BFS dan DFS

Bahan Kuliah IF2151 Strategi Algoritmik

Oleh: Rinaldi Munir

- Traversal di dalam graf berarti mengunjungi simpul-simpul dengan cara yang sistematik.

- Algoritma traversal di dalam graf:

  1. BFS: Pencarian Melebar (*Breadth First Search)*,

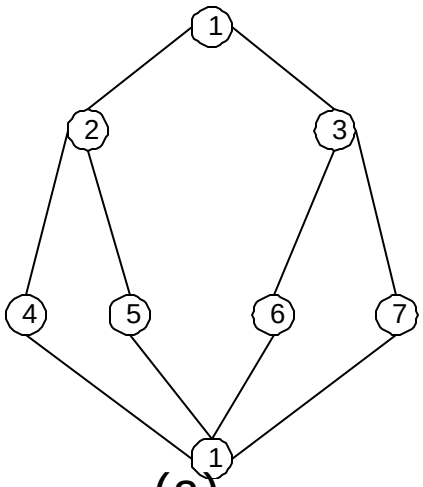
  2. DFS: Pencarian Mendalam (*Depth First Search*).

Algoritma Pencarian Melebar (*BFS*)

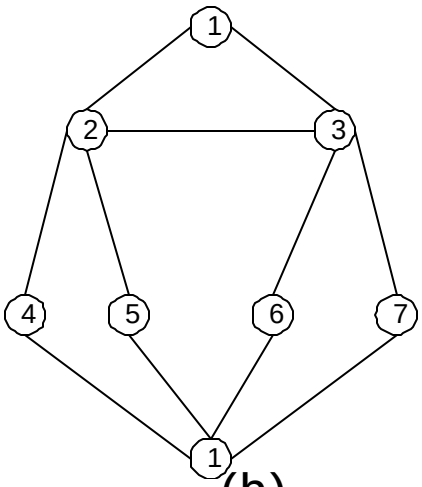
- Traversal dimulai dari simpul *v*.
- Algoritma:
  1. Kunjungi simpul *v*,
  2. Kunjungi semua simpul yang bertetangga dengan simpul *v* terlebih dahulu.
  3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

- Jika graf berbentuk pohor berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum mengunjungi simpul-simpul pada aras $d + 1$.

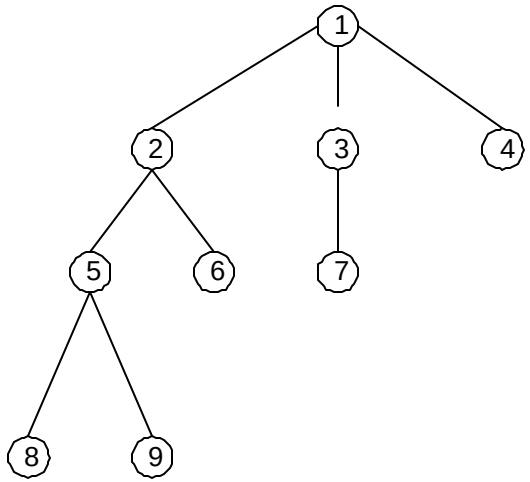
Contoh 1: (misalkan traversal dimulai dari simpul 1)



(a)



(b)



(c)

Gambar (a) BFS(1): 1, 2, 3, 4, 5, 6, 7, 8.

Gambar (b) BFS(1): 1, 2, 3, 4, 5, 6, 7, 8

Gambar (c) BFS(1): 1, 2, 3, 4, 5, 6, 7, 8, 9

- *Pseudo-code* algoritma:
- Diperlukan:
  1. Matriks ketetanggaan $A = [a_{ij}]$ yang berukuran $n \times n$,

     $a_{ij} = 1$, jika simpul i dan simpul j bertetangga,

     $a_{ij} = 0$, jika simpul i dan simpul j tidak bertetangga.
  2. Antrian q untuk menyimpan simpul yang telah dikunjungi.

