Data Crawling

Big Data Analytics

Pengenalan *Data Crawling* (1)

- Big Data adalah terminologi pada komputer yang merujuk kepada pengolahan data yang memiliki ukuran yang besar.
- Salah satu sumber yang dapat menyebabkan ukuran data yang besar adalah *data crawling*.
- *Data crawling* adalah teknik untuk mencari informasi dari berbagai sumber data. *Data crawling* akan melacak informasi ke setiap level sumber data yang dapat diakses.

Pengenalan *Data Crawling* (2)

- Sumber data crawling yang paling umum adalah "website".
  - *Data crawling* yang bersumber dari website biasa disebut *web crawling*.
- *Data crawling* sangat berguna bagi perusahaan dalam mencari berbagai informasi yang tersebar di berbagai website.
- Terutama untuk perusahaan *search engine* yang menggunakan *web crawling* untuk membuat index web dari sebuah mesin pencarian di internet.

Konsep Dasar *Data Crawling* (1)

- *Data crawling* dilakukan menggunakan sebuah aplikasi *crawler* dengan konfigurasi tertentu.
- Aplikasi *crawler* adalah sebuah program yang dibuat untuk menjangkau semua sumber informasi dan melakukan aksi yang sudah ditentukan seperti memeriksa “kesegaran” dari sebuah informasi atau mengambil data dari sebuah sumber informasi.

Konsep Dasar *Data Crawling* (2)

- Langkah-langkah dasar dari sebuah proses *data crawling* adalah seperti terlihat pada gambar berikut:

Konsep Dasar *Data Crawling* (3)

- Langkah-langkah *Data Crawling*:
  - Menentukan sumber-sumber informasi
    - Langkah pertama adalah menentukan atau membuat daftar sumber-sumber informasi. Misalnya untuk *web crawling*, membuat daftar-daftar URL website yang akan diambil informasinya. Daftar website harus kredibel dan hindari website yang tidak mengijinkan *automated crawling* pada konfigurasi robots.txt atau di halaman TOS.
  - Mengkonfigurasi aplikasi *crawler*
    - Langkah kedua membutuhkan kemampuan teknis khususnya kemampuan pemrograman untuk memahami struktur data pada sumber informasi dan mengidentifikasi poin-poin yang bisa diambil dari sumber informasi tersebut sesuai dengan tugas yang sudah ditentukan.

Konsep Dasar *Data Crawling* (4)

- Melakukan *cleansing* dan menghilangkan duplikasi data
  - Data awal hasil *crawler* umumnya penuh dengan data-data anomali dan mengandung duplikasi informasi. Kondisi ini dapat mempengaruhi akurasi dari proses dan analisa data. Karena itu, langkah ini menjadi penting untuk membersihkan data dari data-data anomali serta data yang terduplikasi.
- Restrukturisasi data
  - Data yang didapat dari hasil *cleansing* dan penghilangan duplikasi, perlu diubah struktur-nya ke dalam skema yang dipahami oleh komputer. Dengan data yang terstruktur, akan mempermudah pemrosesan dan analisa lebih lanjut.

Web Crawlers (1)

