

ADP

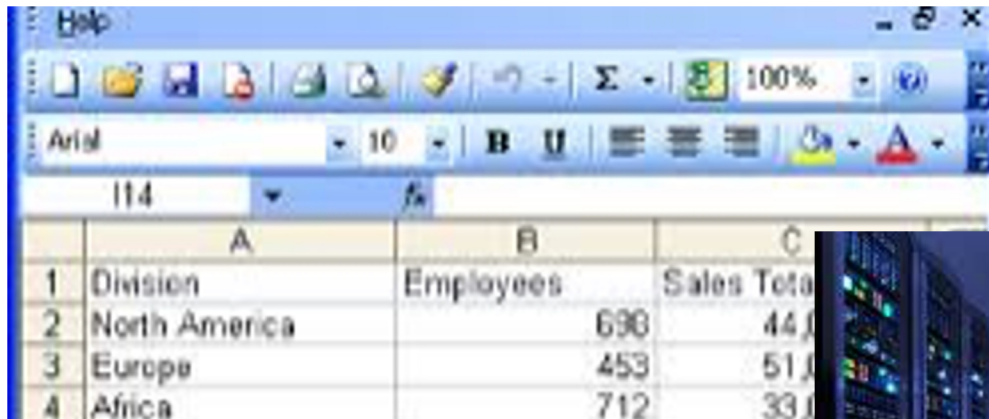
8. Sorting

Lesdoelen

- Het belang van sorteren uit kunnen leggen.
- **Selection sort** kunnen toepassen en uitprogrammeren.
- **Insertion sort** kunnen toepassen en uitprogrammeren.
- **Merge sort** kunnen toepassen en uitprogrammeren.
- **Quick sort** kunnen toepassen en uitprogrammeren.

INTRODUCTIE

Sorteren.. Belangrijk?



A screenshot of a Microsoft Excel spreadsheet. The spreadsheet has three columns labeled A, B, and C. The data is as follows:

	A	B	C
1	Division	Employees	Sales Total
2	North America	690	44,000
3	Europe	453	51,000
4	Africa	712	33,000



Hoe doen ze het?

Insert-sort with Romanian folk dance

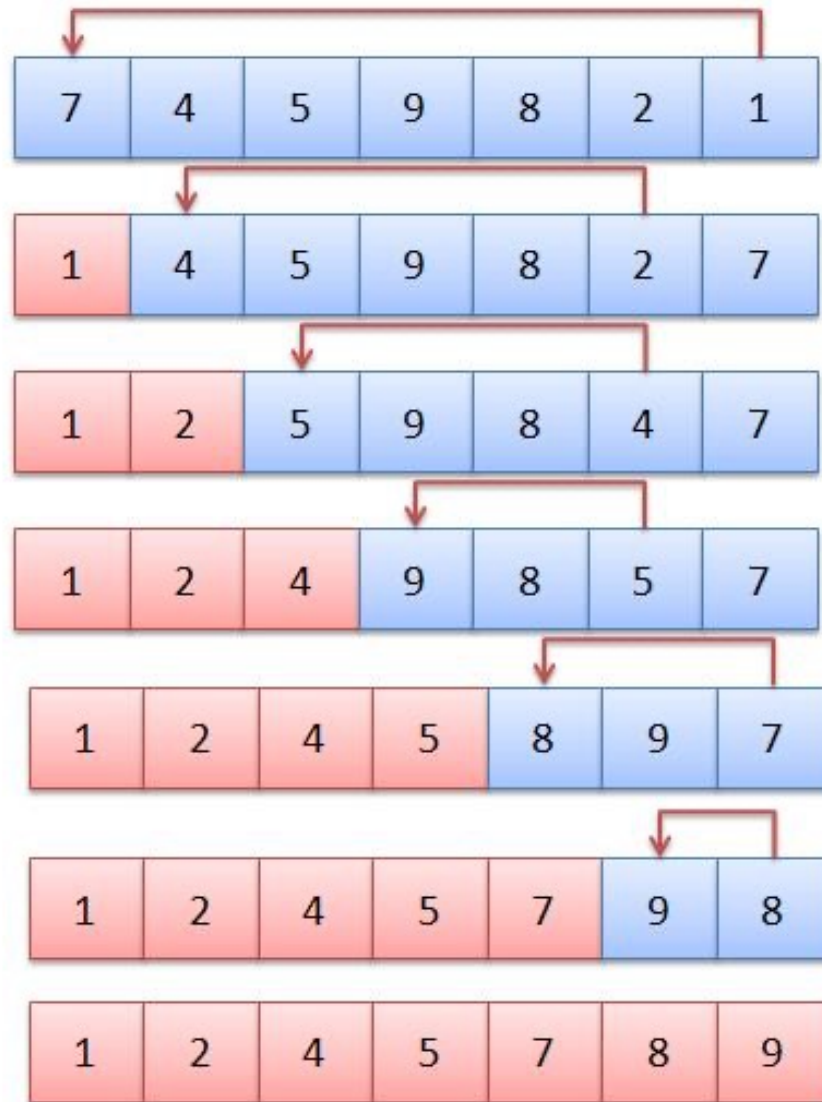
<https://www.youtube.com/watch?v=ROaIU379I3U>

