Dziel i zwyciężaj!

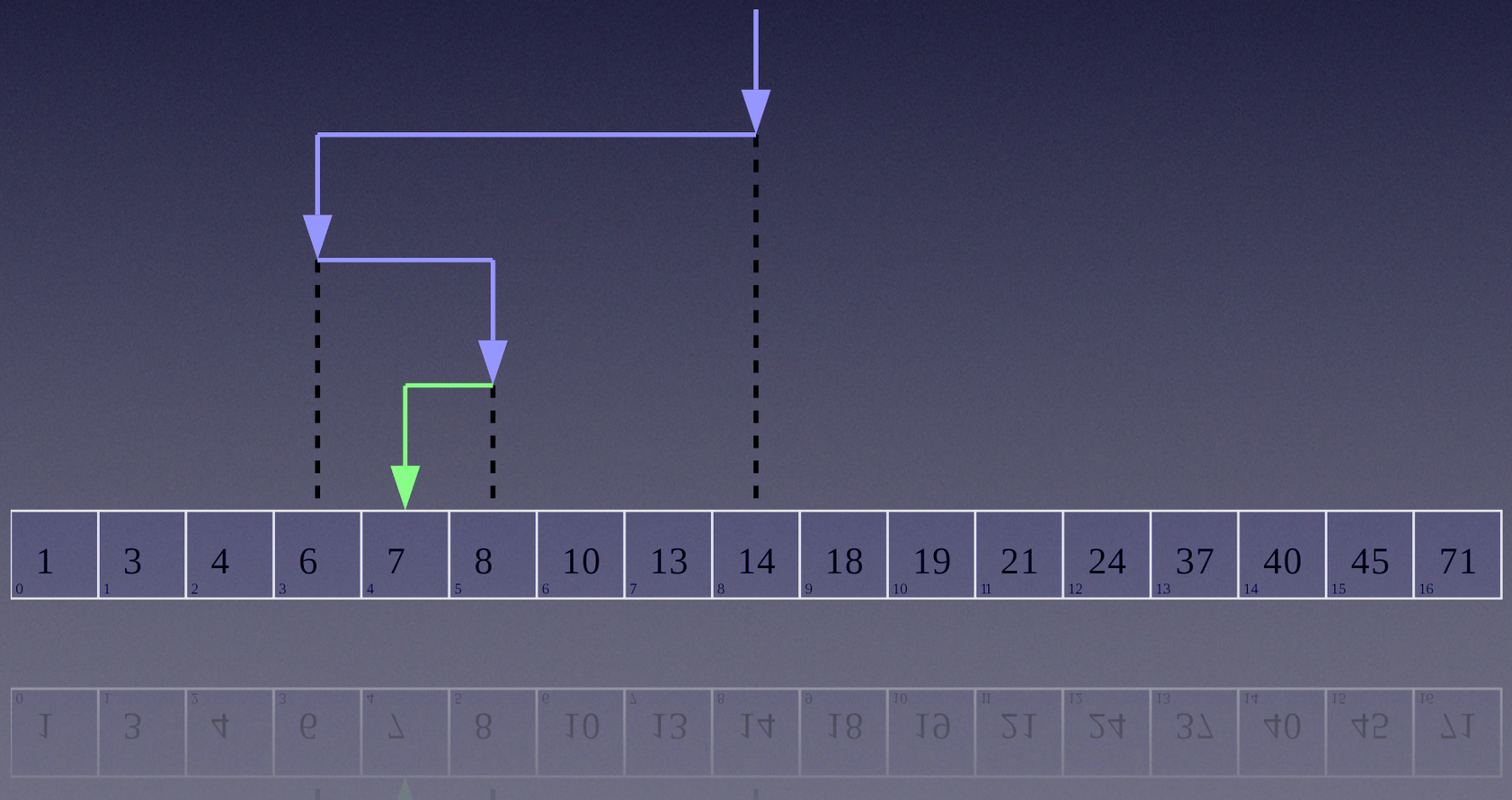
Kamil Kozioł

O co chodzi?

Chodzi o to aby problem dzielić na dwa lub więcej mniejszych podproblemów tak długo, aż fragmenty staną się wystarczająco proste.

Binary Search

- Dobrym przykładem na pokazanie działania jest właśnie binary search



Działanie Binary Search

- arr = [2,7,8,9,10]
- binarySearch(arr, 9) -> 3
- O(log(n))

Dlaczego?