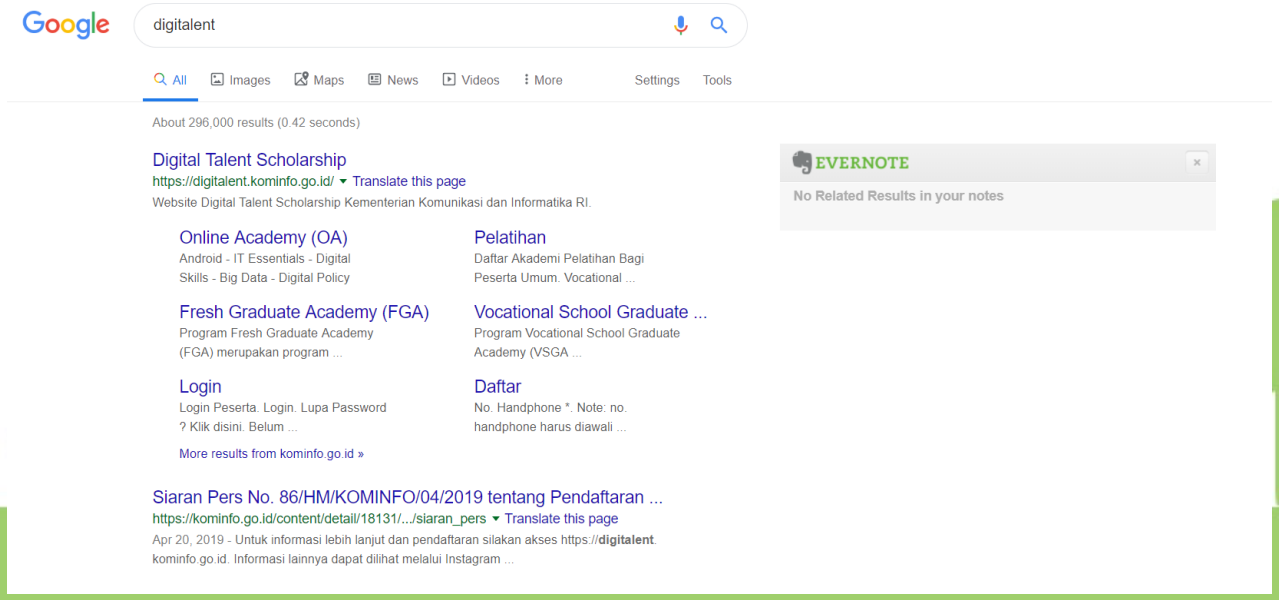
- Untuk memperdalam pemahaman tentang *Data Crawling*, selanjutnya akan dibahas salah satu contoh yaitu *Web crawling*.
- *Web crawling* secara spesifik mencari suatu informasi dari sekumpulan website yang ada di Internet.
- Aplikasi yang digunakan untuk melakukan *web crawling* adalah *web crawlers*.

Web Crawlers (2)

- *Web crawler* adalah
  - sebuah proses atau program yang digunakan oleh mesin pencari untuk mengunduh halaman-halaman dari web untuk diproses lebih lanjut dalam membuat index web.
