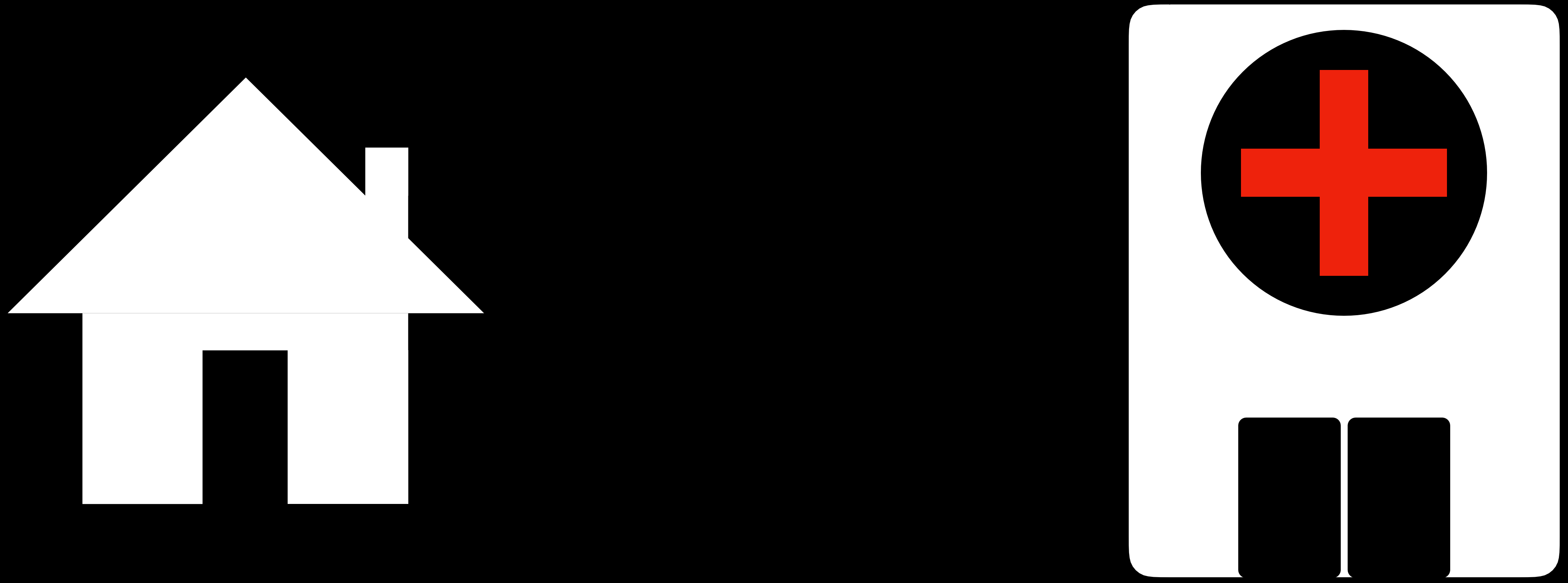
choosing the best option from a set of options

local search

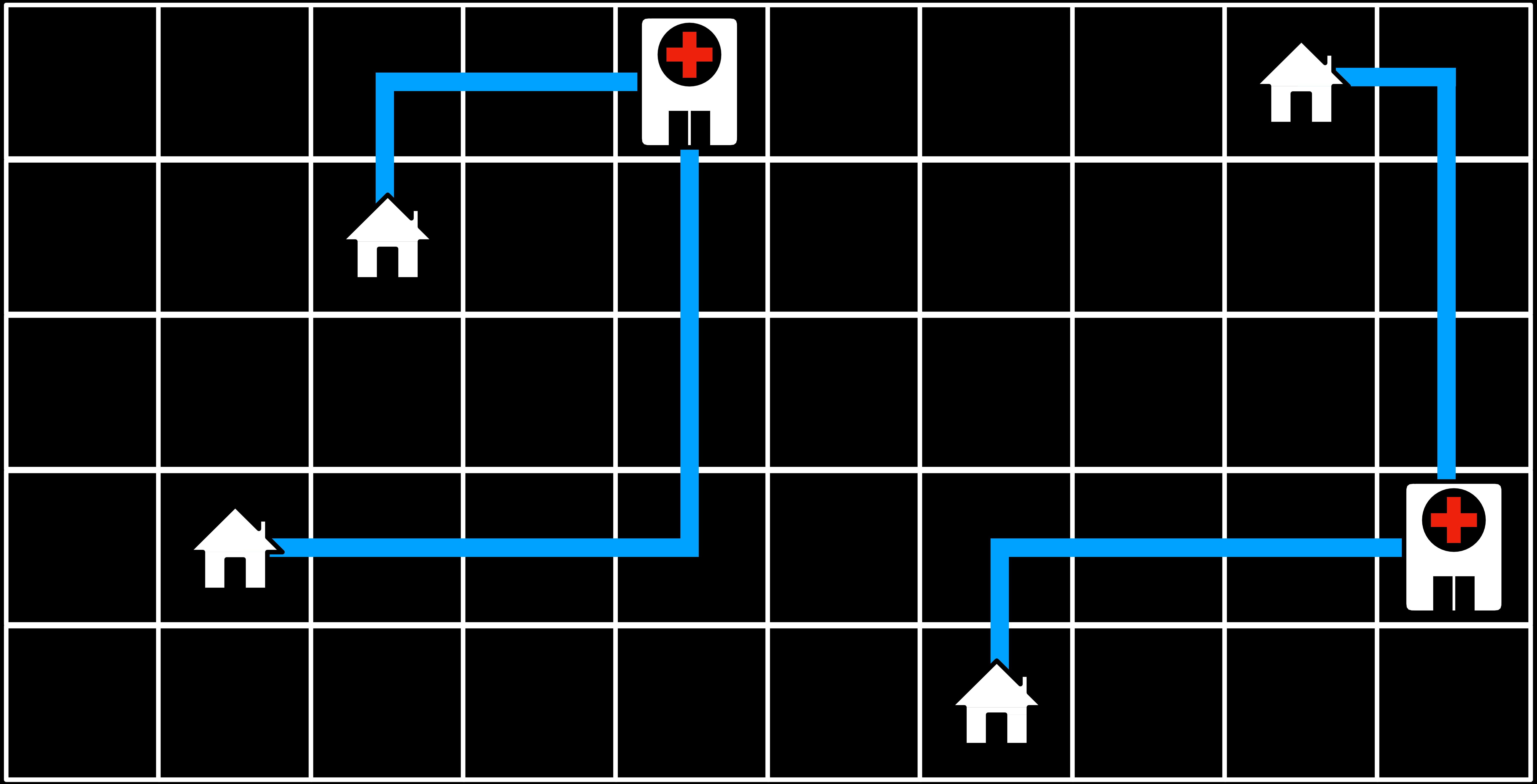
search algorithms that maintain a single node and searches by moving to a neighboring node


