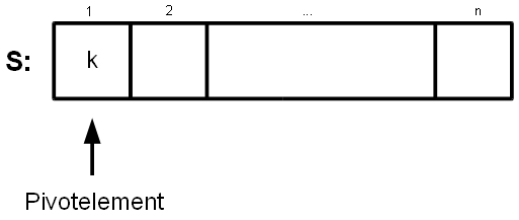
# Quicksort

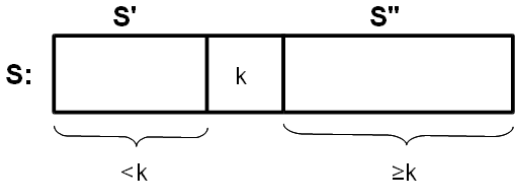
Der Name Quicksort (von engl. „schnell“ und to sort „sortieren“) ist dadurch motiviert, dass Quicksort in der Praxis das schnellste Sortierverfahren ist. Ausserdem ist Quicksort ein Sortierverfahren, welches ohne zusätzlichen Speicherplatz auskommt („in-place“).

## Idee:

1. Wähle ein beliebiges Element $k$ (das Pivotelement) aus dem zu sortierenden Array $S$, z.B. das erste Element $S[1]$.




2. Die Elemente von $S$ werden durch Vertauschungen so gespeichert, dass zuerst alle Elemente die kleiner als $k$ sind in $S$ stehen, dann $k$ und anschließend alle Elemente größer oder gleich $k$.
   Diesen Schritt nennt man Partitionierung.



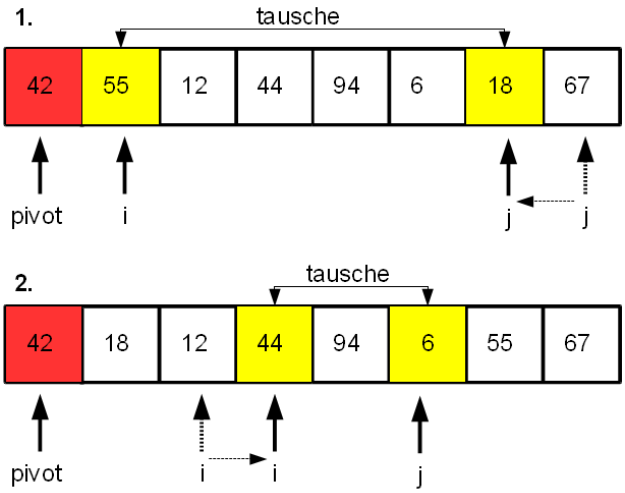
3. Wende das Verfahren rekursiv auf den vorderen Abschnitt $S'$ und den hinteren Abschnitt $S''$ an.

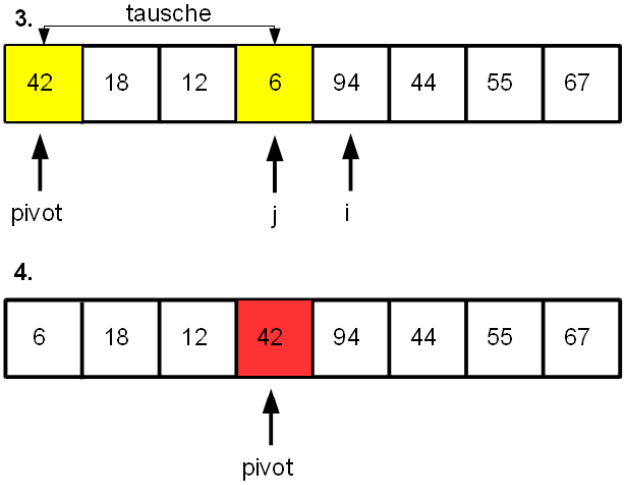
# Implementierung:

Paritionierung:

```
i:=2; j:=n; k:=S[1];
repeat
    while S[i] <= k do i:=i+1;
    while S[j] >  k do j:=j-1;
    if i < j then
        "vertausche S[i] und S[j]"
        i:=i+1;
        j:=j-1;
    fi;
until i > j;
"vertausche S[1] und S[j]"
```

**Beispiel:**






```
procedure quicksort(int l, int r)
(*sortiert das Teilfeld S[l...r] in aufsteigender Reihenfolge*)
{
    int i, j, k;

    begin
        i:=l+1; j:=r; k:=S[l];

        (*Partitionierung*)
            repeat ...
            until i>j;
            "vertausche S[l]und S[j]"

        (*rekursive Aufrufe für S' und S''*)
            if l < j-1    (* |S'| > 1 *)
            then quicksort(l,j-1);

            if r > j+1    (* |S''| > 1 *)
            then quicksort(j+1,r);
    end
}
```

Aufruf im Hauptprogramm: **quicksort(1,n)**

# Analyse von Quicksort

Kosten = Anzahl der Vergleiche zwischen den $S[i]$'s.