    - Index web kemudian digunakan untuk mempercepat proses pencarian informasi di Internet.
  - sebuah program atau script otomatis yang merambah website di internet dengan cara yang terstruktur.
  - juga dikenal sebagai aplikasi *web spider* dan *web robots*.
  - Istilah lain yang jarang digunakan: *ants*, *bots* dan *worms.*

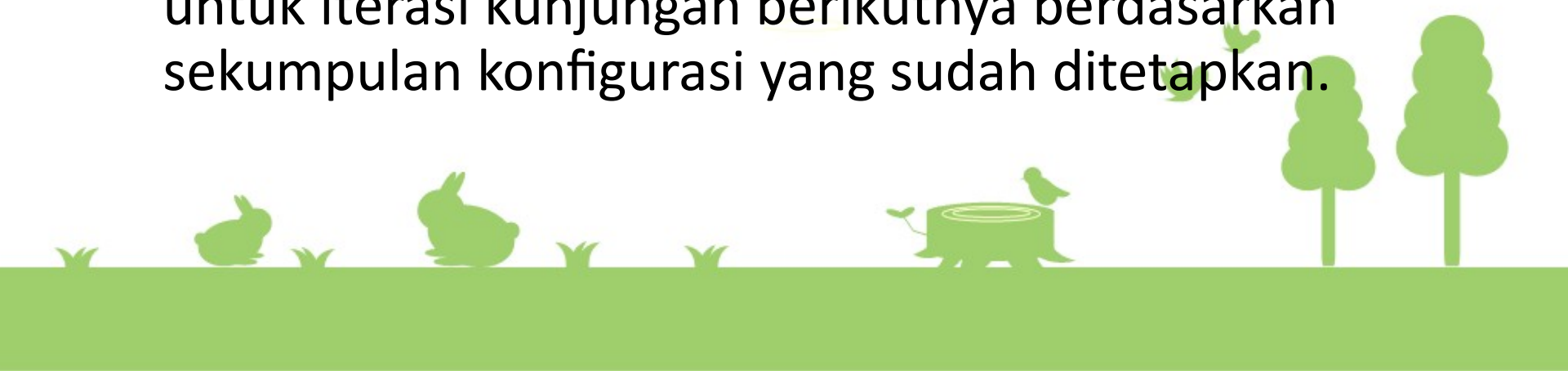
Web Crawlers (3)

- *Web crawlers* sangat berguna karena:
  - Internet memiliki segudang website yang mengandung berbagai jenis informasi.
  - Mencari informasi yang relevan dari sekumpulan website di Internet membutuhkan mekanisme yang efisien.



