Henning Thielemann home.pages.de/~lemming

Vortrag im Seminar "Textalgorithmen und Datenkompression"

Wörterbuchbasierte Packalgorithmen

Motivation

Bisher:
- Huffman, Voraussetzung: manche Zeichen kommen deutlich häufiger vor als andere
- RunLength, Voraussetzung: gleiche Zeichen folgen oft aufeinander

Aber:
- auf abstrakten Daten wie Texten, Programmen, weniger Bilder und Sounds, ist Übersetzen von Wörtern in Token viel besser

Nachteil bei statistischen Methoden:
- Packen und Entpacken sehr ähnlich, daher fast gleich langsam

Wortorientierte Verfahren:
- Entpackroutine kann um einiges schneller sein, da kein Wörterbuch durchsucht werden muß, und über lange Strecken einfach nur Zeichen kopiert werden
- statistische Methoden lassen sich immer noch mit Wörterbuchmethoden kombinieren

Generelle Probleme

- je präziser Analyse, desto langsamer Packer, umso mehr Platz verbraucht die Beschreibung der Struktur
- im Extremfall nimmt der Programmcode des Packers gerade den Speicherplatz ein, den der Packer durch komprimieren der Daten spart
- allgemein gute Packer gibt es nicht
- zu jedem Packer gibt es sehr viele Datensätze die durch Packen länger werden

Im folgenden:
- Texte mit vielen Wiederholungen betrachtet
- ein Packer heiße gut, wenn er solche Texte gut packt

Betrachte im folgenden einzelne Aspekte der Wörterbuch−basierten Kompression und stelle Lösungsideen vor. Viele Ideen lassen sich mit anderen kombinieren.

Literalgröße
- hängt von zu packenden Daten ab
- Byte: ASCII−Texte, 8−Bit−Samples, 256−Farben−Bilder, Programmcode von 8−Bit−Rechnern
- Word: UniCode, 16−Bit−Samples, Programmcode von 16−Bit−Rechnern etc.

Wörterbücher

vorgefertigtes Wörterbuch
- wird mit Worten der Sprache des Textes gefüllt (z.B. deutsch oder englisch)
- jedes bekannte Wort im Text wird durch seine Position im Wörterbuch codiert (Token)
- Probleme
  Aktualität
  allgemeine Verwendbarkeit
  lange Tokennummern
  englische Sprache ohne Beugungen bereits über 500000 Wörter
  Tokenlänge wäre bereits 19 Bit, also etwa 2½ Zeichen

dynamisches Wörterbuch
- werden für jede Datei neu aufgebaut
- Computer hat oft ungewöhnliche Assoziationen wie
  "Drinking contest ... king console" wo "king con" übereinstimmt
- im folgenden ist n Suchpuffer−Länge, m Vorschaupuffer−Länge
- n im Bereich Kilobytes, m einzelne Bytes
- Beispiel: <The_sad_cat_sat_on_t><he_mat.>

Realisierung dynamischer Wörterbücher

Binärbaum (LZ77)
- Suche und Einfügen in O(m*log n)
- eingefügt werden Wörter der Länge m des Vorschaupuffers
- Wort braucht nicht selbst eingefügt zu werden, lediglich Zeiger in den Text
- damit ist es auch möglich, zyklische Worte zu erkennen und in völlig natürlicher Weise in den Algorithmus einzubinden
- Beispiel: C−Kommentar /**********/

- RunLength−Encoding (RLE) ist hier vollständig enthalten
- in Wörterbuch befinden sich im Grundalgorithmus immer n Wörter
- Beispiel: (die Teilzeichenketten sind bereits alphabetisch sortiert)

```
The_sad_<cat_sat_on_the_m><at.    >
                at_on_t
            at_sat_
           cat_sat
                        e_mat.
                       he_mat.
                          mat.
                    n_the_m
                   on_the_
               sat_on_
                      the_mat
                 t_on_th
             t_sat_o
                         _mat.
                  _on_the
              _sat_on
                     _the_ma
```

- gepackter Text, Bausteinformate (Flag = Original, Länge, Originaltext), (Flag = Kopie, Adresse, Länge)

(Original,	11,	The_sad_cat)	
(Kopie,	8,	3	[_sa])
(Kopie,	4,	2	[t_])
(Original,	4,	on_t)	
(Kopie,	19,	3	[he_])
(Original,	1,	m)	
(Kopie,	11,	2	[at])
(Original,	1,	.)	