B
A

											B

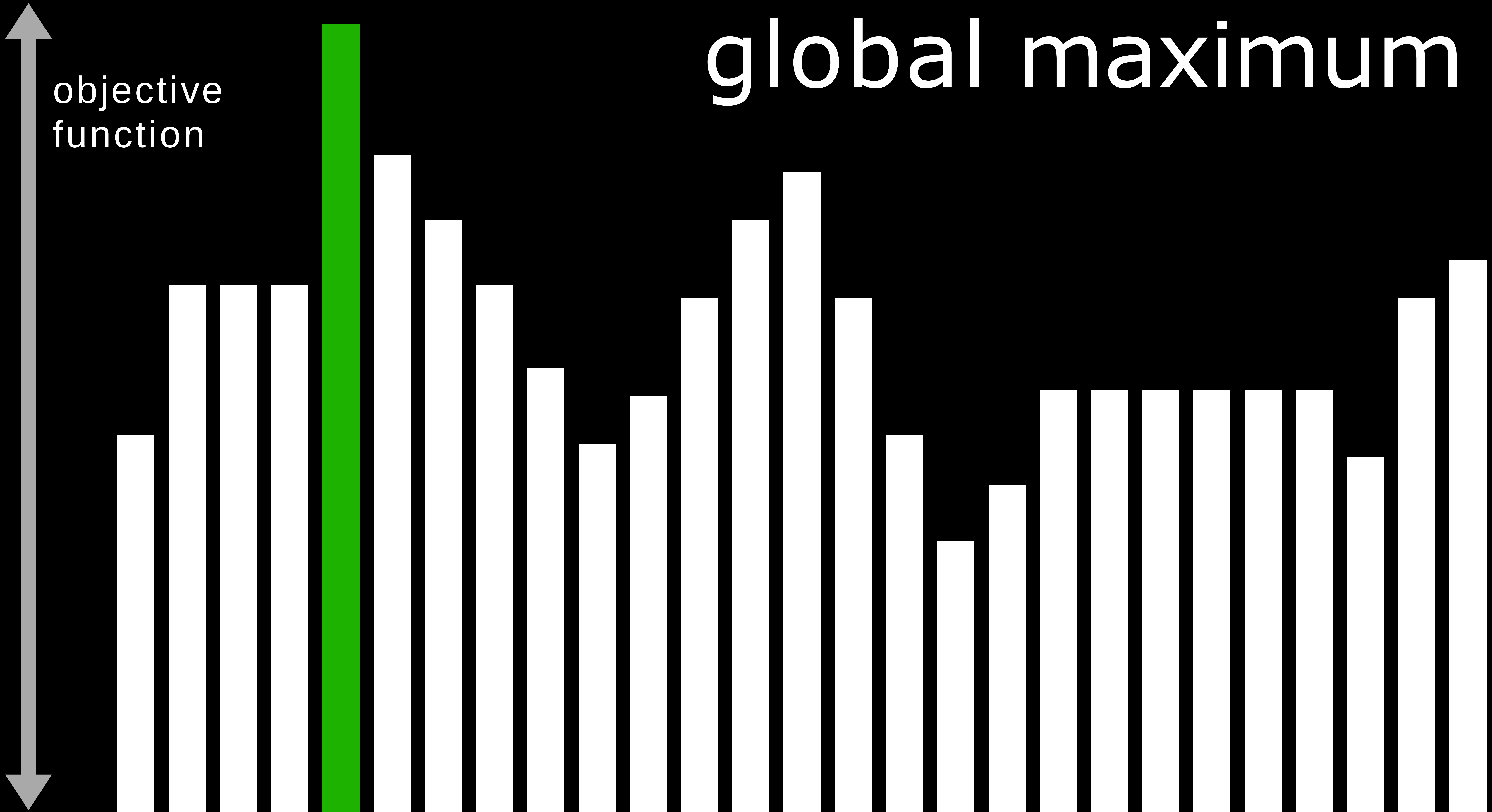


Cost: 17

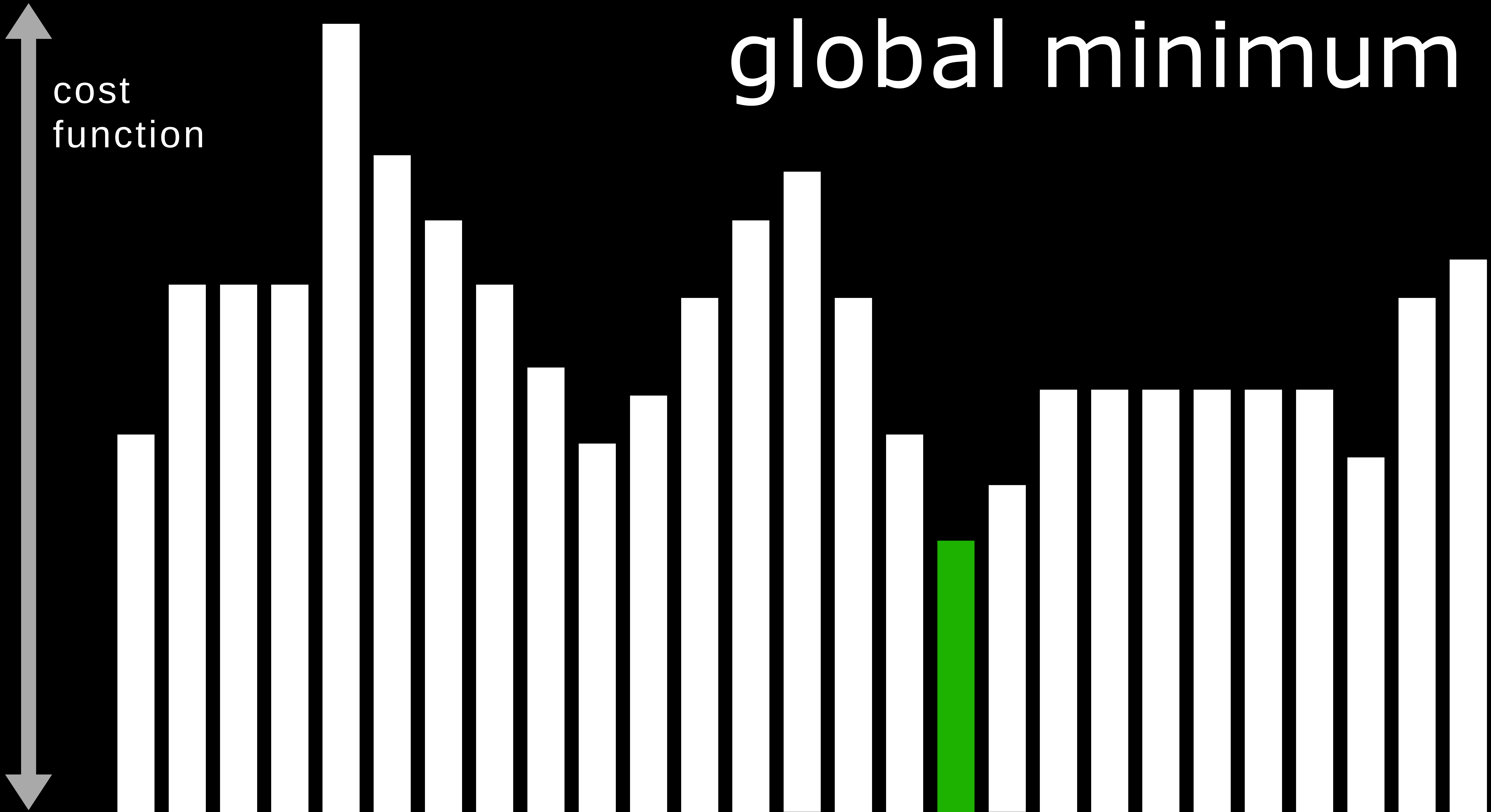


state-space landscape

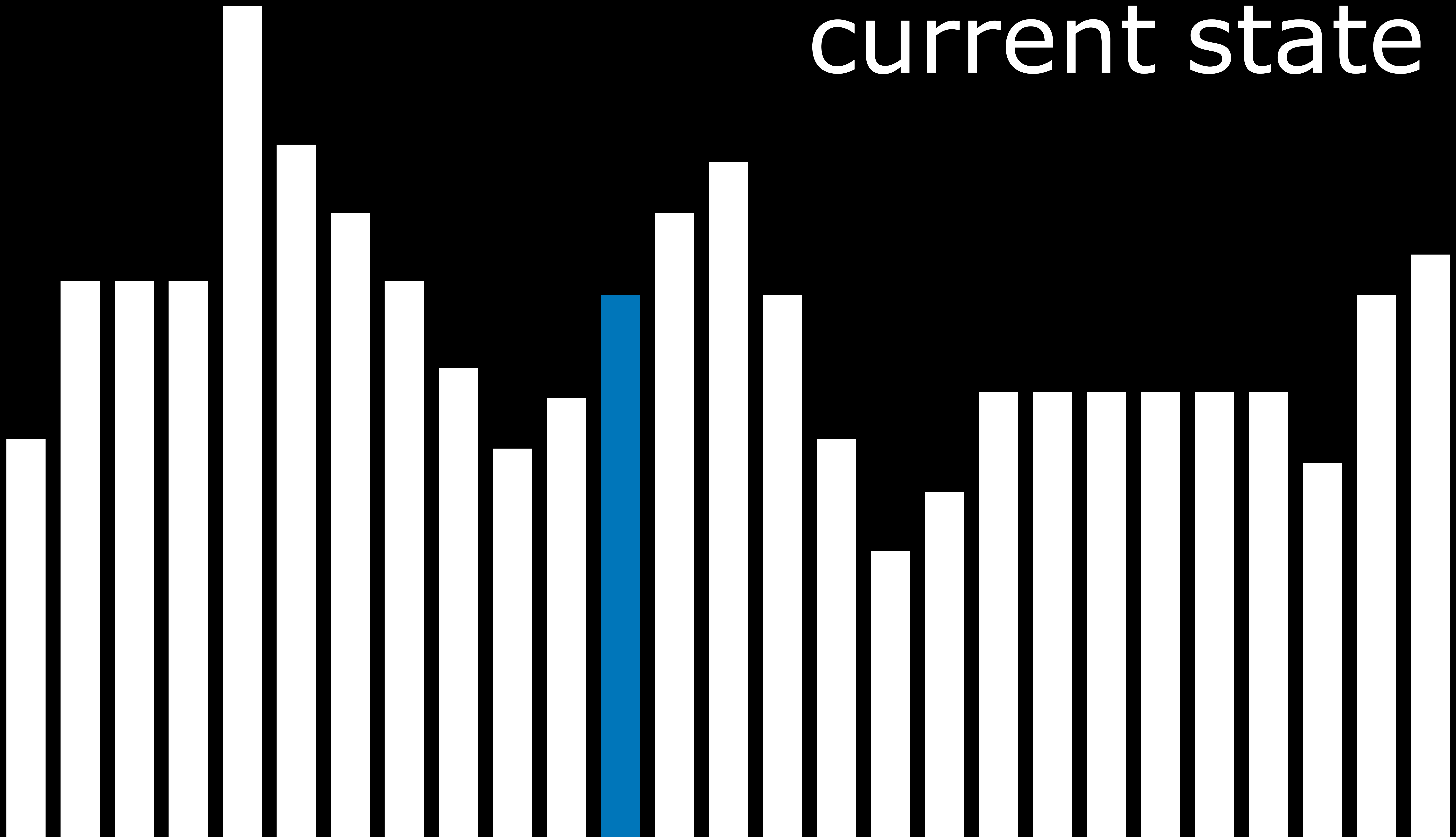
global maximum



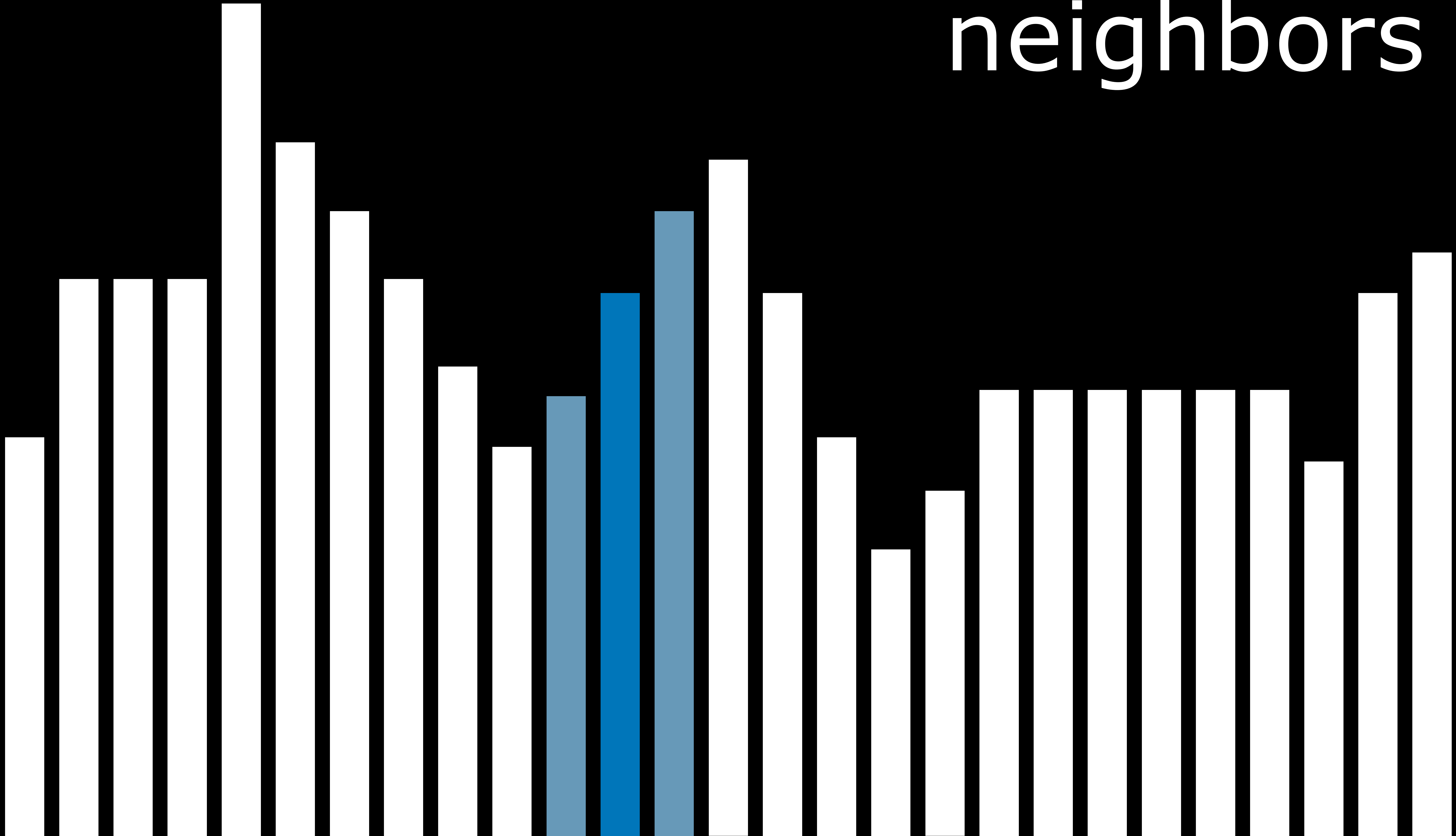
global minimum



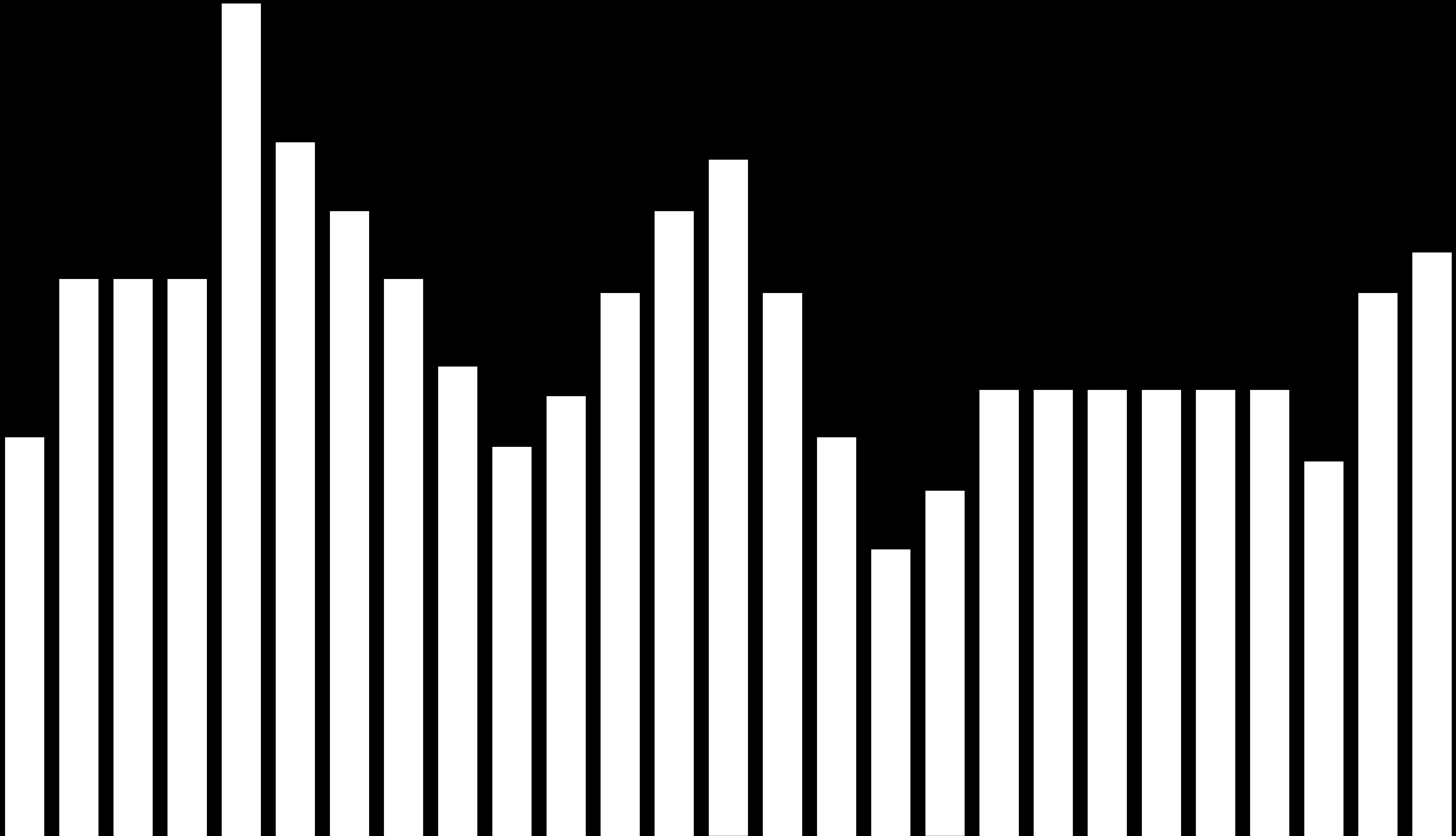
current state

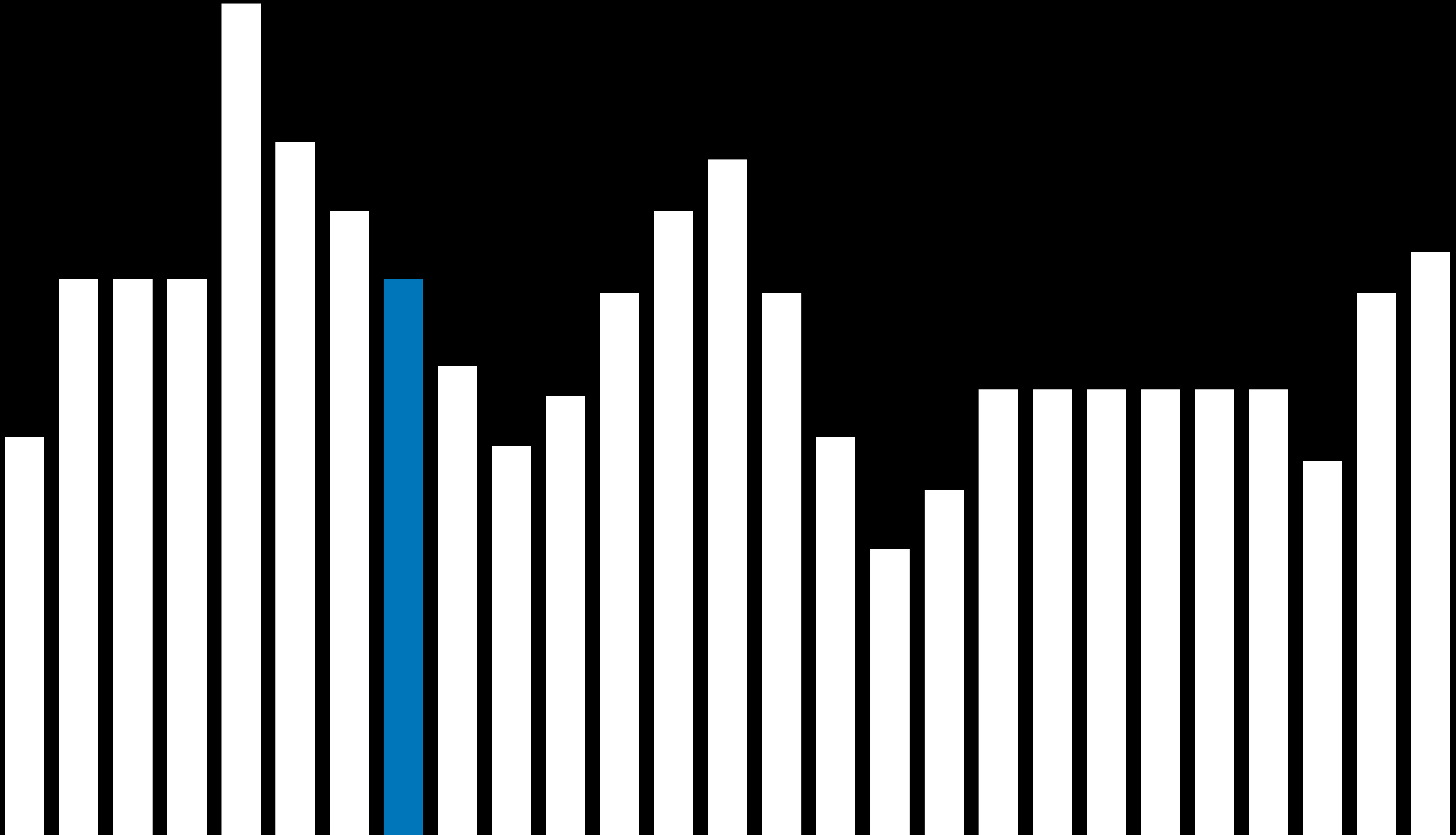


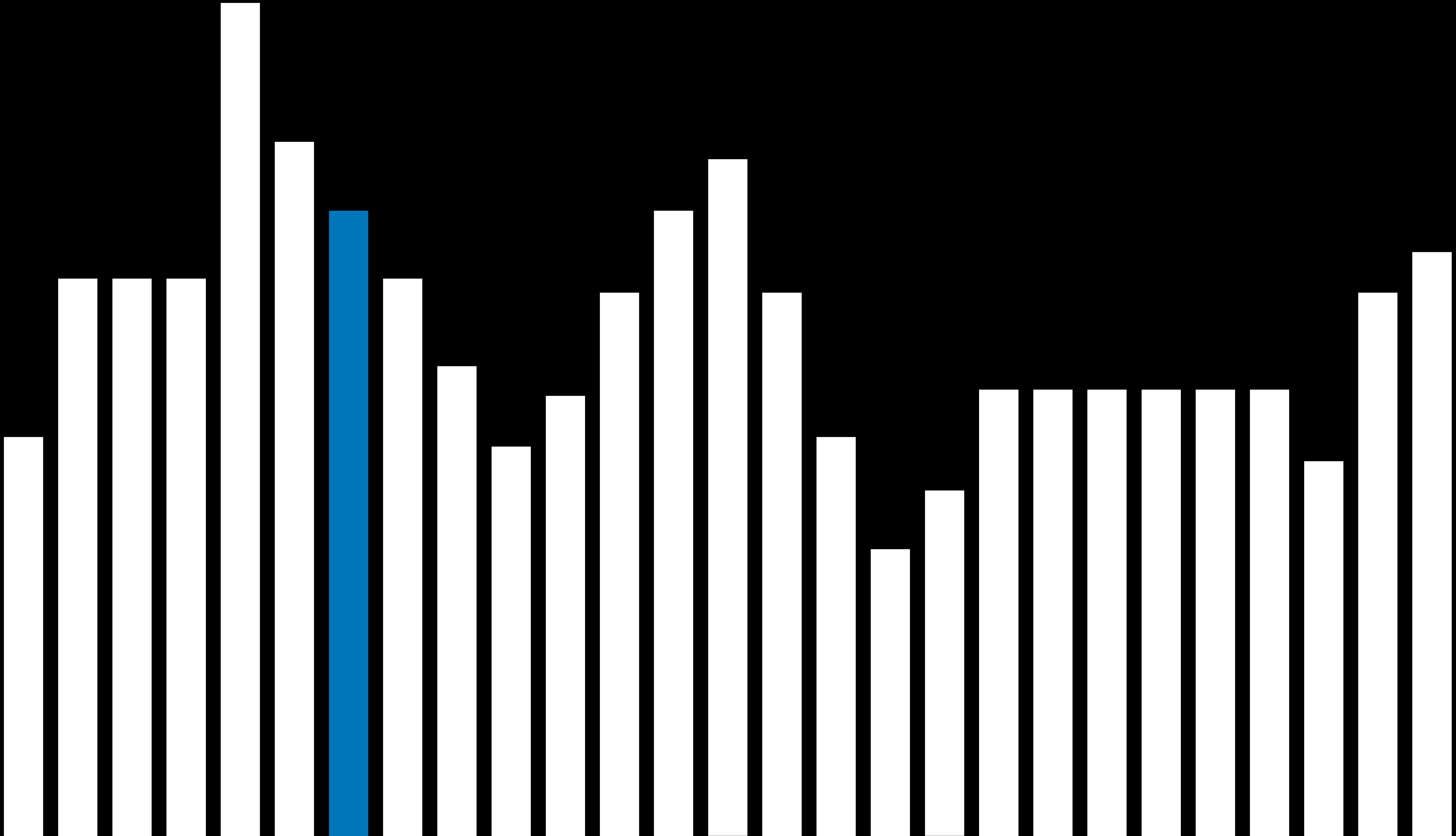
neighbors

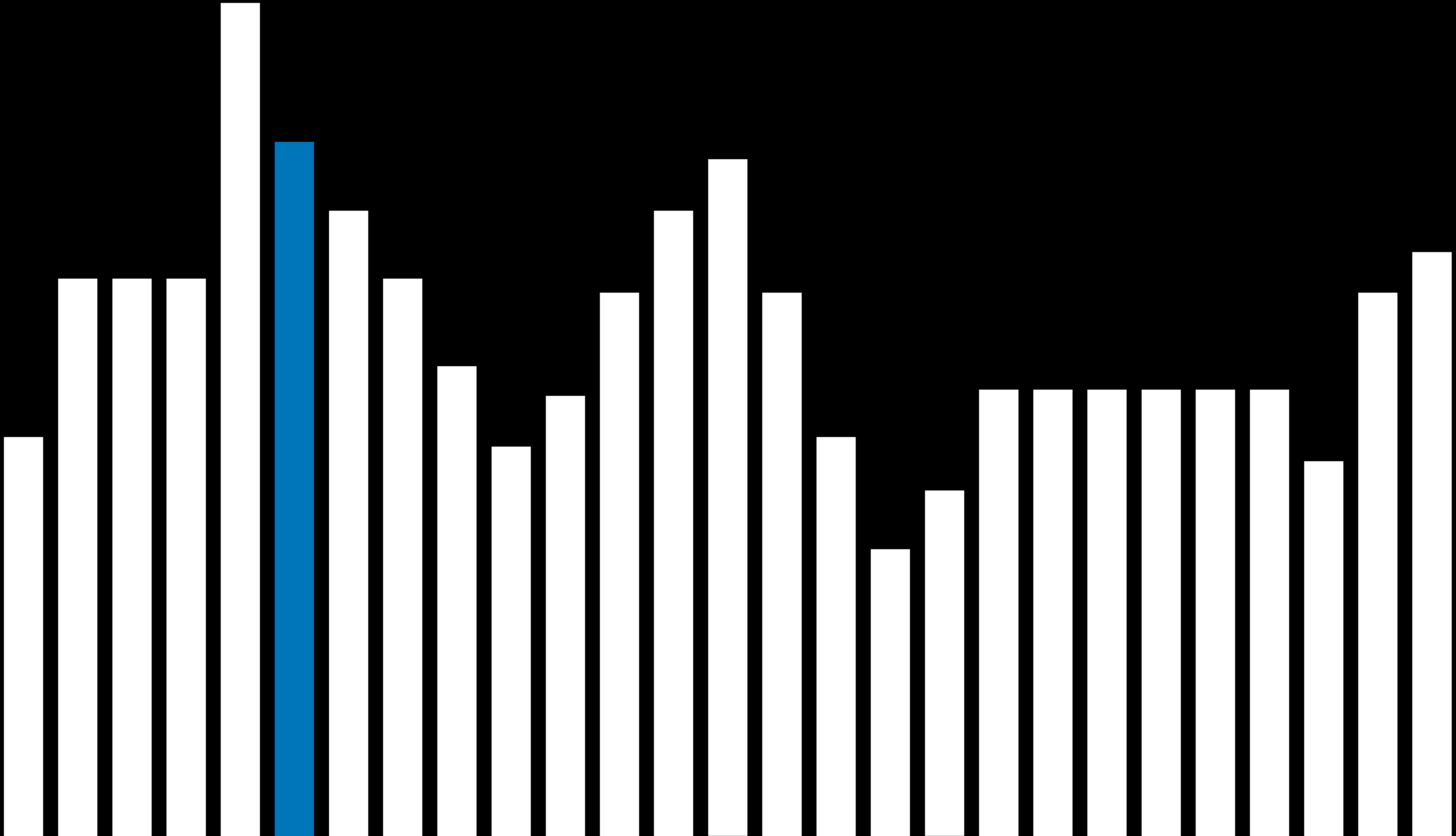


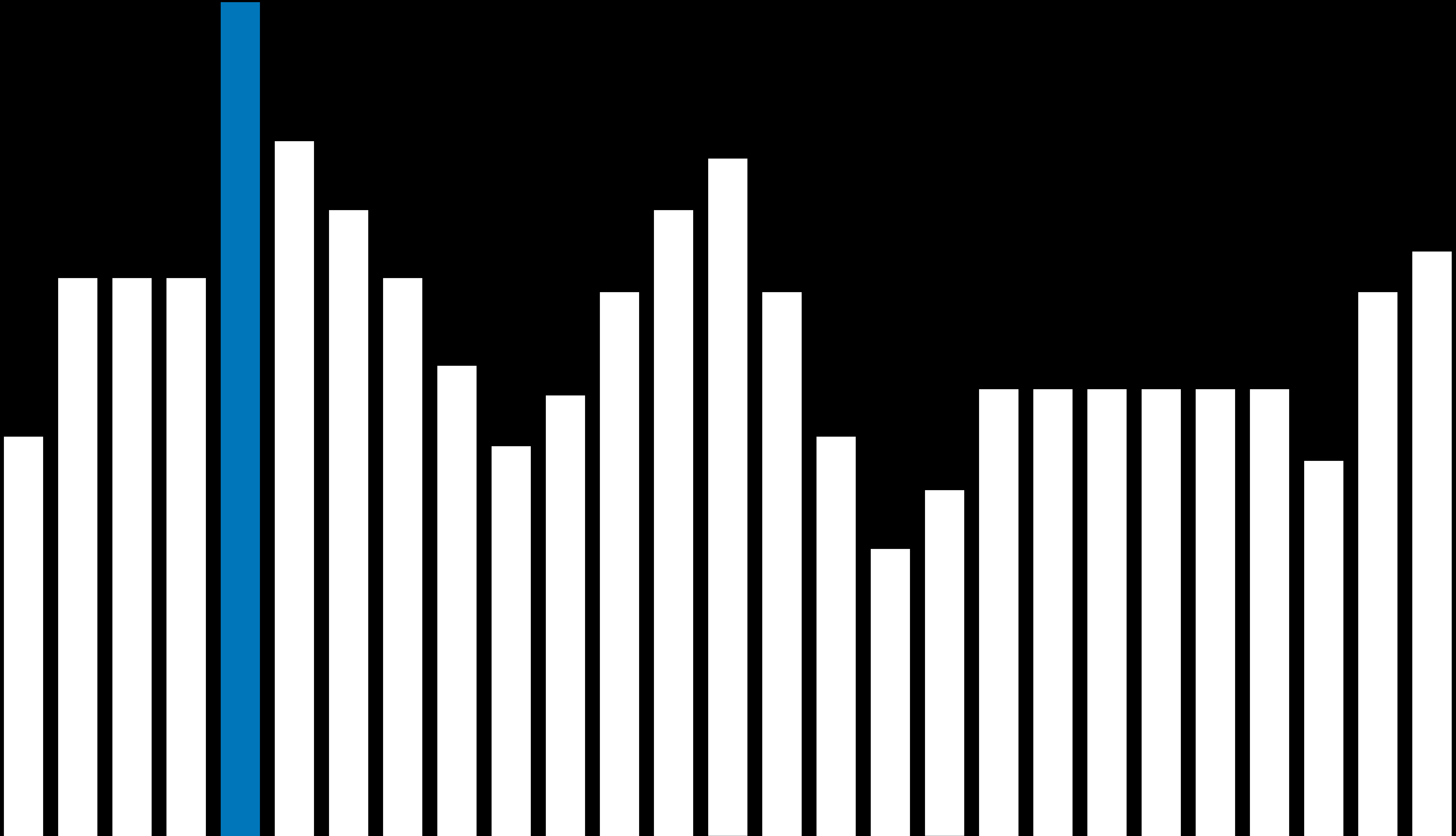
Hill Climbing

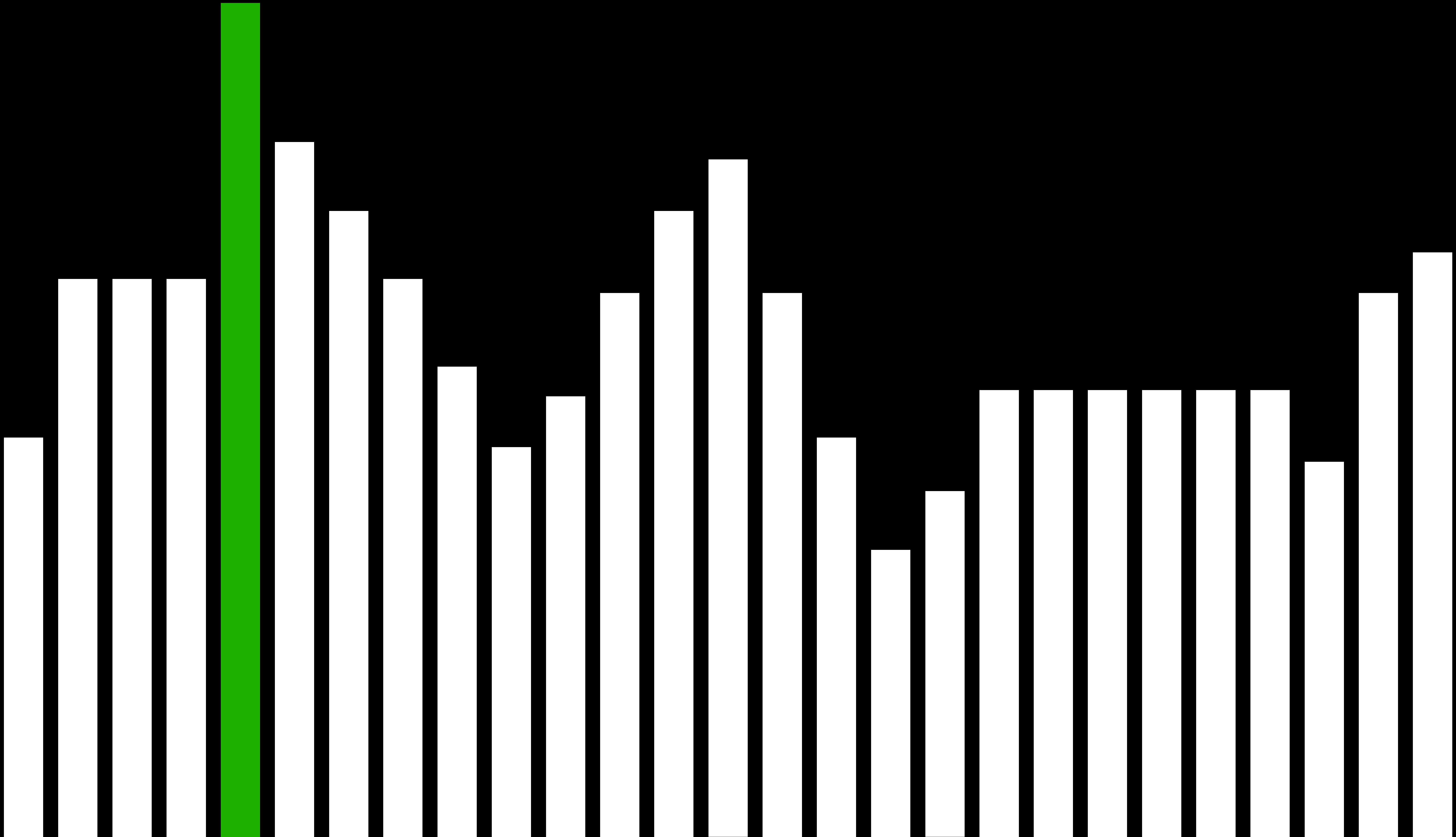