Cara Kerja *Web Crawler* (1)

- *Web crawler* mengawali proses dari daftar URL website yang harus dikunjungi yang biasa disebut *seeds*.
  - Saat *web crawler* mengunjungi setiap website di dalam daftar tersebut, ia akan mengidentifikasi sebuah *hyperlinks* yang ada di dalam website tersebut dan menambahkannya ke dalam daftar URL yang telah dikunjungi, yang disebut *crawl frontier*.
- URL dari daftar *crawl frontier* kemudian menjadi *seeds* untuk iterasi kunjungan berikutnya berdasarkan sekumpulan konfigurasi yang sudah ditetapkan.



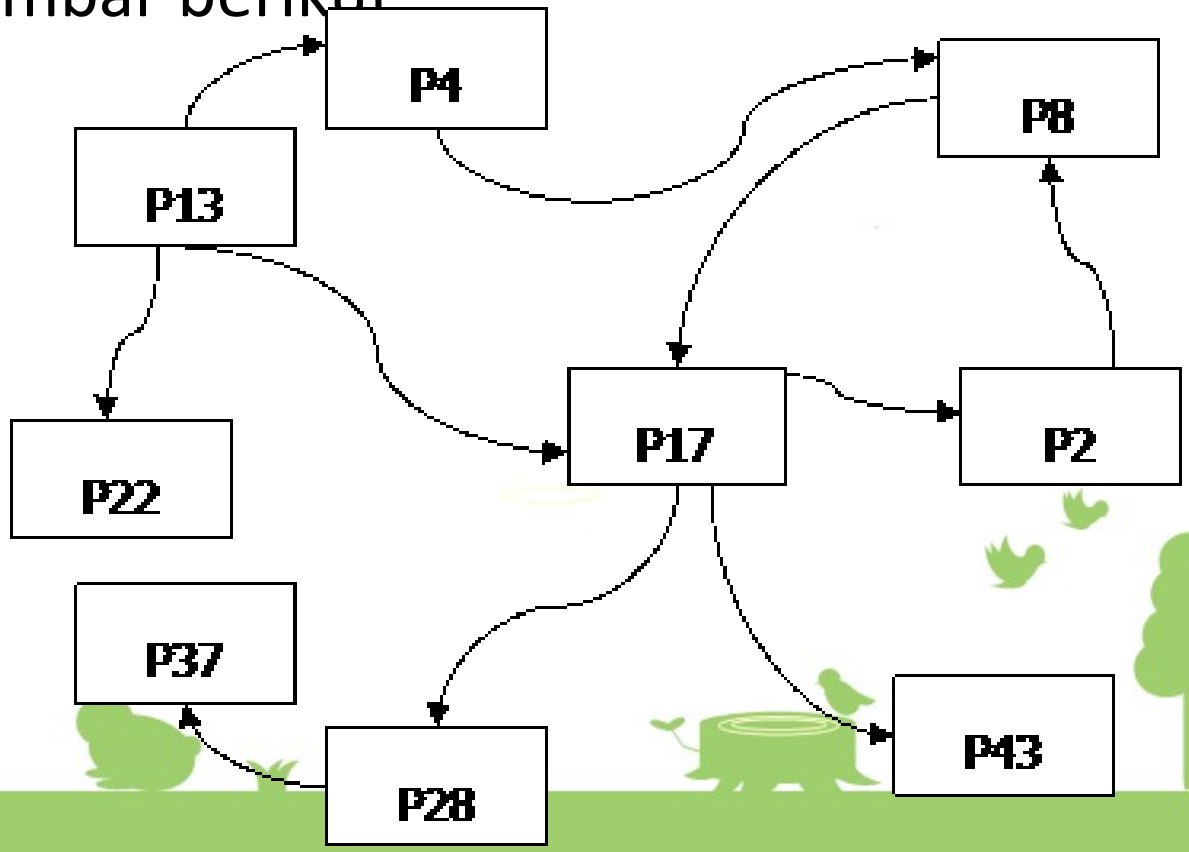
Cara Kerja *Web Crawling* (2)