- Założenie: Znalezienie liczby z 1 mld listy
- Zwykłe wyszukiwanie: MAX -> 1 000 000 000 porównań
- Binary Search: MAX 30 porównań

Merge Sort

O wiele lepszy niż Bubble Sort

Działanie

6 5 3 1 8 7 2 4

Złożoność: O(n*log(n))

Merge Sort

Lista Kroków

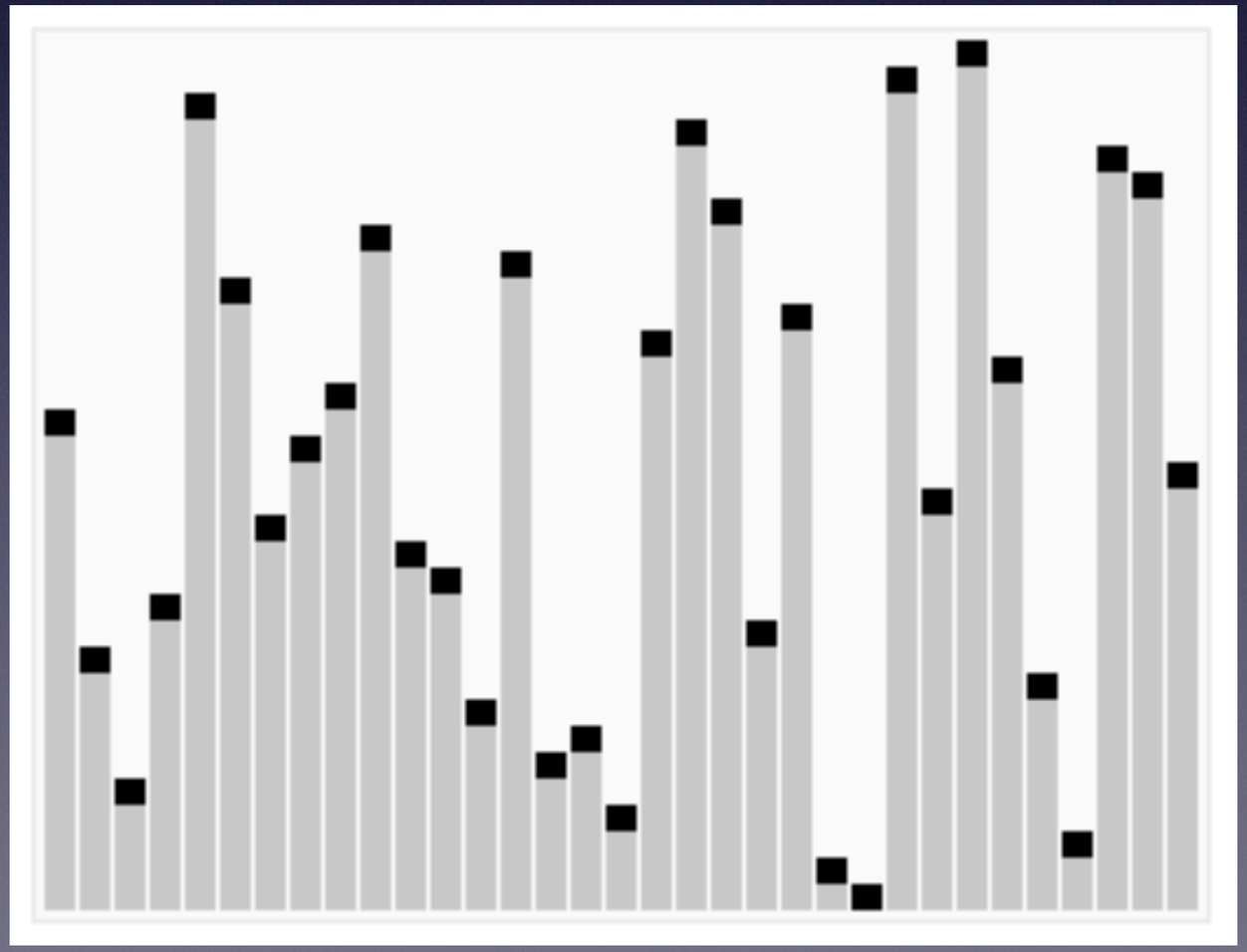
- Podziel zestaw na 2 części
- Zastosuj Merge Sort dla każdej z nich oddzielnie, chyba, że pozostał tylko jeden element
- Połącz posortowane zestawy w jeden ciąg posortowany

Quick Sort

Dla antyfanów mergesort

Quick Sort

Algorytm sortowania szybkiego jest wydajny: jego średnia złożonośc obliczeniowa jest rzędu O(n*log(n)). Jego implementacje znajdują się w bibliotekach standardowych wielu środowisk programowania



Koniec

link do kodów źródłowych:
https://github.com/kamil-koziol/python-nauka/tree/master/Algorytmy