## Kosten im schlechtesten Fall

Die Kosten eines Aufrufes quicksort(l,r) sind

1. die unmittelbaren Kosten bei der Partitionierung:
   $O(r - l + 1)$ Vergleiche
2. die Kosten der rekursiven Aufrufe

Dann gilt:

$$QS(n) = \underbrace{n}_{\text{Partitionierung}} + \max_{1 \le j \le n} \{ \underbrace{QS(j-1)}_{\text{rek. Aufruf für S'}} + \underbrace{QS(n-j)}_{\text{rek. Aufruf für S''}} \}$$

$QS(0) = QS(1) = 0$

Behauptung: $QS(n) = \sum_{i=2}^{n} i = \frac{1}{2} n(n+1) = O(n^2)$

Beweis: durch Induktion. □

Bemerkung: An der Eingabe $[1, 2, 3, ..., n]$ führt Quicksort wirklich $O(n^2)$ Vergleiche durch. Die obere Schranke ist also scharf.

## Analyse des mittleren Verhaltens von Quicksort

Annahme:

1. Alle Elemente von $S$ sind paarweise verschieden.
2. Jede der $n!$ möglichen Permutationen der Eingabe ist gleich wahrscheinlich.

Dann gilt: Die mittlere Laufzeit von Quicksort ist $O(n \log n)$.

Wir können o.B.d.A annehmen, dass die Schlüssel die Zahlen $1, ..., n$ sind und dass $S[1] = k$ mit Wahrscheinlichkeit $\frac{1}{n}$ für $1 \leq k \leq n$.
Dann müssen rekursiv Teilprobleme der Größe $k-1$ und $n-k$ gelöst werden. Diese Teilprobleme sind wieder zufällige Folgen, d.h. sie erfüllen die beiden Argumente der Annahme.

Sei $\overline{QS}(n)$ die mittlere Anzahl der Vergleiche, die Quicksort an einem Feld der Größe $n$ durchführt.

Aus

$$prob(S[1] = k) = \frac{1}{n}$$

schließen wir

$$\overline{QS}(0) = \overline{QS}(1) = 0$$

und

$$\begin{aligned}
\overline{QS}(n) &= n + \text{Erwartungswert}(\overline{QS}(S') + \overline{QS}(S'')) \\
&= n + \frac{1}{n} \cdot \sum_{k=1}^{n} (\overline{QS}(k-1) + \overline{QS}(n-k)) \\
&= n + \frac{2}{n} \cdot \sum_{k=0}^{n-1} (\overline{QS}(k))
\end{aligned}$$

Wir multiplizieren beide Seiten mit $n$ und stellen die gleiche Gleichung auch für $n+1$ auf:

$$n \cdot \overline{QS}(n) = n^2 + 2 \sum_{k=0}^{n-1} \overline{QS}(k) \qquad (*)$$

$$(n+1) \cdot \overline{QS}(n+1) = (n+1)^2 + 2 \sum_{k=0}^{n} \overline{QS}(k) \qquad (**)$$

Um $QS(n+1)$ und $QS(n)$ miteinander in Verbindung zu bringen, subtrahieren wir Gleichung (*) von Gleichung (**):

$$\begin{aligned}
(n+1) \cdot \overline{QS}(n+1) - (n) \cdot \overline{QS}(n) &= (n+1)^2 - n^2 + 2 \cdot \overline{QS}(n) \\
&= 2n + 1 + 2 \cdot \overline{QS}(n) \qquad | + n \cdot \overline{QS}(n)
\end{aligned}$$

Durch Umformung erhalten wir:

$$\begin{aligned}(n+1)\cdot\overline{QS}(n+1) &= 2n+1+(n+2)\cdot\overline{QS}(n) \qquad |:(n+1)\\ \overline{QS}(n+1) &\leq 2+\frac{(n+2)}{(n+1)}\cdot\overline{QS}(n)\end{aligned}$$