- *Web crawlers* bekerja dengan algorima sebagai berikut:

```
Initialize queue (Q) with initial set of known URL's.
Until Q empty or page or time limit exhausted:
  Pop URL, L, from front of Q.
  If L is not an HTML page (.gif, .jpeg, .ps, .pdf, .ppt…)
       exit loop.
  If already visited L, continue loop(get next url).
       Download page, P, for L.
  If cannot download P (e.g. 404 error, robot excluded)
       exit  loop,
  else.
       Index P (e.g. add to inverted index or store cached
 copy).
  Parse P to obtain list of new links N.
  Append N to the end of Q.
```

Cara Kerja *Web Crawling* (3)

- Hasil *web crawler* biasanya berupa graph index seperti pada gambar berikut:



Strategi *Web Crawler* (1)

- Strategi *web crawler* juga bisa disebut sebagai algoritma aplikasi *web crawler*.
- Strategi *web crawler* memiliki asumsi:
  - Web adalah sebuah *directed graph* yang sangat besar, dengan dokumen sebagai *vertices* dan hyperlinks sebagai *edges*.
  - Analisa *directed graph* lebih lanjut menggunakan algoritma *graph traversal*.
  - Dalam algoritma *graph traversal*, *nodes* atau *vertices* digambarkan sebagai kotak dan *directed edges* digambarkan sebagai anak panah.

Strategi *Web Crawler* (2)

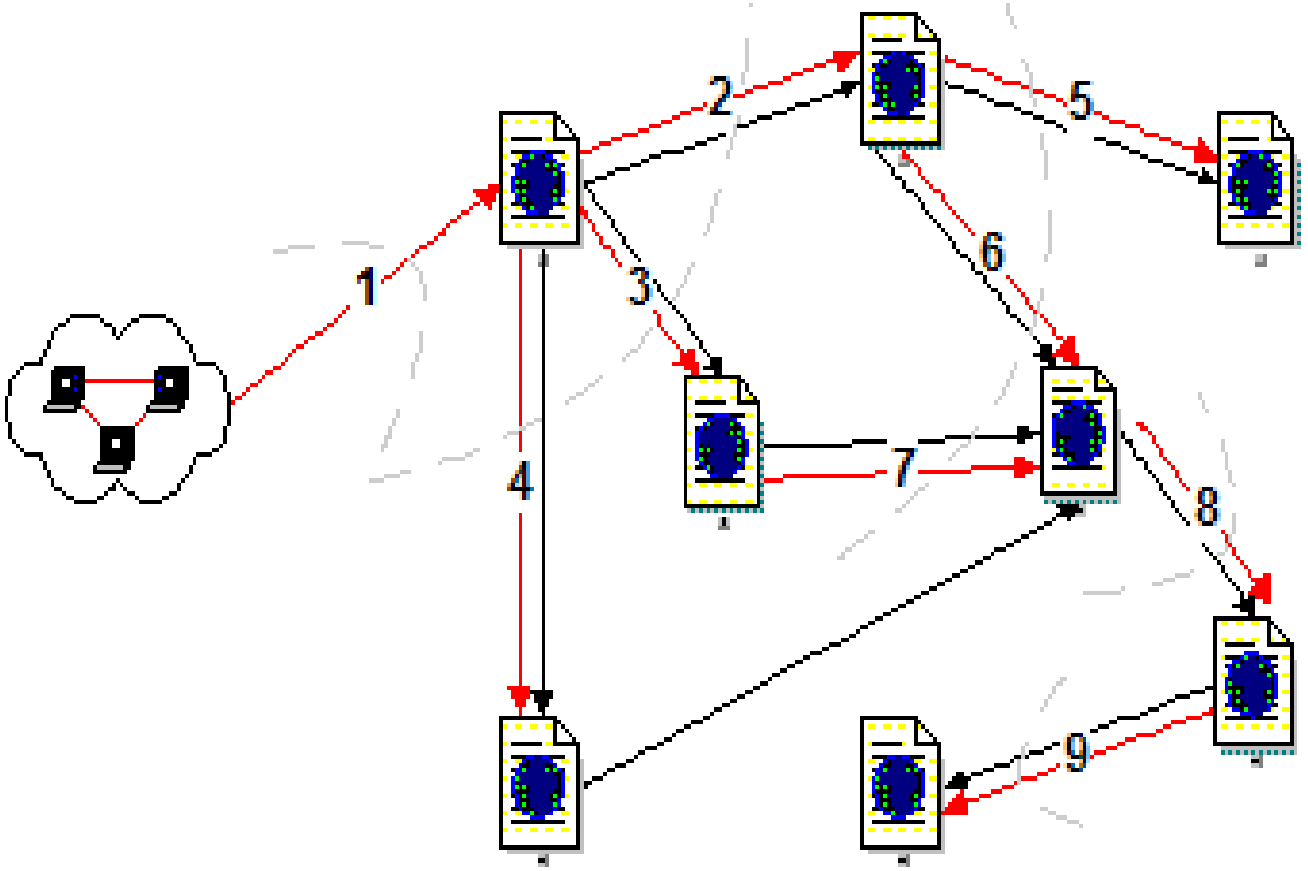
- Strategi *web crawler* yang umum dikenal ada dua:
  - Breadth-First Search Traversal
  - Depth-First Search Traversal