SELECTION SORT

Selection Sort in Java

```
public static void selectionSort(int[] array) {  
    int n = array.length;  
    for (int i = 0; i < n - 1; i++) {  
        int minIndex = i;  
        for (int j = i + 1; j < n; j++) {  
            if (array[j] < array[minIndex]) {  
                minIndex = j;  
            }  
        }  
        int temp = array[minIndex];  
        array[minIndex] = array[i];  
        array[i] = temp;  
    }  
}
```

Voorbeeld



Selection Sort in Java

```
public static void selectionSort(int[] array) {  
    int n = array.length;  
    for (int i = 0; i < n - 1; i++) {  
        int minIndex = i;  
        for (int j = i + 1; j < n; j++) {  
            if (array[j] < array[minIndex]) {  
                minIndex = j;  
            }  
        }  
        int temp = array[minIndex];  
        array[minIndex] = array[i];  
        array[i] = temp;  
    }  
}
```

Complexity?

Selection sort

- Tijd: $O(N^2)$
- Snel voor kleine arrays

Speciale gevallen

Wat is de big Oh van Insertion sort wanneer

- Array al gesorteerd
- Array achterstevoren gesorteerd
- Array gevuld met dezelfde getallen

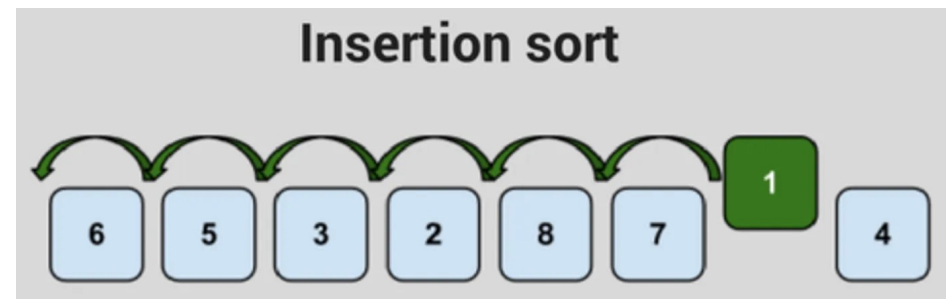
Oefenen met Selection sort

- Sorteert 8 6 0 7 5 3 1 handmatig met Selection sort en laat de tussenstappen zien.