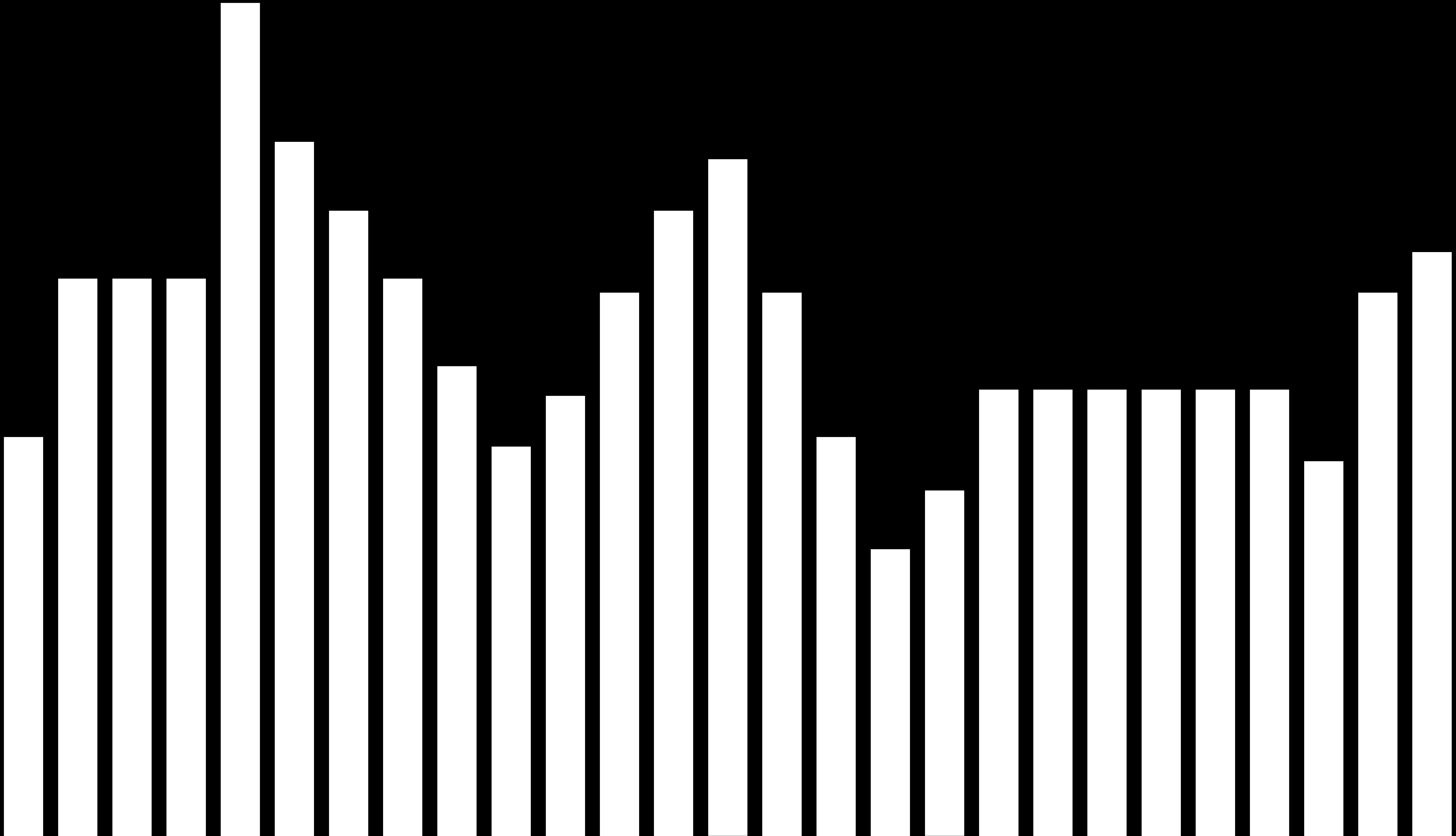


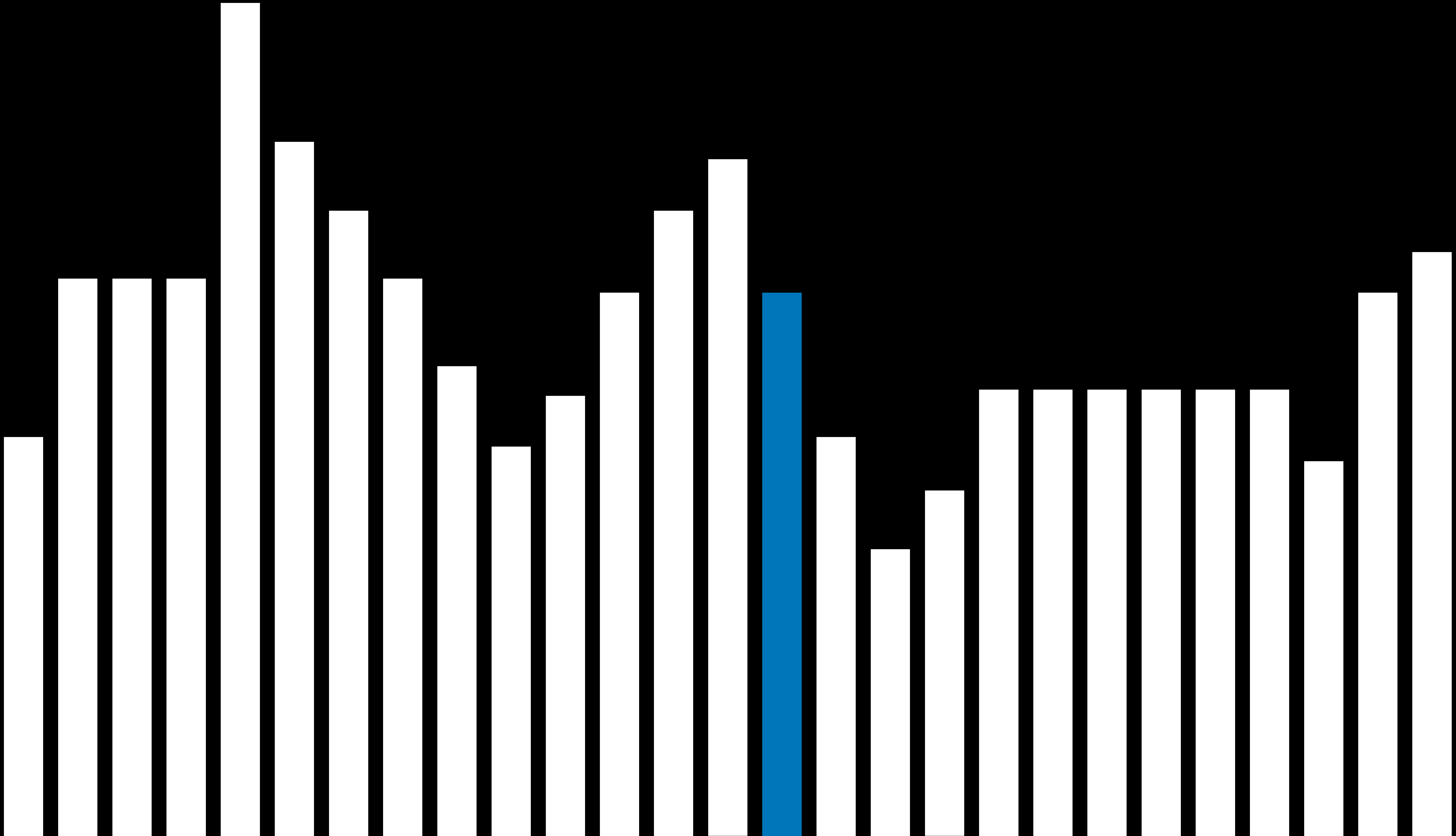


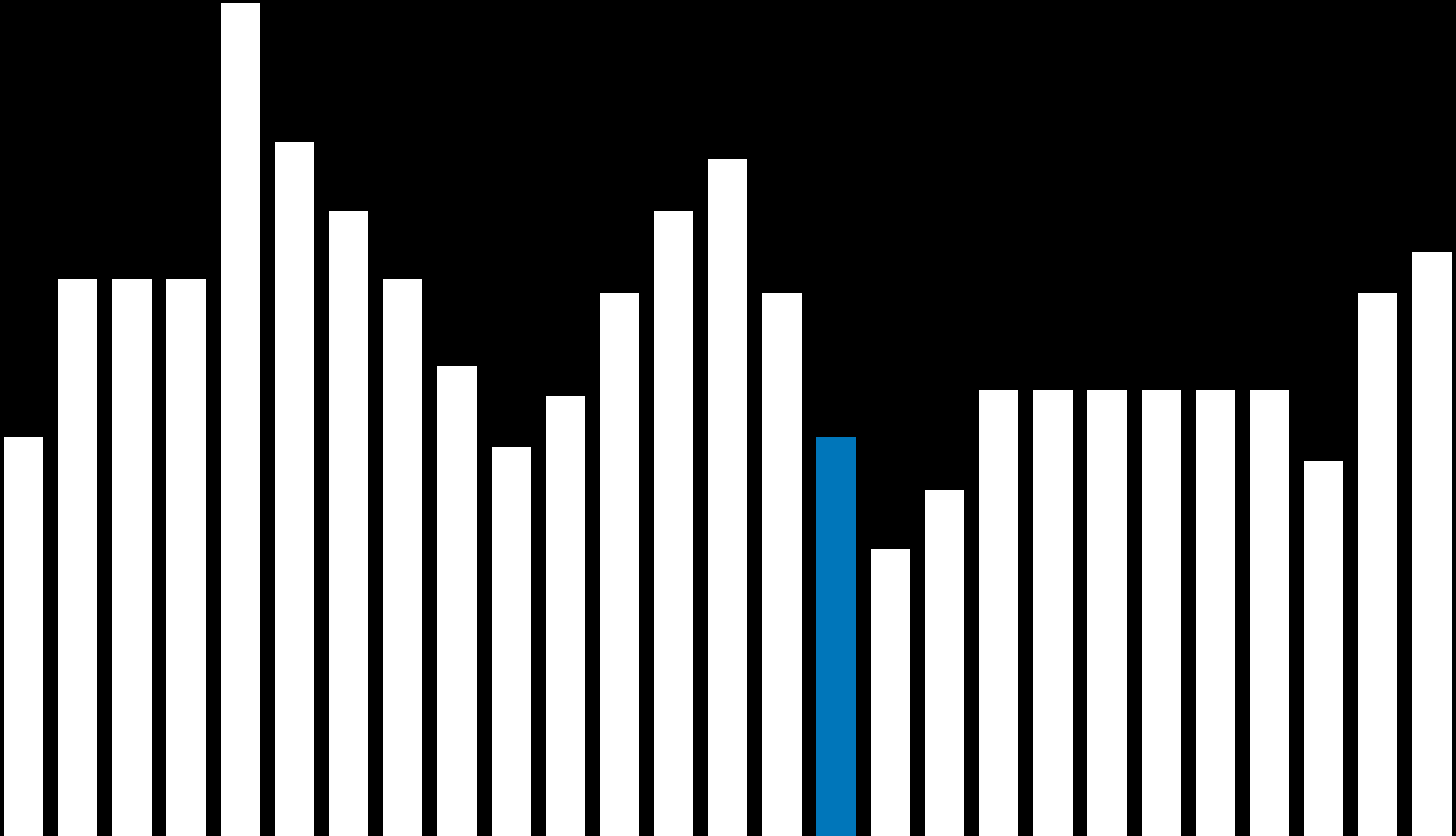


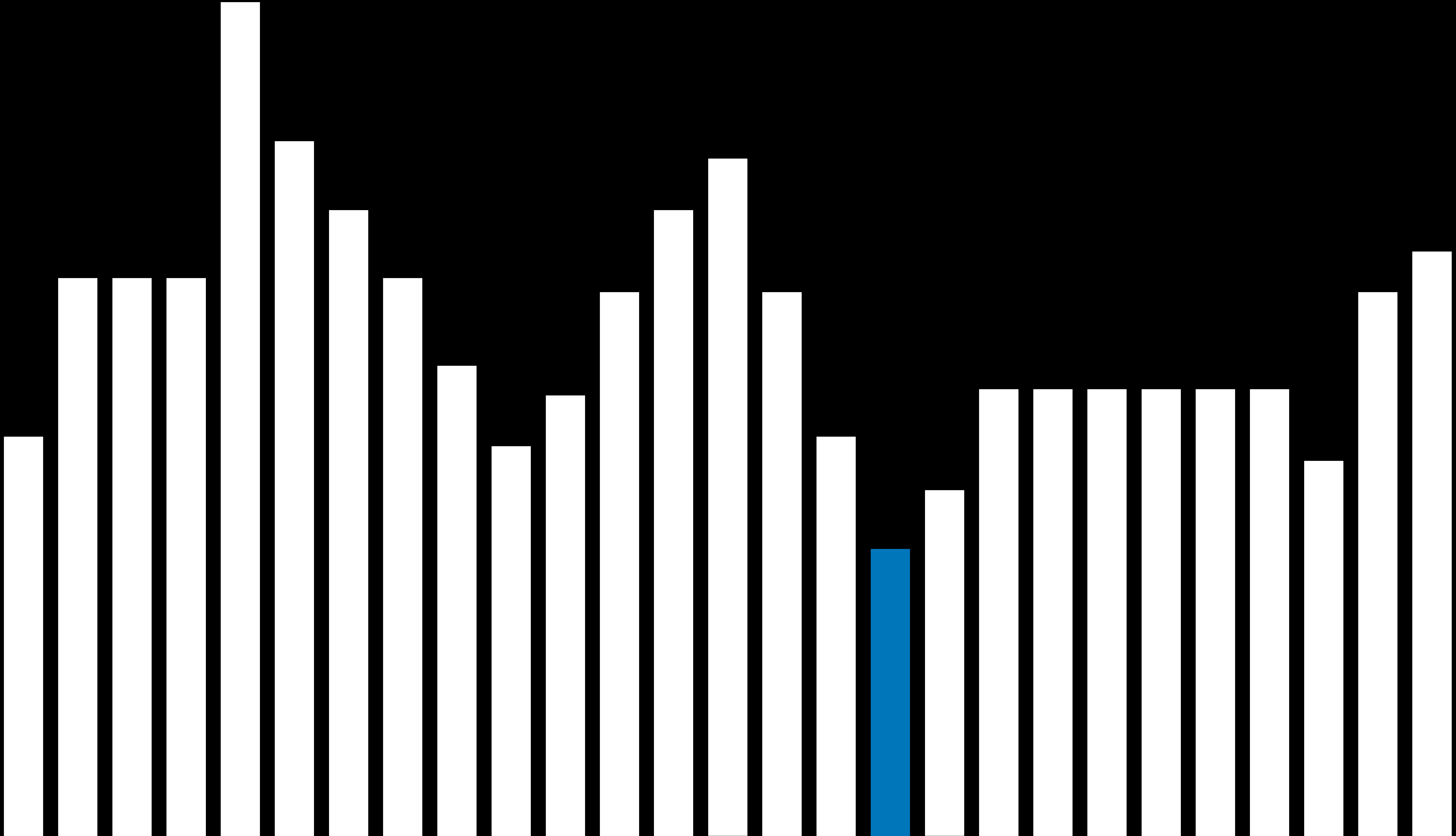


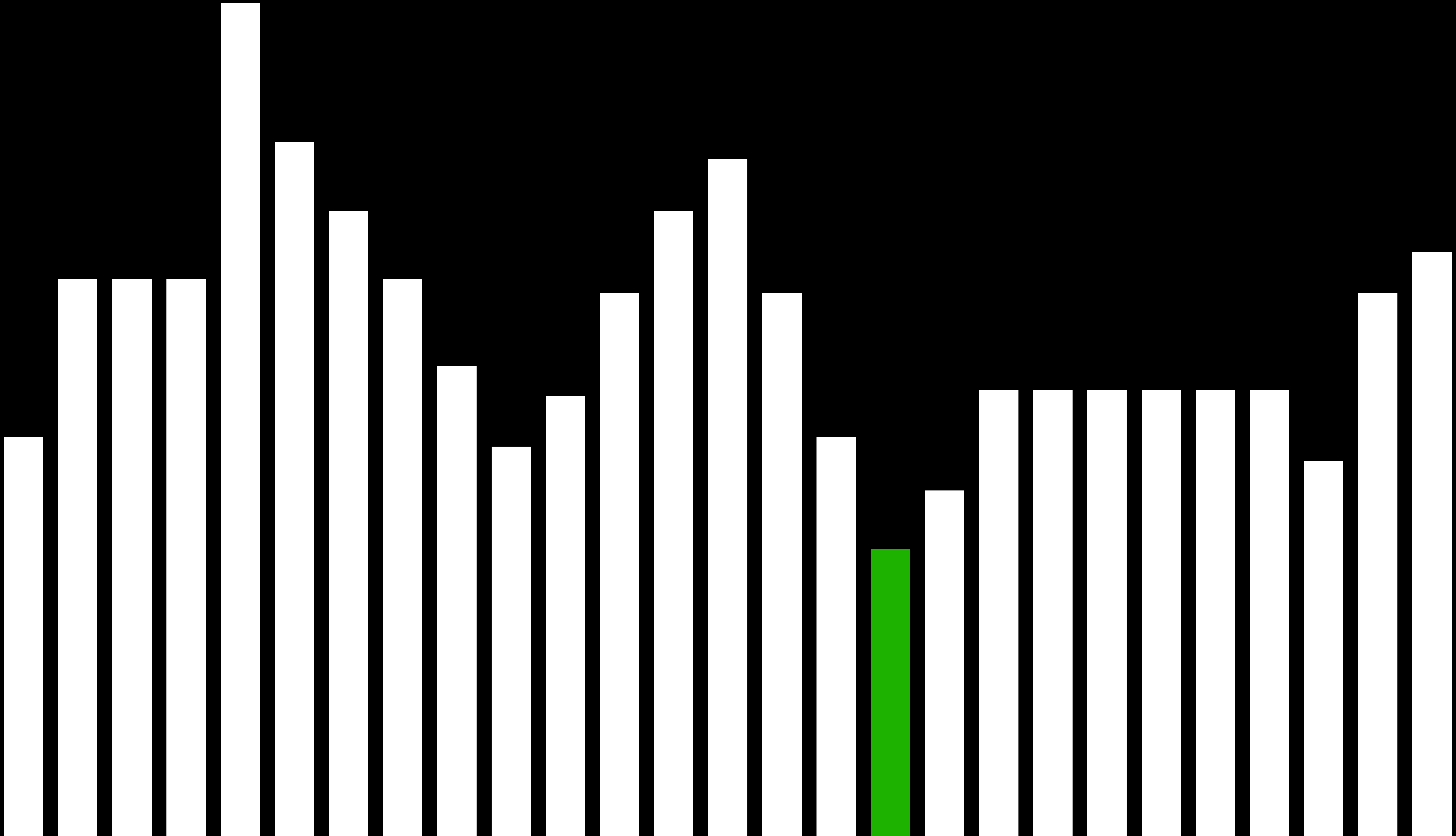








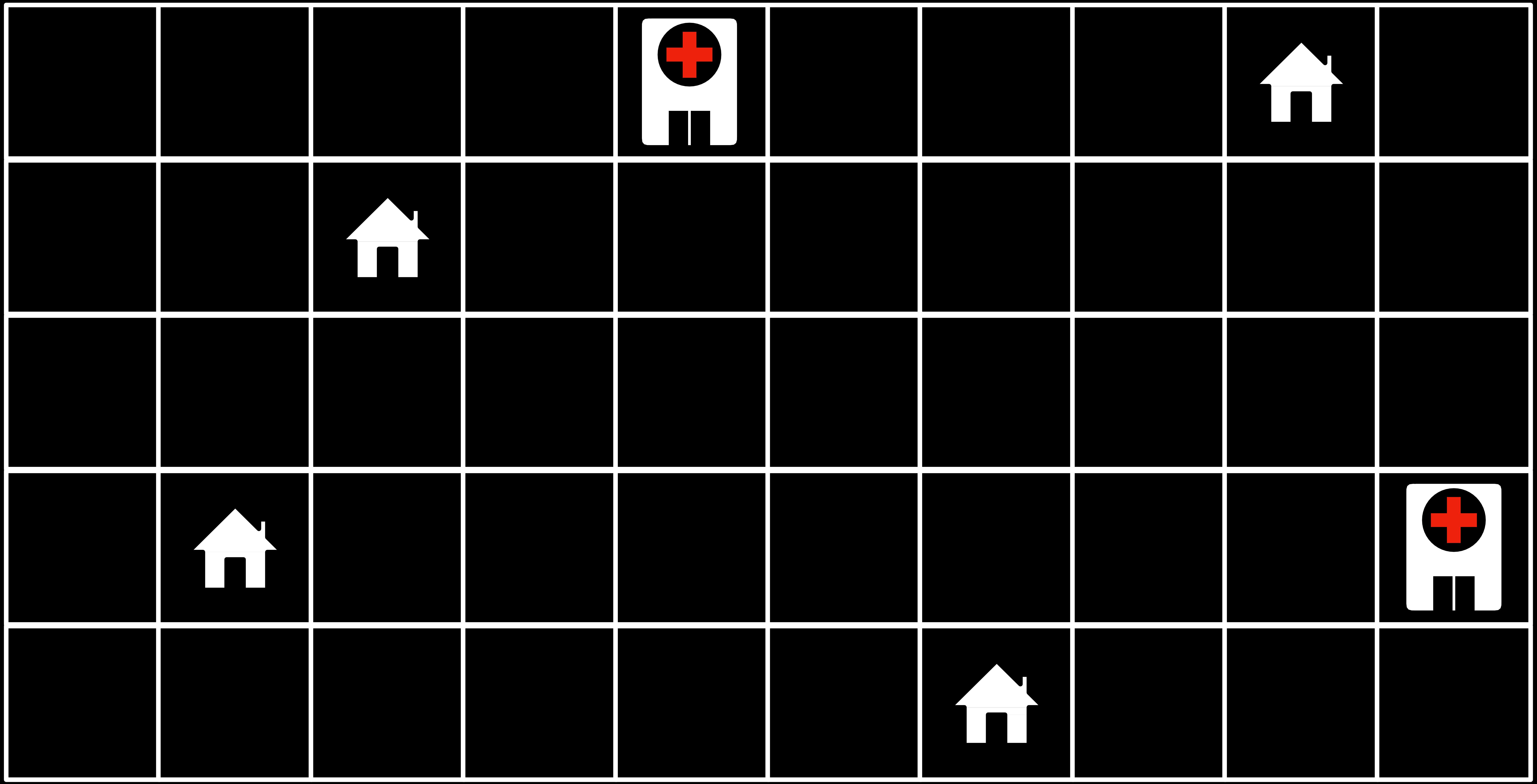




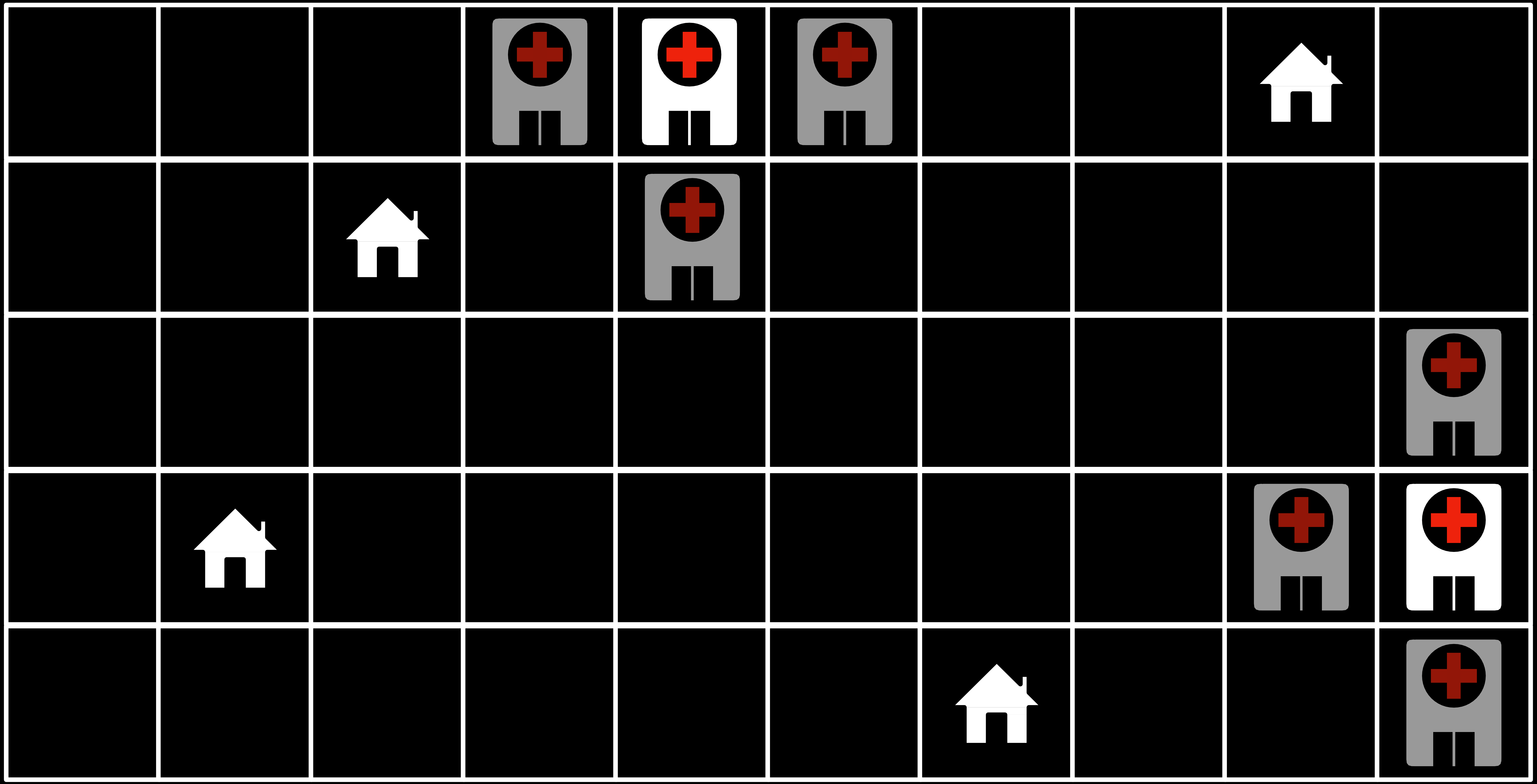
Hill Climbing

```
function HILL-CLIMB(problem):
    current = initial state of problem
    repeat:
        neighbor = highest valued neighbor of current
        if neighbor not better than current:
            return current
        current = neighbor
```