3. Tabel *boolean* yang bernama `dikunjungi`

```
dikunjungi : array[l..n] of boolean
```

dikunjungi[i] = true jika simpul *i* sudah dikunjungi

dikunjungi[i] = false jika simpul *i* belum dikunjungi

Inisialisasi tabel:

```
for i←l to n do
   dikunjungi[i]← false
endfor
```

```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian BFS.

  Masukan: v adalah simpul awal kunjungan
  Keluaran: semua simpul yang dikunjungi dicetak ke layar
}
Deklarasi
   w : integer
   q : antrian;

   procedure BuatAntrian(input/output q : antrian)
   { membuat antrian kosong, kepala(q) diisi 0 }

   procedure MasukAntrian(input/output q:antrian, input v:integer)
   { memasukkan v ke dalam antrian q pada posisi belakang }

   procedure HapusAntrian(input/output q:antrian,output v:integer)
   { menghapus v dari kepala antrian q }

   function AntrianKosong(input q:antrian) → boolean
   { true jika antrian q kosong, false jika sebaliknya }

Algoritma:
   BuatAntrian(q)        { buat antrian kosong }

   write(v)              { cetak simpul awal yang dikunjungi }
   dikunjungi[v]←true    { simpul v telah dikunjungi, tandai dengan
                           true}
   MasukAntrian(q,v)     { masukkan simpul awal kunjungan ke dalam
                           antrian}

  { kunjungi semua simpul graf selama antrian belum kosong }
   while not AntrianKosong(q) do
       HapusAntrian(q,v)  { simpul v telah dikunjungi, hapus dari
                            antrian }
       for tiap simpul w yang bertetangga dengan simpul v do
             if not dikunjungi[w] then
                write(w)    {cetak simpul yang dikunjungi}
                MasukAntrian(q,w)
                dikunjungi[w]←true
             endif
      endfor
   endwhile
  { AntrianKosong(q) }
```

```
procedure BFS(input v:integer)
{ Traversal graf dengan algoritma pencarian BFS.

  Masukan: v adalah simpul awal kunjungan
  Keluaran: semua simpul yang dikunjungi dicetak ke layar
}
Deklarasi
   w : integer
   q : antrian;

   procedure BuatAntrian(input/output q : antrian)
   { membuat antrian kosong, kepala(q) diisi 0 }

   procedure MasukAntrian(input/output q:antrian, input v:integer)
   { memasukkan v ke dalam antrian q pada posisi belakang }

   procedure HapusAntrian(input/output q:antrian,output v:integer)
   { menghapus v dari kepala antrian q }

   function AntrianKosong(input q:antrian) → boolean
   { true jika antrian q kosong, false jika sebaliknya }

Algoritma:
   BuatAntrian(q)       { buat antrian kosong }

   write(v)             { cetak simpul awal yang dikunjungi }
   dikunjungi[v]←true   { simpul v telah dikunjungi, tandai dengan
                          true}
   MasukAntrian(q,v)    { masukkan simpul awal kunjungan ke dalam
                          antrian}

  { kunjungi semua simpul graf selama antrian belum kosong }
   while not AntrianKosong(q) do
       HapusAntrian(q,v)  { simpul v telah dikunjungi, hapus dari
                            antrian }
       for w←l to n do
          if A[v,w] = 1 then     { v dan w bertetangga }
             if not dikunjungi[w] then
                write(w)    {cetak simpul yang dikunjungi}
                MasukAntrian(q,w)
                dikunjungi[w]←true
             endif
          endif
      endfor
   endwhile
  { AntrianKosong(q) }
```

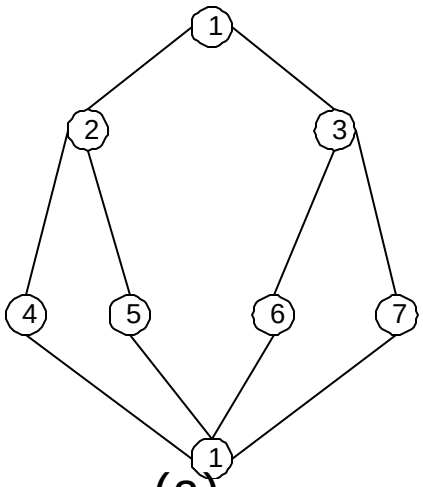
Metode Pencarian Mendalam (DFS)

- Traversal dimulai dari simpul *v*.
- Algoritma:
  1. Kunjungi simpul *v*,
  2. Kunjungi simpul *w* yang bertetangga dengan simpul *v*.
  3. Ulangi *DFS* mulai dari simpul *w*.