INSERTION SORT

Insertion Sort in Java

```
public static void insertionSort(int[] a) {  
    for (int i = 1; i < a.length; i++) {  
        int toBeInserted = a[i];  
  
        int j = i;  
        while (j > 0 && toBeInserted < a[j - 1]) {  
            a[j] = a[j - 1];  
            j--;  
        }  
  
        a[j] = toBeInserted;  
    }  
}
```



Voorbeeld

figure 8.3

Basic action of
insertion sort (the
shaded part is sorted)

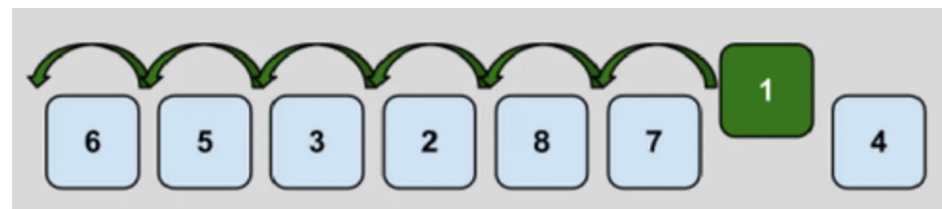
Array Position	0	1	2	3	4	5
Initial State	8	5	9	2	6	3
After a[0..1] is sorted	5	8	9	2	6	3
After a[0..2] is sorted	5	8	9	2	6	3
After a[0..3] is sorted	2	5	8	9	6	3
After a[0..4] is sorted	2	5	6	8	9	3
After a[0..5] is sorted	2	3	5	6	8	9

Insertion Sort in Java

```
public static void insertionSort(int[] a) {  
    for (int i = 1; i < a.length; i++) {  
        int toBeInserted = a[i];  
  
        int j = i;  
        while (j > 0 && toBeInserted < a[j - 1]) {  
            a[j] = a[j - 1];  
            j--;  
        }  
  
        a[j] = toBeInserted;  
    }  
}
```

Complexity?

kaarten in de hand sorteren



Insertion sort

- Tijd: $O(N^2)$
- Snel voor kleine arrays

Speciale gevallen

Wat is de big Oh van Insertion sort wanneer

- Array al gesorteerd
- Array achterstevoren gesorteerd
- Array gevuld met dezelfde getallen