- Wörter werden auch wieder entfernt
  1. sobald Wörterbuch voll
  2. das zuerst aufgenommene Wort wird zuerst wieder entfernt
  3. Benutzungszähler für alle Wörter anhand dessen man über die Löschung des nächsten Wortes entscheiden kann (Prioritätenwarteschlange) geht nur wenn Wörterbuch Kopien enthält (was Erkennung von Zyklen zuwiderläuft)
  optional: die Benutzungshäufigkeit könnte Prioritäten in Binärbaum vorgeben −> Treaps
- Vorteil Lokalität des Packers
  entspricht eher der Struktur von realen Dokumenten
  in bestimmten Abschnitten häufig gebrauchte Wörter kommen in anderen vielleicht gar nicht vor
  Beispiel: gemischtsprachige Texte (Bedienungsanleitung)
  macht die Suche für den Packer einfacher = schneller
  erlaubt kürzere Adressierungen der Worte

<u>Hashing</u>

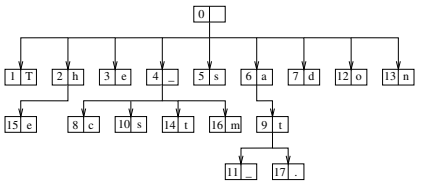
- Hashwerte aus vorangehenden oder nachfolgenden Symbolen als Adressen im Wörterbuch
- Suche in O(m)
- ein paar Symbole des Vorschaupuffers werden gehasht
- man kann auch die jüngsten Symbole des Suchpuffers nehmen −> Vorhersage−Hash
  Vorteil: Worte mit gleichem Anfang verdrängen sich nicht gegenseitig
- mit Hashwert springt man ins Wörterbuch
- Wörterbucheinträge zeigen wiederum auf Stellen im Suchpuffer
- es wird immer der jüngste Hash−Eintrag gefunden (klar, da neue alte einfach überschreiben)
- dadurch findet man zwar nie einen besonders guten Eintrag, aber man findet bedeutend schneller überhaupt was
- Beispiel: MC68000 pro byte: 13 Befehle Kompression, 4 Befehle Dekompression
- Beispiel: The_sad_<cat_sat_on_the_m><at. >
  Adressen: aa, ab, ac, ad, ..., at, ..., __
  direktes Hash: Zelle "at" enthält "at_on_t" statt "at_sat_"
  Vorhersagehash: Zelle "_s" = "at_on_t", Zelle "_c" = "at_sat_"
  Vorteil: keine Überschneidung
  Nachteil: Beide Einträge werden nicht gefunden
  Lösung: bei genügend langen Worten würde hinterer Teil eines Wortes im Wörterbuch gefunden werden können
  Beispiel: Vorschaupuffer "Computer", Wörterbuch "sputen", "putzen", "Computer"
- bessere Suchergebnisse:
  jedem Hasheintrag mehrere Zeiger spendieren (Feld)
  man sucht dann innerhalb dieses Feldes den Zeiger auf die längste Übereinstimmung

Trie (LZ78)

