2.1 BÄUME

Die LISTE ist in der Form, wie wir sie bisher betrachtet haben, als Datenstruktur bei der Durchführung mancher Methoden unpraktisch. Ein gutes Beispiel ist das Suchen eines Eintrages. Selbst wenn wir eine sortierte LISTE betrachten wird das Suchen je nach Länge der LISTE zu einem aufwändigen Unterfangen.
Betrachten wir als Beispiel eine LISTE, die die Wörter eines Wörterbuchs wiederspiegelt:

Beim Suchen in dieser LISTE gibt es verschiedene Szenarien:

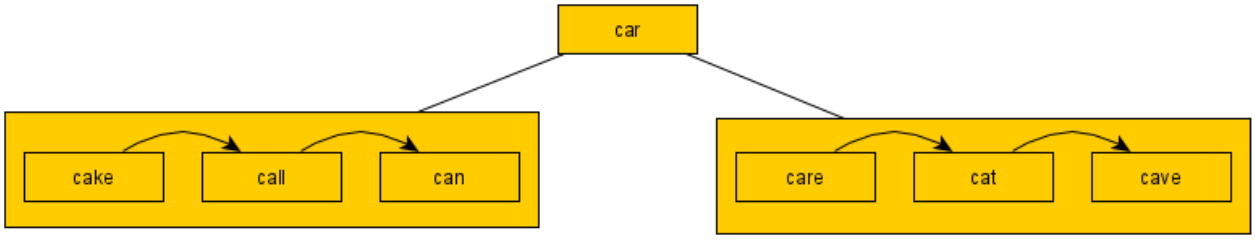
Fall	Vergleiche	Allgemein
worst Case	7	ganze Liste durchsuchen
best Case	1	erstes Element ist gesucht
average Case	4	Erwartungswert

Dabei berechnen wir den Durchschnitt anhand des Erwartungswerts aus der Stochastik. Zur Vereinfachung gehen wir dabei davon aus, dass der Eintrag in der LISTE vorkommen muss und jedes Wort mit gleicher Wahrscheinlichkeit gesucht wird. Jedes Ereignis bekommt somit die Anzahl der Versuche als Zufallsvariablen-Wert und die Wahrscheinlichkeit ist bei allen gleich. Es ergibt sich die Formel

$$E(x)=1\cdot\frac{1}{7}+2\cdot\frac{1}{7}+3\cdot\frac{1}{7}+4\cdot\frac{1}{7}+5\cdot\frac{1}{7}+6\cdot\frac{1}{7}+7\cdot\frac{1}{7}=\frac{28}{7}=4$$

Bei unserer LISTE scheint das jetzt nicht so problematisch zu sein, überlegt man sich jedoch, dass LISTEN dieser Art meist mehrere Millionen Einträge haben, so merkt man schnell, dass besonders ein „worst case“ von mehreren Millionen Vergleichen problematisch werden kann.

Bei unserem Beispiel sehen wir nun, dass durch die Sortierung nach Alphabet (Lexikographische Ordnung) wir den Suchprozess beschleunigen können. Beispielsweise könnte man sich zuerst anschauen, ob das Wort in der ersten Hälfte oder in der zweiten Hälfte vorkommen kann. Alle anderen Elemente muss man dann nicht mehr durchsuchen. Da bedeutet in unserem Beispiel würde man zuerst mit dem Element „car“ vergleichen und danach mit der linken oder rechten Seite der Liste weitermachen. Es ergibt sich folgende Struktur:



Beim Suchen in dieser Struktur gibt es nun folgende Szenarien:

Fall	Vergleiche	Allgemein
worst Case	4	„car“ und dann 3 Elemente
best Case	1	„car“ ist gesucht
average Case	3	Erwartungswert

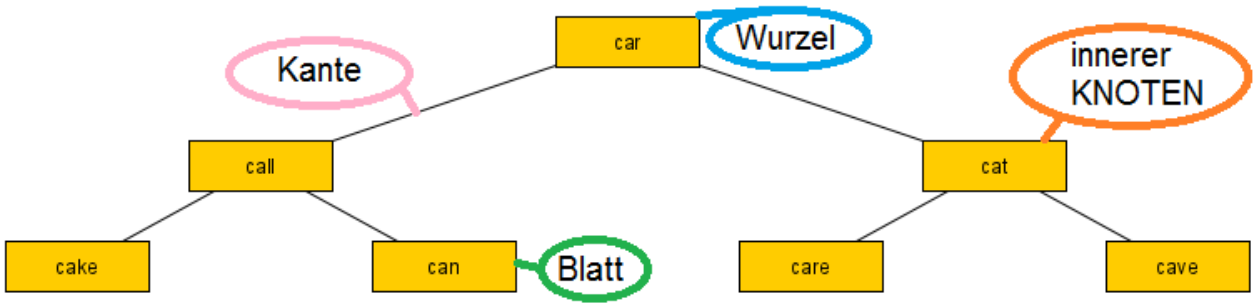
Für den Erwartungswert gehen wir wieder davon aus, dass alle Wörter gleich wahrscheinlich sind. Damit ergeben sich, folgende Wahrscheinlichkeiten:

X (Vergleiche)	1	2	3	4
P(X)	$\frac{1}{7}$ (car)	$\frac{2}{7}$ (cake oder care)	$\frac{2}{7}$ (call oder cat)	$\frac{2}{7}$ (can oder cave)

$$E(X)=1\cdot\frac{1}{7}+2\cdot\frac{2}{7}+3\cdot\frac{2}{7}+4\cdot\frac{2}{7}=\frac{1}{7}\cdot(1+4+6+8)=\frac{19}{7}\approx 2{,}7\approx 3$$

Wir sehen insbesondere, dass der „worst case“ sich stark verbessert hat. Übertragen wir das Prinzip auf eine Liste mit 1 000 001 Einträgen wäre der „worst case“ nicht mehr 1 000 001 wie bei der LISTE, sondern 500 000. Wir halbieren also den Aufwand.

Nun stellt sich quasi rekursiv die Frage, warum wir nicht auch beim Suchen in den neu entstandenen Teilbäumen links und rechts das gleiche Prinzip anwenden. Wir suchen links und rechts wieder die Mittleren Elemente und bauen neue Bäume auf. Das führt uns zu folgender Struktur:



Man nennt diese voll ausgebildete Struktur einen BAUM. Die Untersten „Endelemente“ wie z.B. „cake“ und „can“ nennt man Blätter des BAUMes, innere Elemente die inneren Knoten und den obersten KNOTEN nennt man Wurzel. Die „Ebene“ in der sich ein KNOTEN befindet nennt man auch die Tiefe des KNOTENs, die Wurzel hat dabei Tiefe 0.
Es ergibt sich für die Szenarien folgende Berechnung:

Fall	Vergleiche	Allgemein
worst Case	3	„car“ und dann 3 Elemente
best Case	1	„car“ ist gesucht
average Case	3	Erwartungswert

Für den Erwartungswert gehen wir wieder davon aus, dass alle Wörter gleich wahrscheinlich sind. Damit ergeben sich, folgende Wahrscheinlichkeiten:

X (Vergleiche)	1	2	3
P(X)	$\frac{1}{7}$ (car)	$\frac{2}{7}$ (call oder cat)	$\frac{4}{7}$ (cake, can, care oder cave)

$$E(X)=1\cdot\frac{1}{7}+2\cdot\frac{2}{7}+3\cdot\frac{4}{7}=\frac{1}{7}\cdot(1+4+12)=\frac{17}{7}\approx 2{,}4\approx 3$$

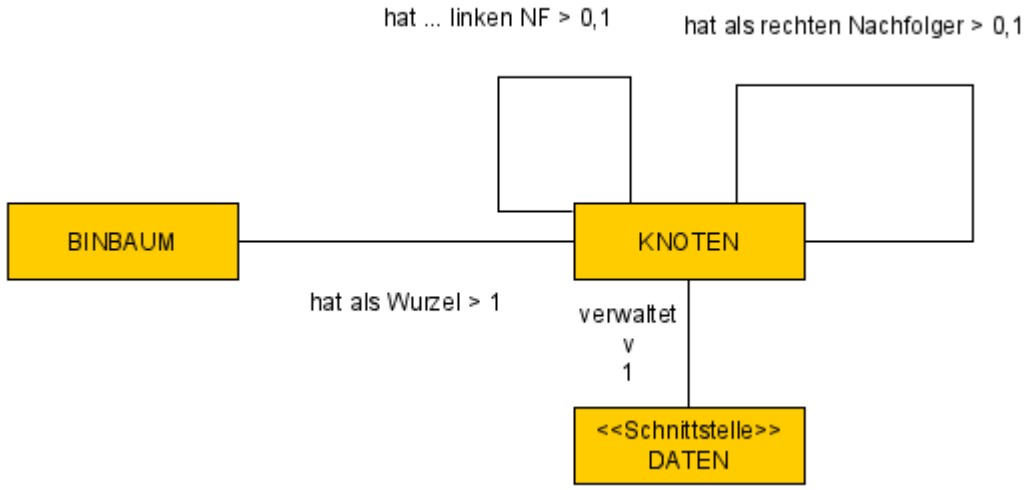
Sehr viel hat sich hier nicht mehr verbessert, jedoch ist der „worst case“ nochmal besser geworden. Betrachtet man unser Extrembeispiel, dann stellt man fest, dass bei 1 000 001 Einträgen der „worst case“ auf $\log_2(1000001)\approx 19{,}9\approx 20$ sinkt, was eine unglaubliche Verbesserung im Vergleich zu den vorherigen Iterationen ist. (Die Berechnung des Erwartungswertes darf gerne zur Übung für Mathematik selber durchgeführt werden. Das Ergebnis ist $E(X)=\frac{1}{1000001}\cdot(\sum_{n=1}^{18} n\cdot 2^n+475714\cdot 19)=\frac{17951464}{1000001}\approx 17{,}95\approx 18$)

Wir sehen daraus, dass wir den Suchaufwand stark verringern können.