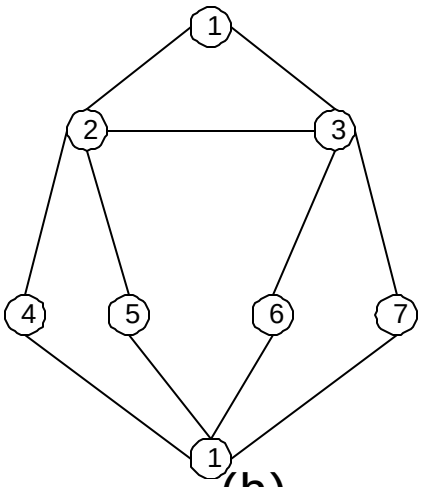
4. Ketika mencapai simpul *u* sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul *w* yang belum dikunjungi.

5. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

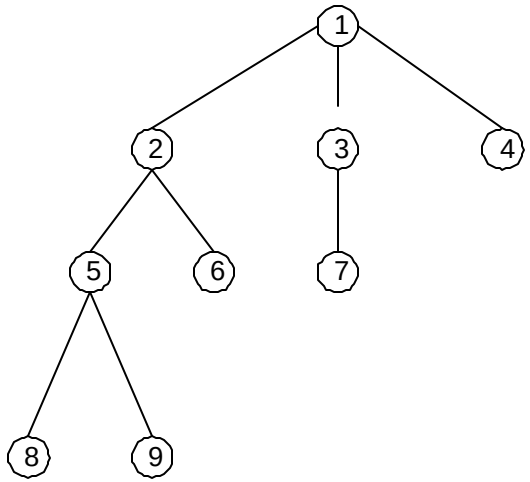
Contoh 2: (misalkan traversal dimulai dari simpul 1)



(a)



(b)



(c)

Gambar (a) DFS(1): 1, 2, 4, 8, 5, 6, 3, 7

Gambar (b) DFS(1): 1, 2, 3, 6, 8, 4, 5, 7

Gambar (c) DFS(1): 1, 2, 5, 8, 9, 6, 3, 7, 4

```
procedure DFS(input v:integer)
{Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS

Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpulyang dikunjungi ditulis ke layar
}
Deklarasi
   w : integer
Algoritma:
   write(v)
   dikunjungi[v]←true
   for tiap simpul w yang bertetangga dengan simpul v do
      if not dikunjungi[w] then
          DFS(w)
      endif
   endfor
```

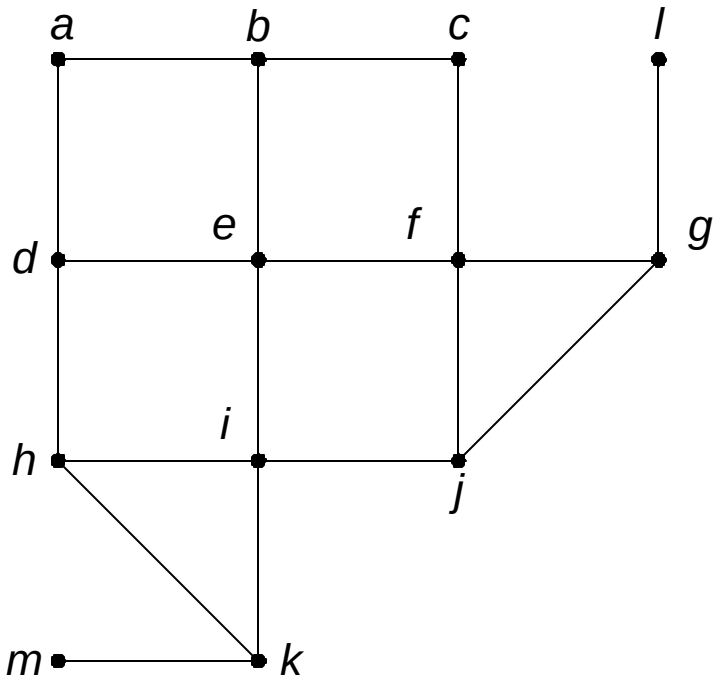
Algoritma DFS selengkapnya adalah:

```
procedure DFS(input v:integer)
{Mengunjungi seluruh simpul graf dengan algoritma pencarian DFS

Masukan: v adalah simpul awal kunjungan
Keluaran: semua simpulyang dikunjungi ditulis ke layar
}
Deklarasi
   w : integer

Algoritma:
   write(v)
   dikunjungi[v]←true
   for w←l to n do
      if A[v,w]=l then {simpul v dan simpul w bertetangga }
          if not dikunjungi[w] then
             DFS(w)
          endif
      endif
   endfor
```

Latihan: Gunakan algoritma *BFS* dan *DFS* untuk menemukan pohon merentang (*spanning tree*) dari graf *G* di bawah ini jika traversalnya dimulai dari simpul *e*. Dalam menjawab soal ini, perlihatkan traversal *BFS/DFS* sebagai pohon berakar dengan *e* sebagai akarnya.



