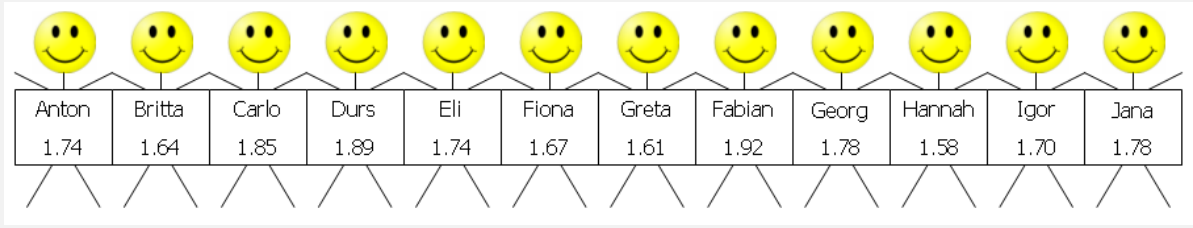
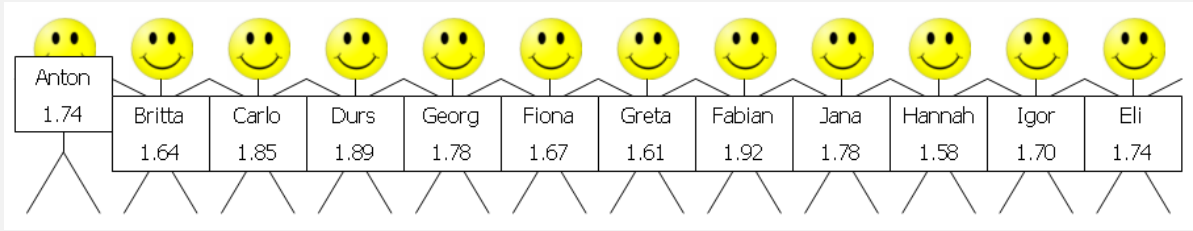
QUICKSORT