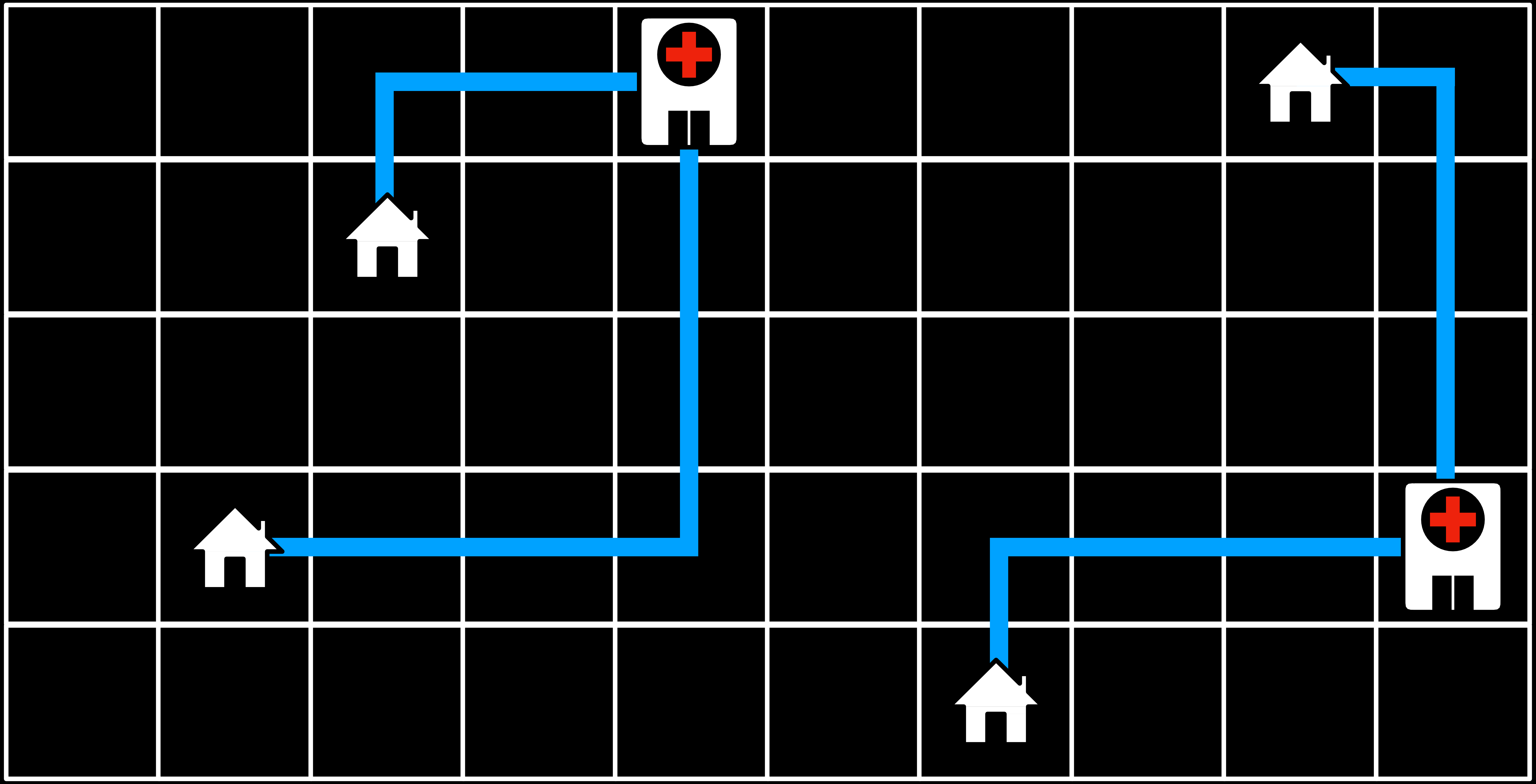
Cost: 17



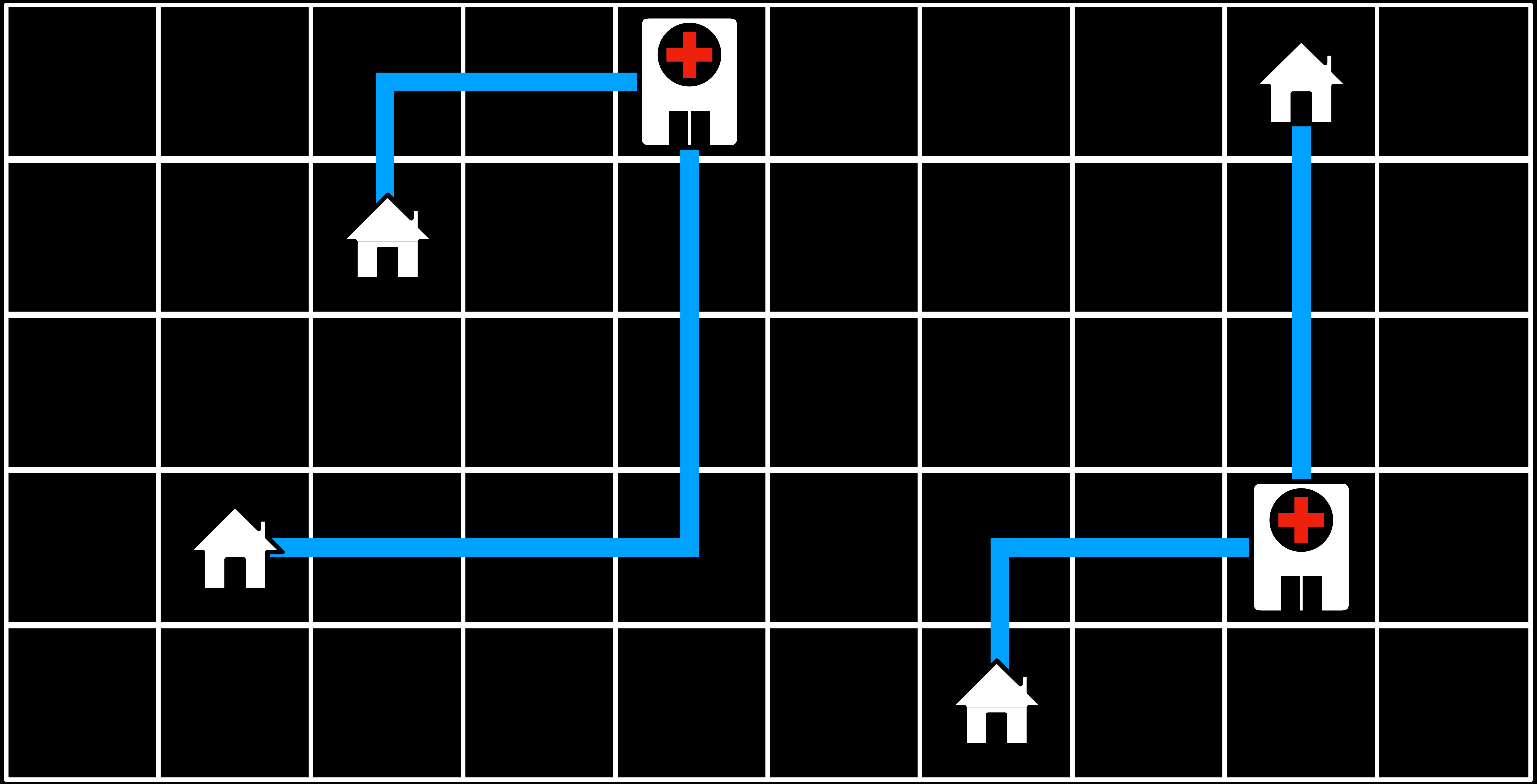
Cost: 17



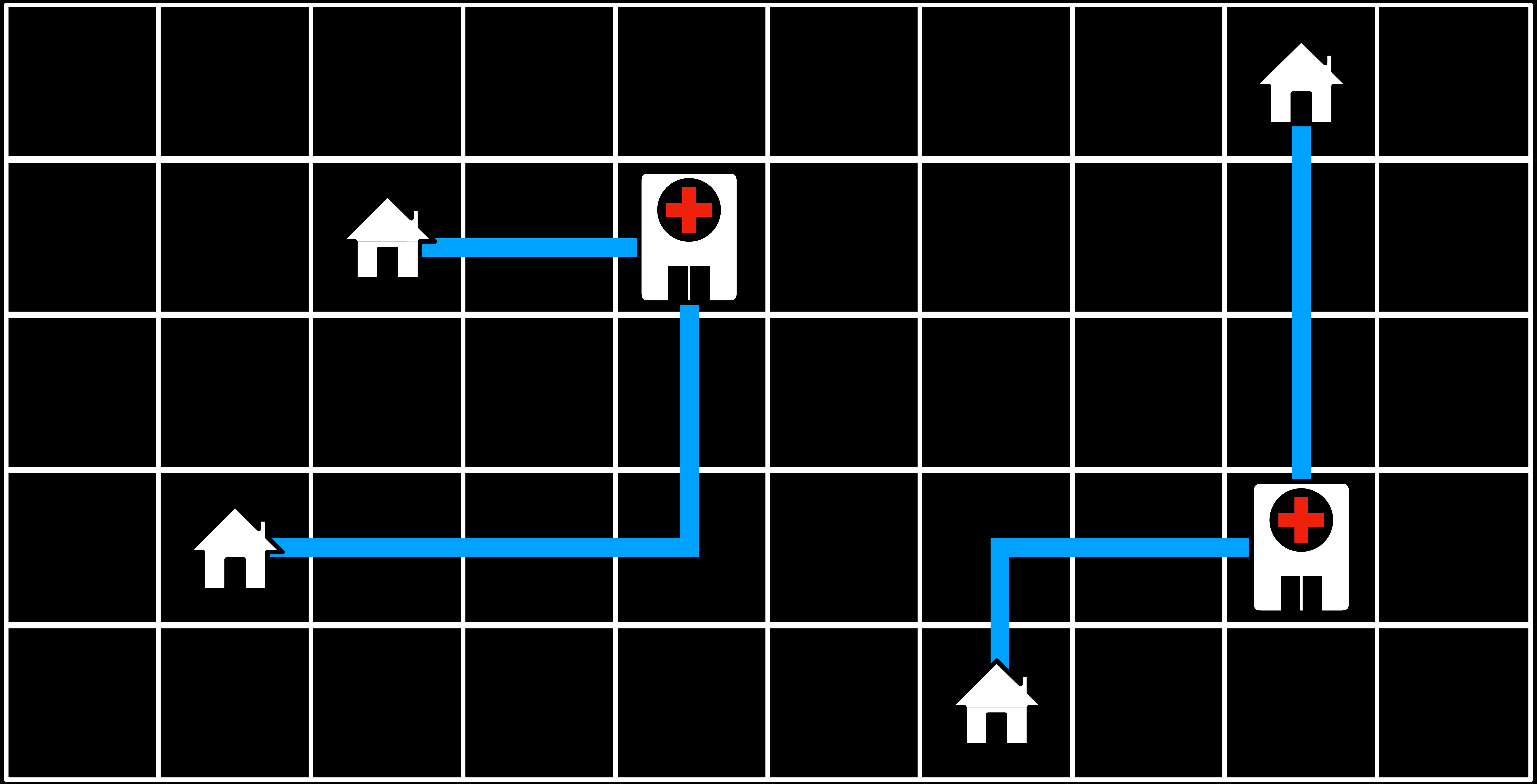
Cost: 17



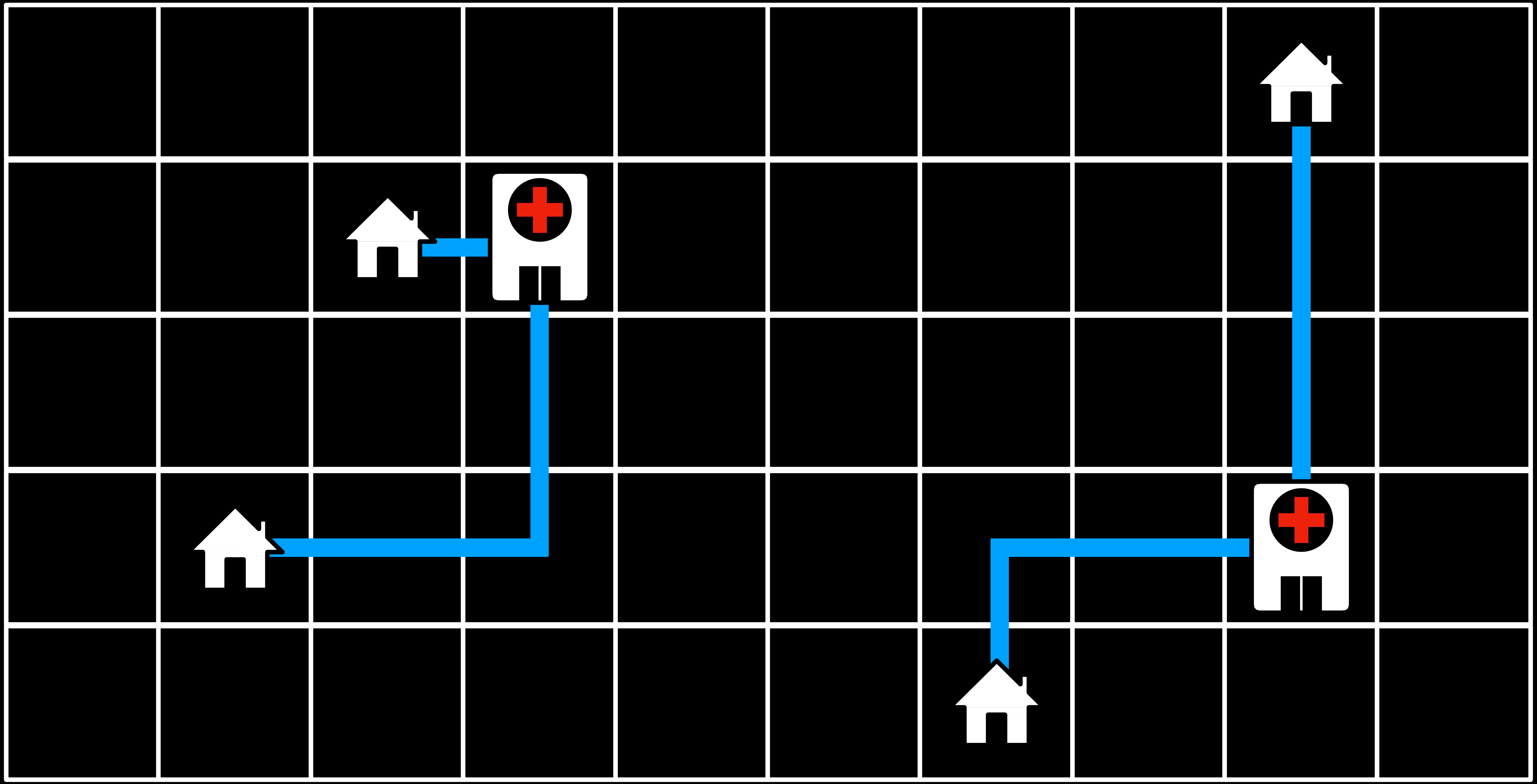
Cost: 15



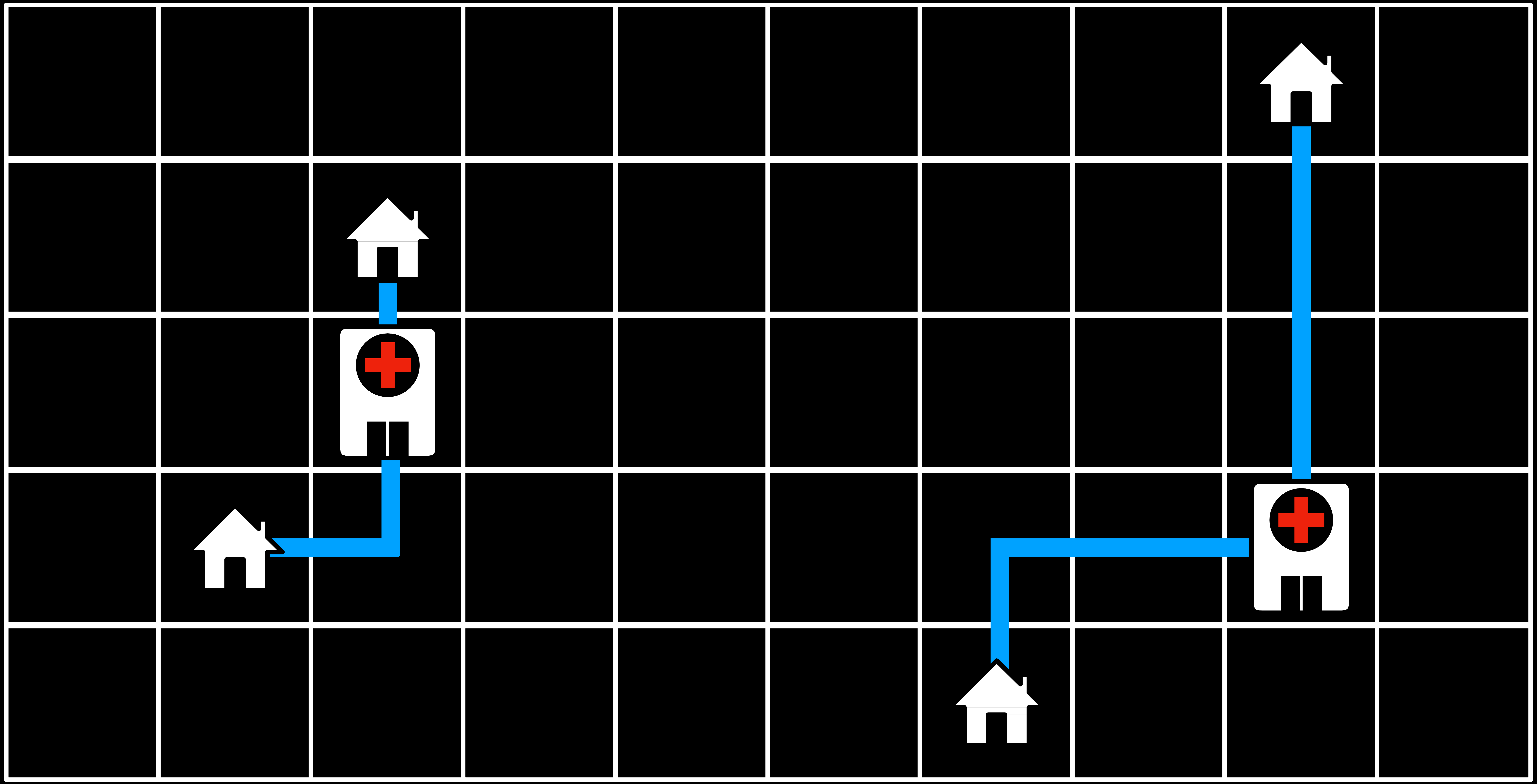
Cost: 13

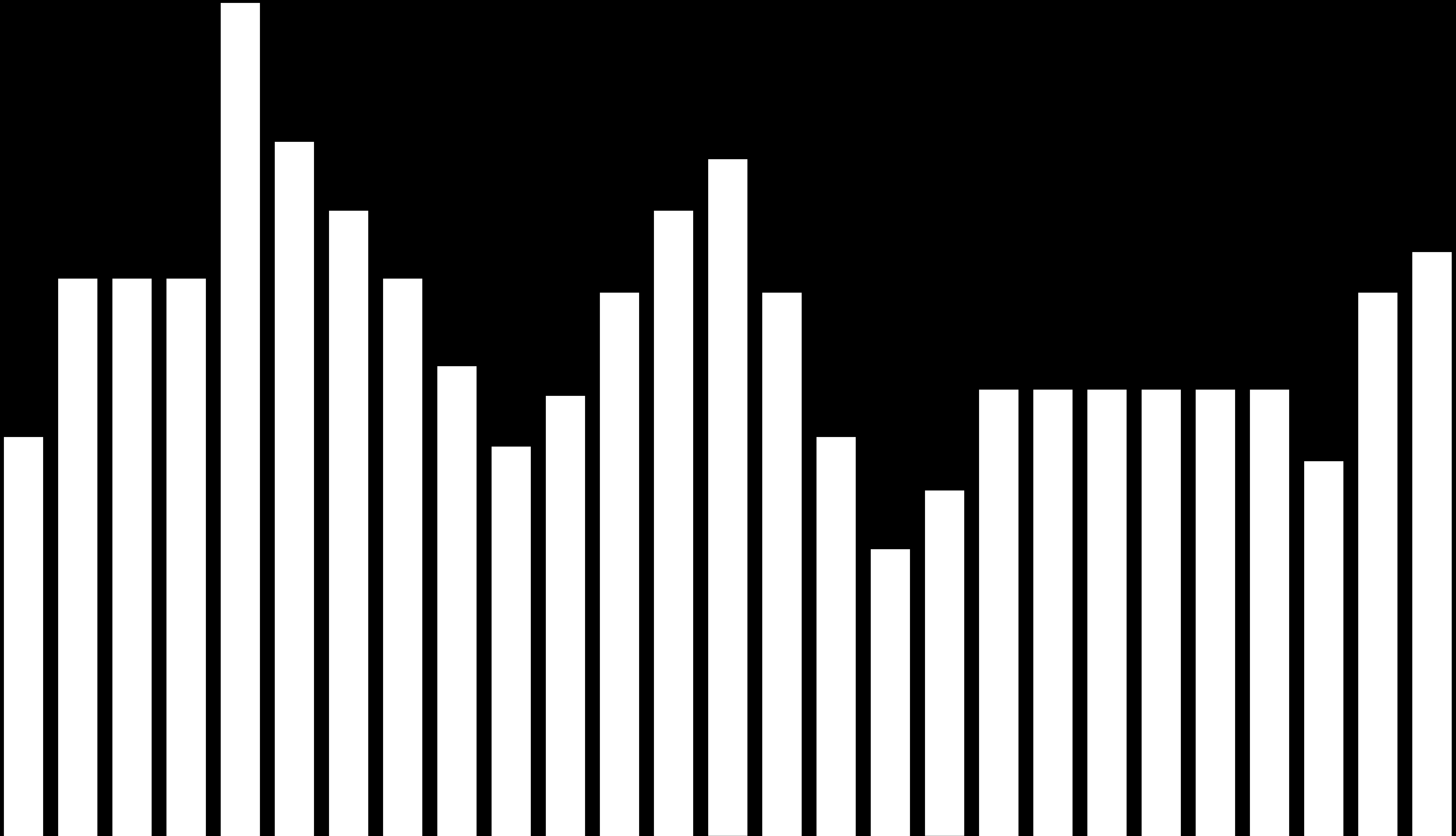


Cost: 11

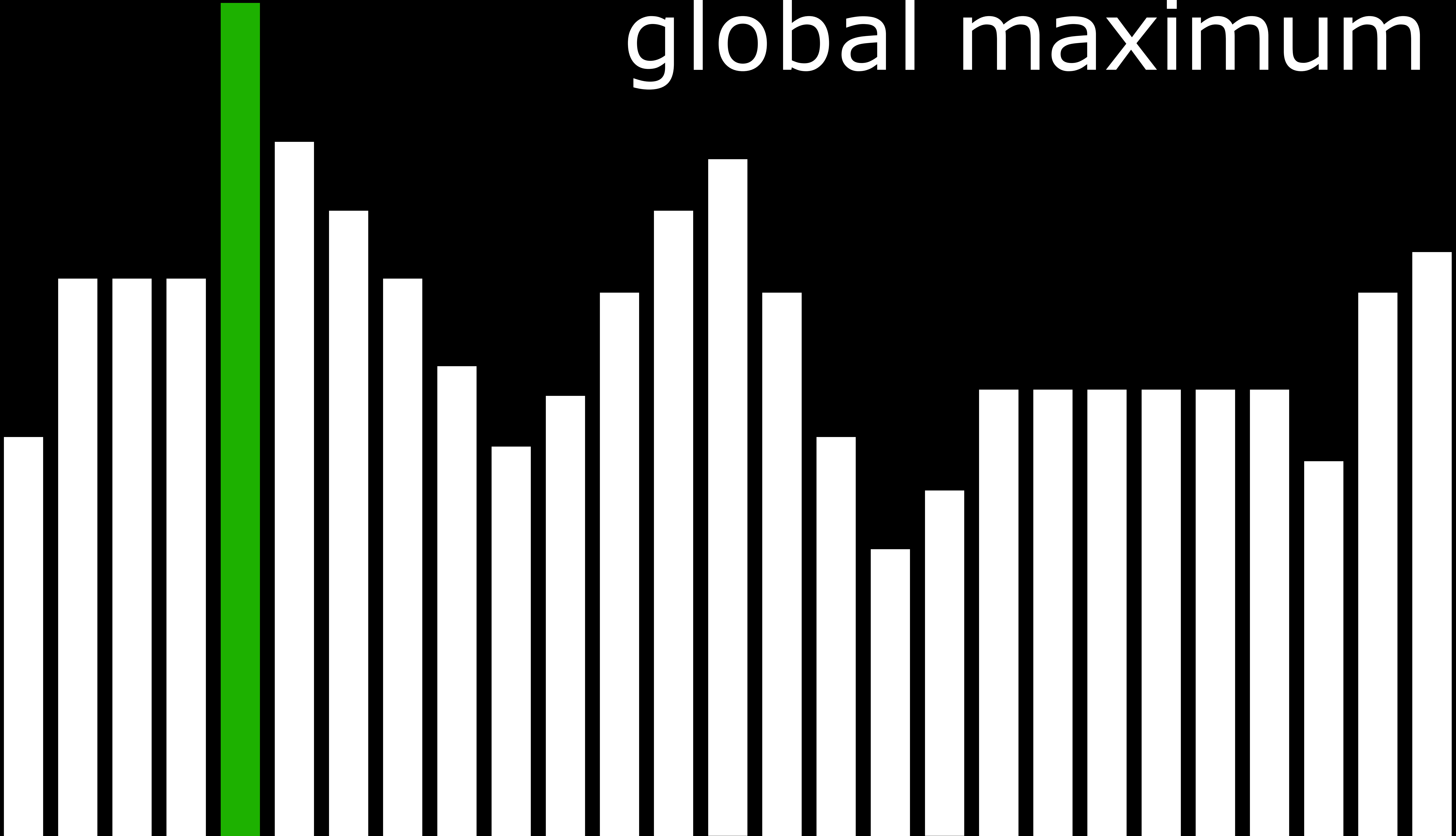


Cost: 9

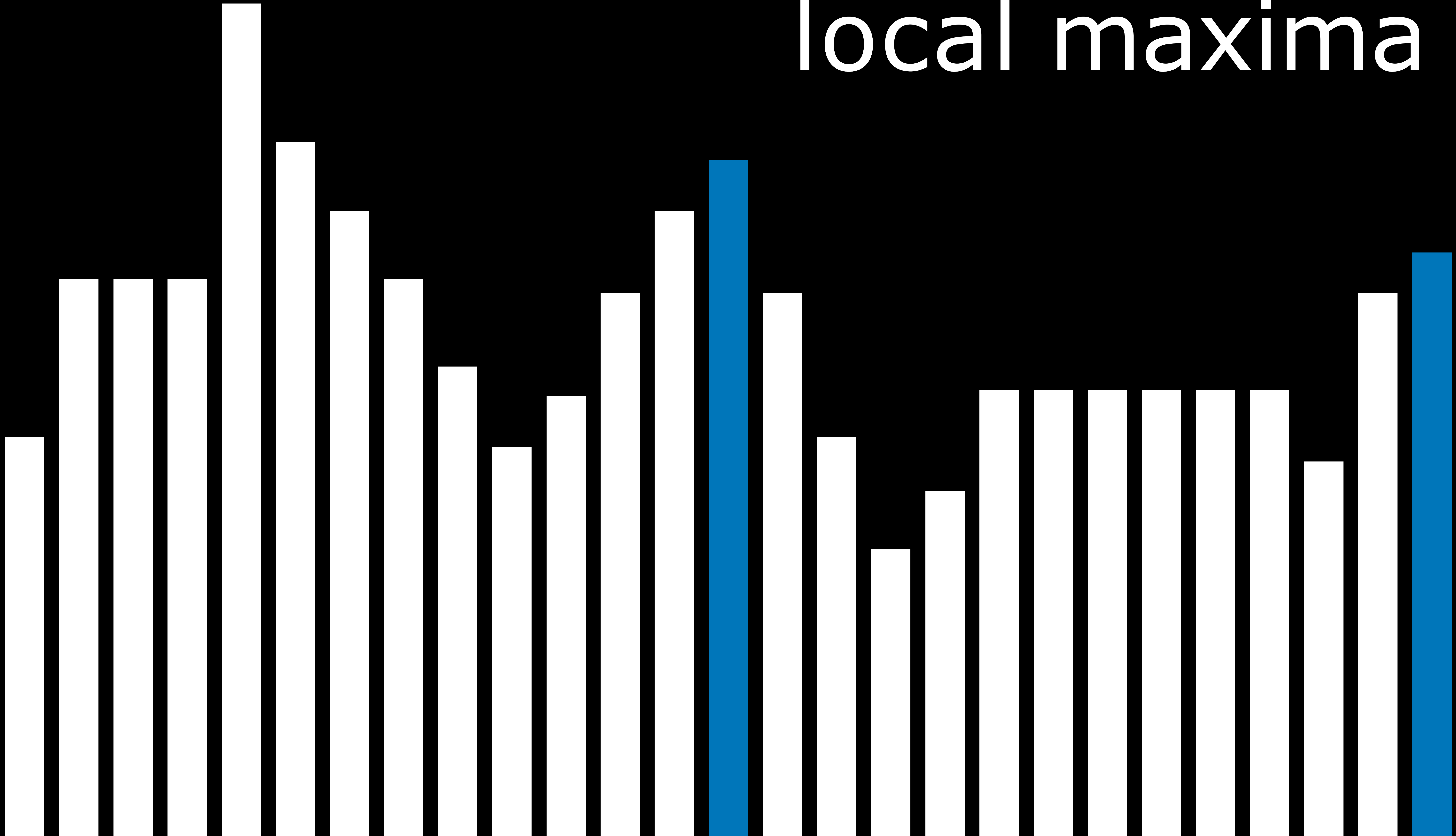




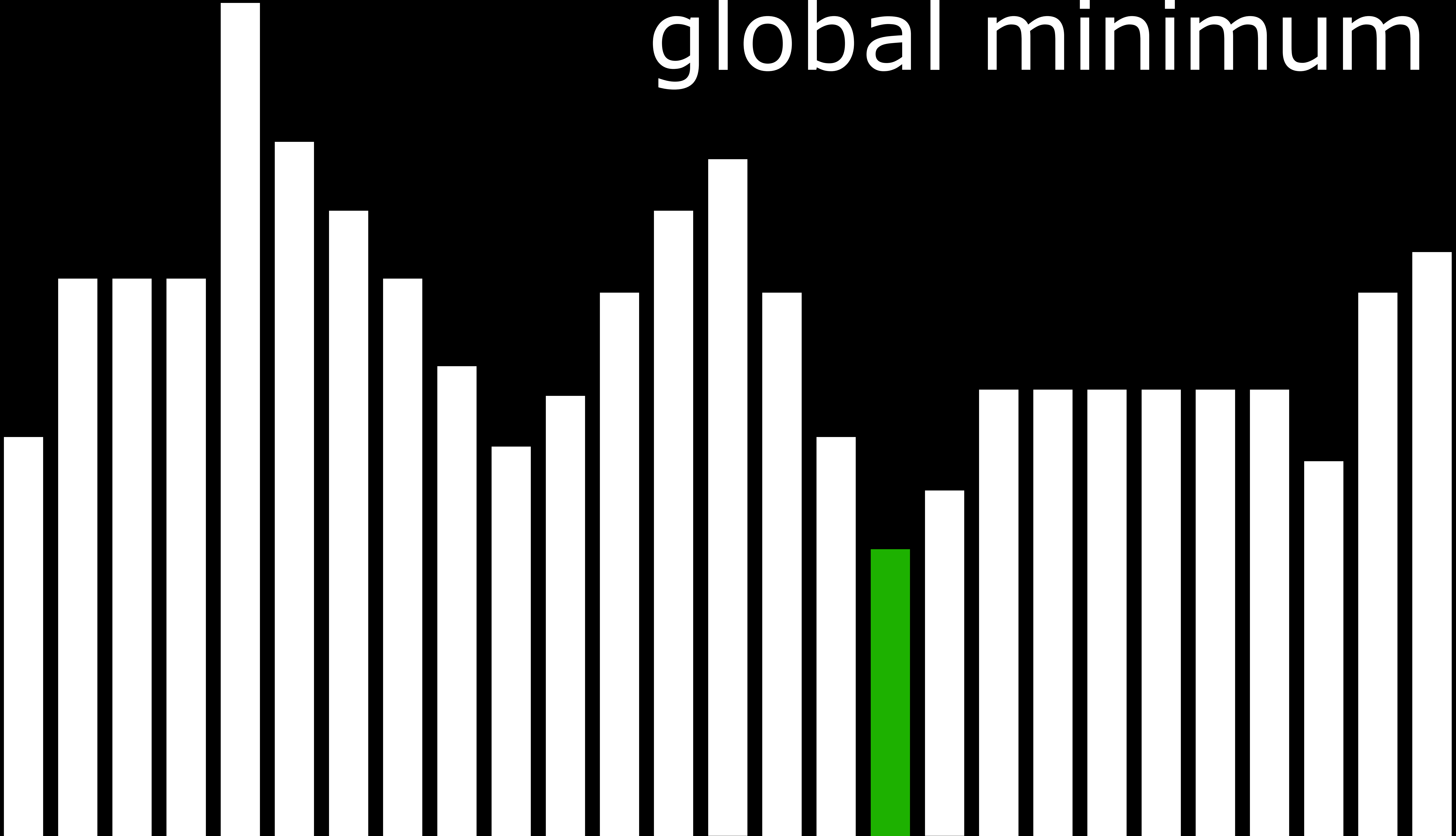
