Nama **: Farkhan**
NPM **: 20081010060**
Dosen Pengampu **: Faisal Muttaqin, S.Kom, M.T**
Kelas **: B**

PENCARIAN DATA DAN PENGURUTAN DATA

Pencarian data (*searching*) sering disebut juga *table look-up* atau *storage and retrieval information* adalah suatu proses untuk mengumpulkan sejumlah informasi di dalam memori komputer dan kemudian mencari kembali informasi yang diperlukan secepat mungkin.

Dalam kehidupan sehari-hari sebenarnya kita sering melakukan pencarian data. Sebagai contoh, saat kita mencari baju yang hendak kita gunakan dari dalam lemari. Contoh lain ketika kita mencari minum dari dalam kulkas dan masih banyak lagi contohnya.

Algoritma pencarian (*searching algorithm*) adalah algoritma yang menerima sebuah argumen kunci dan dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut. Setelah proses pencarian dilakukan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (*successfull*) atau tidak ditemukan (*unsuccessfull*).

Metode pencarian data dapat dilakukan dengan dua cara, yaitu pencarian internal dan pencarian eksternal. Pada pencarian internal, semua rekaman yang diketahui berada dalam memori komputer sedangkan pada pencarian eksternal, tidak semua rekaman yang diketahui berada dalam pengingat komputer, tetapi ada sejumlah rekaman yang tersimpan dalam penyimpan luar, misalnya pita atau cakram magnetis.

Selain itu, metode pencarian data juga dapat dikelompokkan menjadi pencarian statis dan pencarian dinamis. Pada pencarian statis, banyaknya rekaman yang diketahui dianggap tetap, pada pencarian dinamis, banyaknya rekaman yang diketahui bisa berubah-ubah yang disebabkan oleh penambahan dan penghapusan suatu rekaman.

Ada macam-macam teknik pencarian, yaitu pencarian sekuensial dan pencarian biner. Perbedaan dari dua teknik ini terletak pada keadaan data. Pencarian sekuensial digunakan apabila data dalam keadaan acak atau tidak terurut. Sebaliknya, pencarian biner digunakan pada data yang sudah dalam keadaan urut dan tambahannya yaitu pencarian beruntun dengan sentinel jika pencarian bertujuan untuk menambahkan elemen baru setelah elemen terakhir larik, maka terdapat sebuah varian dari metode pencarian beruntun yang mangkus.

1. Pencarian Linear atau Sekuensial

Pencarian berurutan sering disebut pencarian linear merupakan metode pencarian yang paling sederhana. Pencarian berurutan menggunakan prinsip sebagai berikut : data yang ada dibandingkan satu per satu secara berurutan dengan yang dicari sampai data tersebut ditemukan atau tidak ditemukan.

Pada dasarnya, pencarian ini hanya melakukan pengulangan dari 1 sampai dengan jumlah data. Pada setiap pengulangan, dibandingkan data ke-i dengan yang dicari. Apabila sama, berarti data telah ditemukan. Sebaliknya apabila sampai akhir pengulangan tidak ada data yang sama, berarti data tidak ada. Pada kasus yang paling buruk, untuk n elemen data harus dilakukan pencarian sebanyak n kali pula.

2. Pencaian Biner

Salah satu syarat agar pencarian biner dapat dilakukan adalah data sudah dalam keadaan urut. Dengan kata lain, apabila data belum dalam keadaan urut, pencarian biner tidak dapat dilakukan. Dalam kehidupan sehari-hari, sebenarnya kita juga sering menggunakan pencarian biner. Misalnya saat ingin mencari suatu kata dalam kamus.

Prinsip dari pencarian biner dapat dijelaskan sebagai berikut : mula-mula diambil posisi awal 0 dan posisi akhir = n – 1, kemudian dicari posisi data tengah dengan rumus (posisi awal + posisi akhir) / 2. Kemudian data yang dicari dibandingkan dengan data tengah. Jika lebih kecil, proses dilakukan kembali tetapi posisi akhir dianggap sama dengan posisi tengah –1. Jika lebih besar, proses dilakukan kembali tetapi posisi awal dianggap sama dengan posisi tengah + 1. Demikian seterusnya sampai data tengah sama dengan yang dicari.

3. Pencarian Beruntun Dengan Sentinel

Jika pencarian bertujuan untuk menambahkan elemen baru setelah elemen terakhir larik, maka terdapat sebuah varian dari metode pencarian beruntun yang mangkus. Nilai x yang akan dicari sengaja ditambahkan terlebih dahulu. Data yang ditambahkan setelah elemen terakhir larik ini disebut sentinel.

