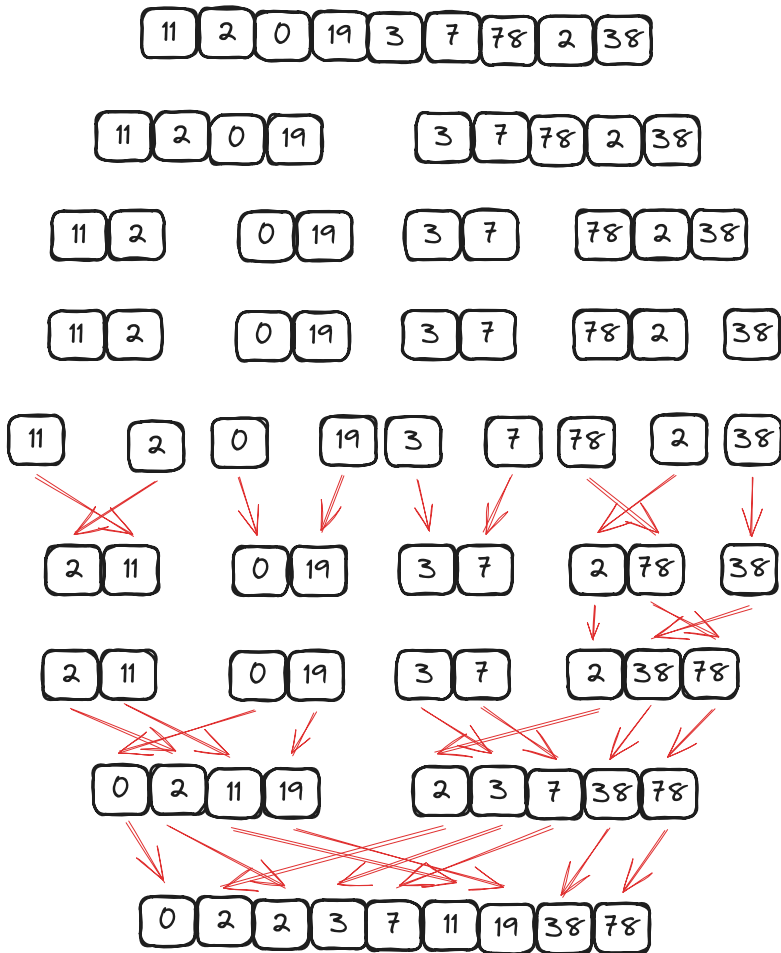


AuD Übung 06

[PDF](#)**10****1.**

```
def find_maximum(arr):  
    # Base case: if the array contains only one element, return that element  
    if len(arr) == 1:  
        return arr[0]  
  
    # Divide the array into two halves  
    mid = len(arr) // 2  
    left_half = arr[:mid]  
    right_half = arr[mid:]  
  
    # Recursively find the maximum in both halves  
    left_max = find_maximum(left_half)  
    right_max = find_maximum(right_half)  
  
    # Conquer: return the maximum of the two halves  
    return max(left_max, right_max)  
  
L = [-3, 5, 1, -6, 10, 4, -7, 8]  
print(find_maximum(L))
```

Copy
Run**3.****a)**



b)

1. **Initiale Folge:** (11, 2, 0, 19, 3, 7, 78, 2, 38)

- Wählen des Pivot-Elements, nehmen wir z.B. das erste Element (11).
- Teilfolge F1 (echt kleiner als 11): (2, 0, 3, 7, 2)
- Teilfolge F2 (größer oder gleich 11): (19, 78, 38)

2. **Sortieren von F1:** (2, 0, 3, 7, 2)

- Pivot-Element 2.
- Teilfolge F1: (0)
- Teilfolge F2: (3, 7, 2)
- Rekursives Sortieren von F1 (0) und F2 (3, 7, 2).

3. **Sortieren von F2 (3, 7, 2):**

- Pivot-Element 3.
- Teilfolge F1: (2, 2)
- Teilfolge F2: (7)
- Rekursives Sortieren von F1 (2, 2) und F2 (7).

4. **Rekursion beendet:**

- F1 (0) ist sortiert.
- F1 (2, 2) ist sortiert.
- Zusammensetzen: (0, 2, 2, 3, 7)

5. Sortieren von F2: (19, 78, 38)

- Pivot-Element 19.
- Teilfolge F1: ()
- Teilfolge F2: (78, 38)
- Rekursives Sortieren von F1 () und F2 (78, 38).

6. Sortieren von F2 (78, 38):

- Pivot-Element 78.
- Teilfolge F1: (38)
- Teilfolge F2: ()
- Zusammensetzen: (38, 78)

7. Rekursion beendet:

- F1 () ist sortiert.
- Zusammensetzen: (19, 38, 78)

8. Finales Zusammensetzen:

- F1 sortiert: (0, 2, 2, 3, 7)
- Pivot-Element: 11
- F2 sortiert: (19, 38, 78)