- eine Art Suffixbaum
- Suche in O(m), Einfügen in O(1) pro Wort
- kommt bei folgendem Packverfahren zum Einsatz:
  initialisiere Wörterbuch mit einem leeren String
  finde in Wörterbuch Eintrag mit übereinstimmendem Präfix maximaler Länge
  schreibe die Nummer des Eintrages in Ausgabe gefolgt von dem folgenden Letter des Vorschaupuffers
  füge neues Wort ein, daß aus dem Wörterbuchwort + folgendem Letter gebildet wird
- Beispiel: (obere Zeile jeweils Ausgabe des komprimierten Textes, untere Zeilen Wörterbucheinträge)
  The_sad_cat_sat_on_the_mat.

```
(0,T) (0,h) (0,e) (0,_) (0,s) (0,a) (0,d) (4,c) (6,t)
 1 T   2 h   3 e   4 _   5 s   6 a   7 d   8 _c  9 at
(4,s) (9,_)  (0,o) (0,n) (4,t) (2,e) (4,m) (9,.)
10 _s 11 at_ 12 o  13 n  14 _t 15 he 16 _m 17 at.
```

- Aufbau der benutzten Datenstruktur
  jeder Pfad von der Wurzel nach unten stellt ein Wort dar
  jeder Knoten repräsentiert Wort, das sich ergibt, wenn man die Lettern von der Wurzel bis zu diesem Knoten liest
  jeder Knoten bekommt eine Nummer, die als Index des assoziierten Wortes im Wörterbuch dient
- Beispiel:



- in jedem Schritt kommt nur ein Letter zu einem Wörterbuch−Wort
- Trennungen können unglücklich sein, wodurch gegenüber Binärbaum−Variante lange Ersetzungen verlorengehen
- Beispiel: steht im Wörterbuch "Prozessor" kann Wort "Parallelprozessor" davon nicht profitieren, denn beide Worte stimmen erst mit "rozessor" überein
  Intellenztest: Wie lang muß ein Eingabetext mindestens sein, damit das Wort "Prozessor" im Wörterbuch vorkommen kann?

- Wörterbuch braucht sehr lange zum Aufbau und wird dazu noch sehr groß
- Anzahl der Wörter in Wörterbuch wächst mindestens proportional zu Sqrt(n) (Beispiel: Text besteht nur aus n−mal dem gleichen Zeichen) maximal aber linear
- Speicherbedarf des Wörterbuches (Anzahl aller Zeichen) wächst linear mit der Anzahl der Zeichen im Text
- Lösungen:
  1. Wörterbuch einfrieren, sobald aufgefüllt (damit ist Lokalität nicht mehr gewährleistet)
  2. Wörterbuch löschen sobald voll und mit neuem anfangen
  bessere Lokalität
  dadurch wird Text im Prinzip in unabhängige Blöcke zerlegt
  3. die ältetesten Worte herausnehmen
  aus Baum können nur Blätter, also nur lange Worte, entfernt werden
- der Entpacker muß auch wieder Wörterbuch aufbauen, daher ähnlich komplex wie Packer
- bevorzugt in Verbindung mit integrierten Literal+Kopie−Bausteinen

Bausteine

getrennte Original− und Kopie−Bausteine

- auf Original−Baustein kann immer nur Kopie−Baustein folgen, es sei denn, die Längencodierung für einen Original−Baustein reicht nicht aus
- kein Problem bei längenvariablen Zahlencodierung
- Struktur immer 1×Original, n×Kopie
- es reicht Struktur <Original|n|Kopien>
- andererseits folgen oft nicht so viele Kopie−Bausteine aufeinander, so daß man auch jeden Baustein mit Bit einleiten kann, das Bausteintyp angibt
- Plazierung dieser Bits
  1. vor jedem Baustein
  2. Bits gesammelt in Blöcken (Steuerblöcke zu 8 oder 16 Bit) wegen der ausgerichteten Speicherzugriffe