Wir entwickeln die Formel $QS(n)$ ein weiteres mal und multiplizieren aus:

$$\begin{aligned}\overline{QS}(n+1) &= 2+\frac{(n+2)}{(n+1)}\left(2+\frac{n+1}{n}\left(QS(n-1)\right)\right)\\ &= 2+(n+2)\left(\frac{2}{n+1}+\frac{(n+1)}{n\cdot(n+1)}\left(QS(n-1)\right)\right)\\ &= 2+(n+2)\left(\frac{2}{n+1}+\frac{1}{n}\cdot\left(QS(n-1)\right)\right)\end{aligned}$$

Durch ein weiteres Entwickeln der Formel erhalten wir:

$$\begin{aligned}\overline{QS}(n+1) &= 2+(n+2)\left(\frac{2}{n+1}+\frac{1}{n}\cdot\left(QS(n-1)\right)\right)\\ &= 2+(n+2)\left(\frac{2}{n+1}+\frac{1}{n}\cdot\left(2+\frac{n}{n-1}\left(QS(n-2)\right)\right)\right)\\ &= 2+(n+2)\left(\frac{2}{n+1}+\left(\frac{2}{n}+\frac{1}{n-1}\left(QS(n-2)\right)\right)\right)\end{aligned}$$

Wir können jetzt erahnen, wie die Folge aussieht. Um sie korrekt herzuleiten, müsste ein Induktionsbeweis erfolgen. Uns reicht jedoch das „intuitive“ Ergebnis:

$$\begin{aligned}\overline{QS}(n+1) &= 2+(n+2)\left(\frac{2}{n+1}+\frac{2}{n}+\frac{2}{n-1}+...+1\right)\\ &= 2\cdot\left(1+(n+2)\cdot\left(\frac{1}{n+1}+\frac{1}{n}+\frac{1}{n-1}+...+\frac{1}{2}\right)\right)\end{aligned}$$

Dadurch können wir $QS(n)$ wie folgt abschätzen:

$$\begin{aligned}
\overline{QS}(n) &= 2 \cdot \left(1 + (n+2) \cdot \sum_{i=2}^{n+1} \frac{1}{i}\right) \\
&\leq 2 + 2 \cdot (n+2) \cdot \left(\sum_{i=2}^{n} \frac{1}{i}\right) \\
&\text{beginne die Summe bei } i = 1 \text{ und ziehe } \tfrac{1}{1} \text{ ab} \\
&= 2 + 2 \cdot (n+2) \cdot \left(\sum_{i=1}^{n} \frac{1}{i} - 1\right) \\
&\text{durch Ausmultiplizieren erhalten wir} \\
&= 2 + 2 \cdot (n+2) \cdot \sum_{i=1}^{n} \frac{1}{i} - 2 \cdot (n+2) \\
&= 2 \cdot (n+2) \cdot \underbrace{\sum_{i=1}^{n} \frac{1}{i}}_{\leq 1 + \int_1^n \frac{1}{x} dx = 1 + \ln n} - 2 \cdot n - 2 \\
&\text{Abschätzung der harmonischen Reihe ergibt} \\
&\leq 2 \cdot (n+2) \cdot (1 + \ln n) - 2 \cdot n - 2 \\
&\leq 2 \cdot (n+2) \cdot (1 + \ln n) \\
&= O(n \log n)
\end{aligned}$$

**Satz:**

Die mittlere Komplexität von Quicksort ist $O(n \log n)$.

Bemerkung:
Die Laufzeit von Quicksort ist sehr stark von der Wahl des Pivotelementes abhängig. Wie bereits erwähnt wird im bereits sortierten Fall die schlechtmöglichste Laufzeit $O(n^2)$ erreicht. Die beste Laufzeit wird erzielt, wenn das Pivotelement das zu sortierende Feld in zwei gleich große Teilfelder partitioniert.

Möglichkeiten für die Wahl des Pivotelements:

1. Wähle das erste, letzte, oder mittlere Element des zu sortierenden Feldes. (Nachteil: recht ineffizient.)

2. Vergleiche die drei oben genannten Elemente und wähle das Element mit dem mittleren Wert als Pivotelement.

3. Wähle ein zufälliges Element als Pivotelement (randomisiertes Quicksort). (Vorteil: Wahrscheinlichkeit, dass der Worst-Case Fall auftritt ist sehr gering!)