Aplikasi *DFS* dan *BFS*

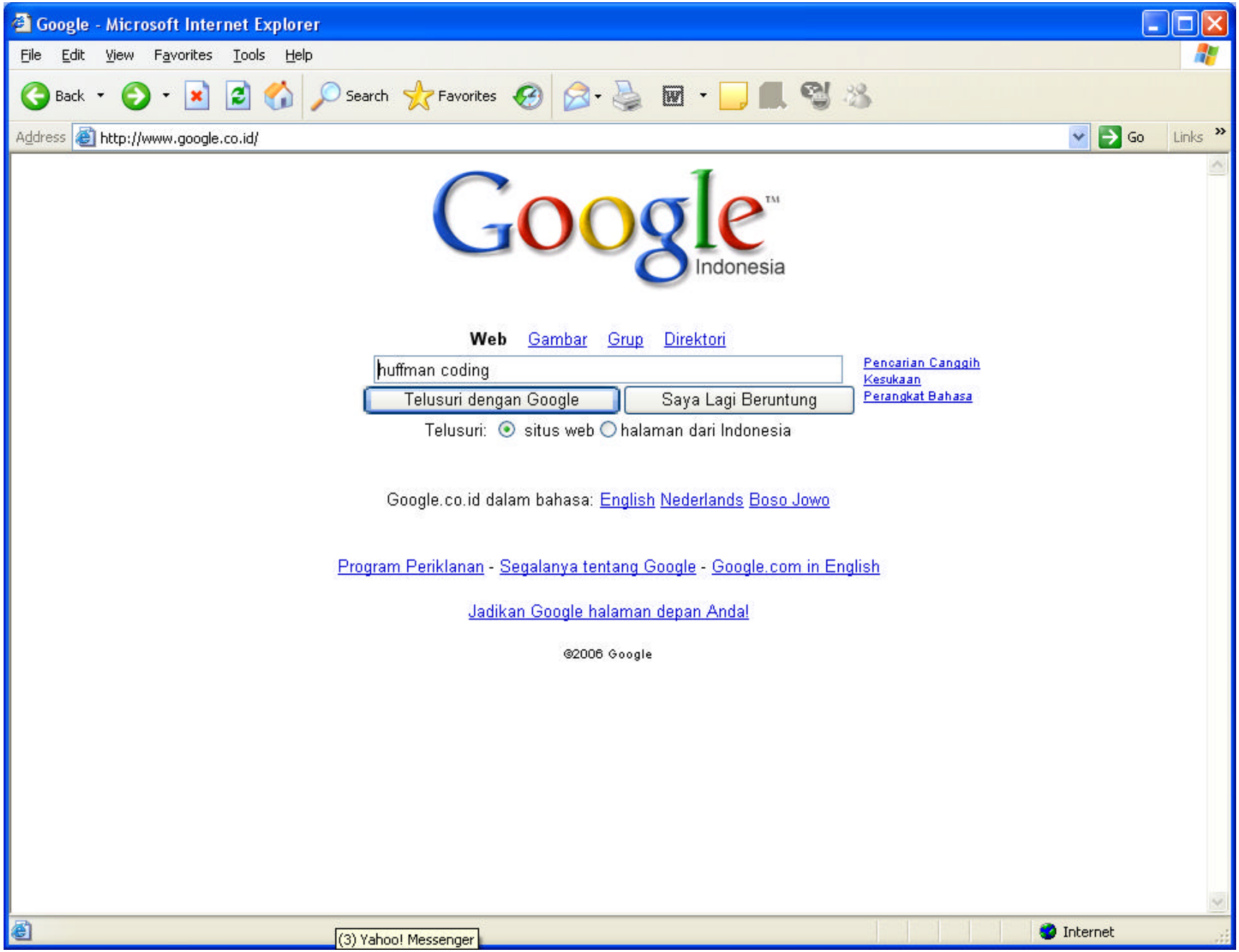
1. Search Engine (*google*, *yahoo*, *altavista*)