Andere bekende $O(N^2)$ sorteeralgoritmes

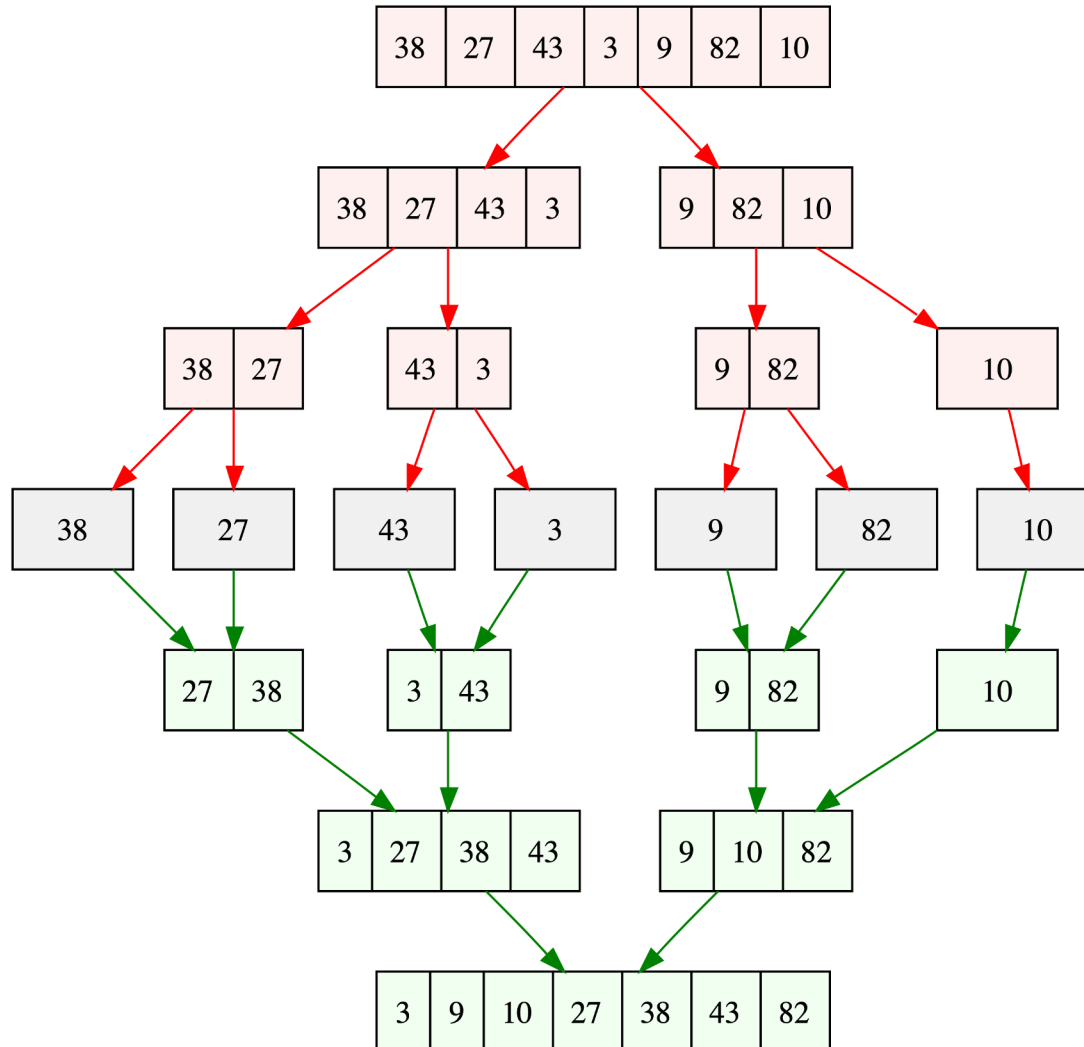
- Bubble sort (not in the course)

Oefenen met Insertionsort

- Sorteer 8 6 0 7 5 3 1 handmatig met Insertionsort en laat de tussenstappen zien.

MERGESORT

Voorbeeld



Merge sort (iets anders dan boek)

```
public static void mergeSort(int[] a) {  
    int[] tmpArray = new int[a.length];  
    mergeSort(a, tmpArray, 0, a.length-1);  
}
```

```
private static void mergeSort(int[] a, int[] tmpArray, int left, int right) {  
    if (left < right) {  
        int center = (left + right) / 2;  
        mergeSort(a, tmpArray, left, center);  
        mergeSort(a, tmpArray, center+1, right);  
        merge(a, tmpArray, left, center+1, right);  
    }  
}
```

