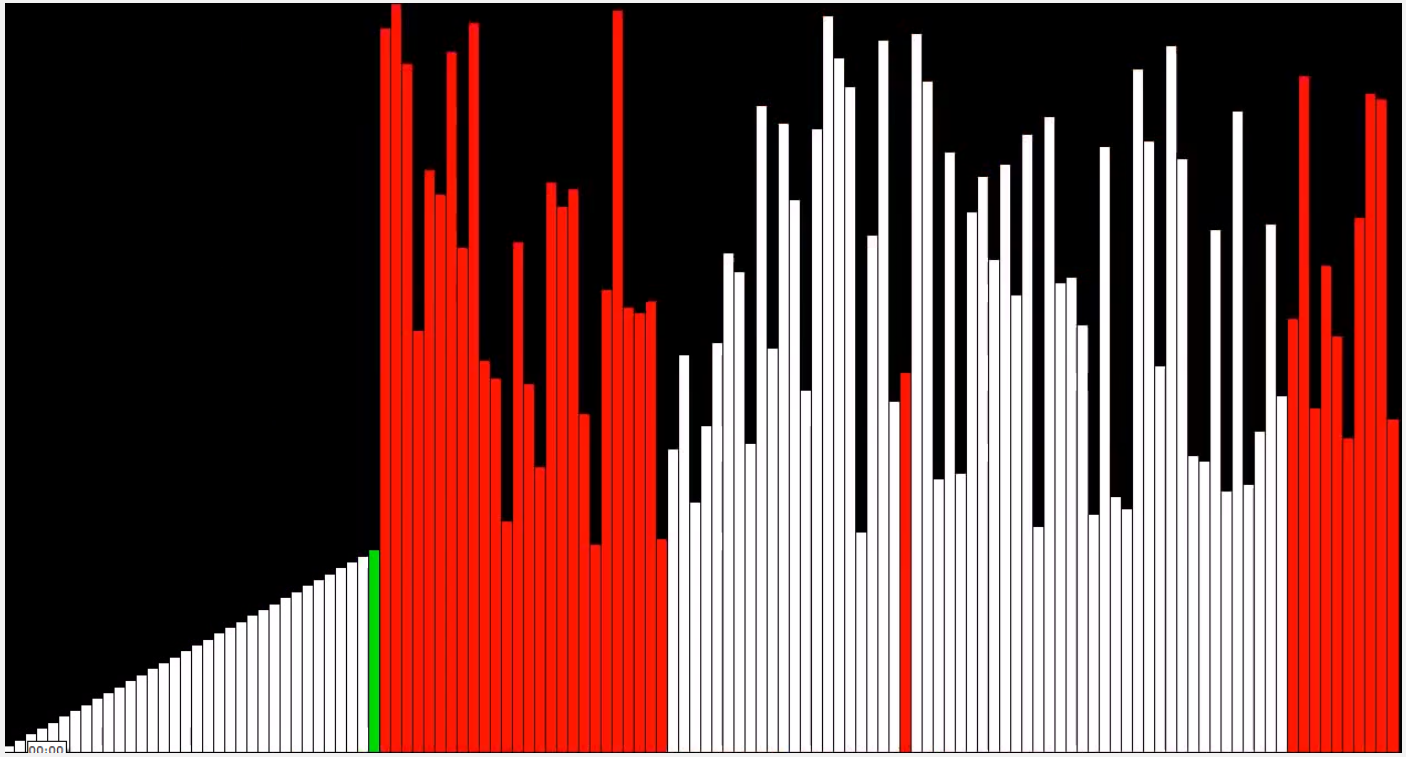
SORTIEREN

#OrdnungMussSein #esistDeutschlandhier

ZUM WARMWERDEN



SORTIEREN

- Sortierverfahren
  - auf Listen, Feldern …
  - Ordnung muss bekannt sein (Größe, lexikographisch…)
  - Unterschiede in Effizienz, Platzbedarf ...

