5. Merkkijonohakuun perustuvat menetelmät

"text scanning", "full text scanning/search"

- haku ilman indeksointia
  - editorin tms. FIND-komento
  - algoritmisesti tärkeä:
    sanat, alkuosat, loppuosat, osajonot
- yksinkertainen tapaus:
  yksi tiedosto
  yksi haun kohde (hakutermi)
- yleistyksiä:
  monta tiedostoa
  monta hakutermiä, Boolen logiikka
  haku likimääräisesti: 1..k virhettä
- käyttökelpoisuus: ainakin 5 -10 MB

Käyttötapoja:
- esiintyykö 'xyz'?
- missä kohdassa 'xyz' esiintyy?
- missä kaikissa kohdissa 'xyz' esiintyy?
  → tuloksen esittämisen vaikeudet

5.1 Merkkijonoalgoritmit
(vrt. merkkijonomenetelmien kurssi)

- haetaan merkkijonon P (hahmon; pattern) esiintymä(t)
  merkkijonosta S |S| >> |P|
  - merkit numeroidaan 1, 2, ...

1° Perusalgoritmi

```
S = a b c a d c a c d a b
P = a d c
      a d c
        a d c
          a d c
                . . .
```

- pahin tapaus: O(m x n) vertailua

```
S = a a a a a a a ... a a a b
P = a b
      a b
        a b
          .....          a b
```

- keskimääräinen tapaus: O(n)
  täsmäämättömyys havaitaan O(1) vertailulla

Perusmenetelmässä m-pituinen 'ikkuna' liukuu koko tekstin yli: merkkien vertailu ja siirto merkin verran. Kehittyneemmissä varianteissa pyritään pitempiin siirtoihin; lisäinformaatiota tarvitaan.

2° KMP-algoritmi (Knuth, Morris, Pratt)

- siirretään kyselyn merkkijonoa kerralla
  enemmän oikealle

```
S =  b a b c b a b c a b c a a b c a
P =  a b c a b c a c a b
       a b c a b c a c a b
```

- a b c ei sisällä toistuvia osia

```
S =  b a b c b a b c a b c a a b c a
           a b c a b c a c a b
             a b c a b c a c a b
```

- a b c a toistuu => ei voida siirtää koko
  täsmäävän osan ohi, vaan on verrattava
  alku- ja loppuosia

```
S =  b a b c b a b c a b c a a b c a
                 a b c a b c a c a b
```

- yleinen tapaus: hahmon merkeille määrätään next-arvot: next[j] ilmaisee, kuinka pitkä on pisin P[1:j-1]:n alkuosa, joka on myös loppuosa

```
next:   0 0 0 0 1 2 3 1 0 1
        a b c a b c a c a b
```


Algoritmin perusaskeleet (P[j] = T[i] ?)
- jos merkit samat, siirry seuraaviin
- jos merkit eroavat, siirry hahmossa next[j]
  merkkiä eteenpäin
  (ero alussa: kummassakin eteenpäin)

- paras tapaus: voidaan siirtyä m askeleen hyppäyksin; suoritusaika O(n)
- pahin tapaus: O(m+n) vertailua
- hahmon esiprosessointi (next): O(m)

3° BM-algoritmi (Boyer & Moore)

- merkkivertailu oikealta vasemmalle,
  siirto mahdollisesti m merkkiä kerrallaan

S = banana cream ice
P = cream

n ↔ m ? n ei esiinny P:ssä =>

S = banana cream ice
cream

e ↔ m ? e esiintyy P:ssä =>
siirretään vain 2 positiota

S = banana cream ice
cream (match)

- pahin tapaus O(n+m)
- käytännössä vertailuja vähän: O(n x log(m) / m)
  (alle n: kaikkia merkkejä ei tutkita ollenkaan)
- hahmon esiprosessointi: O(m+σ),
  σ on aakkoston koko
- paljon variantteja …
- muita menetelmiä …
  esim. shift-or (nopeutuu kertoimella1/w, missä
  w = koneen sananpituus)
  (taustana usein automaattien teoria)