Merge sort algoritme

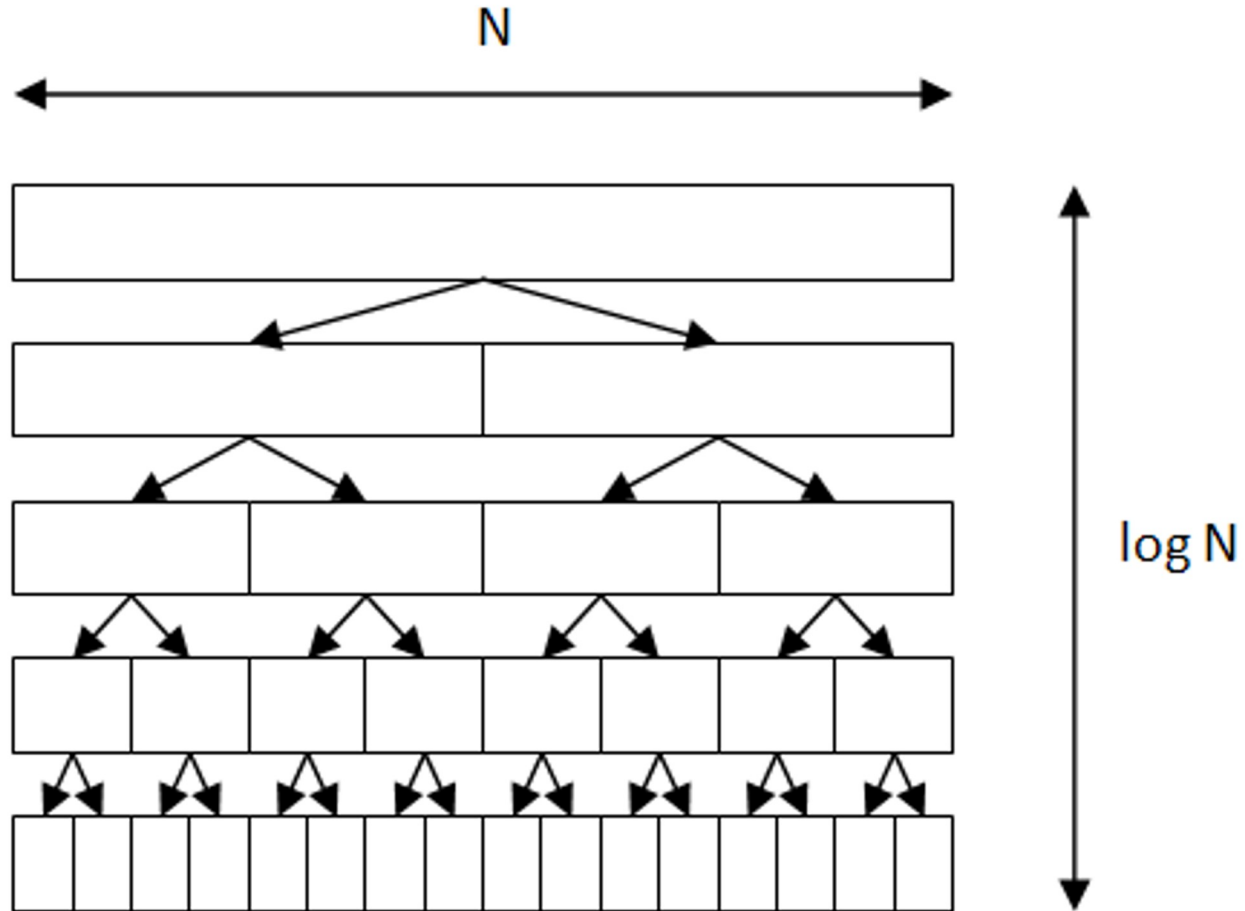
Recursief algoritme!

1. **Als het aantal te sorteren elementen ≤ 1 stop.**
2. **Sorteer linkerhelft recursief.**
3. **Sorteer rechterhelft recursief.**
4. **Merge linker- en rechterhelft.**

Merge

Wat moet er gebeuren in de merge?

Mergesort is $O(N \log N)$



Analyse Merge sort

- *Divide and conquer*-algoritme
- Tijdsduur $O(N \log N)$.
- Merge routine gebruikt extra array (dus 2x zoveel geheugenruimte nodig).
- Veel kopieeracties is duur.
- Minste aantal vergelijkingen nodig (bij object vergelijkingen is dit een voordeel).
- Voor primitieve types is Quicksort in het voordeel.

Speciale gevallen

Wat is de big Oh van Merge sort wanneer

- Array al gesorteerd
- Array achterstevoren gesorteerd
- Array gevuld met dezelfde getallen
- Let op!
 - Er zwerven “elegante” recursieve versies van Merge sort op internet rond, waarin de tmpArray in elke recursieve aanroep opnieuw wordt gemaakt.
 - Dan heb je $O(n \log n)$ geheugen nodig!

Oefenen met Merge sort

- Sorteert 8 6 0 7 5 3 1 handmatig met Merge sort en laat de tussenstappen zien.