global maximum



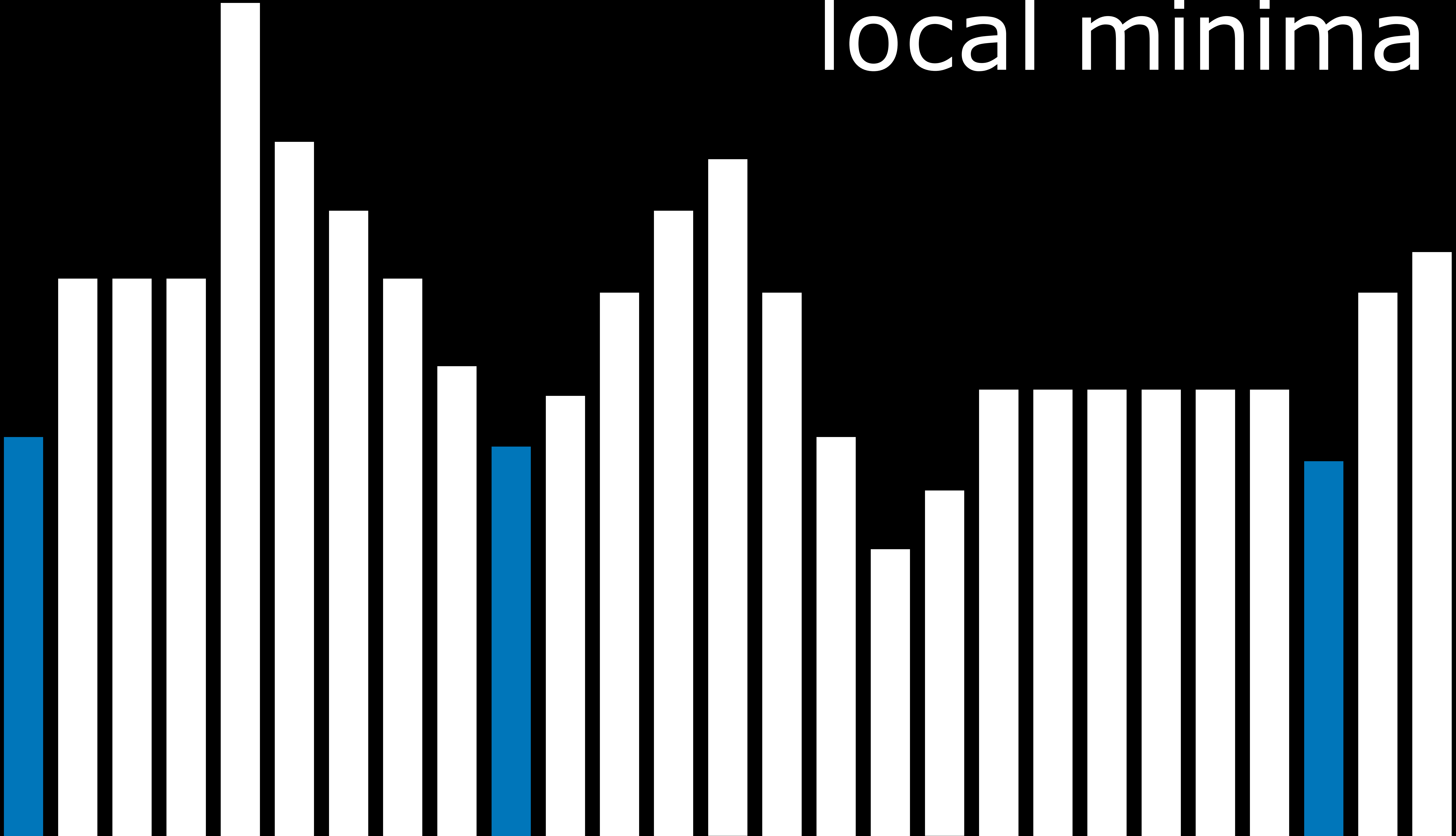
local maxima

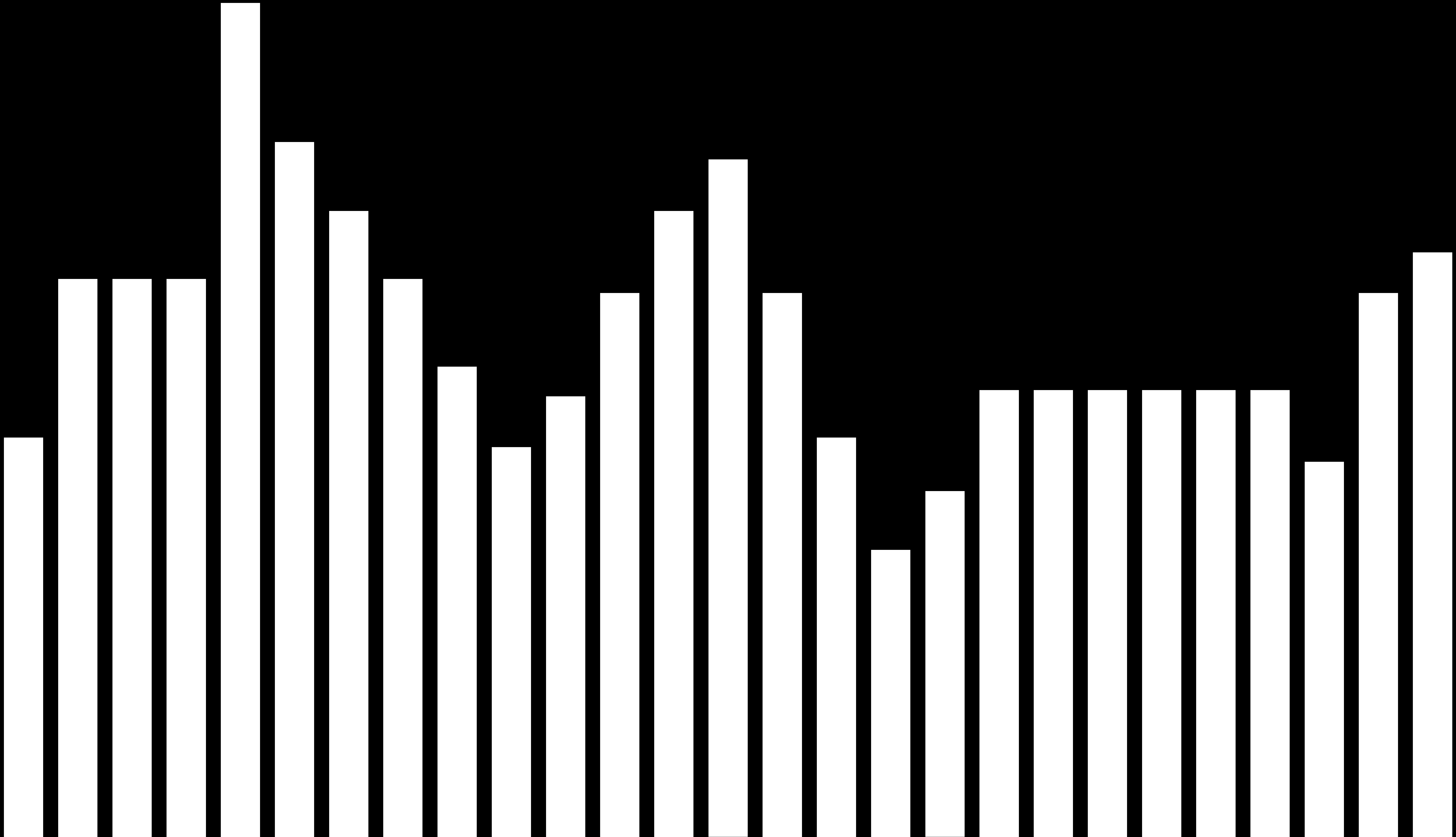


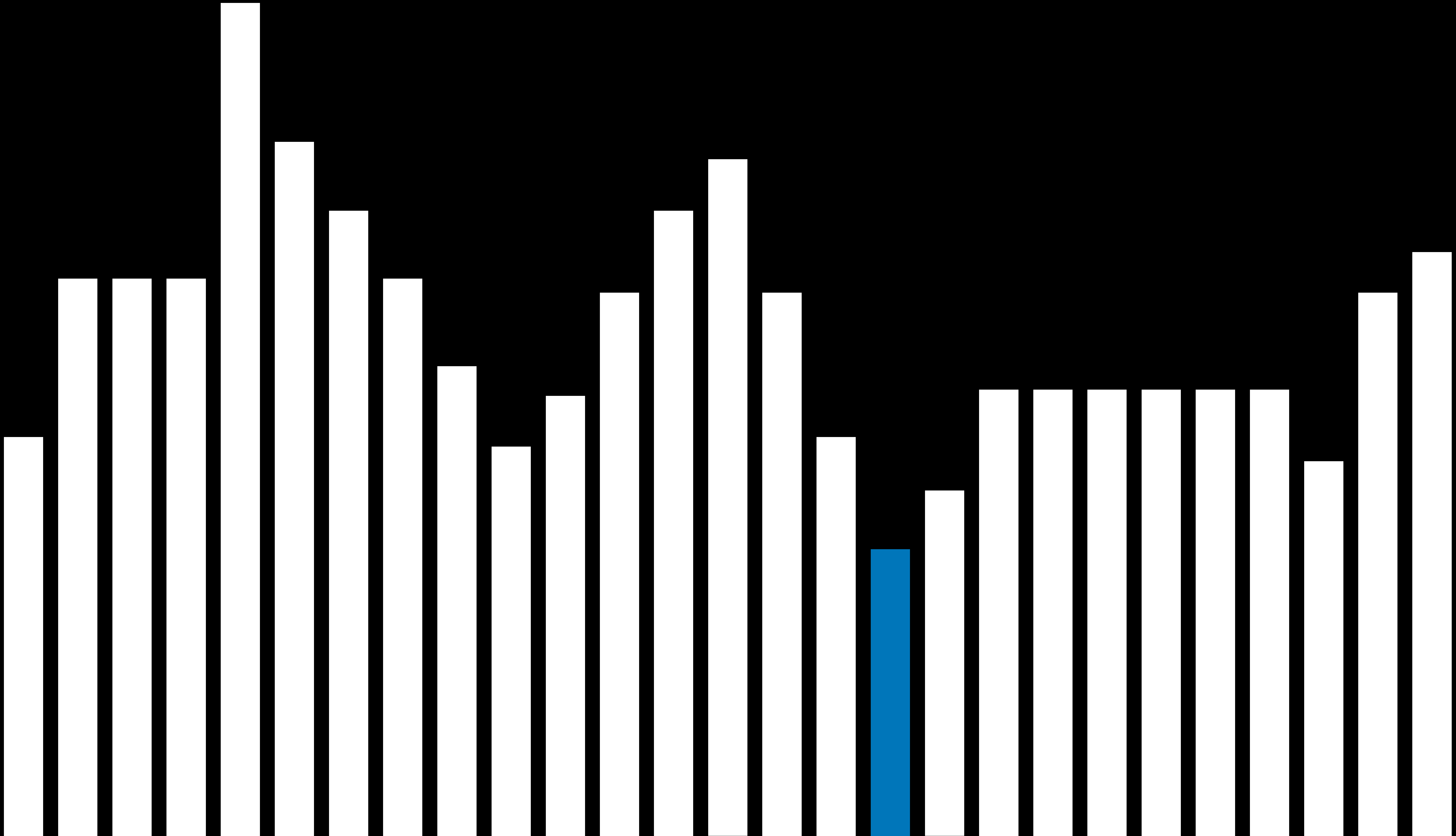
global minimum

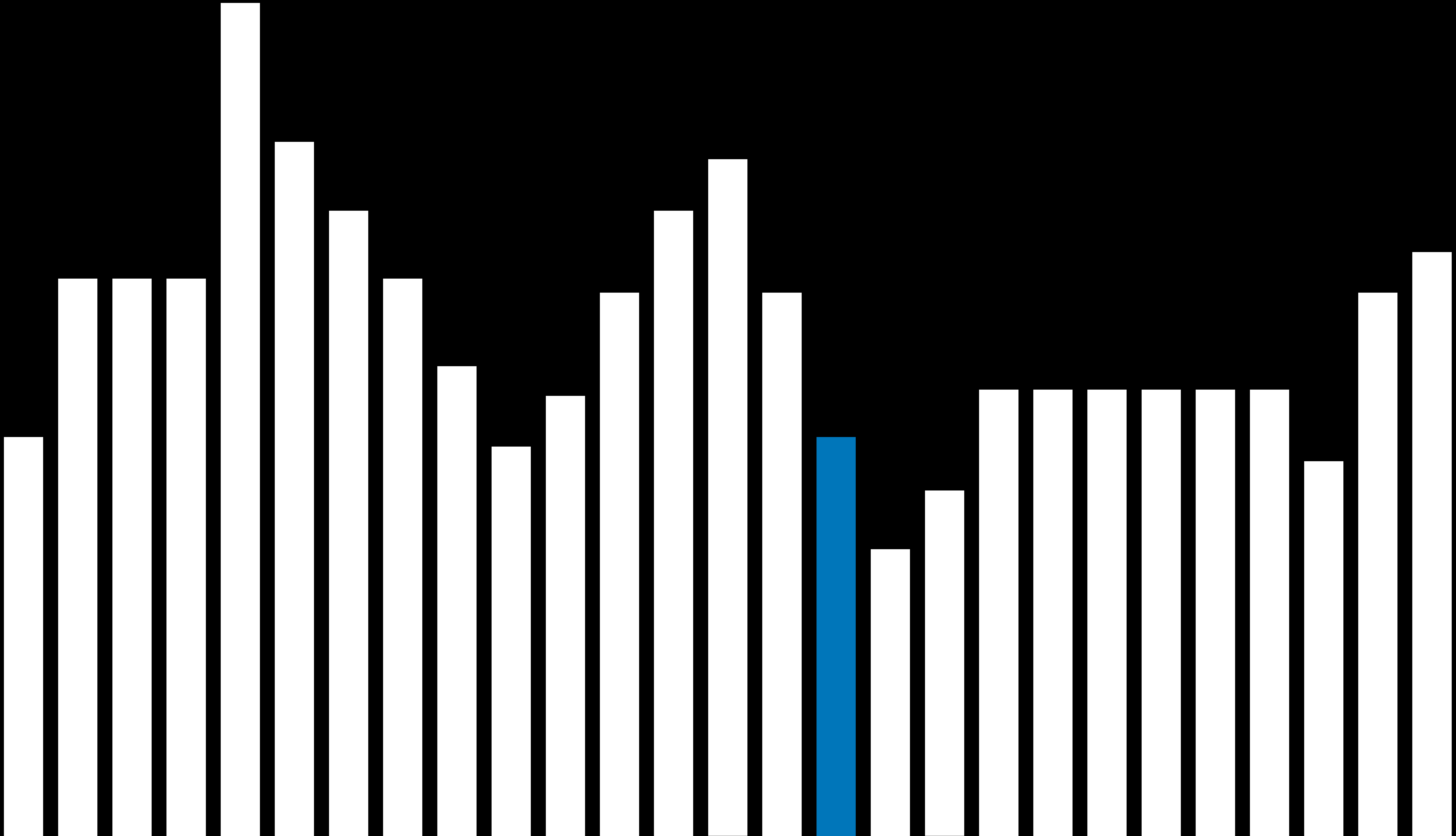


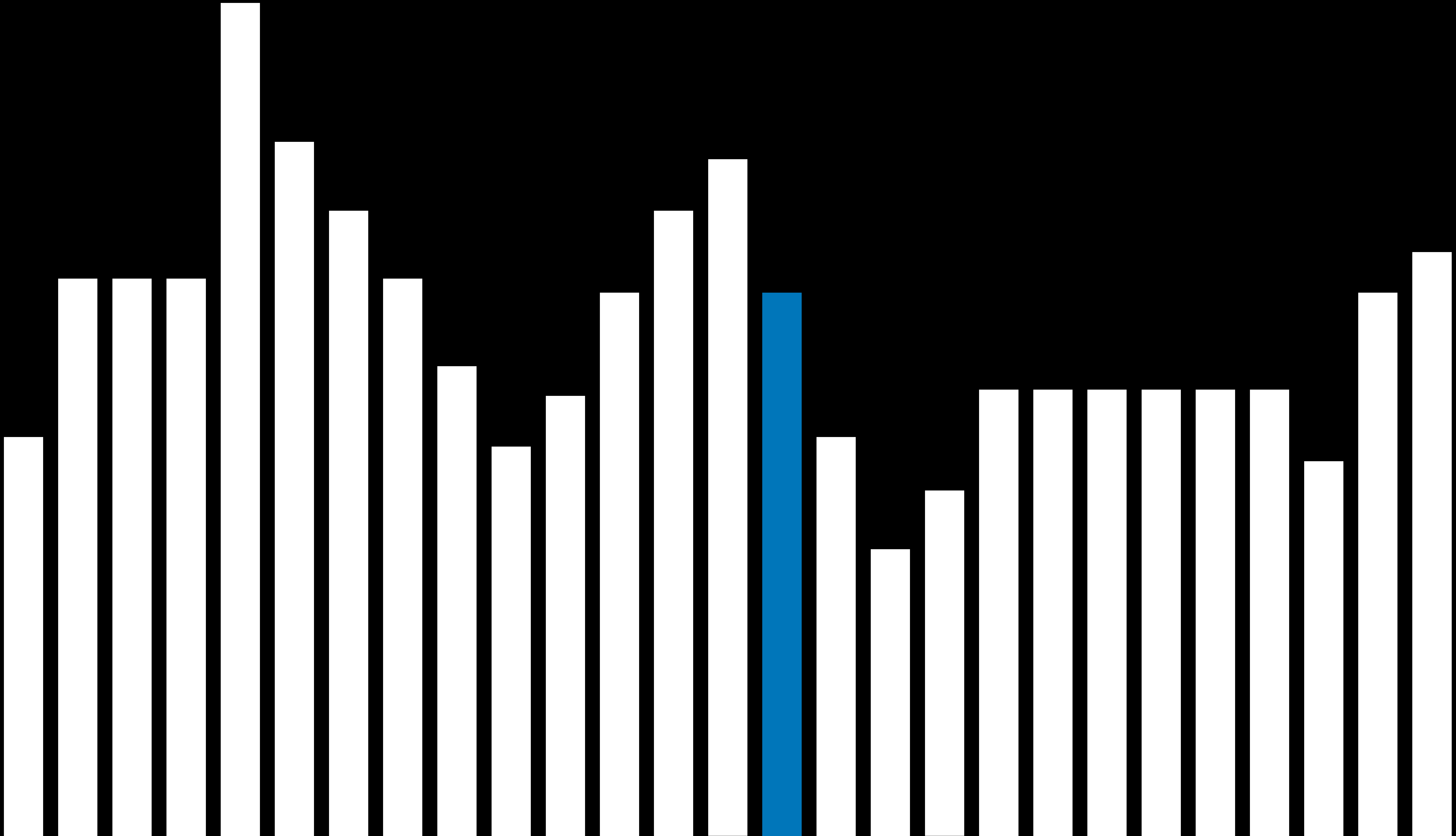
local minima

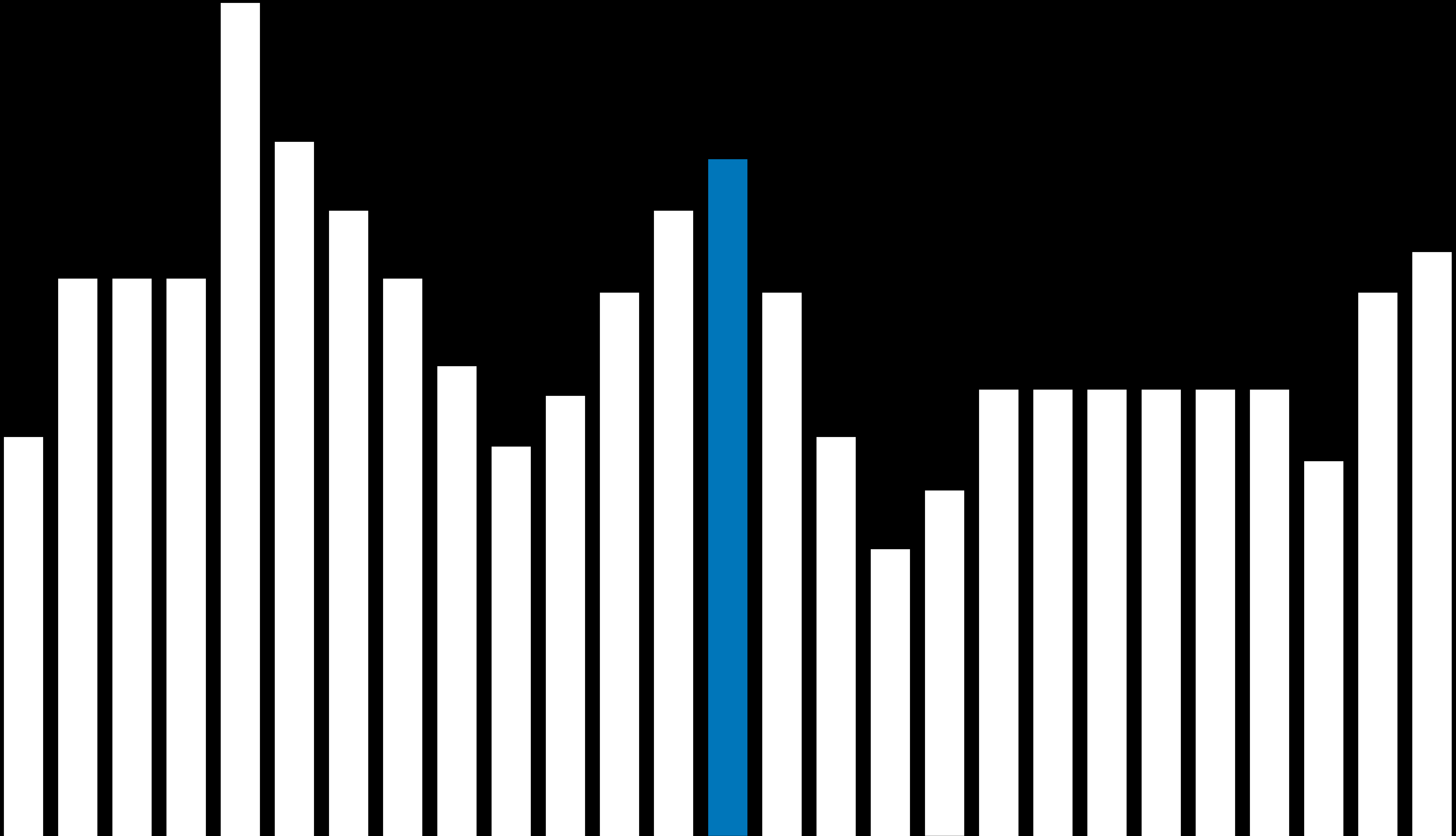




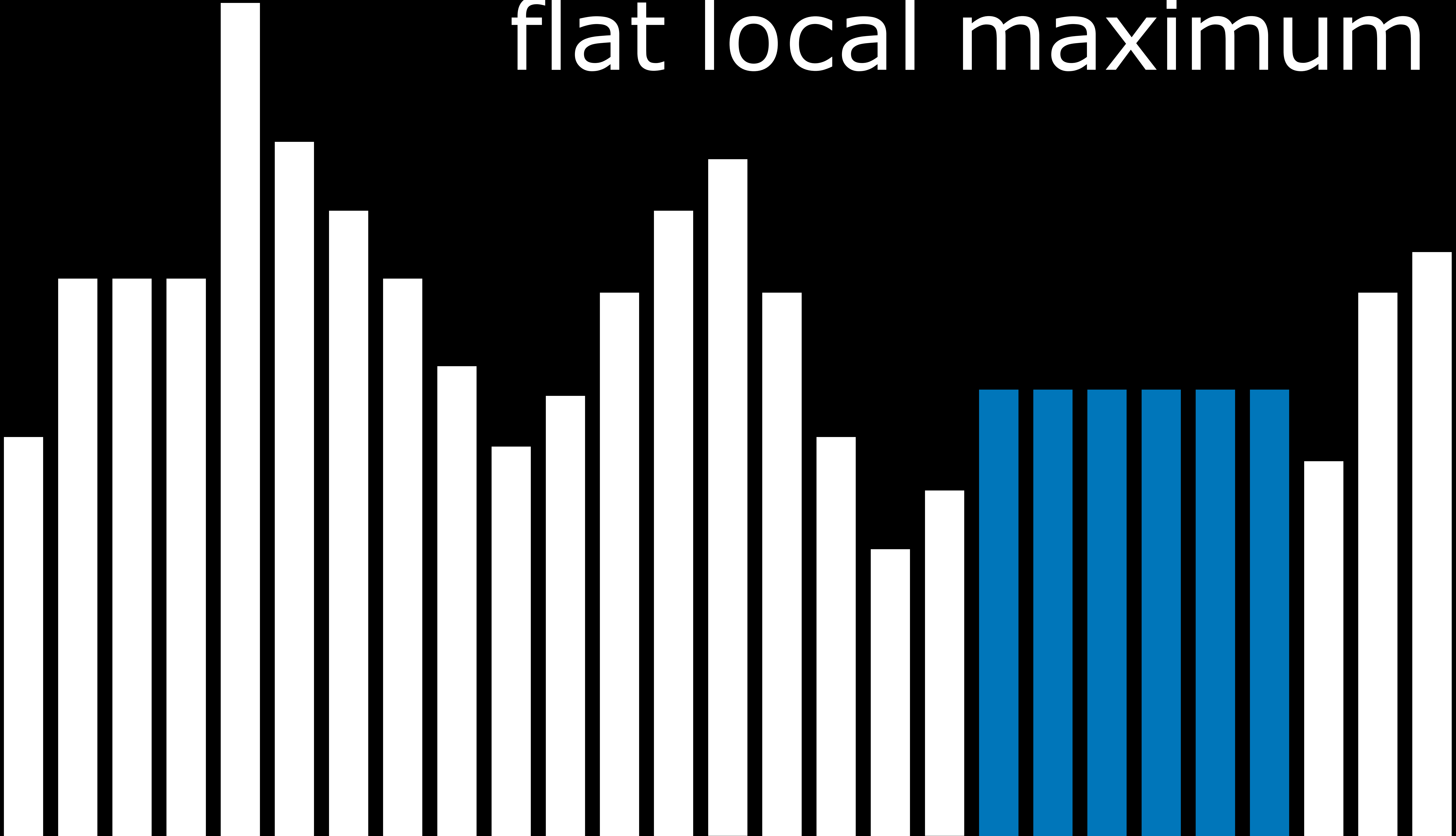




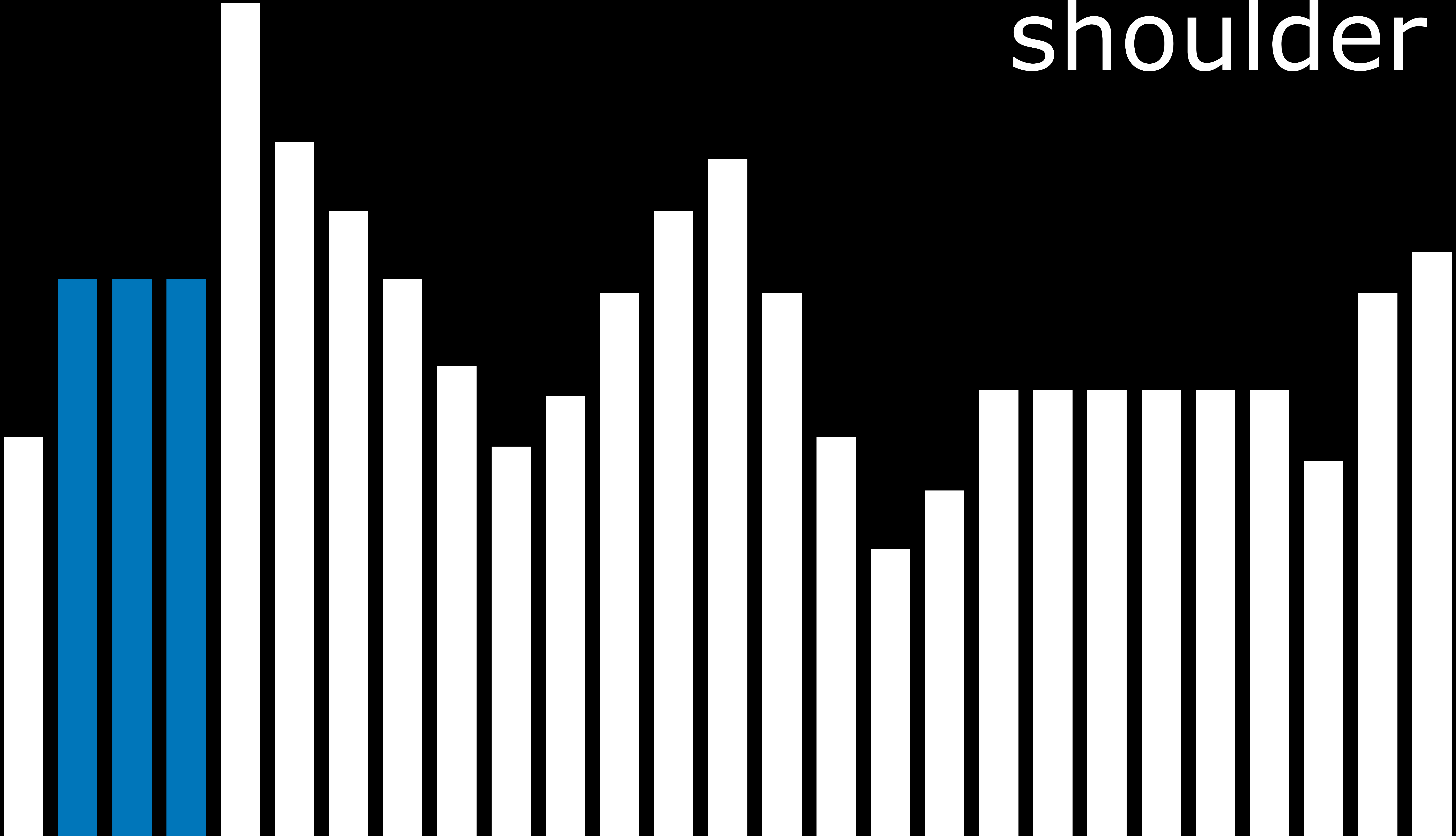




flat local maximum



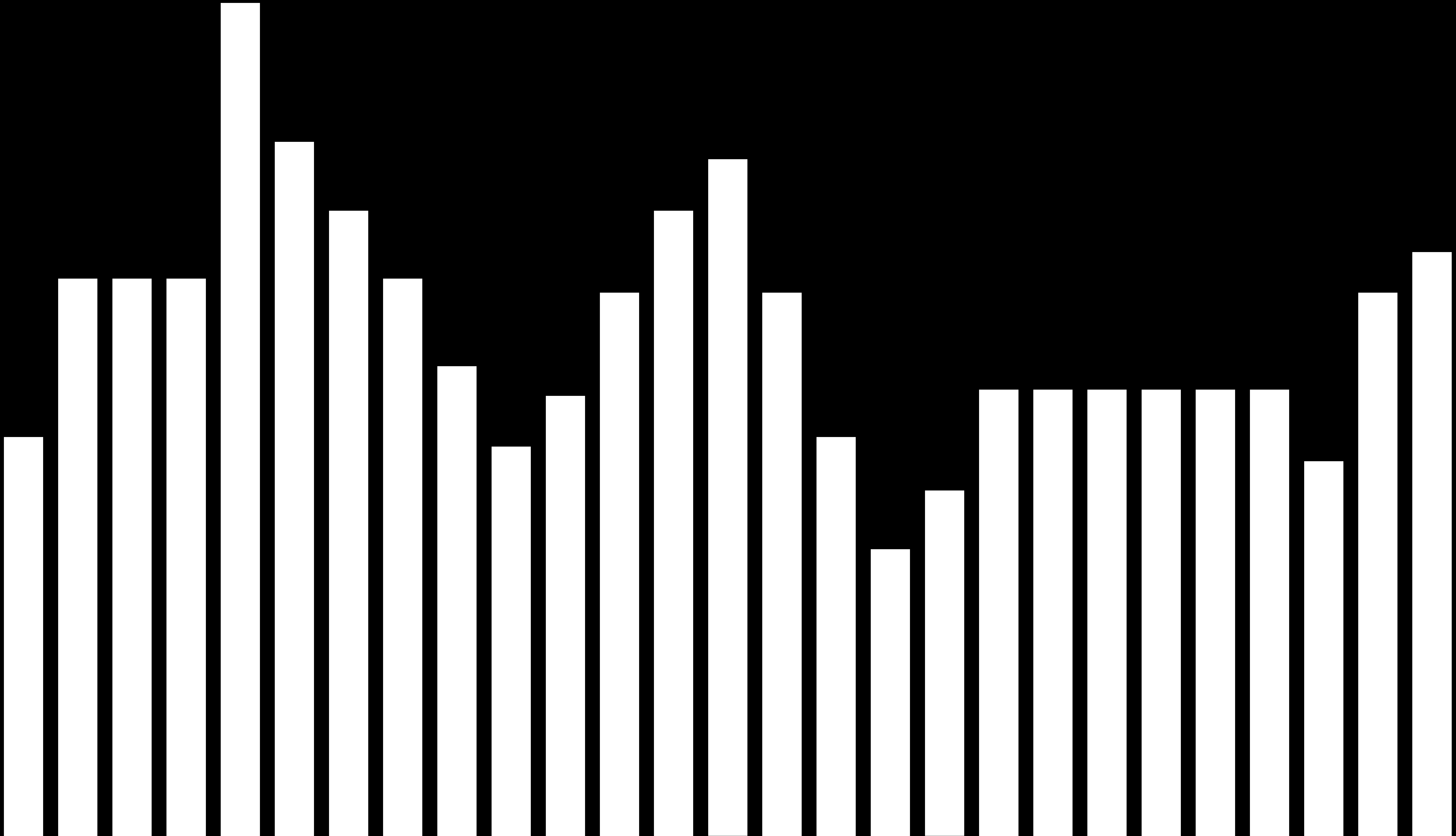