Kopie und Original in einem Baustein

- Beispiel: Paar von Wörterbucheintrag und folgendem Original

Zahlencodes

· benötigt für Adressen und Längenangaben

binär

· feste Bitbreite

unär

· variable Bitbreite aber sehr speicherintensiv

unär/binär

· unäre Codierung der Ziffernzahl + binäre Zahl

eine Unär–Eins bedeutet eine Binärstelle

· führende Eins zur Unär–Terminierung mitverwendet
  unterstützt Zahlen in den Formaten *, **, ***, ****, ***** etc.
  die ersten Zahlen:
  1, 01, 0010, 0011, 000100, 000101, 000110, 000111
· Beispiel: 0000010101 = 21
  5 Nullen als Längenangabe (Länge 5),
  Eins ist Ende der Unärzahl und Beginn der Binärzahl
· Zahl wird im Vergleich zur Binärzahl doppelt so lang

eine Unär–Eins zählt m Binärstellen

· Beispiel m=3
  unterstützt Zahlenformate ***, ******, ********* etc.
  die ersten Zahlen:
  000, 001, 010, 011, 1000100, 1000101, 1000110, 1000111
· Beispiel: 110010101 = 21
  2 Einsen als Längenangabe (Länge 6),
  Null ist Ende der Unärzahl
  Binärzahl besteht aus 6 Ziffern 010101

eine Unär–Eins zählt m Binärstellen, es müssen mindestens k Stellen sein

· Beispiel m=3, k=4
  unterstützt Zahlenformate ****, *******, ********** etc.
  die ersten Zahlen sind
  0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 10001000
· alle drei sind nach oben offene Codes und von daher sogar noch flexibler als reiner Binärcode

eine Unär–Eins zählt m Binärstellen, es müssen mindestens k Stellen sein, aber maximal n

· Beispiel m=4, k=8, n=32
  unterstützt Zahlenformate ********, **********, ..., *^32 etc.
  erlaubt Speicherung von 32 Bit–Zahlen
  weniger als 6 Einsen müssen mit einer Null terminiert werden
  kommen 6 Einsen hintereinander, beginnt Binärzahl sofort im Anschluß, ohne terminierende Null
· die letzten drei Codes sind im Gegensatz zum ersten nicht mehr eineindeutig –> Redundanz

kaskadiert binär (Levenstein–Code)

· Idee: Länge der Binärzahl wird binär codiert, diese Längeangabe wiederum usw.
· Basisalgorithmus
  LevensteinCode von n in Binärdarstellung
  Wenn n=0 gib e zurück
  Andernfalls
  Zähle die Binärstellen ohne die führende Eins
  Entferne führende Eins und ersetze sie durch den Levensteincode der Länge
  Vor diese Zahl setze noch eine Null (damit die Anzahl der Dekodierebenen klar ist) und gib das Ergebnis zurück
· Null wird als einzelne Eins kodiert
· die ersten Zahlen sind
  1, 01, 0010, 0011, 0001000, 0001001, 0001010, 0001011, 00011000, 00011001, 00011010, 00011011
· Beispiel: $123456789_{dec} = 75BCD15_{hex} = 111\ 0101\ 1011\ 1100\ 1101\ 0001\ 0101_{bin}$
  Levensteincodierung (angedeutet durch die eckigen Klammern):

```
[123456789]  = 0[26]11010110111100110100010101
             = 00[4]10101101011011110011010010101
             = 000[2]00101011010110111100110100010101
             = 0000[1]000101011010110111100110100010101
             = 00000100010101101011011110011010010101
```

  da $26_{dec} = 1A_{hex} = 11010_{bin}$, $4_{dec} = 100_{bin}$, $2_{dec} = 10_{bin}$

· Zahlensysteme mit variabler Stellenanzahl kaum geeignet für Kompressionsformate, die spezielle Speicheradressen–Ausrichtung ausnutzen sollen

Vorverarbeitung

· Differenzieren sinnvoll bei Bildern und digitalisierten Klängen, wo ein Wert nur wenig vom vorangehenden abweicht

Verlustbehaftete Packer

· keine exakten Übereinstimmungen bei Suche in Wörterbuch
· Vorsicht: Packraten bei verlustbehafteten auch Packern manchmal Augenwischerei, da Rate auf Ausgangsbild bezogen wird – wenn Ausgangsbild nur 256 statt 16 Mio. Farben hat, ist auch Packrate nur ein Drittel
· Problem: unscharfe Suche im Wörterbuch eher langsamer als exakte Suche, günstige Datenstruktur?