Strategi *Web Crawler* (3)

- Breadth-First Traversal
  - Dengan sembarang *graph* dan sekumpulan *seeds*, *graph* akan bisa dilalui dengan algoritma:
    1. Masukkan daftar URL di dalam *seeds* ke dalam antrean;
    2. Persiapkan daftar *nodes* yang telah dikunjungi (daftar ini awalnya adalah kosong);
    3. Selama antrean masih berisi data URL:
       a. Hilangkan *node* pertama dari antrean;
       b. Tambahkan *node* tersebut ke dalam daftar *node* yang telah dikunjungi.
       c. Untuk setiap *edge* yang dimulai dari *node* tersebut:
          i. Jika *node* pada akhir *edge* telah terdapat di dalam daftar *node* yang telah dikunjungi atau sudah ada di dalam antrean, jangan lakukan apa-apa terhadap *edge* tersebut;
          ii. Sebaliknya, tambahkan *node* pada akhir *edge* ke bagian akhir dari antrean.

Strategi *Web Crawler* (4)

- Ilustrasi strategi breadth-first crawlers:

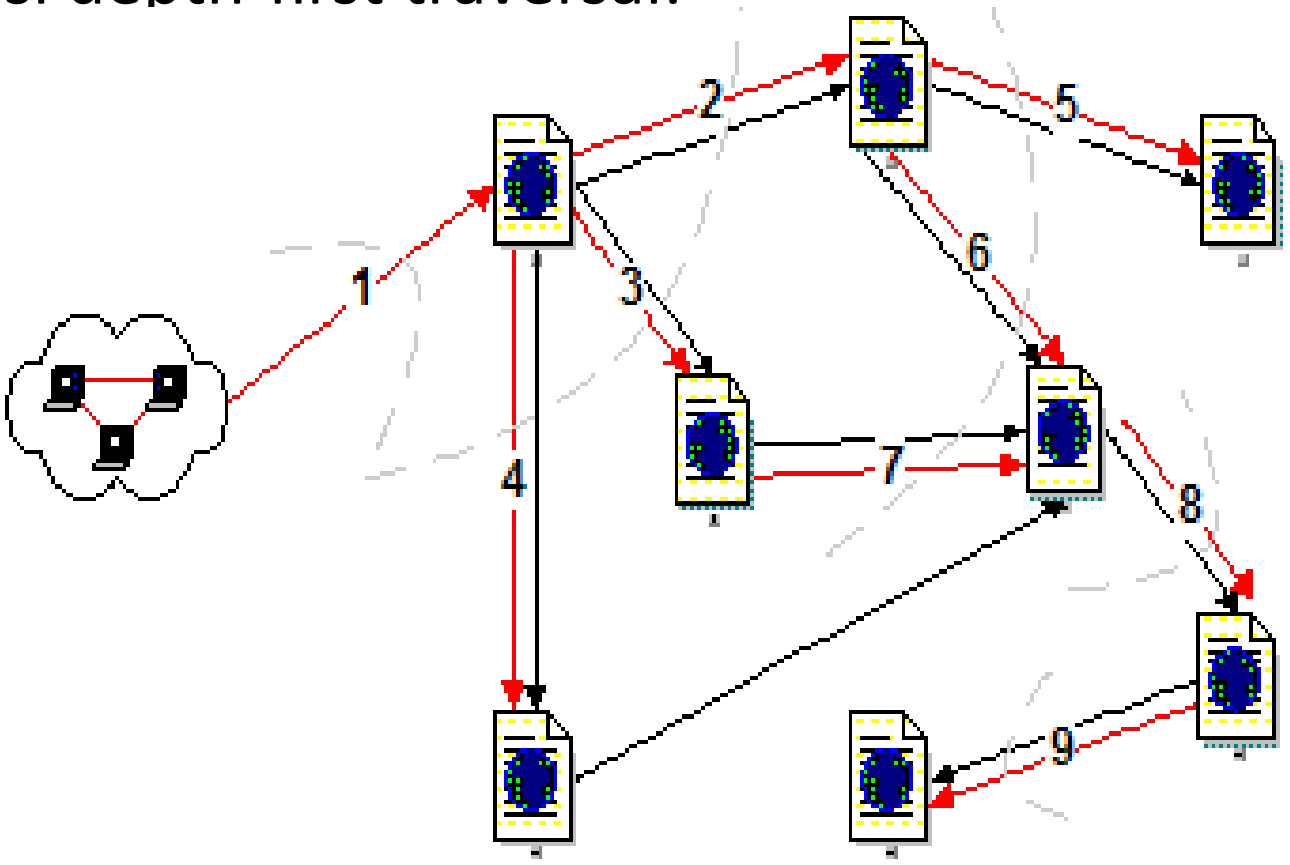


Strategi *Web Crawler* (5)

- Algoritma depth-first search (DFS):
  - Ambil link pertama yang belum dikunjungi dari halaman awal.
  - Kunjungi link tersebut dan ambil link pertama yang belum dikunjungi.
  - Ulangi langkah-langkah di atas hingga tidak ada lagi link yang belum dikunjungi.
  - Kunjungi link yang belum dikunjungi berikutnya yang terdapat di dalam level sebelumnya dan ulangi langkah kedua.

Strategi *Web Crawler* (6)

- Ilustrasi depth-first traversal:

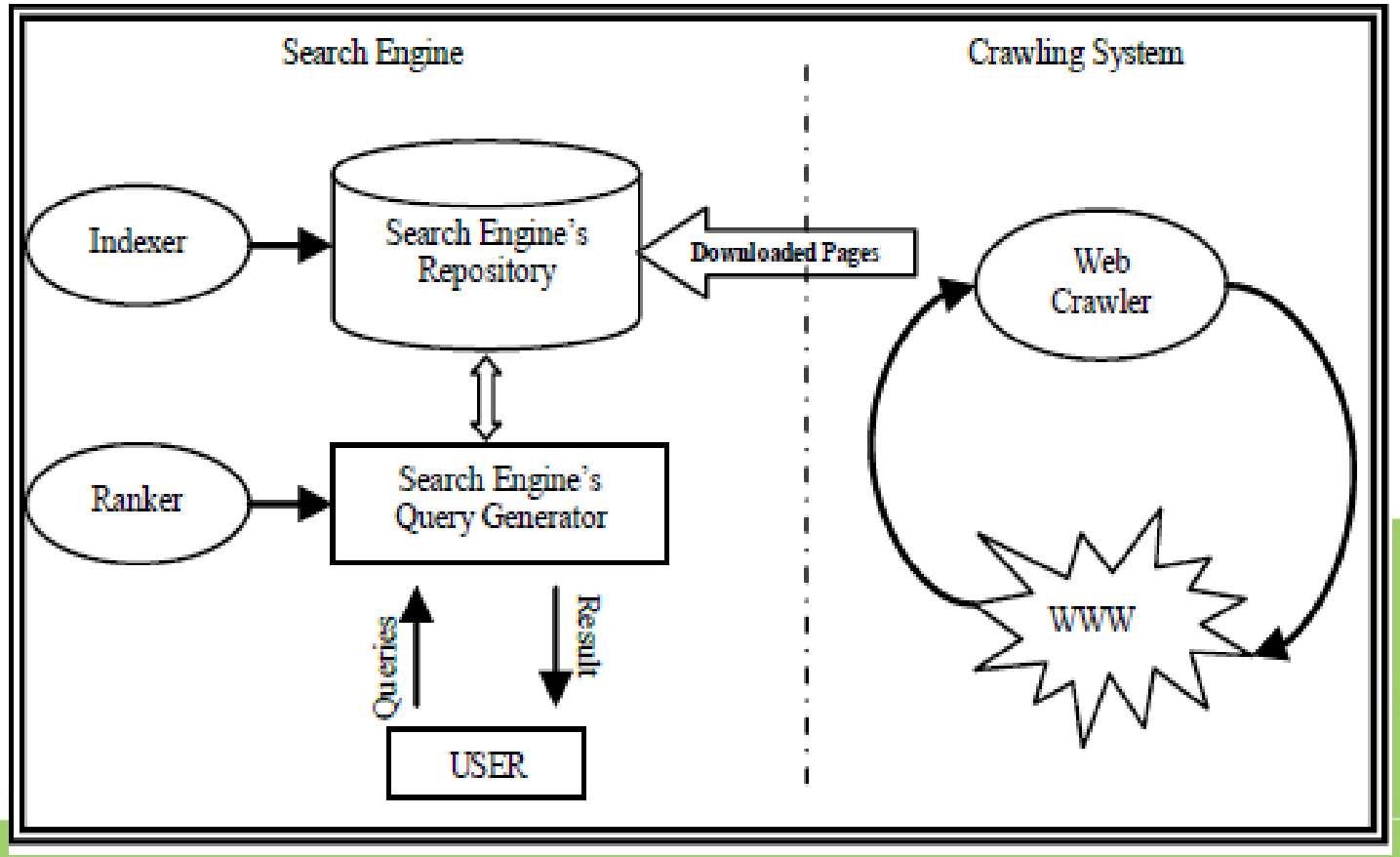


Strategi *Web Crawler* (7)

- Depth-First vs Breadth-First:
  - Depth-First bekerja pada satu *vertices* hingga mencapai sebuah *edge*
    - Tidak cukup bagus jika *edge* berada di *vertices* yang berbeda
    - Tidak lengkap atau optimal
    - Menggunakan ruang yang lebih sedikit di banding Breadth-First
      - Lebih sedikit *node* yang telah dikunjungi yang perlu di awasi
      - Ukuran lebih kecil
  - Breadth-First memeriksa semua alternative dengan lebih teliti
    - Lengkap dan optimal
    - Membutuhkan memory yang cukup besar