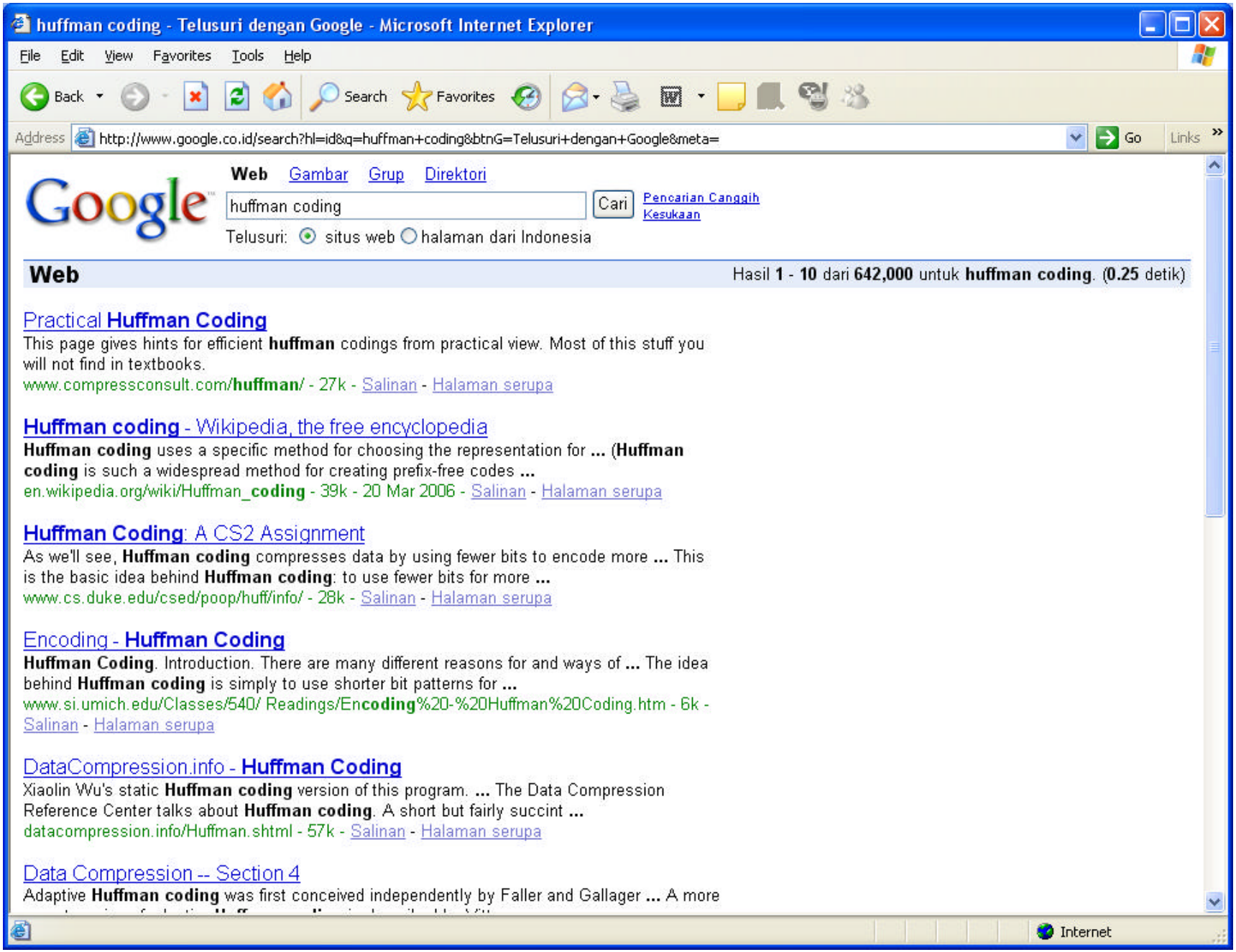
Komponen *search engine*:

1. *Web Spider* : program penjelajah *web* (*web surfer*)
2. *Index*: basisdata yang menyimpan kata-kata penting pada setiap halaman *web*
3. *Query*: pencarian berdasarkan string yang dimasukkan oleh pengguna (*end- user*)

Secara periodik (setiap jam atau setiap hari), *spider* menjejalahi *internet* untuk mengunjungi halaman-halaman *web*, mengambil kata-kata penting di dalam *web*, dan menyimpannya di dalam *index*.

Query dilakukan terhadap *index*, bukan terhadap *website* yang aktual.