Merge sort (extra)

- Handig voor opdracht

[Video](#)

0th Version Naive Parallelization

- $\Theta(n)$ runtime
- $\Theta(n \lg n)$ space

David Taylor: Algorithms with Azubu, 2020 Merge Sort

QUICKSORT

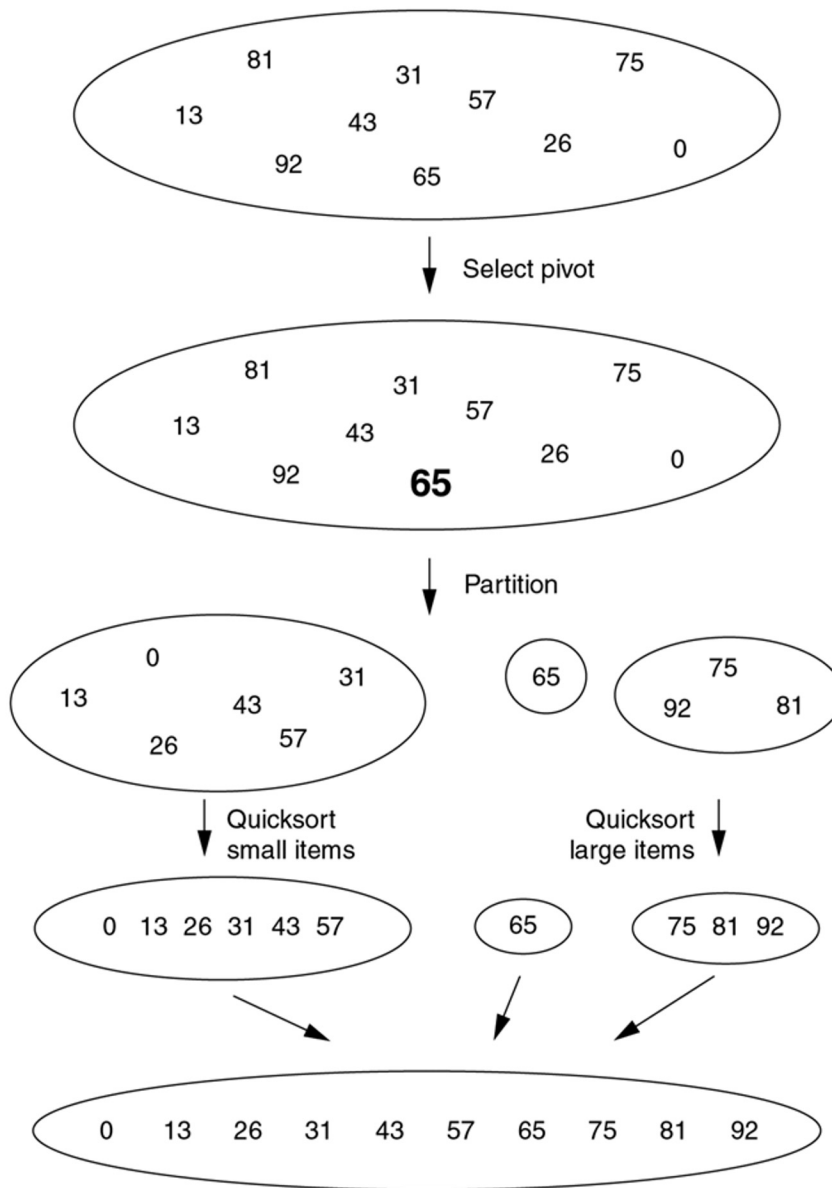


figure 8.10

The steps of quicksort

Quicksort algoritme

- **Rekursief!**
- **voor Quicksort(S) geldt:**
 1. **Als het aantal elementen in S is 0 of 1: stop**
 2. **Kies een element uit S, noem dit de *pivot***
 3. **Partition (splits) S in twee groepen. Een groep kleiner dan *pivot* (L), en een groep groter dan *pivot* (R)**
 4. **Return: Quicksort(L) + pivot + Quicksort (R)**