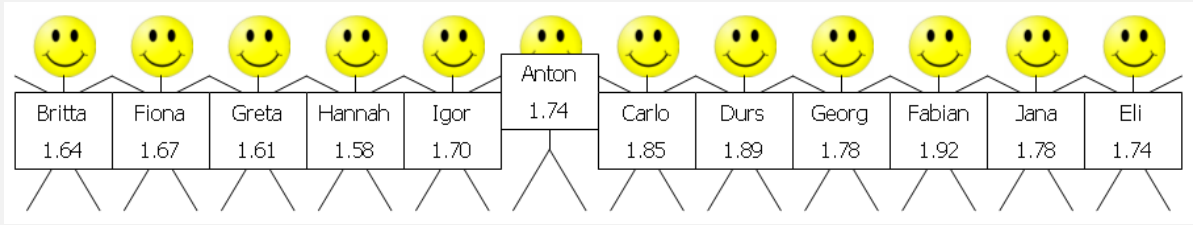
Sorting Done Fast

PRINZIP – TEILE UND HERRSCHE

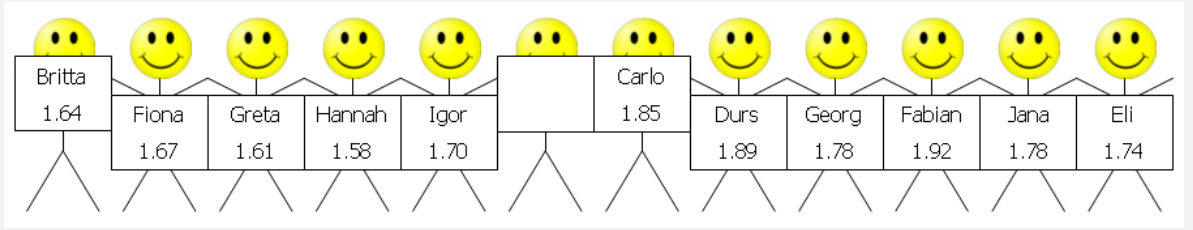




Pivotelement suchen

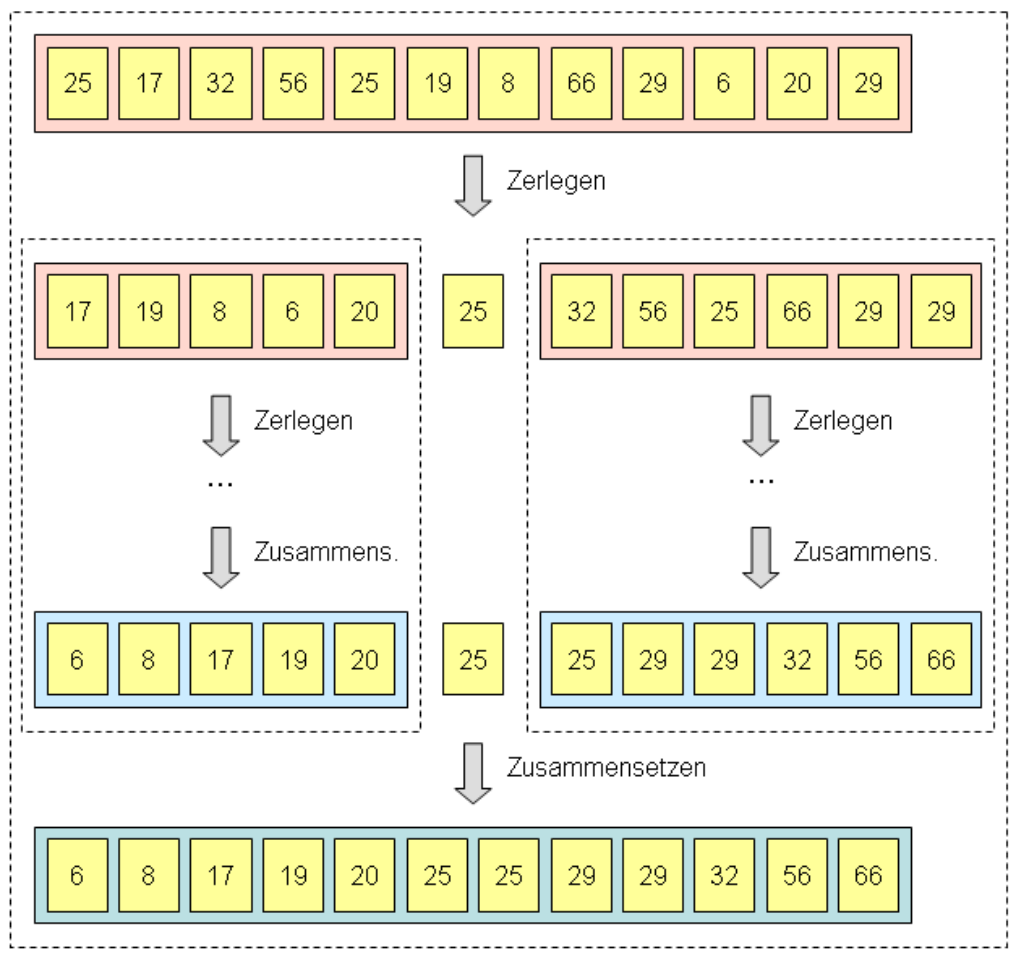


Partitionieren



Mit Teillisten weiterverfahren und zusammensetzen

IDEE


25 17 32 56 25 19 8 66 29 6 20 29
Zerlegen
17 19 8 6 20
25
32 56 25 66 29 29
Zerlegen
...
Zusammens.
Zerlegen
...
Zusammens.
6 8 17 19 20
25
25 29 29 32 56 66
Zusammensetzen
6 8 17 19 20 25 25 29 29 32 56 66

ÜBUNG

- Führe das Sortierverfahren „von Hand“ mit folgender Liste durch:
- 35 28 41 7 14 50 33 21 21 60 18 12
- Das Pivotelement soll immer das erste der Liste sein

```
ALGORITHMUS quicksort
Übergabe: Liste L
wenn die Liste L mehr als ein Element hat:
    # zerlegen
    wähle als Pivotelement p das erste Element der Liste aus
    erzeuge Teillisten K und G aus der Restliste (L ohne p) mit:
    - alle Elemente aus K sind kleiner als das Pivotelement p
    - alle Elemente aus G sind größergleich als das Pivotelement p
    # Quicksort auf die verkleinerten Listen anwenden
    KSortiert = quicksort(K)
    GSortiert = quicksort(G)
    # zusammensetzen
    LSortiert = KSortiert + [p] + GSortiert
sonst:
    LSortiert = L
Rückgabe: LSortiert
```

```python
def qsort(Liste):
    if len(Liste)>1:
        pivot=Liste[0]
        Liste.remove(pivot)
        Llist,RList=partition(Liste, pivot)
        #Rekursive Anwendung von
         Quicksort auf die linke und rechte Teilliste
        RechtsSortiert=qsort(RList)
        LinksSortiert=qsort(Llist)
        #Liste Zusammensetzen
        #LSortiert=[]
        LinksSortiert=LinksSortiert+[pivot]
        LSortiert=LinksSortiert+RechtsSortiert
    else:
        LSortiert=Liste
    return LSortiert
```

```python
def partition(L, p): #L -Liste p - Pivotelement
    RList=[]
    LList=[]
    for ele in L:
        if ele<p:
            LList.append(ele)
        else:
            RList.append(ele)
    return LList, RList
```