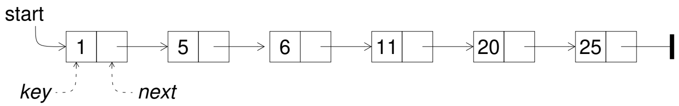
J Datenstrukturen und Algorithmen

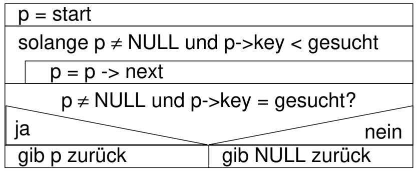
J.1 Sortierte verkette Listen



- **Sortierkriterium:** Für jedes Element gilt: alle Schlüssel der Liste, auf die next zeigt, sind größer oder gleich dem eigenen Schlüssel

- Suchalgorithmus:

suche(start, gesucht)



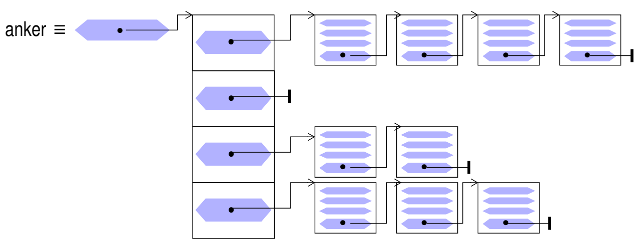

```
struct list {
    int key;
    struct list *next;
};
```

- **Problem:** Verkette Listen erlauben nur die ineffiziente lineare Suche!

J.2 Hashing

- Feld von Ankern für verkettete Listen

- Listenelemente werden abhängig von ihrem Inhalt in eine der Listen eingetragen



- Vorteil gegenüber einer Liste:
  Suchaufwand sinkt auf einen Bruchteil (Anzahl der Anker)

J.2 Hashing (2)

- Index in das Anker-Feld wird aus dem Inhalt des einzutragenden Listenelements ermittelt
  - ◆ Kriterium, nach dem Listenelemente hinterher gesucht werden, nutzen
    - ➤ erster Buchstabe des Nachnamens
    - ➤ Geburtsdatum
    - ➤ Matrikelnummer
    - ➤ Übungsgruppe

- Eintragen von Listenelementen
  - ➤ einfach am Anfang der entsprechenden Liste eintragen

- Suchen von Listenelementen
  - ➤ entsprechende Liste durchlaufen und Elemente einzeln abprüfen

J.2 Hashing (3)

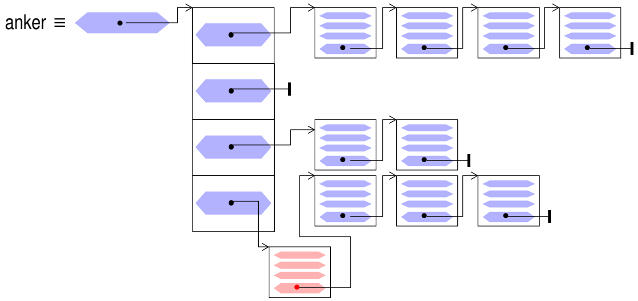
- Beispiel
  - ➤ Feld mit 32 Ankern
  - ➤ Code an Codeskizze aus H.16 angelehnt

```
struct eintrag *anker[32];
struct eintrag *akt_liste;
...
/* Index aus erstem Buchstaben des Nachnamens ermitteln */
index = akt_eintrag->stud.nachname[0] % 32;
akt_liste = anker[index];

/* akt_liste zeigt jetzt auf die Liste,
   in die akt_eintrag eingetragen wird */
akt_eintrag->naechster = aktliste;
aktliste = akt_eintrag;
```

J.2 Hashing (4)

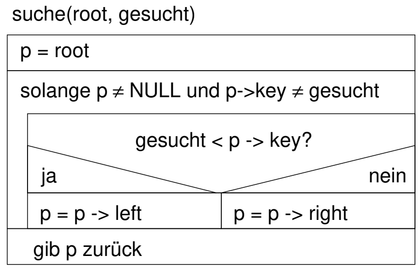
- … Beispiel
  - ➤ Resultat



J.3 Sortierte Binärbäume (2)

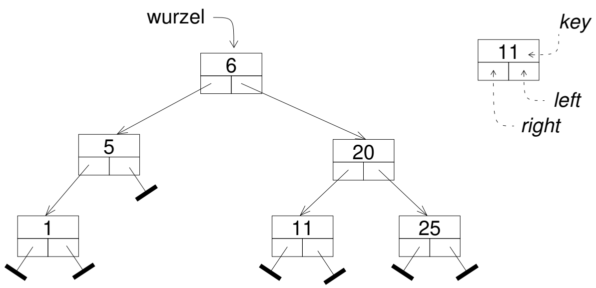
- Suchalgorithmus:

```
struct tree {
   int key;
   struct tree *left;
   struct tree *right;
};
```



- Vorteil gegenüber Listen: sehr effiziente Suche
- Nachteil gegenüber Listen: Einfügen und Löschen von Elementen aufwendiger (Problem: Ausgleich des Baums!)

J.3 Sortierte Binärbäume



- Sortierkriterium: Für jeden Knoten gilt: Alle Schlüssel des linken Teilbaums sind kleiner als der eigene Schlüssel, alle Schlüssel des rechten Teilbaums sind großer oder gleich dem eigenen Schlüssel

J.4 Vergleich von Listen und Bäumen

- Annahme: ausgeglichene Bäume

Anzahl enthaltener Elemente	Liste: mittlere Zahl von Vergleichen	Bäume: max. Zahl von Vergleichen
1	1	2
3	2	4
7	4	6
15	8	8
31	16	10
63	32	12
127	64	14
255	128	16
511	256	18
…	…	…
1048575	524288	40
	$O(n)$	$O(\log n)$