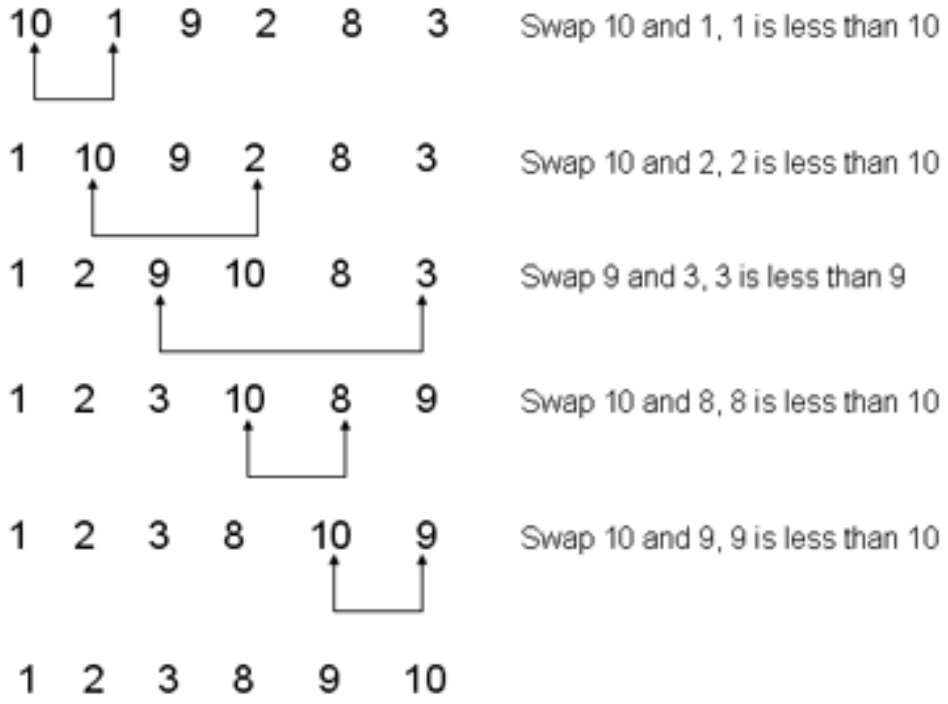
Algorithmen und Datenstrukturen

Aufgabe 2

Selection Sort

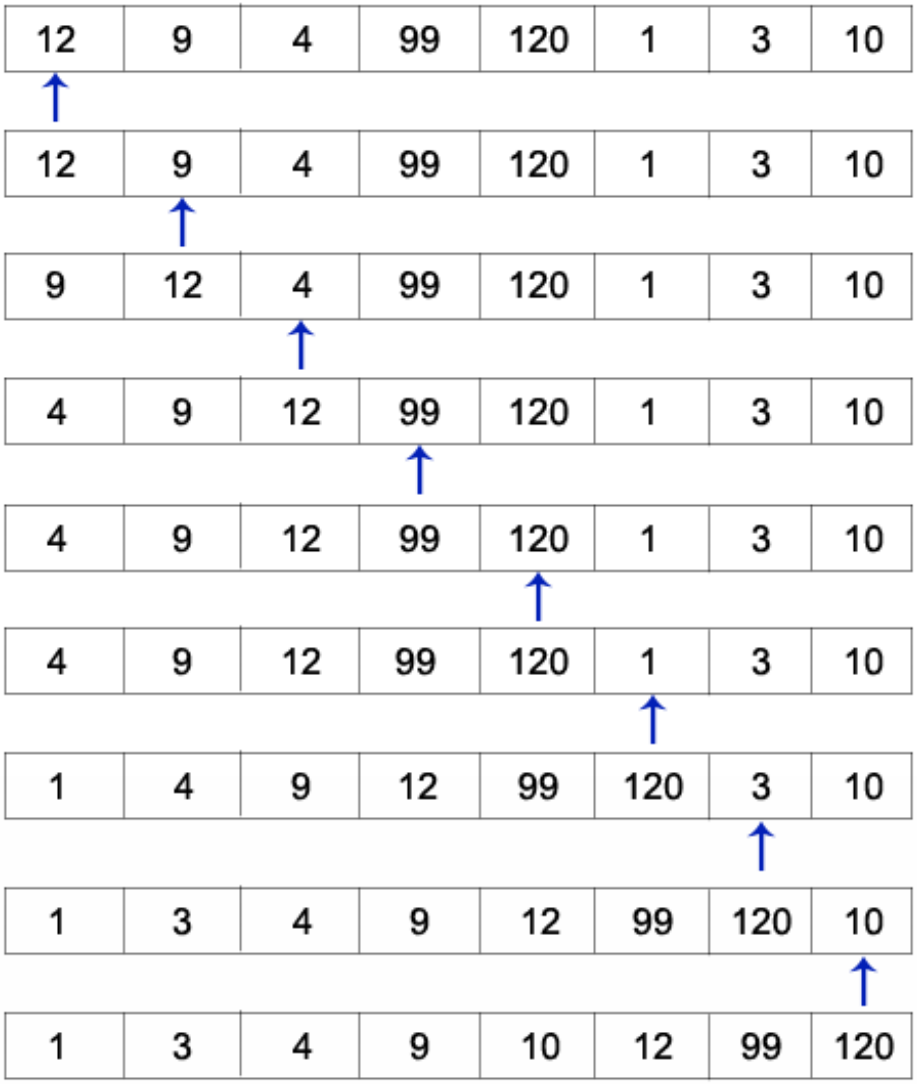
```
SelectionSort() {
    int minimum = 0;
    for ( int i = 1; i ≤ N-1 ; i++ ) {
        minimum = findMinStartingAt(i);
        swap(i,minimum);
    }
}

int findMinStartingAt(int i) {
    int min = i;
    for ( int j = i+1; j≤ N; j++) {
        if(a[j].key < a[min].key)
                min = j;
    }
}
```