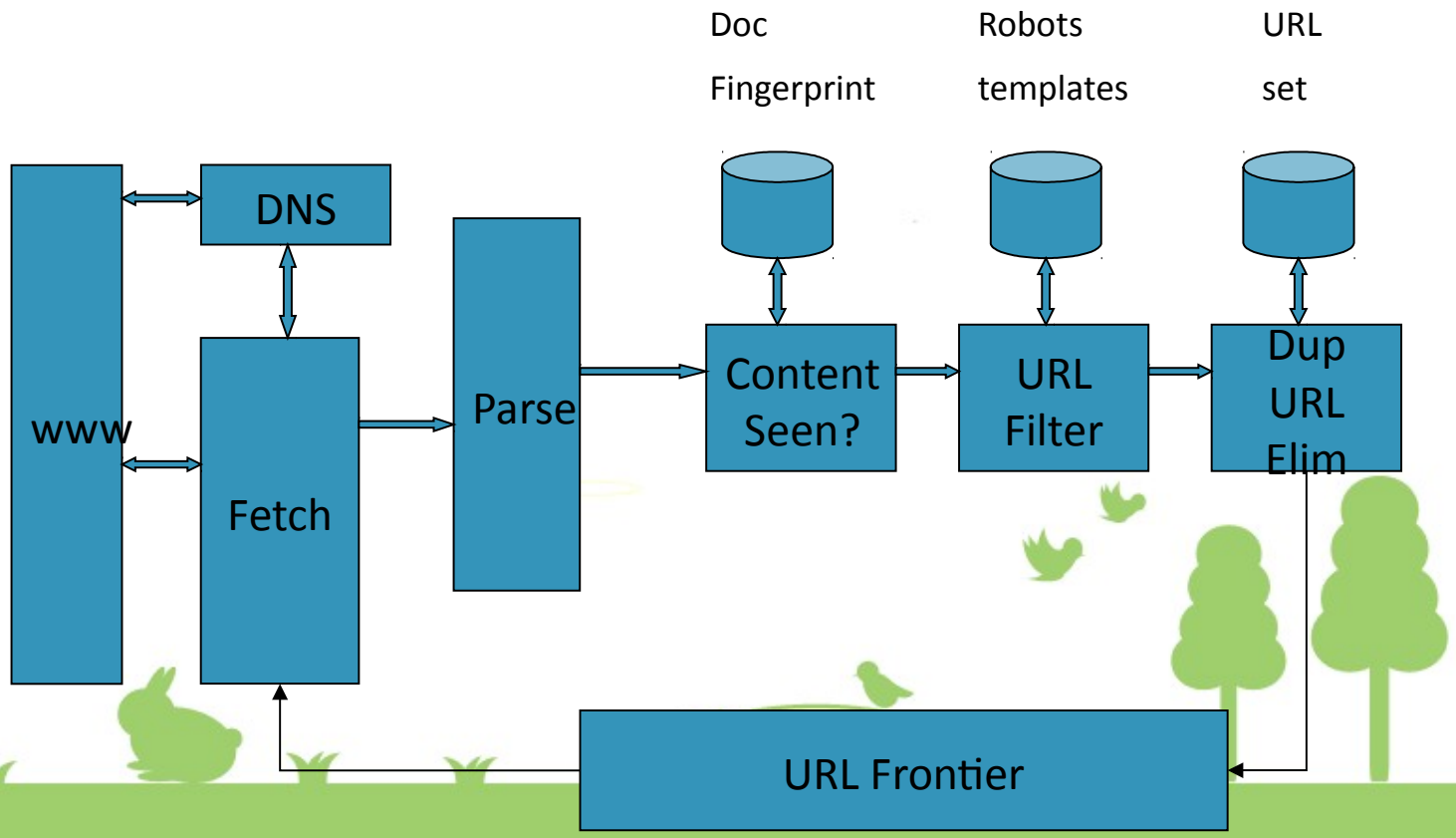
Arsitektur *Web Crawler* (1)

- Ilustrasi arsitektur *search engine*:



Arsitektur *Web Crawler* (2)

- Ilustrasi arsitektur *web crawler*:



Arsitektur *Web Crawler* (3)

- **URL Frontier**: mengandung URL yang belum dikunjungi. Awalnya, sebuah *seeds* disimpan pada URL Frontier dan sebuah *web crawler* memulai proses dari URL yang berada di dalam URL frontier.
- **DNS**: singkatan dari Domain Name Service yang memetakan sebuah domain name dengan alamat IP.
- **Fetch**: umumnya menggunakan protokol HTTP untuk mengakses URL.
- **Parse**: halaman web di urai dan mengambil informasi text dan link.
- **Content Seen?**: menguji apakah sebuah halaman web dengan isi yang sama telah diakses melalui URL yang lain.

Arsitektur *Web Crawler* (4)

- **URL Filter:**
  - Untuk memeriksa apakah URL yang terbaca harus dikeluarkan dari daftar URL Frontier (pengaruh konfigurasi robots.txt).
  - URL harus dinormalisasi.
- **Dup URL Elim**: URL diperiksa untuk menghilangkan duplikasi.

Kebijakan *Web Crawler*

- Kebijakan *web crawler* terdiri dari:
  - Kebijakan *selection* yang menentukan halaman mana yang bisa diakses.
  - Kebijakan *re-visit* yang menentukan kapan sebuah halaman di kunjungi kembali untuk mendeteksi adanya pembaharuan isi.
  - Kebijakan *politeness* untuk menentukan bagaimana menghindari proses yang memberi kelebihan beban bagi website.
  - Kebijakan *parallelization* mengatur cara mengkoordinasi *web crawler* yang terdistribusi.