5.2 Merkkijonohakuun perustuva kokonaisjärjestelmä: GLIMPSE

(GLobal IMPlicit Search) Manber & Wu, 1994

- merkkijonomenetelmiä:
  likimääräishaku
  sanat / merkkijonot
  säännölliset lausekkeet

- indeksit
  erilaisia vaihtoehtoja
  pienempi tilantarve kuin
  käänteishakemistoissa

- kaksivaiheinen toiminta: haku indeksistä,
  kohdetiedostoista
  (samojakin rutiineja, ei näy käyttäjälle)

- yleispiirteitä
  nopeus 'riittävä'
  indeksit 'riittävän pieniä'
  monipuoliset toiminnot
  - optioita aitoon Unix-tyyliin ...

- kohteena tekstitiedostot
  käsittely hakemistoittain
  erityisiä tiedostoja jätetään pois
  ... .o, ...gz, ...Z, ...zip, ...tar
  luettelotyyppiset (heuristisesti)
  -w 1000: enintään 1000 uutta sanaa
  (muuten pidetään sanaluettelona)
  parametrina ilmaistut

- uusi versio: WebGlimpse (1997)

Indeksointi

3 indeksityyppiä:

- mini-indeksi (tiny): sanan lohko-osoitteet
  lohko voi olla useita tiedostoja
  2-4 % tiedoston koosta
- pieni indeksi (small):
  kullekin tiedostolle oma lohko
  7-8 %
- suuri indeksi (medium):
  sanan esiintymille tavuosoitteet
  20-30 %

- indeksointiaika
  esim. 200 MB (15000 tiedostoa) / 20 min
  yöaikaan
  inkrementaalinen indeksointi mahdollinen
  tiedostojen lisäys
  muuttuneet/uudet tiedostot

- indeksoinnin kohteet
  sanoja
  fraaseja ei indeksoida
  haku1: sanat indeksistä
  haku2: fraasit haun 1 tulostiedostoista

  erikoistapauksena lukuja
  glimpseindex -n tdstot
  vuosiluku, päivämäärä
  lukuja (väh. puolet) sisältävät tiedostot
  ilmoitetaan

- hukkasanalistat
  - dynaamisia, indeksivaihtoehdon mukaan

  mini-indeksi: ei käytetä
  hyöty olisi pieni

  pieni indeksi: sana, joka esiintyy vähintään
  k %:ssa tiedostoja (esim. k = 80)
  glimpseindex -S 80 ...

  suuri indeksi: sana, jolla vähintään
  k esiintymää / 1 MB (esim. k = 500)

- sananmuodot / stemmaus
  ei käytetä; joustavat hakumahdollisuudet
  (sopivuus suomen kieleen ?)

Haku (varsinainen glimpse - ohjelma)

glimpse [- almost all letters] file(s) pattern

= joustava grep (agrep)
- likimääräinen samanlaisuus
- Boolen operaatiot
- säännölliset lausekkeet

· kohteena rivi, muutettavissa (esim. kappale)

· haettava hahmo

merkkijono 'Arizona'
- sana: glimpse -w ... parent
  - ei: transparent, parenthesis
- rivin alku (^), loppu ($)
- metamerkit, niiden haku: esim. \$\$
  - tai -k 'a(b<c)*d'
- jokerimerkit . # esim. H#ki

merkkiluokat
- [aeiop-t] [^a-c]

Boolen operaatiot
- {political,computer};science

säännölliset lausekkeet (rajoituksia)

(Yleisesti:
- e1
- e2
- (e1|e2)
  - kis (sa|a)
- (e)*
  - (abrakad)* abra

ym.)

tarkan ja likimääräisen osan yhdistelmä
(kun likimääräisyys muuten ilmaistu)

<mathematic>s mathe<matics>

- täsmääkö 'mathematica' yhdellä virheellä ?

kohteen määritys (muu kuin rivi):

-d <rajoitin>

-d '$$'	kappaleraja
-d '^From\ '	sähköpostiviesti

· likimääräinen haku:

- case: glimpse -i 'parent' Part PART

- sallittujen virheiden määrä

glimpse -1 .... 'Arzona' 'Arisona'

myös tiedostonimessä:
glimpse -F '-1 \.html' jotain .shtml .htm

glimpse -2 .... 'Oriona' ei: 'Orion'

glimpse -D2	poisto = 2 virhettä
-I2	lisäys = 2 virhettä
-S3	korvaus = 3 virhettä

(painotusten merkitys?)