Insertion Sort

```
InsertionSort() {
    for ( int i = 2; i ≤ N ; i++ ) {
        int j = i;
        Datensatz t = a[i];
        int k = t.key;
        while(a[j-1].key > k) {
            a[j] = a[j-1];
            j = j-1;
        }
        a[j] = t;
    }
}
```



Mergesort

```
mergeSort(int left, int right) {
    if (left < right) {
        int mid = (right + left)/2;

        mergeSort(left, mid);
        mergeSort((mid+1), right);

        merge(left, mid, right); } }

merge(int left, int mid, int right) {
    Datensatz tmp[(right-left)+1]; int tmp_i = 0;
    int i = left; int j = mid+1;

    while ((i <= mid) && (j <= right)) {
        if (a[i].key < a[j].key) { tmp[tmp_i] = a[i]; i++;
        } else { tmp[tmp_i] = a[j]; j++; };
        tmp_i++; }

    while (i <= mid) { tmp[tmp_i] = a[i]; tmp_i++; i++; };
    while (j <= right) { tmp[tmp_i] = a[j]; tmp_i++; j++; };

    tmp_i = 0;
    for (i=left; i <= right; i++) {
        a[i] = tmp[tmp_i]; tmp_i++; } }
```

Bottom-Up: in **Schlange** verwalten
Top-Down: in **Stapel** verwalten

Quicksort

```
quickSort(int ilinks, int irechts) {
 if (ilinks < irechts) {
       int i = quickSwap(ilinks,irechts);

       quickSort(ilinks,i-1);
       quickSort(i+1,irechts); }
}

int quickSwap(int ilinks, int irechts) {
   int i = ilinks;
   int j = irechts-1;
   int pivot = a[irechts].key;

   while(i <= j) {
       while((a[i].key ≤ pivot) && (i < irechts)) i++;
       // a[i].key > pivot
       while((ilinks ≤ j) && (a[j].key > pivot)) j--;
       // a[j].key ≤ pivot
       if ( i < j ) swap(i,j); }

   swap(i,irechts); //Pivotelement in die Mitte tauschen
   return i;}
```

Quicksort

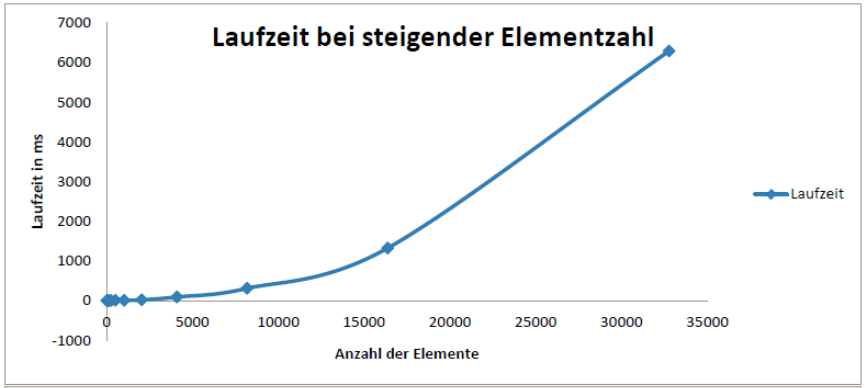
- Die Wahl eines passenden Pivot-Wertes p (Vergleichswerts) für die Partitionierung des Feldes a ist offensichtlich von entscheidender Bedeutung für die Effizienz von Quicksort.
- Optimal wäre ein Element, das a in zwei annähernd gleich große Teile partitioniert: Dies müsste ein, dem Wert nach, mittleres Element von a sein. Um ein solches Element zu bestimmen, wäre jedoch ein Aufwand erforderlich, der die Laufzeitvorteile von Quicksort wieder zunichte machen würde.
- In den bekannten Implementierungen beschränkt man sich daher auf die Inspektion von drei naheliegenden Elementen von a (Median of 3):
  - a[li] das linke Element von a;
  - a[mid] das mittlere Element von a mit mid = (li+re)/2;
  - a[re] das rechte Element von a;

