Algorithmen und Datenstrukturen
Klausuraufgaben und ihre Lösung

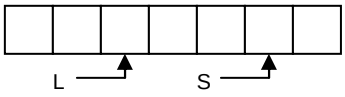
- **Nennen Sie die drei Suchverfahren in linearen, sequentiell gespeicherten Listen und geben Sie deren mittlere Anzahl von notwendigen Vergleichen bei erfolgreicher und nicht erfolgreicher Suche an.**

  1. Sequentielle Suche → keine Voraussetzungen
     $\overline{S}_e = \frac{n+1}{2}$ $\overline{S}_n = n$
  2. Binäre Suche → Voraussetzung: sortierter Datenbestand u. wahlfreier Zugriff
     $\overline{S}_e = O(ld\,n)$ $\overline{S}_n = O(ld\,n)$
  3. m-Wege-Suche
     $\overline{S}_e = \frac{g+1}{2} + \frac{n_g+1}{2}$ $n = g \cdot n_g$ $S_{n\max} = g + n_g$

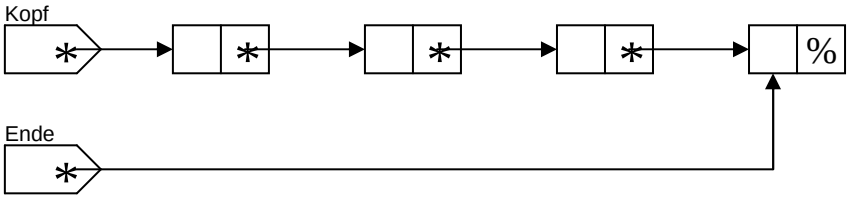
- **Definieren Sie den Begriff „Warteschlange“ (queue). Welche englische Abkürzung gibt es alternativ für diese Datenstruktur? Geben Sie zwei mögliche Implementierungsformen an und skizzieren Sie diese. Geben Sie mögliche Anfangsbedingungen für die leere Warteschlange an!**

  Definition: Eine Warteschlange ist eine lineare Liste, bei der an dem einen Ende eingefügt wird und an dem anderen Ende entnommen wird.
  Alternative Engl. Abkürzung: FIFO (**F**irst**I**n**F**irst**O**ut)

  1. Implementierung: Sequentielle Implementierung:
     (Auch als Ringpuffer bekannt)
     Mögliche Anfangsbedingung: L=S (Lese- und Schreibindex)

  2. Implementierung: Gekettete Implementierung:
     Mögliche Anfangsbedingung: K=E=NULL (K := Kopfanker, E:= Endeanker)

- **Ein *Anker* weist auf einen Stapel. Die Elemente des Stapels bestehen jeweils aus einem Schlüssel *Key* und einen Zeiger *Pnext* auf das nächste Element sind vom Typ *TElem*. Geben Sie in der Programmiersprache C die Anweisungen zum Entfernen eines Elementes vom Stapel an.**

```
typedef struct TElem{
                int key;
                struct TElem *Pnext;
        }TElem;
```

```
TElem *Anker, *PHilf;
if (Anker != NULL)
{
    PHilf = Anker;
    Anker = PHilf -> Pnext;
    free(PHilf);
}
```

- **In einem ursprünglich unsortierten Feld A der Ausdehnung N+1 seien die ersten i-1 Elemente von j=1, 2, …, i-1 bereits aufsteigend sortiert. Geben Sie eine C-Funktion *void Insert (int *A, int N, int i)* an, die das i-te Element des Feldes nach der Methode des direkten Einfügens in das Feld einsortiert. Benutzen Sie dabei das nicht belegte 0-te Element als Marke, um den Algorithmus zeitoptimiert auszuführen.**

```
void Insert (int *A, int N, int i)
{
    int j;
    A[0] = A[i];
    for (j=i-1; A[j] > A[0]; j--)
    {
            A[j+1] = A[j];  // Aufschieben
    }
    A[j+1]=A[0];
}
```

- **Wie lautet die Sortierbedingung für einen Min-Heap? Auf welcher Weise wird ein Min-Heap in ein absteigend sortiertes Feld überführt? Die Funktion zur Korrektur evtl. verletzter Heap-Bedingungen braucht nur genannt zu werden.**

Sortierbedingung:	Der Schlüssel in einem Knoten ist immer kleiner oder gleich dem Schlüssel der beiden Nachfolger!
Umsortierung:	Der Min-Heap kann durch Dekrementieren von N und anschließender Korrektur mit CorrDownHeap(O,A) zu einem absteigend sortierten Feld überführt werden. Die einzelnen Elemente des Heaps werden „getauscht“.

- **Geben Sie die vier grundsätzlichen Traversierungsmöglichkeiten in binären Wurzelbäumen an. Bei welchem Verfahren werden bei einem sortierten Wurzelbaum die Knoten in Sortierfolge aufgesucht?**

Vier grundsätzliche Traversierungsmöglichkeiten:

- PREORDER
- INORDER
- POSTORDER
- LEVELORDER

Davon ist die INORDER-Möglichkeit die einzige Variante, bei der die Knoten in Sortierfolge aufgesucht werden.

- **Warum kann man bei gestreuter Speicherung, offener Adressierung und Anwendung der linearen Sondierung ein Datenobjekt nicht einfach löschen? Was ist stattdessen zu tun?**

  Bei der gestreuten Speicherung kann das zu löschende Objekt nicht einfach gelöscht werden, da sonst bei einer Suche an der entsprechenden Stelle, die Suche abgebrochen würde. Das zu löschende Objekt muss bei der gestreuten Speicherung als gelöscht markiert werden.

- **Beschreiben Sie die wesentlichen Merkmale eines B-Baumes. Warum werden für die Dateiorganistation keine Binärbäume verwendet?**

  Beim B-Baum ist jeder Knoten gleichgroß bzw. jeder Knoten belegt die gleiche Größe an Speicherplatz. Außerdem enthält jeder Knoten mindestens n und maximal 2n Schlüssel.
  Bei der Dateiorganisation werden die B-Bäume nicht verwendet, weil dort nicht die Anzahl der Vergleiche ausschlaggebend ist, sondern die Anzahl der Zugriffe (k-närer Baum).

-