shoulder

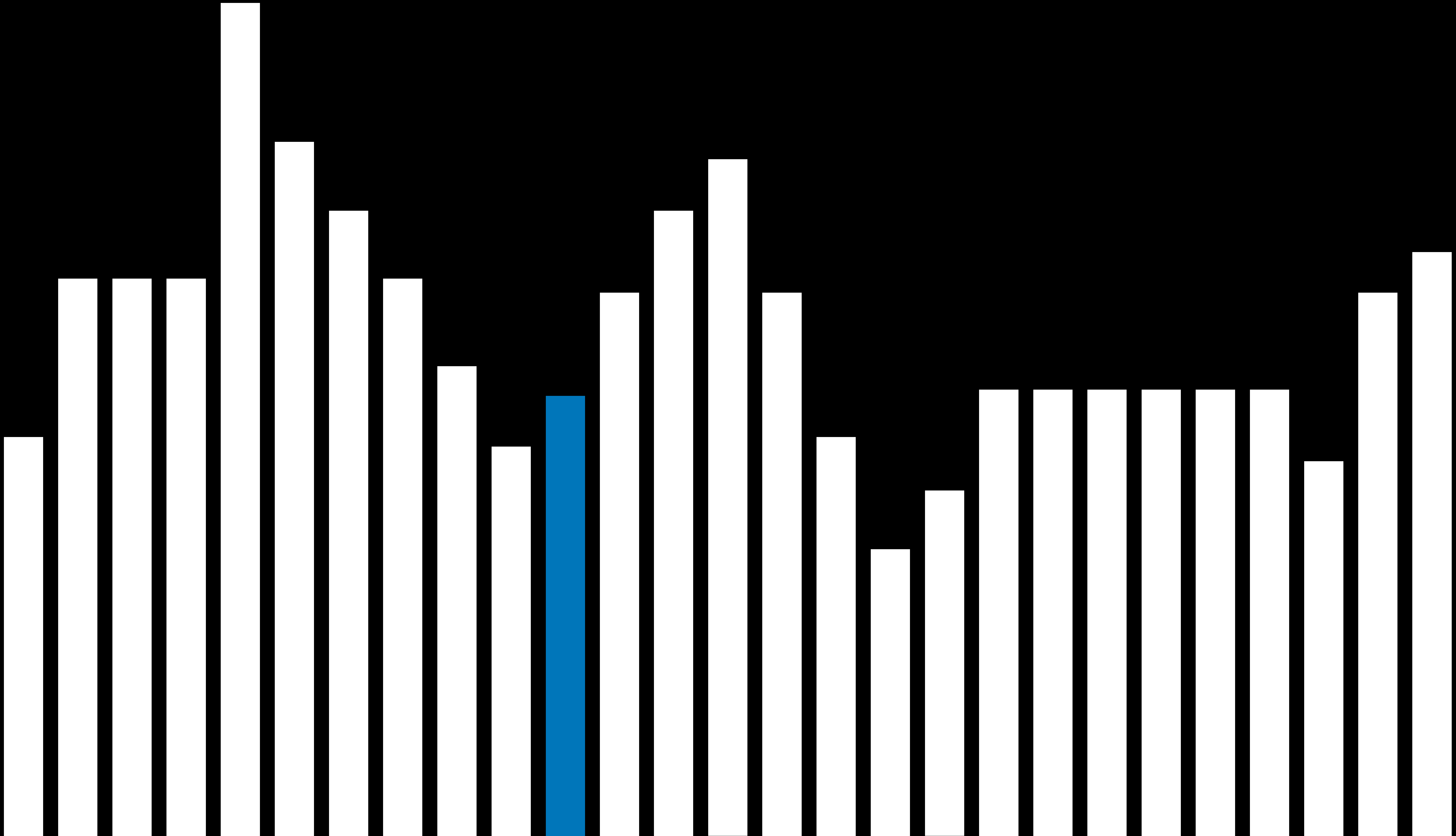


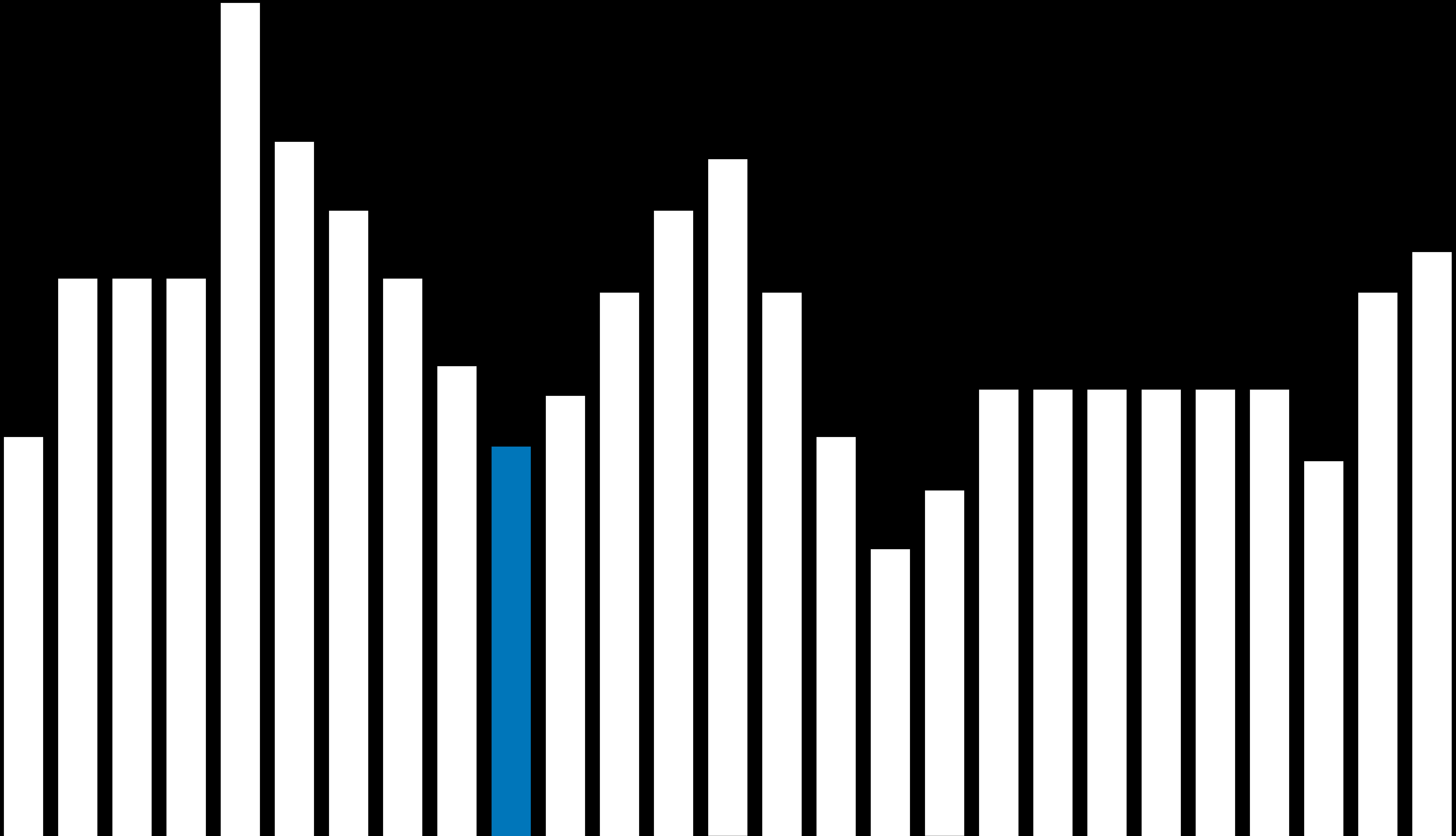
Hill Climbing Variants

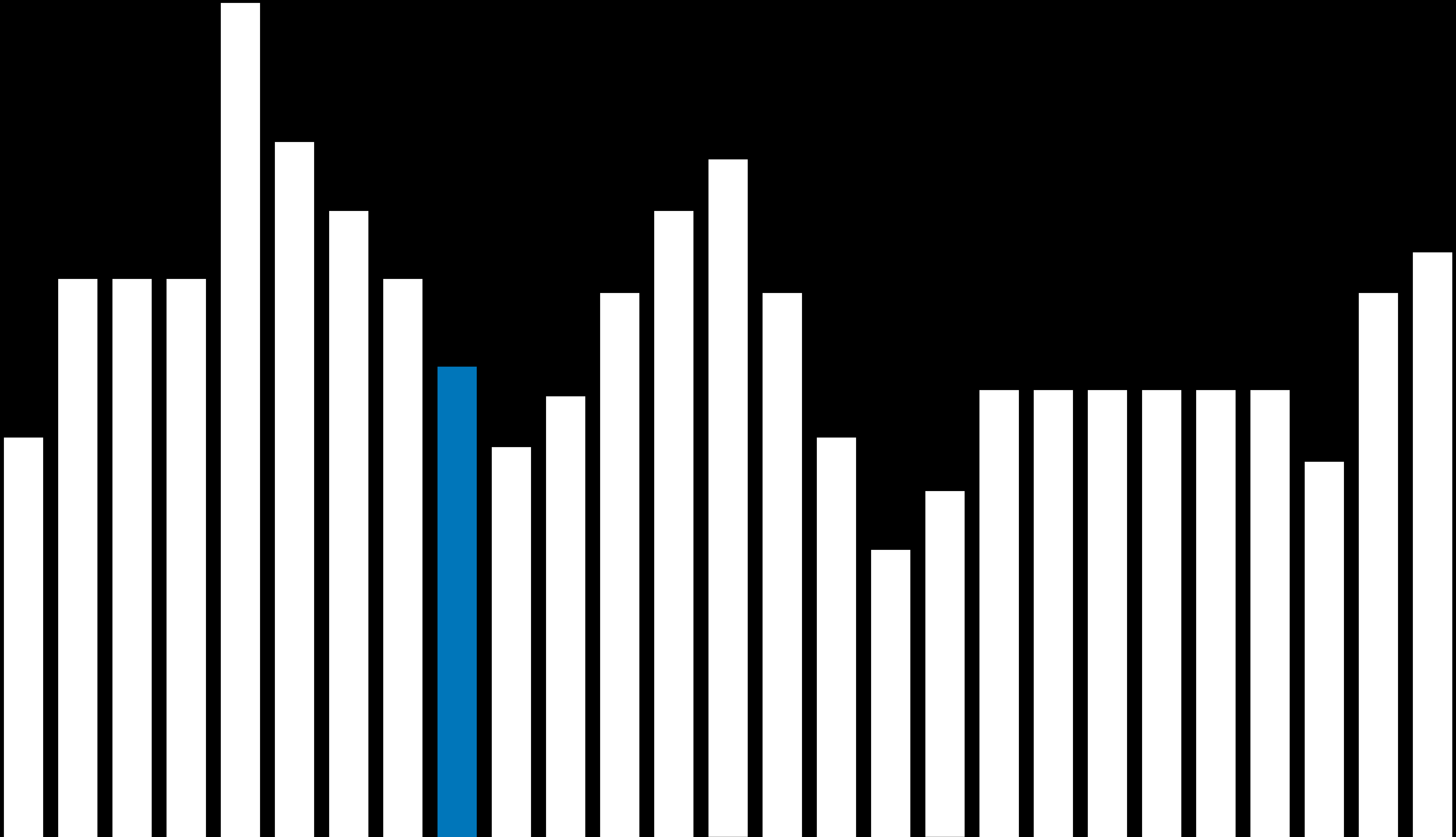
Variant	Definition
steepest-ascent	choose the highest-valued neighbor
stochastic	choose randomly from higher-valued neighbors
first-choice	choose the first higher-valued neighbor
random-restart	conduct hill climbing multiple times
local beam search	chooses the *k* highest-valued neighbors