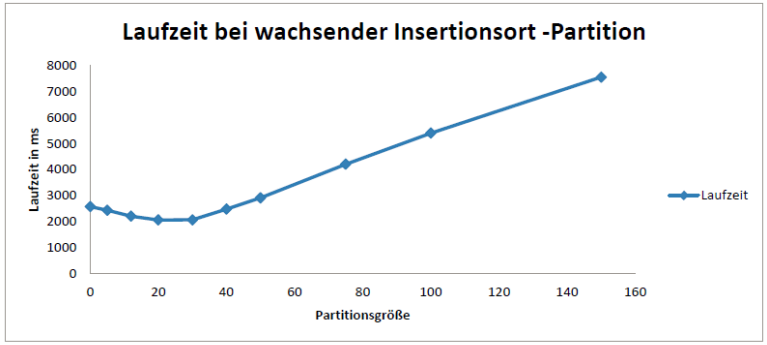
  Man kann entweder eines dieser drei Elemente nehmen und die anderen unberücksichtigt lassen – oder aber unter den drei genannten Werten dasjenige auswählen, welches dem Wert nach in der Mitte liegt.
- Eine weitere Möglichkeit wäre eine randomisierte (zufällige) Wahl des Pivot-Elementes: randomisiertes Quicksort (Paradigma Zufallsstichproben)

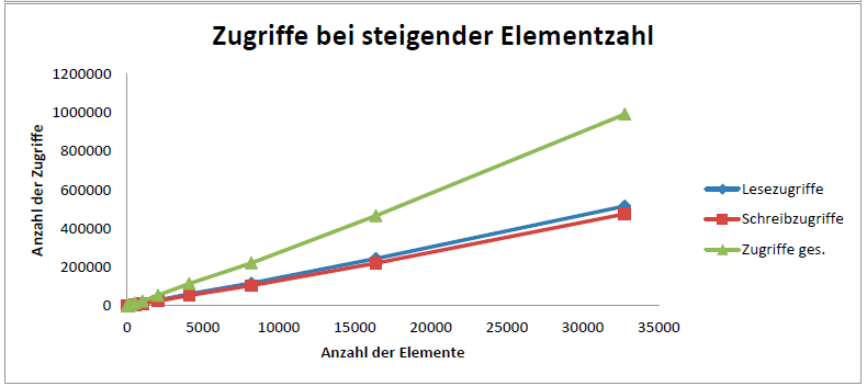
Messergebnisse

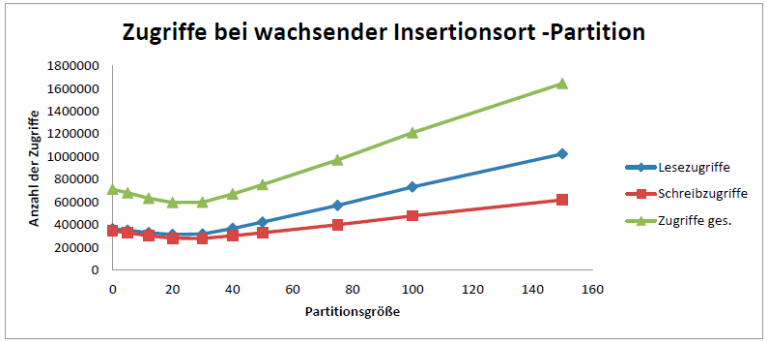
5000	MedianOf3	random	1	30	547	105376	532	16430	103107	107863
5000	Random	random	1	30	511	110252	807	15331	106243	119665
5000	Rechts	random	2	30	351	108091	340	10555	103251	116726
5000	Links	random	2	30	493	110729	501	14819	105986	121424
5000	MedianOf3	random	2	30	339	106158	339	10173	102972	110553
5000	Random	random	2	30	353	111436	372	10590	106783	117969
5000	Rechts	random	3	30	336	108134	344	10082	102754	113425
5000	Links	random	3	30	488	110452	504	14655	106814	115008
5000	MedianOf3	random	3	30	304	105768	302	9137	102921	112643
5000	Random	random	3	30	312	111666	309	9381	107682	117012
5000	Rechts	random	4	30	339	108347	343	10170	103557	116172
5000	Links	random	4	30	487	111052	482	14617	107002	118536

Zeitmessungen	Zeit in ns				
1	16418690	5728651984	101955974	646516	663640442
2	295984	7674994	51982052	295216	252765694
3	129694	5513707	20277634	305553	221819676
4	60133	4334657	14212454	192411	205394459
5	56990	3823215	[illegible]	196183	190127269
6	50844	4144063	[illegible]	185357	314059545
7	57130	4815231	[illegible]	185148	318884774
8	61110	5979334	[illegible]	206309	333303048
9	59853	5059114	[illegible]	12921726	154631128
10	61879	4484955	12243782	151275	131882667
Durchschnitt	1725230,7	577448125	30686217,7	1528569,4	278650870









Messdaten

zahlen.dat mit 20013 Zufallszahlen; Quicksort verzweigt ab 75 Elementen auf den Insertionsort.

Zeitangabe in ms; Gesamtlaufzeit (exklusive Einlesen der Datei); es wurde die eigene ADT verwendet.

Team	Insertionsort	Quicksort (left/right/median3/random)
01:	-	- / - / - / -
??:	-	- / - / - / -