横浜国立大学 大学院 電子情報工学専攻 専門科目

# ディジタルドキュメント論(7)

Digital Document Processing

森 辰則

mori@forest.dnj.ynu.ac.jp

## 文書内の文字列の検索手法

二つの観点

- □全文検索か決められた単位での検索か
- □索引を使うか使わないか

組み合わせると

- □全文+索引を使わない → BM法
- □全文+索引を使う → サフィックスアレイ(suffix array)法
- □決められた単位+索引を使う → 今回の主題

## 索引を使う検索

索引をどのように構成するか?

- □転置ファイル(Inverted file)方式
  - ○語(固定長文字列)→文書出現位置
- □署名ファイル(Signature file)方式
  - ○語(固定長文字列)→文書

## 転置ファイル(Inverted file)

「検索される可能性のある」文字列と，その文字列が出現している文書の組を管理するファイル

- □各文字列について，その文字列が出現している文書(その文書の中のその文字列の出現位置)の情報をまとめて管理する．
- □索引ファイル(Index file) と 配置ファイル(Posting file)の組で管理されることが多い．
  - ○索引ファイル: 文字列を「ポインタ」に対応づける．ポインタは配置ファイル内の特定箇所を指し示す(先頭から数えた位置など)．
  - ○配置ファイル: 文書番号(あるいは文書内の位置情報)の並び．この情報は文書への「リンク」情報と呼ばれる．
    - ▸その文字列に関する統計情報(出現頻度など)も同時に保存しておくこともある．

## 転置ファイル(Inverted file) (cont.)

Index file

| 語 | 文書数 | ポインタ |
|---|---|---|
| | | |
| 宇宙 | 4 | |
| | | |
| ロケット | 2 | |
| | | |
| SF | 3 | |
| | | |

Posting file

| 文書番号 | 語の重み |
|---|---|
| | |
| 3 | 0.5 |
| 5 | 0.2 |
| 20 | 0.6 |
| 53 | 0.1 |
| | |
| 10 | 0.4 |
| 37 | 0.3 |
| | |
| 1 | 0.9 |
| 16 | 0.5 |
| 18 | 0.2 |

## 転置ファイルにおける単語の検索手法(1)

要件

- □新しく与えられた単語が転置ファイルのなかに入っているかどうかを効率良く検索したい．
- □検索に適した索引構造を作成し，その上で探索する．
  - ○検索「文字列」→数字(配置ファイルへのポインタ)

## 転置ファイルにおける単語の検索手法(2)

手法

- □ソート(整列)されたリストを用いる
  - ○索引: ソートされた索引語のリスト，探索: 二分探索
- □ハッシュ(hash)を用いる
  - ○索引: 索引語から作られたハッシュ構造，探索: ハッシュ関数
- □B木(B tree)を用いる
  - ○索引: 索引語から作られたB木，探索: 木の探索
- □トライ(Trie)を用いる
  - ○索引: 索引語から作られたトライ，探索: 木の探索
- □パトリシア木(Patricia tree)を用いる
  - ○索引: 索引語から作られたパトリシア木，探索: 木の探索

## 二分探索

- □索引語を文字列の配列に保存し，二分探索する

配列の添字 索引語の配列 ポインタの配列

1 abcd 10
2 acdef 3
:
n xyz 16

単純であるが以下の難点がある

- □単語中の文字をそのまま辞書に保存するため辞書が大きくなる．
- □探索に際して，文字の比較回数が多い．
  - ○探索文字列の長さをn, 辞書中の文字列の数をm とすると o(n*log(m)) のオーダー

## ハッシュ(Hashing)

検索文字列から「直接」(探索無しに)索引中のその文字列の項目に到達する方法は?

ハッシュ(Hashing)とは

- □検索対象(今の場合は文字列)をそれから一意に決まるある範囲の整数に対応させ，そこに関連するデータを保存するデータ構造
- □対応させる関数Hはハッシュ関数と呼ばれる．
  - ○H: 文字列 → 自然数

## ハッシュ(2)

- □索引の作成
  - ○「文字列」と「ポインタ」の組を必要分だけ用意
  - ○ポインタ情報を格納するための配列 HA[] を用意．
  - ○H('文字列') を計算し，その値で指し示される配列の要素に対応するポインタを保存する．
- □索引の検索
  - ○「文字列」が与えられた時，H('文字列')を計算し，対応するポインタ情報を取り出す．その情報を基に配置ファイル内を調べて必要な情報を取り出す．

## ハッシュ(3)

- □問題は
  - ○関数Hの作り方
- □性能は
  - ○衝突: 違う文字列が同じ添字に割り当てられた
  - ○衝突の頻度(の少なさ)
  - ○未使用記憶領域(の少なさ)
  - ○衝突がなければ，探索のための時間計算量は o(1)