- best match mode: interaktiivinen
  "best match: 2 errors, 14 matches;
  output them (y/n)?"

· muita toimintoja:

-x	täsmäys koko riviin
-c	vain esiintymien lkm tulostetaan
-l	vain tiedostojen nimet tulostetaan
-N	haetaan vain indeksistä nopea, tulos voi olla epätarkka (indeksi pienin kirjaimin, case-virheet ...)
-v	tulokseen ne, joissa ei esiintymää
-W	Boolen operaatio koskee koko tiedostoa eikä riviä (kohdetta)
-Y 7	tutkitaan vain viimeisen viikon aikana luodut tai muutetut tdstot

Glimpse: monen palvelun 'hakumoottori'
- erilaisia käyttöliittymiä (peittävät optiot ...)
  (esim. Tktl:n kirjastojärjestelmä)

WebGlimpse

· joukko URLeja lähtökohdaksi

· paikallisia tai ei-paikallisia sivuja (’naapureita’
mukaan haluttuun linkkietäisyyteen asti
- etäisyys voi vaihdella,
esim. kaikki paikalliset, muut 2 linkkiin asti

· muodostuu linkkiverkko
esim. suosikkisivut

· haku valituille sivuille

5.3 Merkkijonohaun käyttö

- usein ‘riittävän’ nopea, ilman indeksointiakin

- helppo, jos kohde on yhtenäinen
tavallaan automaattinen ‘selausväline’
(selauksen vaihtoehto ?)

esim. haun tuloksena saatu dokumentti,
WWW-sivu, luettelo, ...
= selaimen Find tms.

- sallii sanan, merkkijonon, fraasin

- likimääräinen haku
case/CASE, virheiden määrä

- kätevyys riippuu tavoitteesta:

yksi (tietty) esiintymä: siirto tiettyyn kohtaan

yksi (mahdollinen) esiintymä: varmistus

useita esiintymiä: näyttäminen on ongelma
ensimmäinen, kaikki, paras?

eteenpäin / taaksepäin (vrt. ’oikea’ haku ...)

(Yleisesti yleisen haun ja sivunsisäisen erot
mielenkiintoinen asia.)

5.4 Tekstin tiivistäminen

[BYRN, Ch. 7.4, 8.8]

· laajat materiaalit → tiivistämisen
tarvetta periaatteessa aina

- säilytystila, siirtoaika (i/o, verkon yli), etsintäaika

· tiedonhaku asettaa erityisvaatimuksia:
sanan (tekstikohdan) suorahaku,
voi vaatia koko tekstin laventamisen

· tiedonhaussa tiivistys sanoittain (ei merkeittäin)
suorahaku voi olla mahdollinen

· suorituskykytekijät
tilantarve
tiivistysaika ’kerran’
purkamisaika ’usein’

· käänteishakemiston tiivistys
sanaston sanat
esiintymälistat
nouseva järjestys sallii tiivistyksen:
differenssit pienempiä lukuja kuin
tekstikohdan osoitteet

Sanativiistyksen menetelmiä:

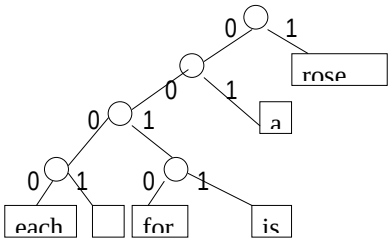
tilastolliset sanastopohjaiset
Huffman Ziv-Lempel

Huffman-koodaus:
- yleisimmille sanoille lyhimmät koodit
- purkaminen koodin bittirakenteen (puun) avulla

Esim. teksti

for each rose, a rose is a rose

Huffman-koodaus sisältää binäärisen trie-rakenteen:



0010 0000 1 0001 01 1 0011 01 1

Peräkkäishaku (ilman käänteishakemistoa):
- haetaan koodisanastosta koodi
- haetaan koodi tiivistetystä tiedostosta