1. Pendahuluan

1.1 Latar belakang

Pengguna komputer kerap menggunakan fungsi ”Search” yang terdapat pada sistem operasi ataupun pada aplikasi khusus untuk mencari data yang terdapat dalam ruang penyimpanan (*harddisk*). Pencarian data itu sendiri dilakukan karena beberapa kasus yang berbeda, antara lain :

1. Data yang dimaksud ada dalam *harddisk* namun pengguna lupa posisi data tersebut.
2. Data yang dimaksud ada dalam *harddisk* namun pengguna tidak yakin apakah di dalam harddisk terdapat data tersebut.
3. Data yang dimaksud ada dalam *harddisk* dan pengguna mengetahui posisi penyimpanan data tersebut namun *user* malas melakukan penelusuran secara manual ke dalam subfolder-subfolder.
4. Data yang dimaksud tidak ada dalam *harddisk* namun pengguna merasa memiliki data tersebut di dalam *harddisk* komputernya.

Pencarian file seperti di atas kerap memakan waktu yang lama mengingat umumnya fungsi ”Search” pada sistem operasi adalah fungsi pencarian dengan metode *real-time*. Penelusuran untuk mencari data benar-benar dilakukan dari folder *root* yang dipilih sampai ke dalam subfolder-subfolder yang ada. Oleh karena itu, waktu yang dibutuhkan untuk menampilkan hasil pencarian data yang dimaksud adalah tergantung dari posisi file tersebut.

Hasil pencarian data juga tergantung dari *query* atau *keyword* yang diinputkan oleh pengguna. Terkadang, hasil pencarian dinilai pengguna tidak terlalu menyelesaikan masalahnya karena hasil pencarian hanya dapat diranking berdasarkan *last modified* saja dan dicari berdasarkan nama file saja bukan berdasarkan isi file. Padahal, biasanya pengguna lebih menyukai file yang sering dibuka olehnya dan isi datanya relevan dengan *keyword*.

Banyaknya jumlah data pada setiap media penyimpanan juga sangat berpengaruh dalam proses pencarian. Semakin banyak jumlah data, semakin besar pula *cost* yang diperlukan untuk mnenyelesaikan proses pencarian. Bisa dibayangkan lama waktu pencarian jika data yang pengguna cari terdapat pada sebuah data set yang ukuran nya sangat besar, mengingat saat ini banyak jenis media penyimpanan dengan ukuran memori yang luar biasa besarnya (160GB, 320 GB, 500 GB, dll).

Ilmu *Information Retrieval* dapat menjadi solusi untuk permasalahan di atas. Dengan menerapkan konsep *Information Retrieval,* hasil pencarian dapat ditampilkan lebih cepat karena pencarian dilakukan pada indeks. *Query* dicocokkan dengan kumpulan *term* yang terdapat pada indeks yang telah dibangun pada inisialisasi awal. Namun pencarian ke indeks dapat menjadi tidak akurat / relevan jika indeks tidak pernah di-*update*. Oleh karena itu perlu dibangun proses *re-updating index* yang sesuai dengan metode *indexing* yang diterapkan.

Terdapat beberapa metode *indexing* yang kerap digunakan untuk permasalahan tersebut, salah satunya adalah dengan metode pembangunan *tree*. *Tree* biasa dipakai karena kemudahan implementasinya dibandingkan teknik yang lain, misalnya *Hash*. Salah satu metode pengindeksan dengan *tree* adalah *B-Tree* atau *balanced tree*. *Balanced Tree* atau pohon seimbang adalah pohon dimana tidak ada simpul daun yang lebih panjang terhadap daun yang lain. *B-Tree* merupakan metode *indexing* yang sering digunakan dalam mengindeks *free text* dan pada jumlah data yang besar. Sebuah *B-tree* dijaga keseimbangannya dengan membuat semua simpul daun berada pada ketinggian yang sama. Ketinggian ini akan meningkat perlahan saat elemen-elemen ditambahkan pada pohon, tetapi peningkatan ketinggian secara keseluruhan jarang terjadi. Hal disebabkan karena sebuah *B-tree* didesain dapat memiliki cabang dalam jumlah besar dan mengandung sejumlah kunci pada tiap simpulnya sehingga ketinggian pohon relatif kecil. [8] Tujuannya yaitu untuk mendapatkan akses yang cepat pada data.

