Inf ProgTech Übung: Trie

Klaus Kusche

Die Aufgabe ist dieselbe wie im Wortindex-Beispiel der Übung 4 und 5,
aber die Wörter sollen diesmal in einem *Trie* gespeichert werden.

Du kannst wahlweise deine eigene vorige Lösung oder meine Musterlösung von Übung 5
als Ausgangsbasis nehmen.

Folgende Dinge wirst du ändern müssen, sonst solltest du möglichst nichts modifizieren:

- Wir geben unseren Trie-Knoten der Einfachheit halber *256 Söhne*,
  einen für jeden möglichen **`unsigned char`**-Wert, d.h. ASCII-Wert.

  Speicher-effizienter wären Knoten mit 63 Söhnen (für jedes zulässige Zeichen)
  mit einem globalen 256-elementigen Hilfs-Array, das die ASCII-Codes
  der tatsächlich zulässigen 63 Zeichen auf einen Index 0-62 abbildet
  (und die der ungültigen Zeichen auf -1), aber diese Mühe sparen wir uns
  (mein Programm nimmt dann eben für einen Index über alles Header-Files
  im System 600 MB in Anspruch...).

  Achtung: Da **`char`** normalerweise *signed* und daher als Index nicht verwendbar ist
  (Umlaute haben ev. *negative* Werte!), muss ein Zeichen vor Verwendung als Index
  auf **`unsigned char`** *gecastet* werden! (ich habe mir dafür ein Makro geschrieben,
  das mir zu einem Trie-Knoten und einem Zeichen diesen Cast macht
  und den entsprechenden Sohn-Pointer des Knotens als Ergebnis liefert).
- Das Wort selbst speichern wir *nicht* in den Trie-Knoten, es lässt sich auf dem Weg
  von der Wurzel zum betreffenden Knoten zeichenweise rekonstruieren.
- Weiters sparen wir uns die Mühe,
  die Nutzdaten eines Wortes in eine eigene Struktur auszulagern
  und von den Knoten im Trie darauf zu verweisen.

  Wir haben ja nur *Anfang und Ende der Positionsliste* als Nutzdaten,
  und diese bringen wir direkt im Trie-Knoten unter.

  Ist ein Trie-Knoten nur Zwischenknoten für längere Wörter,
  aber entspricht selbst keinem gespeicherten Wort,
  sind Anfang und Ende der Positionsliste **`NULL`**.

  `new_word` hat diesmal *keine* Parameter,
  es initialisiert Head und Tail der Positionsliste sowie alle Söhne auf **`NULL`**.

  Demgemäß muss **`add_pos`** nun auch eine Position
  an eine noch *leere Positionsliste* anhängen können!
- Der *oberste* Knoten des Trie, der dem *leeren Wort* "" entspricht,
  wird schon *bei der Initialisierung fix* angelegt.
- Dafür ist **`save_word`** ganz einfach:
  Man geht das Wort *Zeichen für Zeichen* durch
  und steigt pro Zeichen eine Ebene im Trie hinunter:
  Das aktuelle Zeichen legt fest, welcher Sohn gewählt wird.

Ist der entsprechende Sohn **NULL**,
wird ein *neuer* Trie-Konten erzeugt und an dieser Stelle eingehängt.

Ist man am *Ende des Wortes* angelangt, wird an den Knoten,
den man dadurch erreicht hat, eine *Position angehängt*, egal,
ob das ein neuer Knoten, ein bestehender Zwischenknoten ohne Positionen,
oder ein Knoten mit Positionen ist.

- Auch beim Suchen in **`process_word`** wandert man (iterativ, nicht rekursiv)
Zeichen für Zeichen im Trie nach unten.
Existiert der dem Zeichen entsprechende Sohn nicht, gibt es das Wort nicht.

  Auch wenn man am Ende des Wortes den Zielknoten erreicht hat,
aber dessen Positionsliste *leer* ist, existiert das Wort nicht,
der Knoten ist in diesem Fall nur ein Zwischenknoten.

Als *Zusatzaufgabe* kannst du wieder *Statistiken* wie beim Baum
(Gesamtzahl der Wörter, Wörter pro Tiefe, …) ausgeben
und vor allem eine Funktion schreiben, die den gesamten Trie einmal *rekursiv durchläuft*
und *eine sortierte Liste aller Wörter* und ihrer Positionen ausgibt.