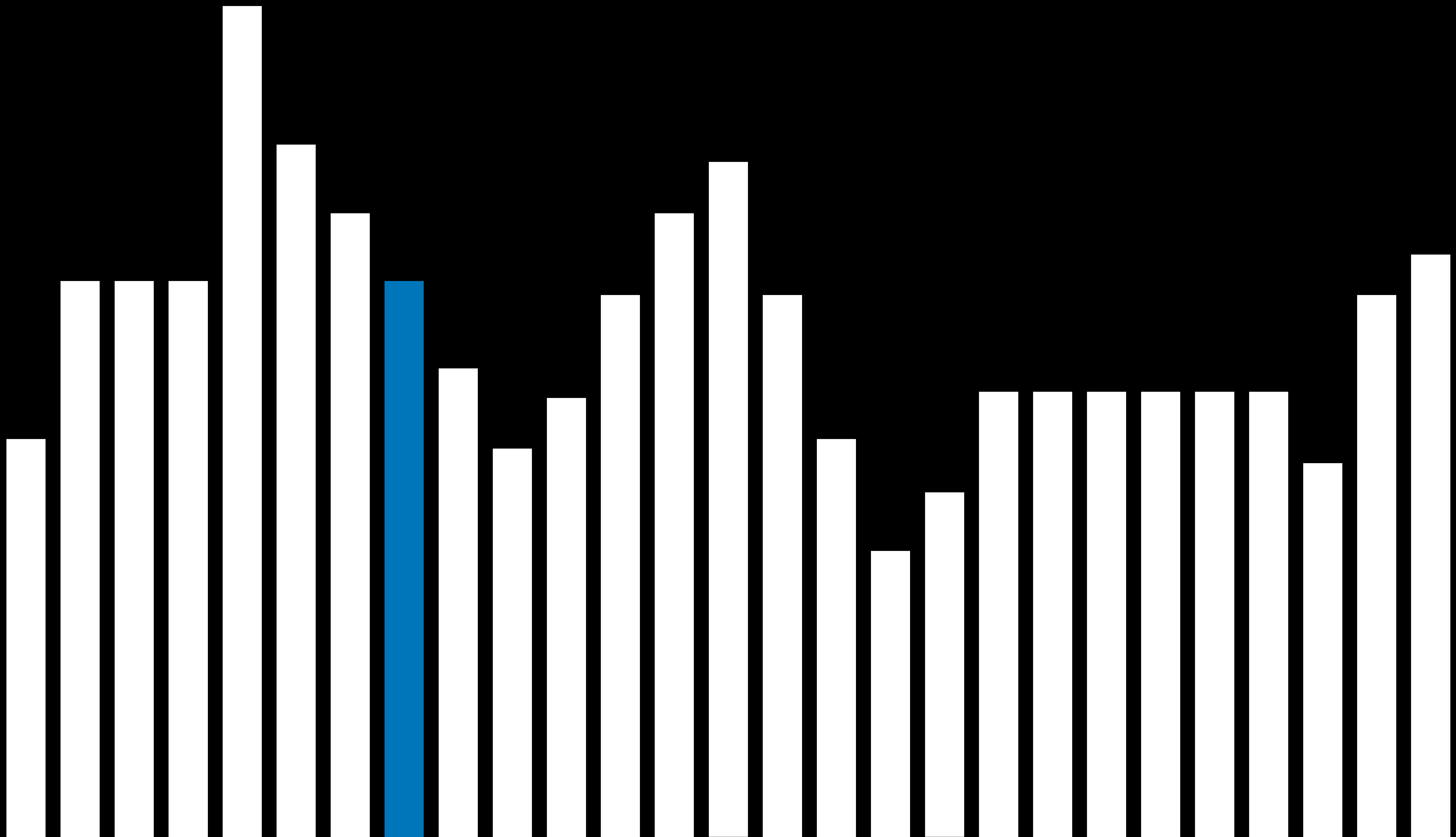
Simulated Annealing

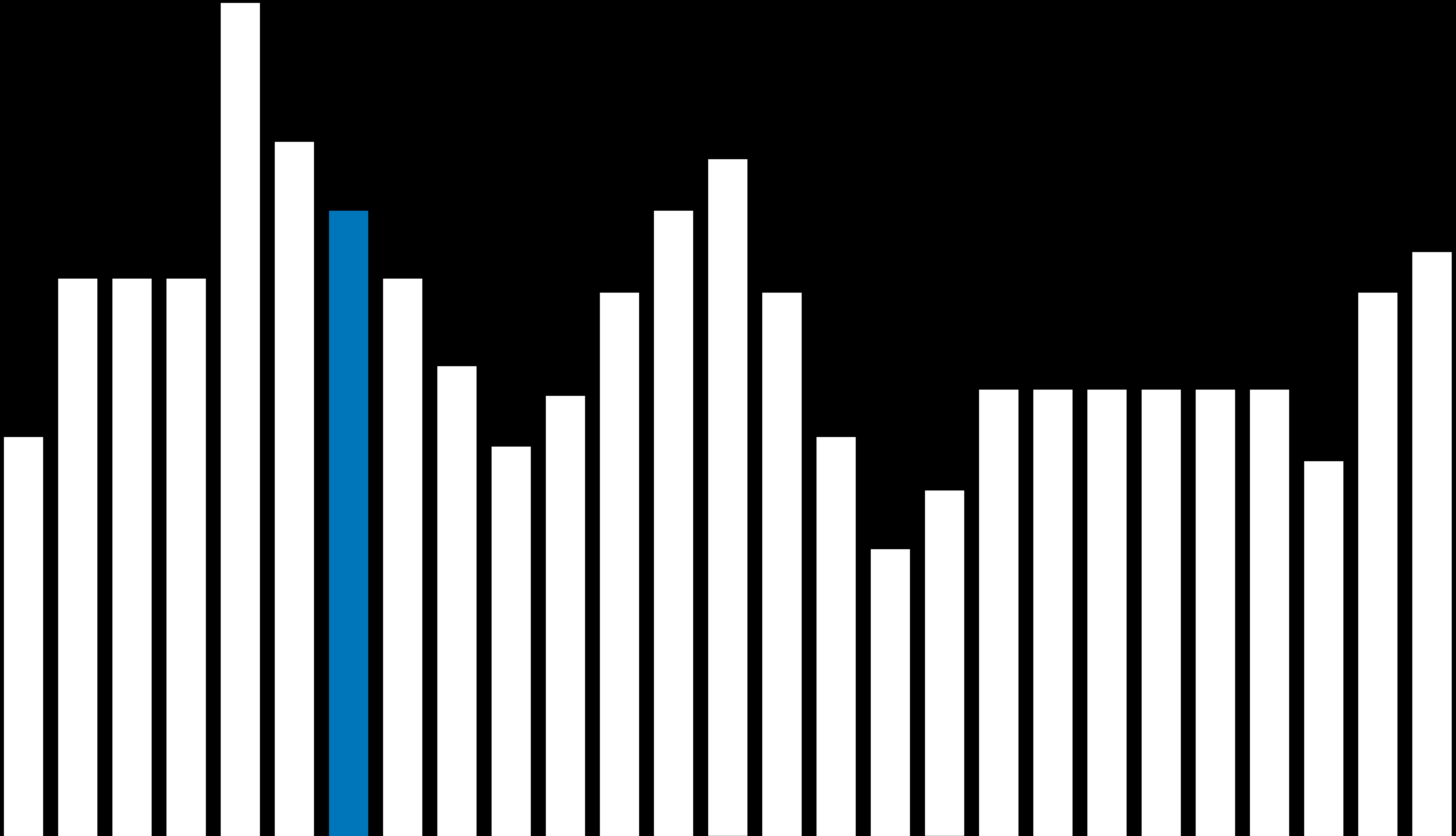


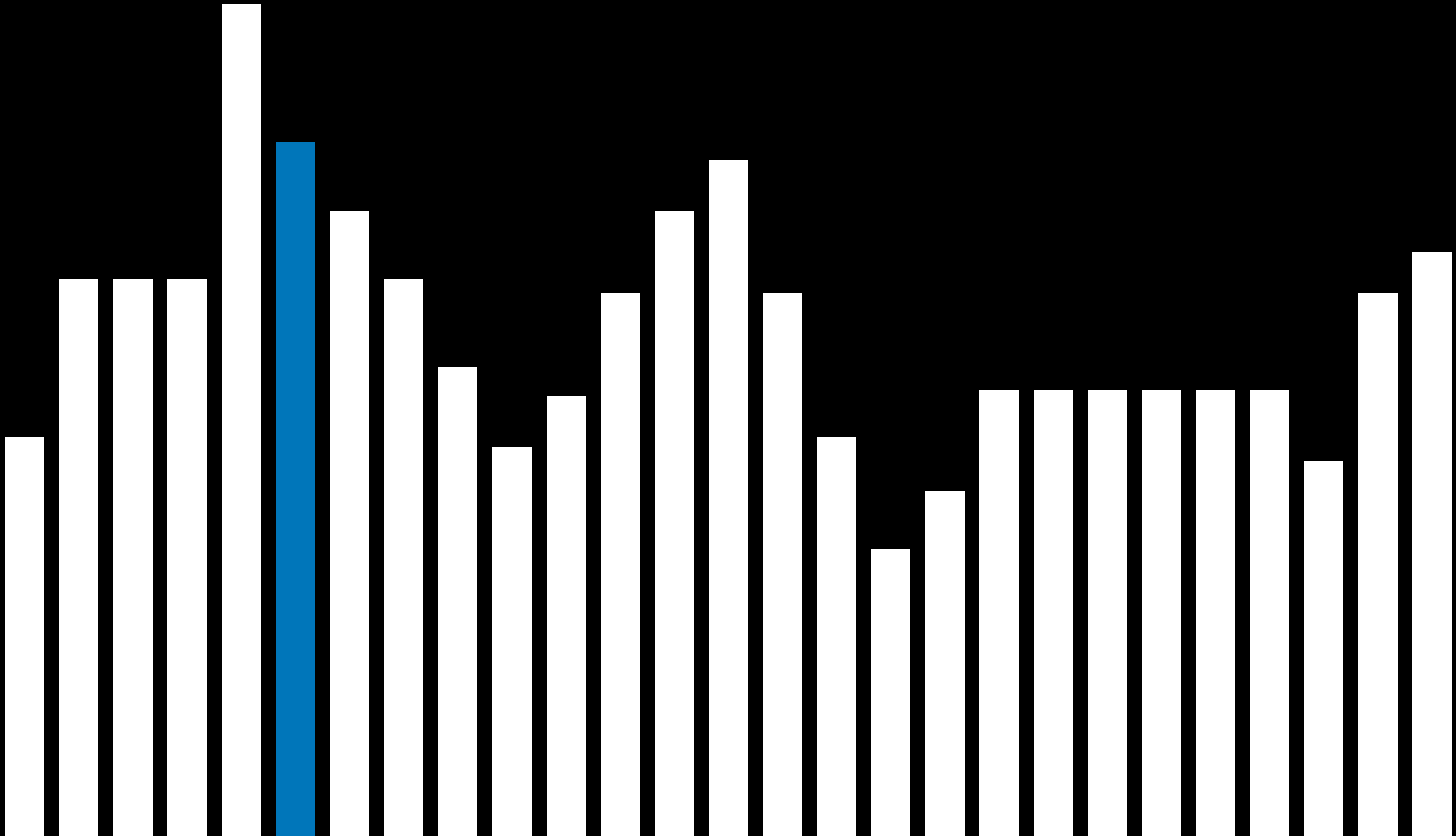


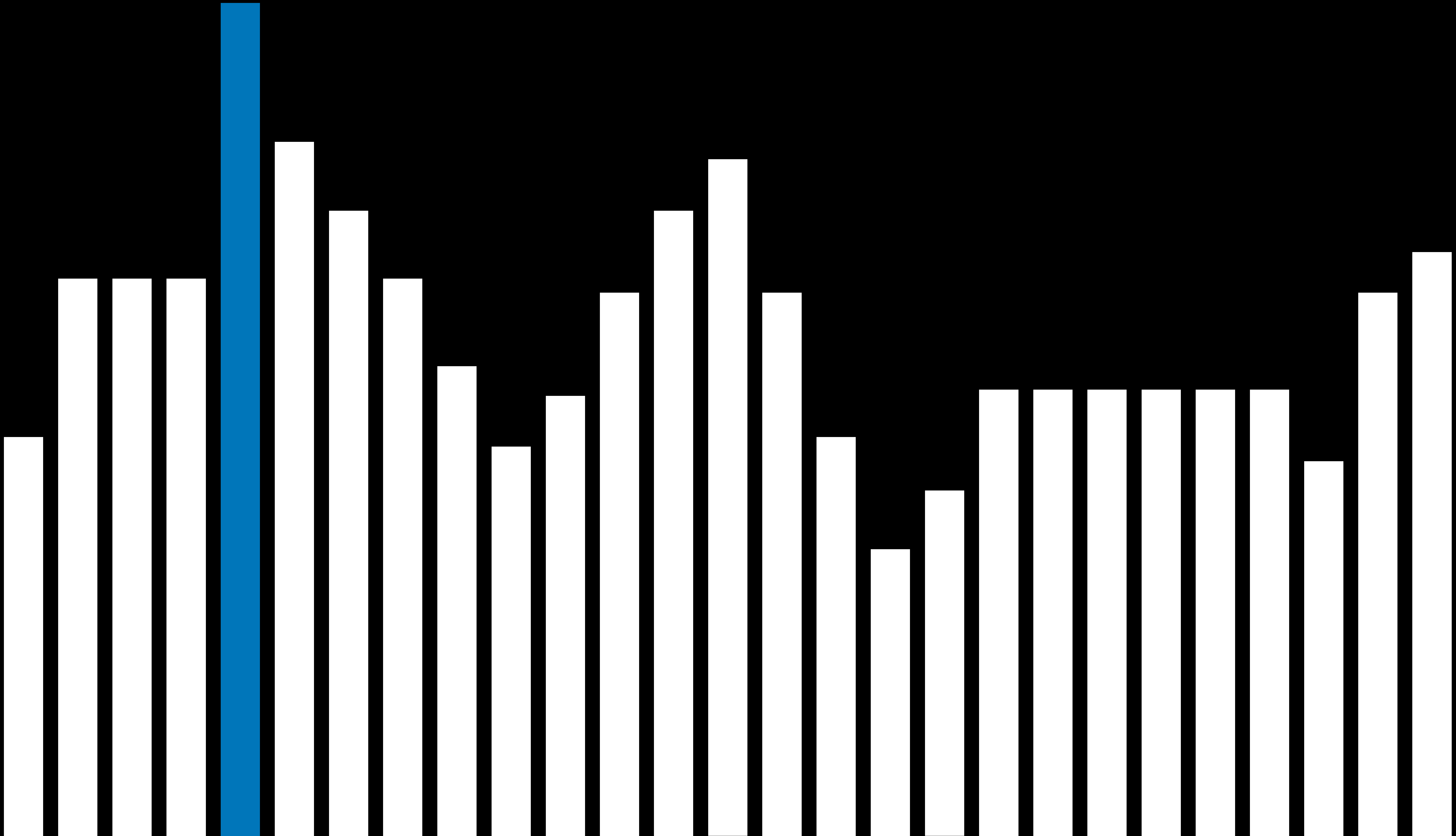












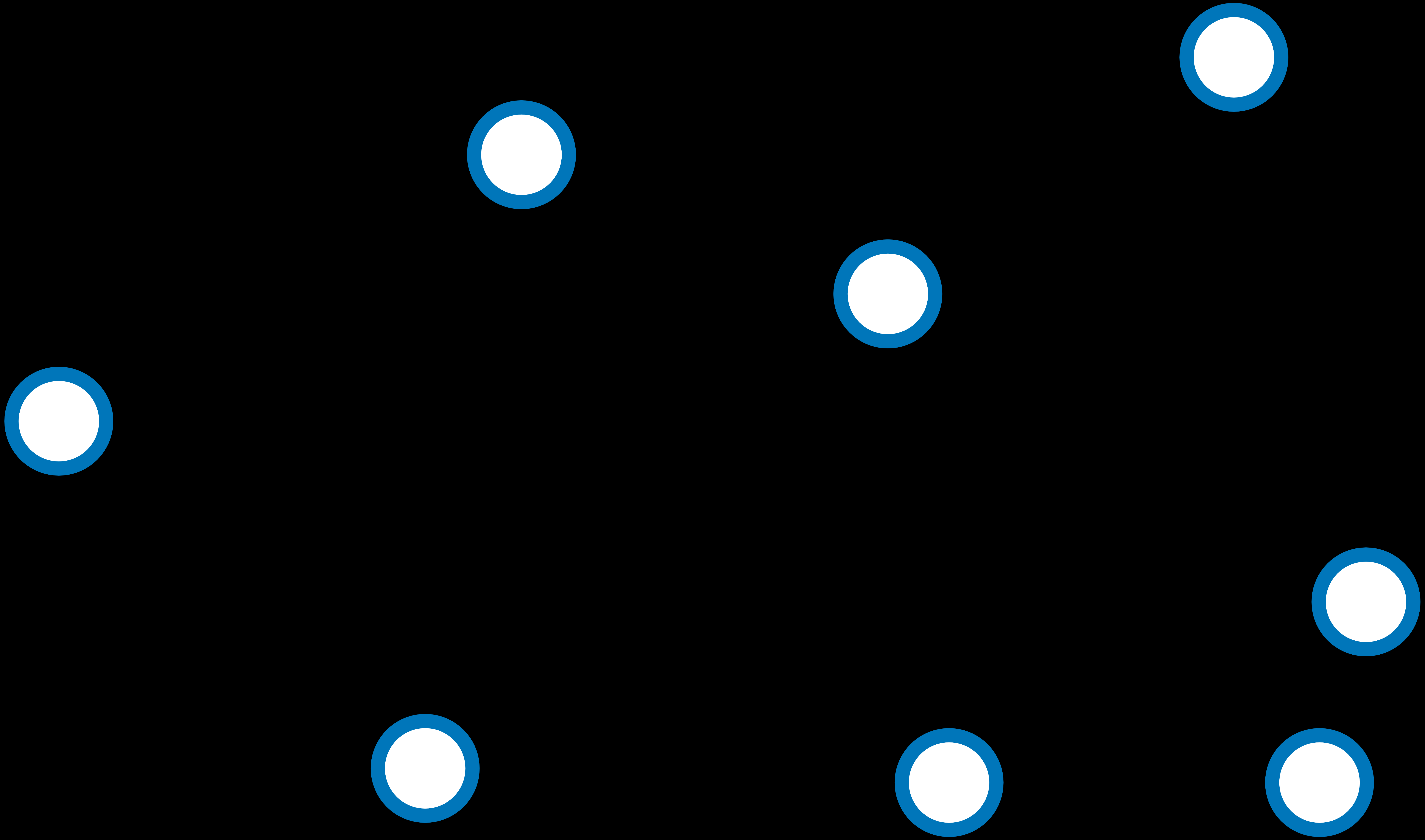
Simulated Annealing

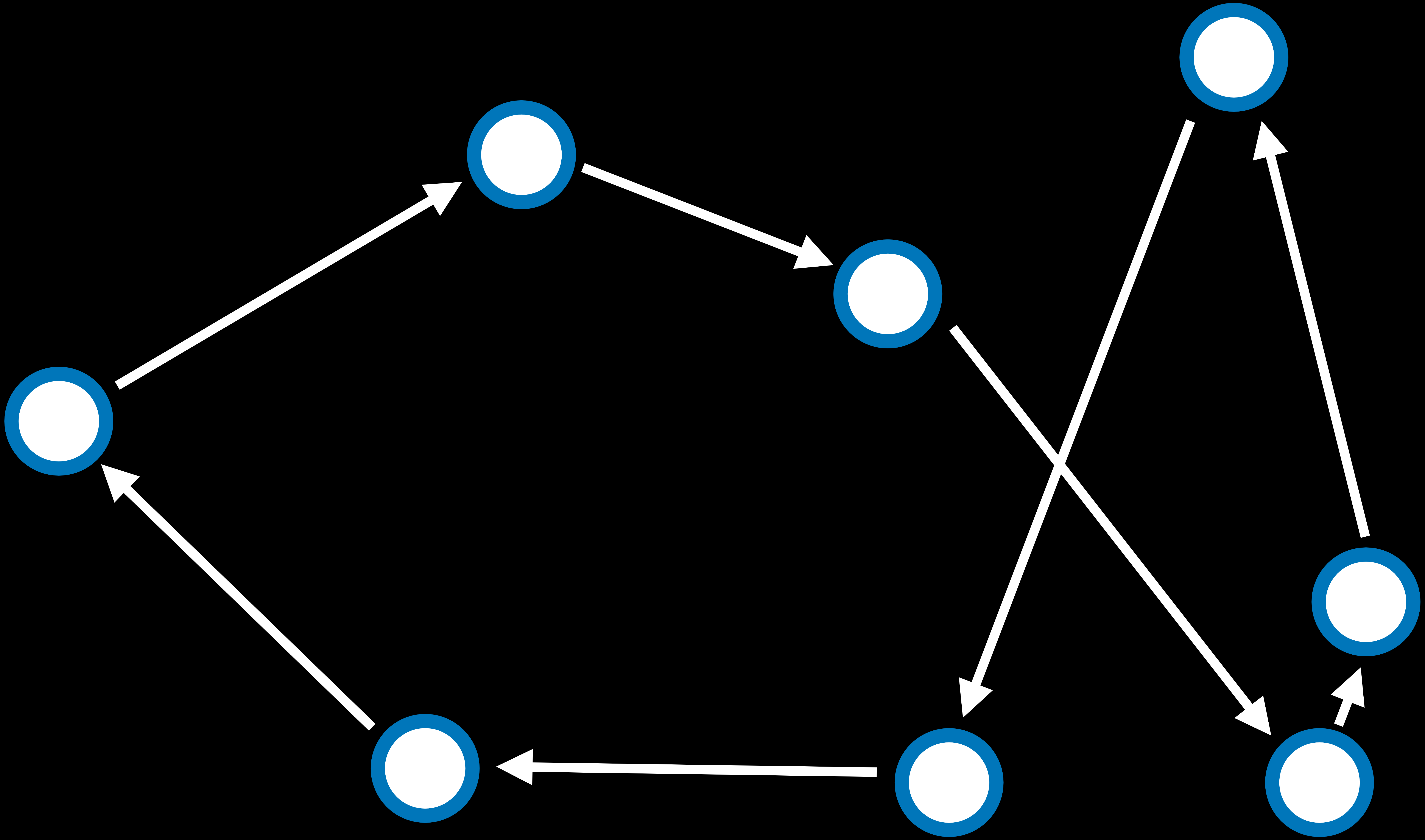
- Early on, higher "temperature": more likely to accept neighbors that are worse than current state
- Later on, lower "temperature": less likely to accept neighbors that are worse than current state

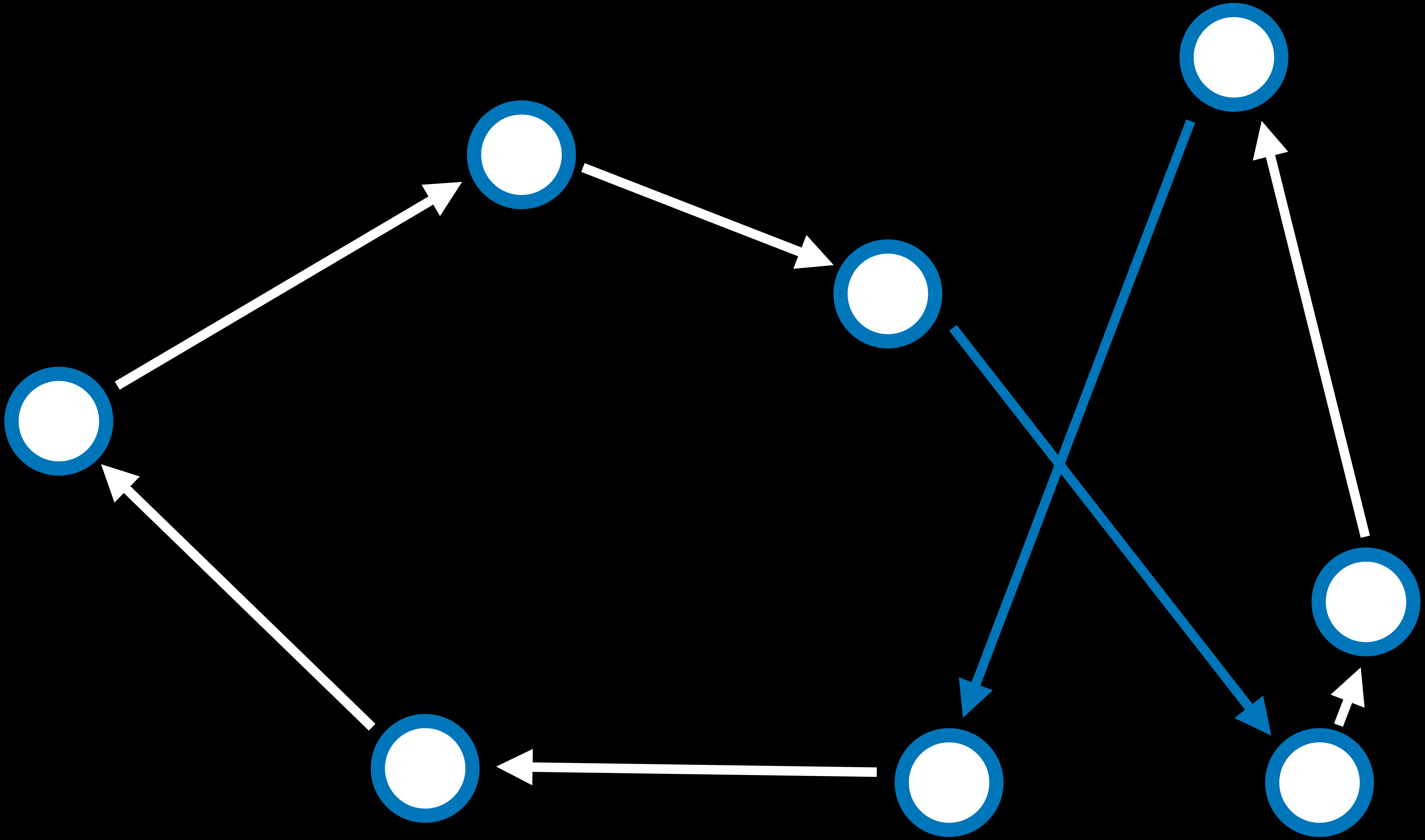
Simulated Annealing

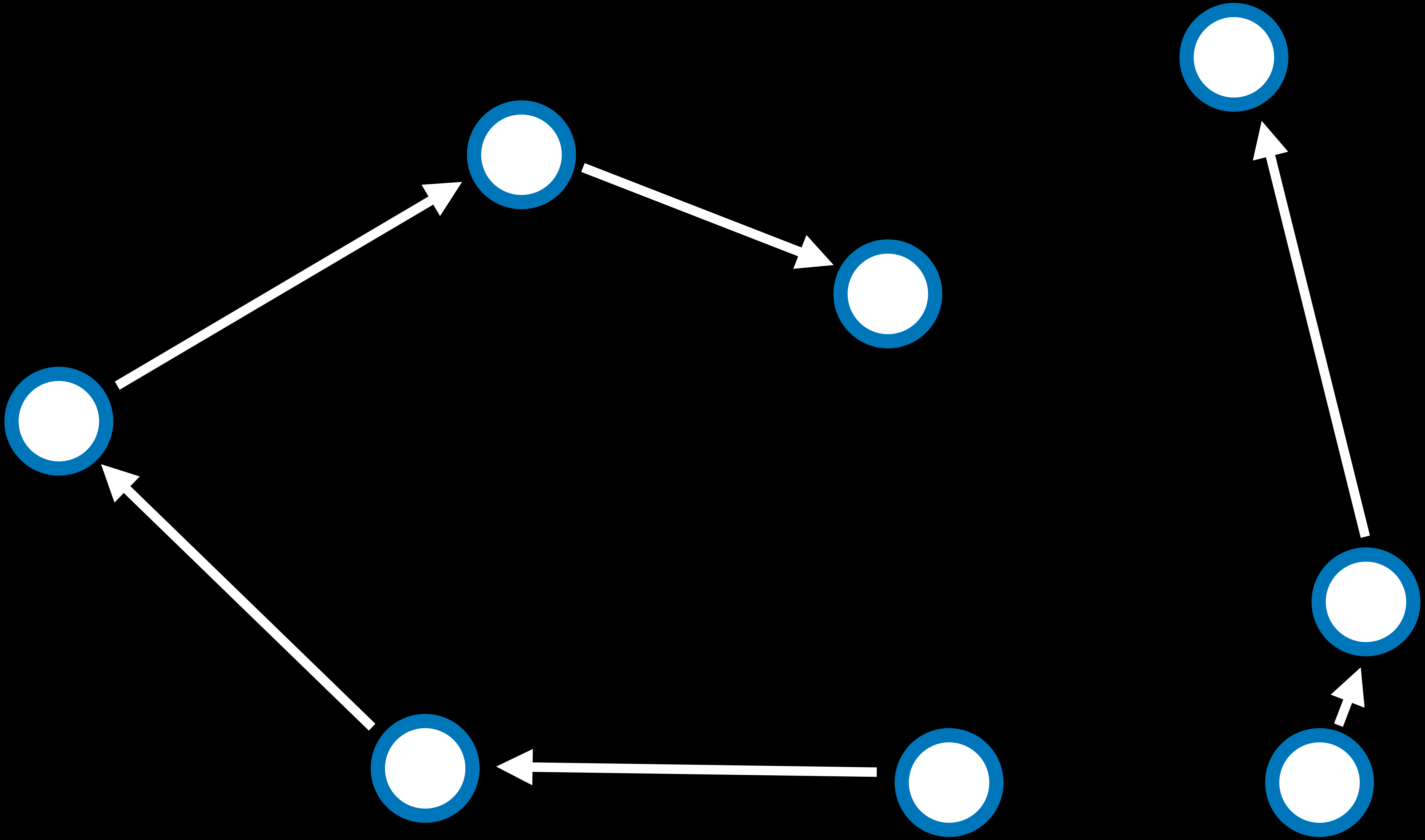
```