Google - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.google.co.id/
Go Links
Google™ Indonesia
Web Gambar Grup Direktori
huffman coding
Pencarian Canggih
Kesukaan
Perangkat Bahasa
Telusuri dengan Google
Saya Lagi Beruntung
Telusuri: situs web halaman dari Indonesia
Google.co.id dalam bahasa: English Nederlands Boso Jowo
Program Periklanan - Segalanya tentang Google - Google.com in English
Jadikan Google halaman depan Anda!
©2006 Google
(3) Yahoo! Messenger
Internet


huffman coding - Telusuri dengan Google - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.google.co.id/search?hl=id&q=huffman+coding&btnG=Telusuri+dengan+Google&meta=
Go Links
Google
Web Gambar Grup Direktori
huffman coding
Cari
Pencarian Canggih
Kesukaan
Telusuri: situs web halaman dari Indonesia
Web
Hasil 1 - 10 dari 642,000 untuk huffman coding. (0.25 detik)
Practical Huffman Coding
This page gives hints for efficient huffman codings from practical view. Most of this stuff you will not find in textbooks.
www.compressconsult.com/huffman/ - 27k - Salinan - Halaman serupa
Huffman coding - Wikipedia, the free encyclopedia
Huffman coding uses a specific method for choosing the representation for ... (Huffman coding is such a widespread method for creating prefix-free codes ...
en.wikipedia.org/wiki/Huffman_coding - 39k - 20 Mar 2006 - Salinan - Halaman serupa
Huffman Coding: A CS2 Assignment
As we'll see, Huffman coding compresses data by using fewer bits to encode more ... This is the basic idea behind Huffman coding: to use fewer bits for more ...
www.cs.duke.edu/csed/poop/huff/info/ - 28k - Salinan - Halaman serupa
Encoding - Huffman Coding
Huffman Coding. Introduction. There are many different reasons for and ways of ... The idea behind Huffman coding is simply to use shorter bit patterns for ...
www.si.umich.edu/Classes/540/ Readings/Encoding%20-%20Huffman%20Coding.htm - 6k - Salinan - Halaman serupa
DataCompression.info - Huffman Coding
Xiaolin Wu's static Huffman coding version of this program. ... The Data Compression Reference Center talks about Huffman coding. A short but fairly succint ...
datacompression.info/Huffman.shtml - 57k - Salinan - Halaman serupa
Data Compression -- Section 4
Adaptive Huffman coding was first conceived independently by Faller and Gallager ... A more
Internet

Bagaimana *spider* menjelajahi (*surfing*) *web?*

- Halaman web dimodelkan sebagai graf berarah
- Simpul menyatakan halaman web (*web page*)
- Sisi menyatakan *link* ke halaman web
- Bagaimana teknik menjelajahi *web*? Secara DFS atau BFS
- Dimulai dari *web page* awal, lalu setiap *link* ditelusuri secara *DFS* sampai setiap *web page* tidak mengandung *link.*
- Pada setiap halaman *web*, informasi di dalamnya dianalisis dan disimpan di dalam basisdata *index*.

2. Referensi pada Makalah/Jurnal

- Pada setiap makalah, ada acuan ke literatur yang digunakan.
- Pada literatur tsb, ada acuan ke makalah/literatur yang lain?
- Bagaimana menelusuri acuan-acuan pada makalah? Secara DFS atau BFS?

in two or three pixels) alignment with the expected watermark signal and is less sensitive to imperfect registration. This is reflected in the two techniques' relative performance after printing and scanning, although the NEC detection following substantial watermark degradation can be improved with post-processing techniques.

Likewise, the results following an attack by Stirmark[3] indicate the NEC detector's potential vulnerability following subtle image distortions. Stirmark is a tool designed to test watermark robustness by introducing geometric distortions similar to those encountered when printing and scanning.

References

1. I.J. Cox et al., *Secure Spread Spectrum Watermarking for Multimedia*, Tech. Report 95-10, NEC Research Institute, Princeton, N.J., 1995.
2. W.R. Bender, D. Gruhl, and N. Morimoto, "Techniques for Data Hiding," *Proc. SPIE: Storage and Retrieval of Image and Video Databases*, Vol. 2,420, Soc. of Photo-Optical Instrumentation Engineers, Bellingham, Wash., Feb. 1995, pp. 164–173.
3. M. Kuhn, "Stirmark—Image Watermarking Robustness Test, Version 1.0, 1997-11-10," electronic copy available at *http://www.cl.cam.ac.uk/~mgk25/stirmark.html/*.