Dalam Tugas Akhir ini dilakukan penelitian tentang bagaimana proses *indexing* dokumen beserta proses *re-updating index* dengan menggunakan metode *B-Tree*. Dan pada Tugas Akhir ini juga dilakukan penelitian tentang bagaimana performansi *indexing* dan *re-updating index* dengan metode *B-Tree* pada *Desktop Search*.

1.2 Perumusan masalah

Masalah yang dirumuskan dalam Tugas Akhir ini berdasar dari latar belakang yang telah dijelaskan sebelumnya, yaitu:

1. Bagaimana tahapan *preprocessing* terhadap dokumen.
2. Bagaimana proses *indexing* dan *re-updating index* dengan metode *B-Tree*.
3. Bagaimana proses *searching* terhadap *B-Tree*.
4. Bagaimana pengaruh metode *B-Tree* terhadap performansi IRS secara keseluruhan.

Batasan masalah pada Tugas Akhir ini adalah :

1. Implementasi dilakukan dengan simulasi menggunakan *desktop search engine* berbasis *desktop*.
2. Data input untuk tahap *preprocessing* berupa file berekstensi *.txt dan *.docx.
3. Data yang digunakan adalah dokumen berita berbahasa Inggris.
4. Pengambilan data diambil dari beberapa portal berita berbahasa Inggris.
5. *Text preprocessing* diimplementasikan dalam Tugas Akhir ini namun tidak menjadi fokus permasalahan dalam Tugas Akhir ini.
6. *Searching sub-system* diimplementasikan secara sederhana untuk pembuktian metode.

1.3 Tujuan

Tujuan yang ingin dicapai dalam pengerjaan Tugas Akhir ini adalah sebagai berikut :

1. Merancang dan mengimplementasikan simulator proses *indexing* dan *re-updating index* pada *desktop search engine*.

2. Menganalisis performansi waktu pemrosesan *indexing*, *re-updating index* dan *searching* dengan metode *B-Tree* dengan menghitung nilai waktu pemrosesan rata-rata (*Average Execution Time*).

1.4 Metodologi penyelesaian masalah

Metodologi penyelesaian masalah yang akan dilakukan dalam Tugas Akhir ini adalah :

1. Studi literatur.
   Mengumpulkan informasi dan referensi dari buku, artikel maupun paper-paper yang ada di internet serta memahami dan mempelajarinya sehingga dapat digunakan sebagai dasar teori dalam penyusunan Tugas Akhir ini. Literatur yang dicari adalah yang berkaitan dengan *text preprocessing*, algoritma mengenai *indexing* dan *re-updating index*, dan lebih mendalam mengenai algoritma *B-Tree*.
2. Pencarian dan Pengumpulan data.
   Mengumpulkan data berupa dokumen teks berita berbahasa Inggris secara *offline* yang dibutuhkan untuk keperluan proses implementasi dan pengujian algoritma *B-Tree* yang diambil dari web-web penyedia berita berbahasa Inggris.
3. Analisis kebutuhan dan perancangan aplikasi yang akan dibangun.
   Melakukan analisis dan perancangan kebutuhan perangkat lunak dalam melakukan *indexing* untuk teks berita berbahasa Inggris menggunakan metode *B-Tree*. Analisis kebutuhan yang dilakukan antara lain kebutuhan fungsional sistem, spesifikasi perangkat lunak dan perangkat keras yang digunakan, serta pemodelan sistem yang akan dibangun.
4. Implementasi dan Pengujian.
   Mengimplementasikan hasil perancangan dengan menggunakan pendekatan objek, yaitu dengan menggambarkan *use-case diagram*, *class diagram*, dan *sequence diagram*. Pada Tugas Akhir ini juga akan dilakukan pengujian dan pengukuran performansi *indexer* yang digunakan, yaitu *B-Tree* berdasarkan waktu pemrosesan. Proses-proses utama dari metode *B-Tree* akan dipaparkan dengan alat bantu berupa diagram alir (*flowchart*).
5. Analisa hasil pengujian dan pengambilan kesimpulan.
   Menganalisa hasil pengujian dan pengukuran performansi berdasar data yang diuji serta mengambil kesimpulan dari hasil yang telah dianalisa. Pengujian metode dilakukan dengan menggunakan sistem yang sebelumnya telah diimplementasikan pada tahap implementasi.
6. Penyusunan laporan tugas akhir
   Pembuatan laporan tugas akhir yang mendokumentasikan tahap-tahap kegiatan dan hasil dalam Tugas Akhir ini. Dijelaskan pula mengenai langkah-langkah secara detail dalam menganalisis kebutuhan dari awal, perancangan sistem, implementasi, pengujian, serta analisisnya.