Quicksort algoritme

```
public static void quickSort(int[] array, int low, int high) {  
    if (low < high) {  
        int pivotIndex = partition(array, low, high);  
        quickSort(array, low, pivotIndex - 1);  
        quickSort(array, pivotIndex + 1, high);  
    }  
}
```

Quicksort algoritme

```
private static int partition(int[] array, int low, int high) {  
    int pivot = array[high];  
    int i = low - 1;  
  
    for (int j = low; j < high; j++) {  
        if (array[j] < pivot) {  
            i++;  
            swap(array, i, j);  
        }  
    }  
  
    swap(array, i + 1, high);  
    return i + 1;  
}
```

```
private static void swap(int[] array, int i, int j) {  
    int temp = array[i];  
    array[i] = array[j];  
    array[j] = temp;  
}
```

Pivot problemen

- Algoritme werkt het beste als pivot de array in twee gelijke delen splitst.
- Is de pivot steeds het grootste of kleinste element dan is de looptijd $O(N^2)$
- Komt statistisch gezien niet vaak voor gelukkig
- Kunnen we slim een pivot kiezen?

Pivot kiezen (1)

- Eerste element uit input kiezen? Geen goed idee... waarom niet?

Pivot kiezen (2)

- Goede keuze: random of middelste

Pivot kiezen (3)

- Beste keuze: Median of three. Kies de mediaan van het eerste, middelste en laatste element.
- 1 8 7 3 3 4 7 8 0 7 2
- De pivot wordt dus de mediaan van 1, 4 en 2 -> 2 dus.

Oefenen met Quicksort

- Sorteer 8 6 0 7 5 3 1 handmatig met Quicksort (Median of Three) en laat de tussenstappen zien

Merge sort vs Quick sort

- **Time Complexity:** Average-case time complexity of $O(n \log n)$.
- **Space Complexity:** can be implemented in-place. Requires less additional memory compared to Merge Sort (advantageous for primitive types where memory efficiency is often a concern.)

Merge sort vs Quick sort


- **Time Complexity:** guarantees a consistent $O(n \log n)$ time complexity in the worst-case scenario
- **Space Complexity:** uses additional memory proportional to the size of the input.
- **Stability:** is a stable sorting algorithm, meaning it maintains the relative order of equal elements in the sorted output.

Quicksort (extras)

[Video Link](#)

Optimization: Hoare's Partition

- Using pivot from middle (not required)
- Partition of small elements on left



David Taylor, Algorithms with Attitude, 2009

QuickSort and QuickSelect

APP: Algoritmen en Datastructuren

IMPLEMENTATIE IN JAVA SDK

Veel sorteeralgoritmen

- <https://www.toptal.com/developers/sorting-algorithms>
- De beste voor specifieke omstandigheden zijn geïmplementeerd in Java:
 - Alle kleine arrays: insertionsort
 - Objectreferenties (grotere arrays): mergesort
 - Primitieve arrays (grotere arrays): quicksort

Overzicht sorteeralgoritmes

- <https://www.toptal.com/developers/sorting-algorithms>
- <https://www.geeksforgeeks.org/sorting-algorithms/>
- <https://brilliant.org/wiki/sorting-algorithms/>
- <https://visualgo.net/bn/sorting?slide=1>
- http://en.wikipedia.org/wiki/Sorting_algorithm
- <https://www.youtube.com/user/AlgoRythmics/videos>

APP: Algoritmen en Datastructuren

AFSLUITING

Lesdoelen

- Het belang van sorteren uit kunnen leggen.
- **Selection sort** kunnen toepassen en uitprogrammeren.
- **Insertion sort** kunnen toepassen en uitprogrammeren.
- **Merge sort** kunnen toepassen en uitprogrammeren.
- **Quick sort** kunnen toepassen en uitprogrammeren.