function SIMULATED-ANNEALING($problem$, $max$):
    $current$ = initial state of $problem$
    for $t$ = 1 to $max$:
        $T$ = TEMPERATURE($t$)
        $neighbor$ = random neighbor of $current$
        $\Delta E$ = how much better $neighbor$ is than $current$
        if $\Delta E > 0$:
            $current$ = $neighbor$
        with probability $e^{\Delta E/T}$ set $current$ = $neighbor$
    return $current$
```

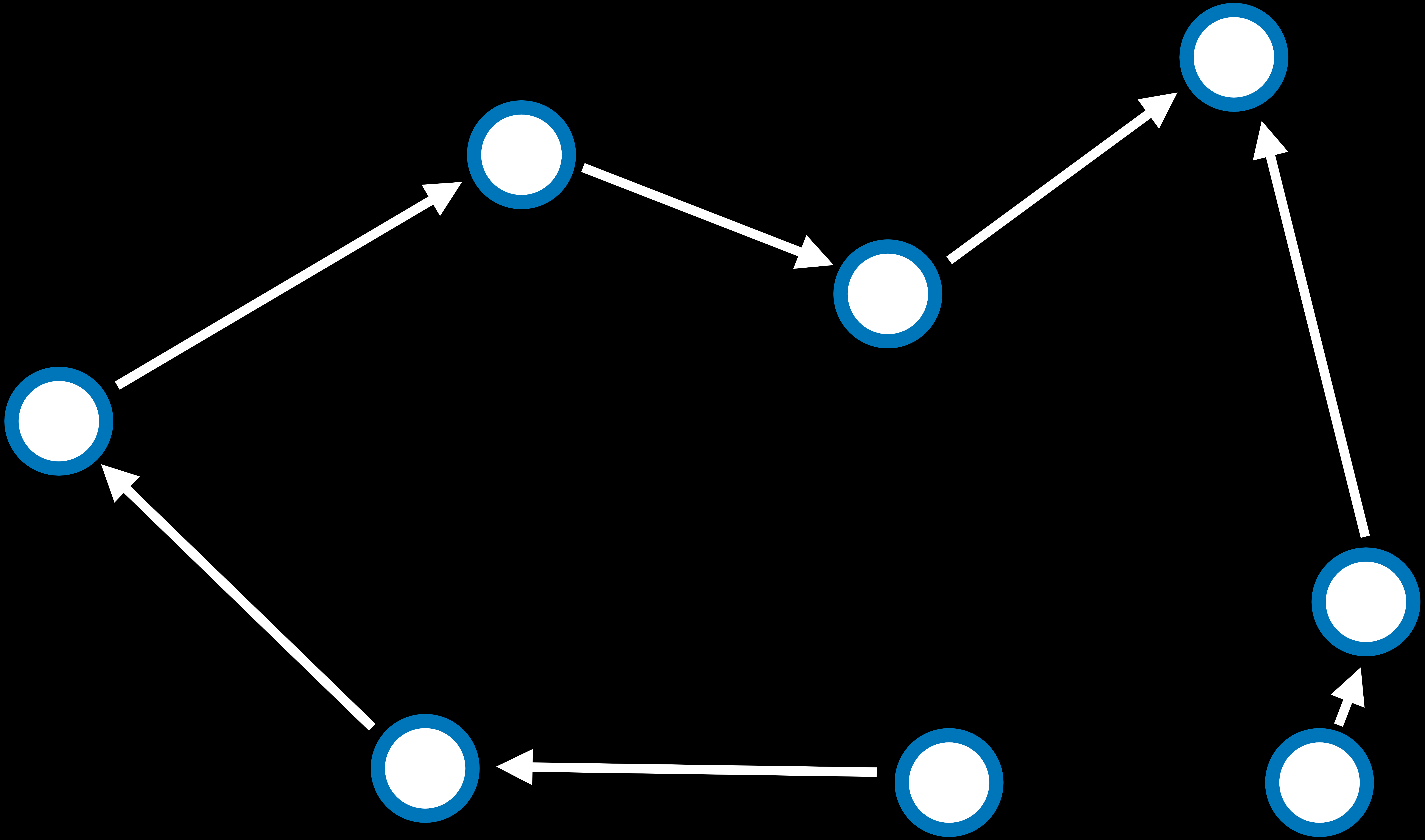
Traveling Salesman Problem

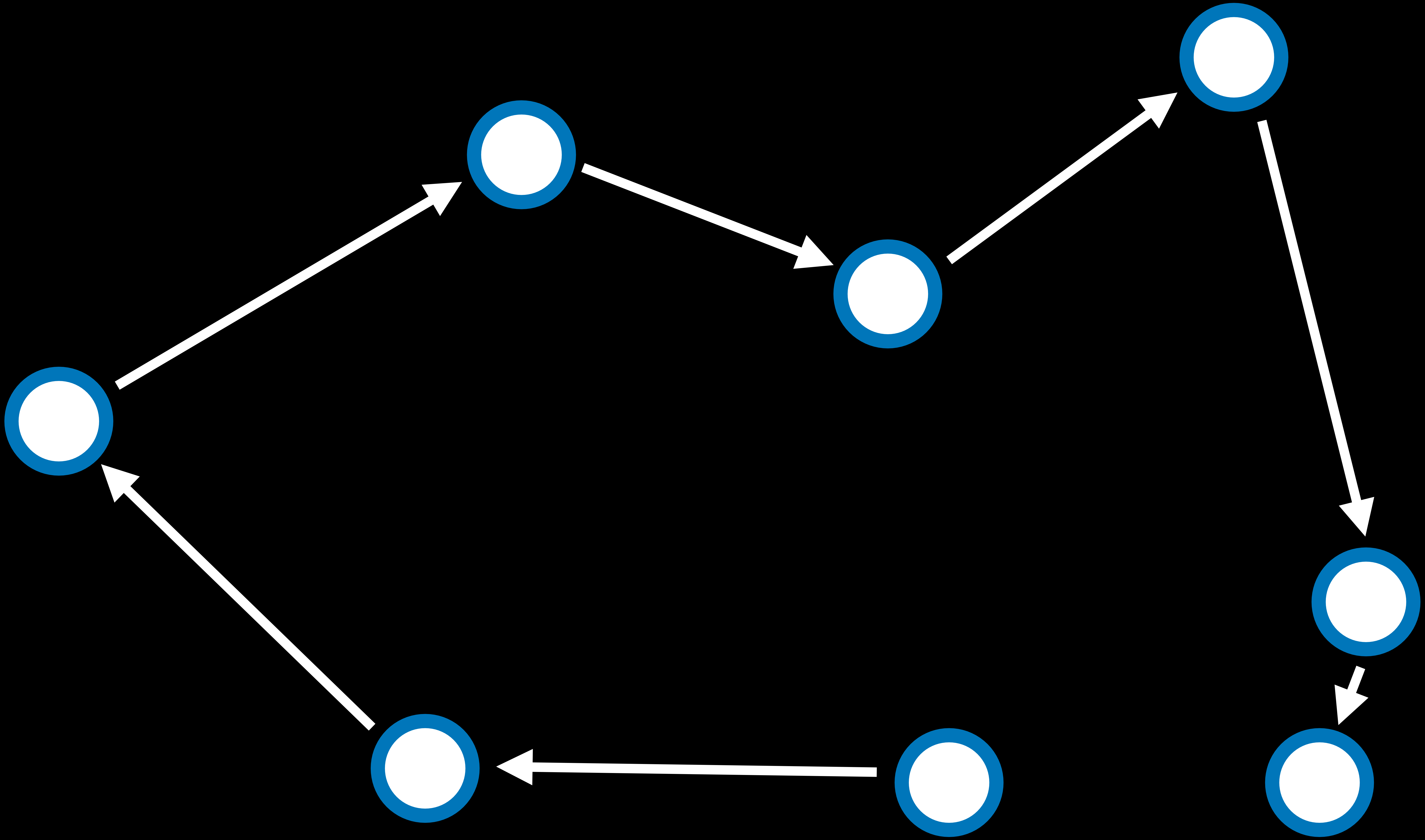


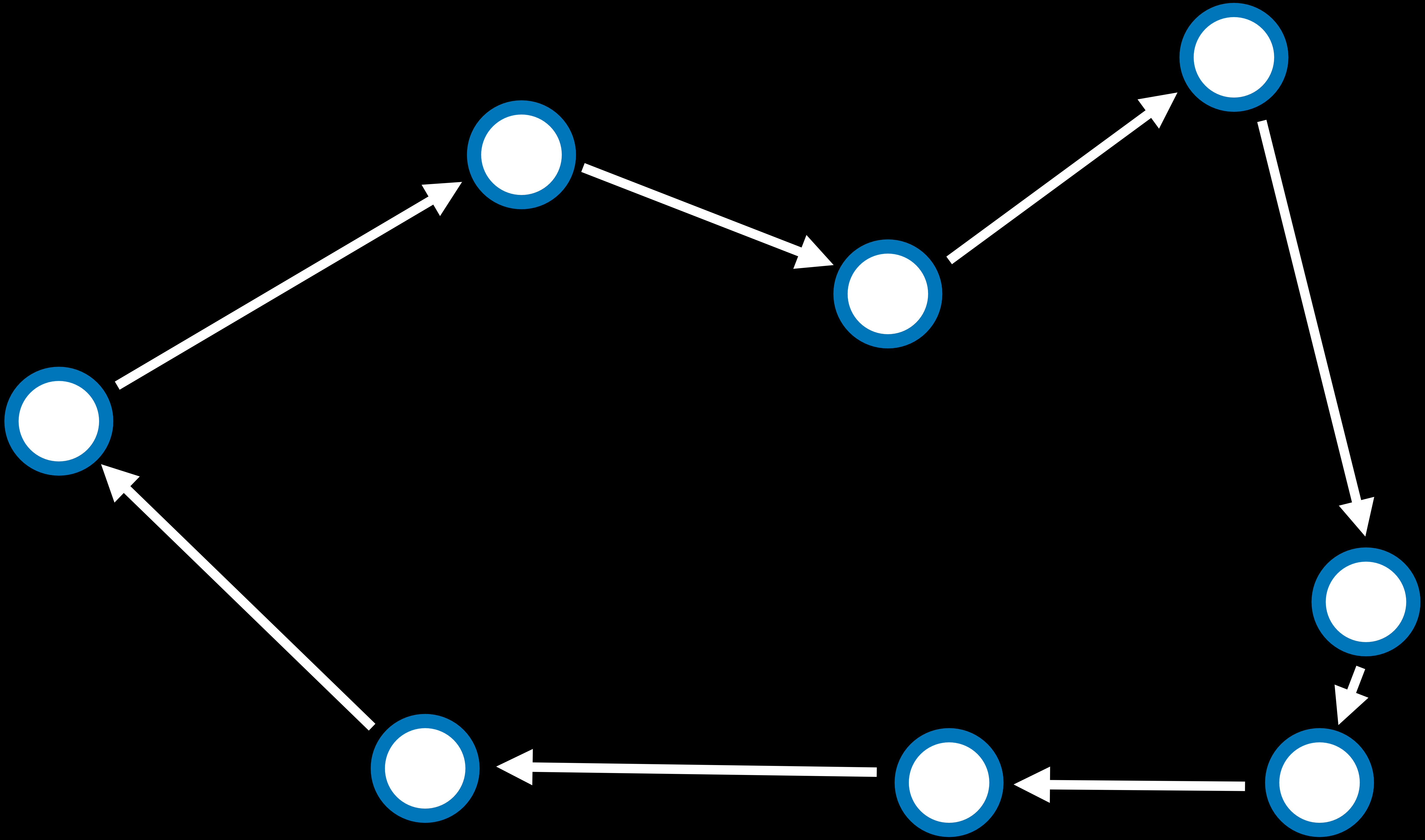












Optimization

Introduction to Artificial Intelligence with Python