## 手法1: チェイン・ハッシュ(Chained Hashing)

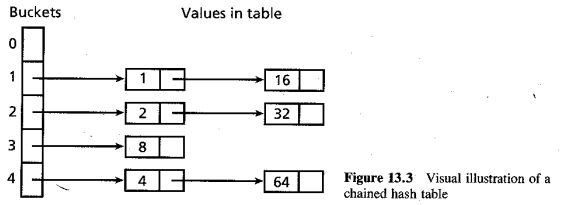
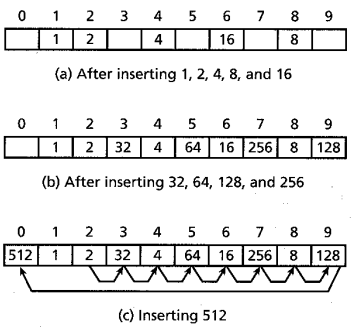



**Figure 13.3** Visual illustration of a chained hash table

## 手法2: オープンアドレス(Open Addressing)




## 木構造型の索引

- □B木，トライ，パトリシア木
- □検索文字列に現れる文字を左から順に見ながら，木の根(root)から葉に向かって下っていく．最終的にたどり着いた葉にポインタがある．

## トライ(1)

- □de la Briandais(1959) 考案
- □Information Retrieval: 命名 Fredkin(1960)
- □多分木構造の一種

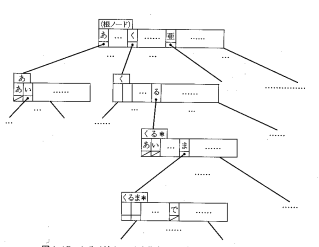


図1.17 トライ法(＊は文字集合として与えられたものを示す)

## トライ(2)

- □各ノードには一文字が割り当てられており，そのノードは木の各々その子は文字種数の長さの配列であり，文字列ポインタは娘ノードの文字によりラベル付けられている．
- □文字列の検索はrootノードから出発する．検索文字列の文字を先頭から調べ，それに応じて娘のノードを順次選択していく
- □時間計算量は辞書の規模によらず，検索文字列の長さ k について o(k)
- □字種が多い言語では各ノードのデータ構造が非常に大きくなる．(cf. Digital Trie)

## 二進トライ(Binary-Trie) (1)

- 1,0を文字とするトライ
- 文字列をbit列(1/0の列)とみる．
  - 文字列→文字の並び，文字→文字コード→ビット列
- 必ず二分木となるためノードのデータ構造が小さくなる．

## 二進トライ(2)

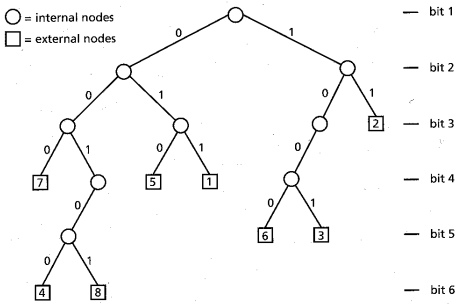



**Figure 2.5** Binary trie (external node label indicates position in the text) for the first eight suffixes in "01100100010111 . . .".

## パトリシア木(Patricia Tree)

- 二進トライから，枝分かれのない枝をとり去ったもの
- Practical Algorithm To Retrieval Information Coded In Alphanumerical
- 共通のプレフィックス(prefix, 先頭の部分文字列)が長い文字列が多い場合は，ノード数が少なくなり，木の高さが低くなるので，空間計算量，時間計算量ともに改善される．

## Patricia Tree

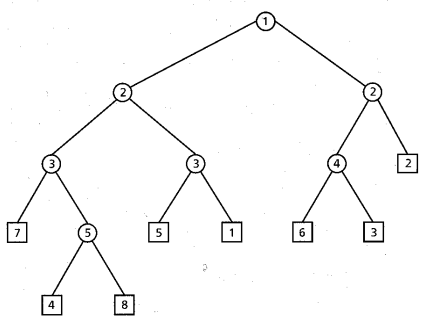


**Figure 2.6** Patricia tree (internal node label indicates bit number).

## 署名ファイル(Signature File)方式(1)

転置ファイル方式との比較

- 転置ファイル方式
  - 索引ファイルが元テキストと同等かそれ以上になることが多い
    - 大規模テキストベースでは無視できない問題
- 署名ファイル
  - 文書の特徴をビット列で表現して，索引ファイルを小さくする．

## 署名ファイル(Signature File)方式(2)

- 各ファイルに対して「署名」を作成する．
- 「署名」：ある語が含まれるかどうか簡単にわかるようにしたビット列
  - 語に対する署名
    - 語の文字列に対してハッシュ関数を適用
  - 文書の署名：1となるビット位置だけが1であるビット列
- 検索：検索要求となるすべての語のsignatureの論理和
  - 検索要求となる単語に対応する署名を求める
  - この署名において，1となっているビット位置が1となっている署名をもつ文書をとり出す．
  - 誤った文書をとり出すことがあって衝突がある可能性があるので，後処理を必要とする．

## 署名ファイル(Signature File)方式(3)

‘情報’ 0000 0000 0000 0010 0000
‘検索’ 0000 0001 0000 0000 0000
‘手法’ 0000 0000 0000 0001 0000

これらをすべて持つ文書の署名

0000 0001 0000 0011 0000