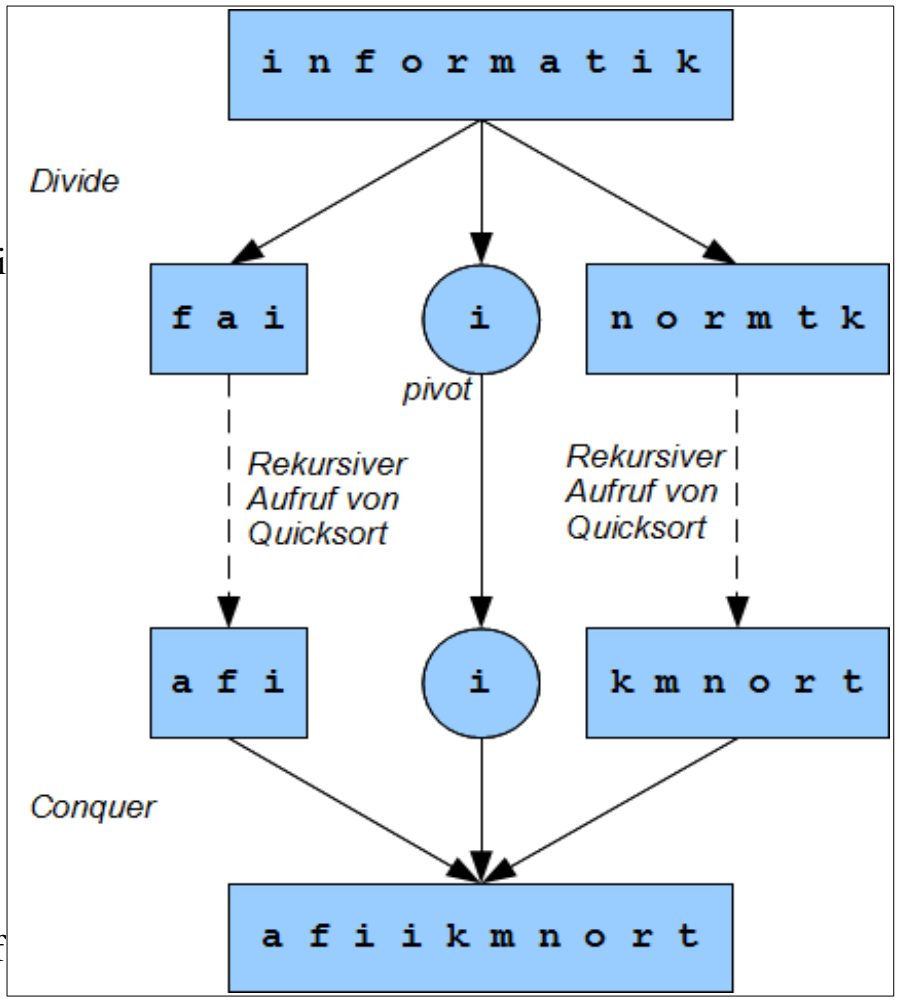
Quicksort



Quicksort arbeitet nach dem **Divide-and-Conquer-Prinzip**:

Zunächst wird die zu sortierende Liste in zwei Teillisten („linke“ und „rechte“ Teilliste) getrennt (=**Divide**). Dazu wählt Quicksort ein sogenanntes **Pivotelement** aus der Liste aus. Alle Elemente, die kleiner als das Pivotelement sind, kommen in die linke Teilliste, und alle, die größer sind, in die rechte Teilliste. Die Elemente, die gleich dem Pivotelement sind, können sich beliebig auf die Teillisten verteilen. Nach der Aufteilung sind die Elemente der linken Liste kleiner oder gleich den Elementen der rechten Liste.

Anschließend wird jede Teilliste in sich sortiert, indem der Quicksort-Algorithmus **rekursiv** auf die linke und die rechte Teilliste angewendet werden. Wenn eine Teilliste nur ein oder kein Element enthält, bricht die Rekursion ab, d. h. für diese Teilliste wird Quicksort nicht mehr aufgerufen.

Am Ende werden die beiden Teillisten und das Pivot-Element zusammengeführt (=**Conquer**).

Implementierung von Quicksort

Fülle die Lücken!

```
public List<Celebrity> quicksort(List<Celebrity> pList){
        List<Celebrity> ergebnis = new List<Celebrity>();
        // Abbruchbedingung
        _____________________________________________________
        _____________________________________________________
        _____________________________________________________
        // das erste Element wird das Pivot-Element
        pList.toFirst();
        Celebrity pivot = pList.getContent();
        List<Celebrity> links = new List<Celebrity>();
        List<Celebrity> rechts = new List<Celebrity>();
        // die weiteren Elemente auf links u. rechts aufteilen
        pList.next();
        while(pList.hasAccess()){
            Celebrity c = pList.getContent();
            _____________________________________________________
            _____________________________________________________
            _____________________________________________________
            _____________________________________________________
            _____________________________________________________
            _____________________________________________________
            pList.next();
        }
        // rekursive Aufrufe fuer links und rechts
        _____________________________________________________
        _____________________________________________________
        // das Ergebnis zusammenbauen
        _____________________________________________________
        _____________________________________________________
        _____________________________________________________
        return ergebnis;
    }
```