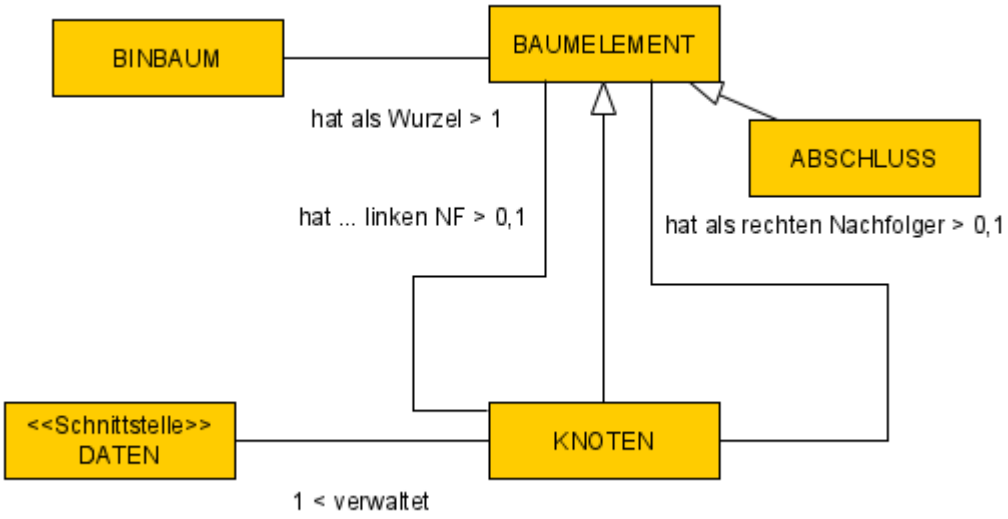
Umsetzung des BAUM

Überlegen wir nun, welche Klassen nötig sind um den BAUM umzusetzen.
Jedes Element außer der Wurzel hat einen Vorgänger und sonst jeder Knoten Nachfolger. In unserem Fall haben wir genau zwei Nachfolger weshalb man von einem sortierten Binärbaum spricht. (Ein allgemeiner BAUM könnte von einem inneren KNOTEN oder der Wurzel zu beliebig vielen Nachfolgern führen. Diese Art von Bäumen soll uns aber hier nicht interessieren.)
Die letzte Ebene an KNOTEN, die Daten enthalten, muss nach unten entweder leere Nachfolger erhalten, oder im Sinne des Kompositum einen ABSCHLUSS erhalten.

Aus diesen Überlegungen ergibt sich folgendes Klassendiagramm:



Möchte man die Umsetzung nicht mit leeren Referenzen machen, verwendet man das Kompositum, um als linke und rechte Nachfolger aller Blätter die Klasse ABSCHLUSS zu verwenden. Das Klassendiagramm ist leicht komplizierter:



Aufgaben:

1) Kopiere die Vorlage aus dem Klassenordner (Swing_BinBaum_01).

2) Überlege anhand des Klassendiagramms, welche Attribute und Methoden die Klassen BINBAUM, BAUMELEMENT, KNOTEN, ABSCHLUSS und DATEN haben müssten. Denke dabei besonders daran, …

- … welche Attribute/Methoden zur Herstellung der Struktur nötig sind.
- … welche Methoden und Attribute zur Darstellung und für die Daten nötig sind.

(Hinweis: Für die Darstellung eines KNOTEN wird die Klasse KNOTENSYMBOL zur Verfügung gestellt. Um einen BAUM komplett anzuzeigen existiert die Klasse BAUMSYMBOL. Dabei ist darauf zu achten, dass die Methoden, die in BAUM- und KNOTENSYMBOL verwendet werden in den eigenen Klasse gleiche Bezeicher haben.)

3) Erstelle die Klassen des Klassendiagramms mit den Methoden und Attributen, die du dir überlegt hast. Lass dabei Methodenrümpfe zuerst leer.
Zur Vereinfachung machen wir als Beispiel ein Wörterbuch. Das bedeutet, dass statt einer Schnittstelle DATEN in den Daten einfach nur einen String abspeichern kannst, der zurückgegeben werden kann.
(Hinweis: Die Klassen KNOTENSYMBOL, BAUMSYMBOL und TESTBAUM sind dafür da, um einen Testbaum zu erstellen und diesen anzuzeigen siehe 2)

4) Erstelle mithilfe der Klasse TESTBAUM einen eigenen BAUM und teste deine Struktur.

5) Überlege, wie die Klasse BAUMSYMBOL vorgeht, um alle KNOTEN darzustellen. Kannst du erklären, wie sie von einem KNOTEN zum nächsten springt? Könnte man die Darstellung auch anders aufbauen?