IDEEN?

- [206, 366, 623, 638, 160, 22, 14, 163, 112, 311, 71, 825, 467, 960, 775, 598, 879, 653, 790, 19]

SORTIEREN DURCH EINFÜGEN – INSERTIONSORT (NAIVE IDEE)

Idee

- geg.: Liste1 – unsortiert, Liste2 – leer

1. Suche das Minimum in Liste1
2. Füge das Minimum in Liste2 ein
3. Lösche das Minimum aus Liste1
4. Führe Schritte 1) -3) durch, bis Liste1 leer
5. Gib Liste2 aus

Aufgabe: Implementiere das Verfahren!

Hilfe: Minimumsuche kann „abgeschaut“ werden. (Selber überlegen macht aber schlauer!)

LÖSUNG

```
def findMin(Liste):
    aktMin=Liste[0]
    for element in Liste:
        if element<aktMin:
            aktMin=element
    return aktMin

def MinSort(Liste):
    ErgListe=[]
    for i in range(0,len(Liste)):
        min=findMin(Liste)
        ErgListe.append(min)
        Liste.remove(min)
    return ErgListe
```

Vorteile:

- einfach

Nachteile:

- langsam
- viel Speicherbedarf (2 Listen)

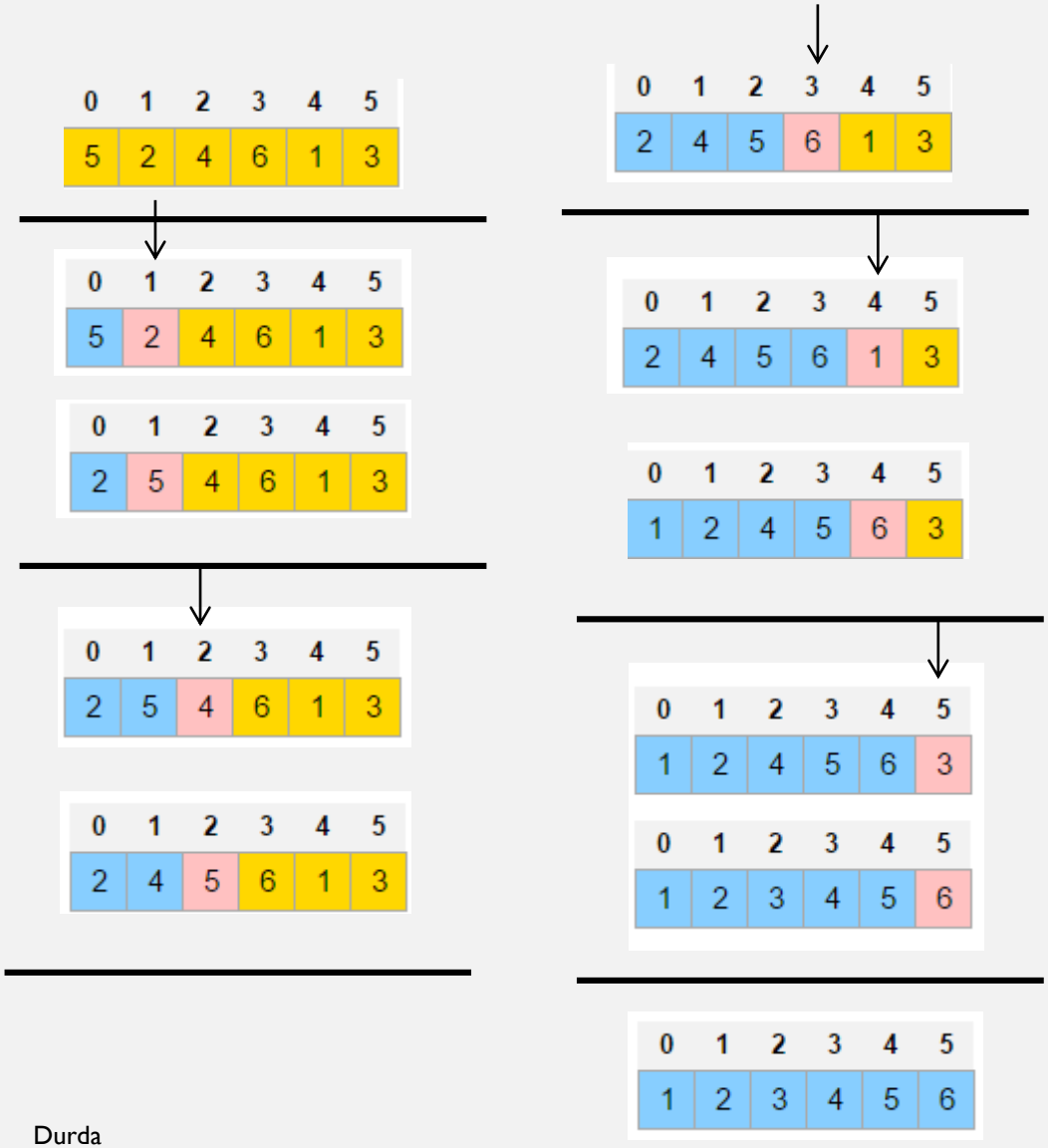
→ Aufgabe:

Wir ändern das Verfahren so, dass nicht eine neue Liste benötigt wird, sondern nur eine Liste umsortiert wird.

BÜCHERREGALE

- Ihr steht zuhause vor eurem (natürlich) unaufgeräumten Bücherregal.
- Ihr beschließt in einem Anfall von Langeweile und Verrücktheit, die Bücher zu ordnen.
- Prämisse: Ihr habt keinen Ebook-Reader.
- Beschreibt euer Vorgehen!

INSERTION SORT



Insertionsort entnimmt der unsortierten Eingabefolge ein beliebiges Element und fügt es an richtiger Stelle in die (anfangs leere) Ausgabefolge ein. Da wir auf einer Liste arbeiten, müssen die Elemente hinter dem neu eingefügten Element verschoben werden. Dies ist die aufwändige Operation von Insertionsort. *(Wikipedia)*

Metapher:
Sortieren eines Kartenspiels

INFORMELLE BESCHREIBUNG

Algorithmus Insertionsort
Übergabe: Liste
sortierter Bereich besteht aus erstem Element
unsortierter Bereich ist Restliste (ohne erstes Element)

SOLANGE der unsortierte Bereich Elemente hat:

1. entferne das erste Element aus dem unsortierten Bereich
2. füge es an der richtigen Stelle im sortierten Bereich ein

Rückgabe: sortierter Bereich

STRUKTOGRAMM, PSEUDOCODE

Zähle i **von** 1 **bis** n-1

- einzusortierender_wert = A[i]
- j = i
- **Solange** j > 0 und A[j-1] > einzusortierender_wert
  - A[j] = A[j-1]
  - j = j - 1
- A[j] = einzusortierender_wert

Einsortieren

```
INSERTIONSORT(A)
1 for i = 1 to (Länge(A)-1) do
2       einzusortierender_wert = A[i]
3       j = i
4       while (j > 0) and (A[j-1] > einzusortierender_wert) do
5             A[j] = A[j - 1]
6             j = j - 1
7       end while
8       A[j] = einzusortierender_wert
9 end for
```

Quelle: Wikipedia, https://de.wikipedia.org/wiki/Insertionsort

LÖSUNG

```
from fill import fill_besser

def insertionSort(Liste):
    for i in range(1,len(Liste)):
        wert=Liste[i]
        j=i
        while (j>0) and (Liste[j-1]>wert):
            Liste[j]=Liste[j-1]
            j=j-1
        Liste[j]=wert
    return Liste


Liste=fill_besser(10,100)
print(Liste)

Liste=insertionSort(Liste)
print(Liste)
```