

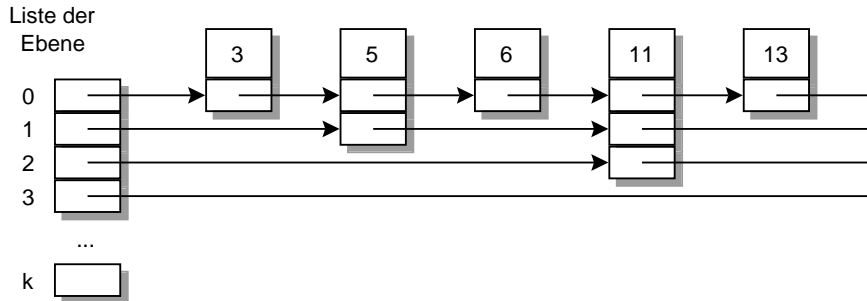
Institut 186 für Computergraphik		Algorithmen und Datenstrukturen 1			VO-Prüfung 18. Juni 1998 Gruppe B	
Kennz.†	Matrikelnr.	Beil.‡	1 (30)	2 (15)	3 (25)	4 (30)
NACHNAME, Vorname					Σ (100)	

† Geben Sie jene Kennzahl an, auf die das Zeugnis ausgestellt werden soll.

‡ Geben Sie an, wieviele Zusatzbl. Sie abgeben (**jedes mit Name & Matr.-Nr. beschriftet!**).

(30) Aufgabe 1 – Skip Lists

Skip Lists sind eine Datenstruktur, die zur schnellen Suche von Datensätzen nach einem Schlüssel dienen. Sie basieren auf sortierten Listen. Die Idee liegt darin, daß jeder Datensatz in einer nach dem Schlüssel sortierten Liste enthalten ist. Dies ist die Liste der Ebene 0. Jeder zweite Datensatz ist in einer weiteren sortierten Liste enthalten, der Liste der Ebene 1. Jeder vierte Datensatz ist in einer weiteren sortierten Liste der Ebene 2 enthalten usw. bis zur Liste der Ebene k , die leer ist. Graphisch kann man sich den Sachverhalt so vorstellen:



a) Entwerfen Sie Datenstrukturen `SkipNode` und `SkipList` für Datensätze und Skiplists von `INTEGER` Zahlen. Gehen Sie davon aus, daß maximal 2^{28} Datensätze in ihren Skiplists gespeichert werden müssen. Sie brauchen dabei nicht darauf zu achten, die Pointer effizient abzuspeichern: sehen Sie in ihren Datenstrukturen genügend Pointer für den schlimmsten Fall vor.

b) Schreiben Sie eine *effiziente* Funktion

```
FUNCTION suche(liste : SkipList; schl : INTEGER) : SkipNodePtr;
```

die nach dem Datensatz mit dem Schlüssel `schl` in der Skiplist `liste` sucht, und einen Pointer auf ihn zurückliefert, wenn er gefunden wurde. Sollte der Datensatz nicht in der Liste sein, so soll `Nil` zurückgegeben werden.

c) Wieviele der Pointer in jedem Datensatz werden durchschnittlich tatsächlich gebraucht? Oder mit anderen Worten: Wenn nun in jedem Datensatz die Pointer der Skipliste *effizient* gespeichert werden, wieviele Pointer sind durchschnittlich pro Datensatz notwendig? (*Geben Sie eine Begründung für Ihr Ergebnis an!*)

(25) **Aufgabe 3 – Zur Fußballweltmeisterschaft**

Ein Programmierer erhält die Aufgabe, Programme zur statistischen Auswertung einzelner Spiele bei der Fußballweltmeisterschaft zu schreiben. Hierzu wurden die Positionen der Spieler in jedem Spiel in regelmäßigen Abständen in Quadrees abgespeichert. Es soll nun herausgefunden werden wieviele Spieler jeder Mannschaft sich zu den aufgezeichneten Zeitpunkten jeweils in den Strafräumen und in den beiden Spielhälften befunden haben. Die Datenstruktur für den Quadtree sieht folgendermaßen aus:

```
TYPE
  TeamTyp = (Eins, Zwei);
  NodePtr = ^Node;
  Node = RECORD
    x, y: REAL;    { Spielerposition }
    team: TeamTyp { welche Mannschaft }
    next: ARRAY[1..4] OF NodePtr;
  END;
```

Jeder Quadtree-Knoten enthält die Information über einen Spieler, sowie 4 Zeiger auf die vier von dieser Spielerposition definierten Quadranten.

Schreiben Sie nun eine *effiziente* Prozedur:

```
FUNCTION countPlayers(spielfeld : NodePtr;
  xmin,xmax,ymin,ymax : REAL
  VAR anzahl eins, anzahlzwei : INTEGER);
```

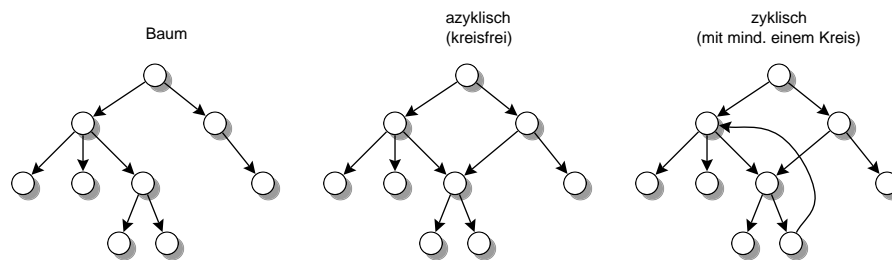
die die Anzahl der Spieler jeder Mannschaft berechnet, die sich im übergebenen rechteckigen Bereich aufhält, und diese in `anzahl eins` bzw. `anzahlzwei` zurückliefert.

(30) Aufgabe 4 – Graphen

Folgende Definition für Adjazenzlisten von gerichteten Graphen sei gegeben:

```
TYPE
  NodePtr = ^Node;
  Node = RECORD
    number : INTEGER;
    next : NodePtr;
  END;
  GraphTyp = ARRAY[1..K] OF NodePtr;
```

Man kann folgende drei Arten von gerichteten Graphen unterscheiden: Bäume, azyklische Graphen und zyklische Graphen. Hier ein Beispiel:



Diese drei Arten von Graphen seien in folgendem Enumerationstyp definiert:

```
GraphArt = (Baum, Azyklisch, Zyklisch);
```

Schreiben Sie nun eine *effiziente, rekursive* Funktion

```
FUNCTION whichGraph(graph : GraphTyp;
  rootNumber : INTEGER) : GraphArt;
```

die, wenn man einen Graphen (**graph**) und einen Wurzelknoten (**rootNumber**) übergibt, berechnet, um welche Art von Graph es sich handelt.

Geben Sie auch alle Hilfsfelder an die Sie benötigen, und mit welchen Werten sie initialisiert werden müssen! *Hinweis:* ein einziges Hilfsfeld vom Typ **BOOLEAN** reicht nicht aus, um alle drei Arten voneinander unterscheiden zu können!