Sorting atau pengurutan merupakan proses yang sering kali harus dilakukan dalam pengolahan data. *Sort* dalam hal ini diartikan mengurutkan data yang berada dalam suatu tempat penyimpanan, dengan urutan tertentu baik urut menaik (*ascending*) dari nilai terkecil sampai dengan nilai terbesar, atau urut menurun (*descending*) dari nilai terbesar sampai dengan nilai terkecil. Pengurutan data memiliki beberapa metode, antara lain :

1. Bubble Sort atau Exchange Sort

Ide dari *Bubble sort* adalah gelembung air yang akan "mengapung" untuk tabel yang terurut menaik (*ascending*). Elemen bernilai kecil akan "diapungkan" (ke indeks terkecil), artinya diangkat ke "atas" (indeks terkecil) melalui pertukaran. Karena algoritma ini melakukan pengurutan dengan cara membandingkan elemen-elemen data satu sama lain, maka *bubble sort* termasuk ke dalam jenis algoritma *comparison-based sorting*.

Metode gelembung (*bubble sort*) sering juga disebut dengan metode penukaran (*exchange sort*) adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu. Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang paling tidak efisien.

Proses dalam *Bubble sort* dilakukan sebanyak N-1 langkah (*pass*) dengan N adalah ukuran *array*. Pada akhir setiap langkah ke – I , *array* L[0 ... N] akan terdiri atas dua bagian, yaitu bagian yang sudah terurut L[0 ... I] dan bagian yang belum terurut L[I+1 ... N-1]. Setelah langkah terakhir, diperoleh *array* L[0 ... N-1] yang terurut menaik.

2. Selection Sort

Algoritma *selection sort* memilih elemen maksimum atau minimum *array*, lalu menempatkan elemen maksimum atau minimum itu pada awal atau akhir *array* (tergantung pada urutannya *ascending* atau *descending*). Selanjutnya elemen tersebut tidak disertakan pada proses selanjutnya. Karena setiap kali *selection sort* harus membandingkan elemen-elemen data, algoritma ini termasuk dalam *comparison-based sorting*. Terdapat dua pendekatan dalam metode pengurutan dengan Selection Sort :

- Algoritma pengurutan maksimum (*maximum selection sort*), yaitu memilih elemen maksimum sebagai basis pengurutan.

- Algoritma pengurutan minimum (*minimum selection sort*), yaitu memilih elemen minimum sebagai basis pengurutan.

3. Insertion Sort

Insertion sort adalah sebuah algoritma pengurutan yang membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan. Karena algoritma ini bekerja dengan membandingkan elemen-elemen data yang akan diurutkan, algoritma ini termasuk pula dalam *comparison-based sort*

Ide dasar dari algoritma *Insertion Sort* ini adalah mencari tempat yang "tepat" untuk setiap elemen *array*, dengan cara *sequential search*. Proses ini kemudian menyisipkan sebuah elemen *array* yang diproses ke tempatnya yang seharusnya. Proses dilakukan sebanyak N-1 tahapan (dalam *sorting* disebut sebagai "*pass*")

Algoritma *insertion sort* pada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian *array* yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari *array* yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tidak ada lagi elemen yang tersisa pada bagian *array* yang belum diurutkan.

4. Quick Sort

Metode *Quick* sering disebut juga metode partisi (*partition exchange sort*). Metode ini diperkenalkan pertama kali oleh C.A.R. Hoare pada tahun 1962. Untuk mempertinggi efektivitas dari metode ini, digunakan teknik menukarkan dua elemen dengan jarak yang cukup besar.

Dasar strateginya adalah "memecah dan menguasai". *Quick sort* dimulai dengan menscan daftar yang disortir untuk nilai median. Nilai ini, yang disebut tumpuan (pivot), kemudian dipindahkan ke satu sisi pada daftar dan butir-butir yang nilainya lebih besar dari tumpuan di pindahkan ke sisi lain.

Pertama-tama x ← L[q] dipakai untuk membagi larik L[p .. .r] menjadi dua bagian (disebut pemartisian) dengan kondisi semua elemen bagian kiri selalu lebih kecil daripada nilai

elemen pivot dan nilai semua elemen bagian kanan selalu lebih kecil daripada nilai elemen pivot.

Pemartisian dilakukan dengan menggunakan variabel i dan j. Dalam hal ini i berupa petunjuk yang bergerak naik, sedangkan j adalah penunjuk bergerak turun. Variabel j bergeser turun secara terus-menerus sehingga L[j] <= elemen pivot, sedangkan i digeser naik secara terus-menerus sampai memenuhi kondisi L[j] >= elemen pivot. Proses pengulangan dilakukan sampai nilai i >= j. Pada keadaan seperti ini nilai balik subrutin partisi berupa j.

5. Heap Sort

Metode *heap sort* adalah metode dari pengembangan *tree*. *Heap sort* melakukan suatu pengurutan menggunakan suatu struktur data yang di sebut *heap*. *Heap* memiliki kompleksitas yang besar dalam pembuatan kodenya, tetapi *heap sort* mampu mengurutkan data-data yang sangat banyak dengan waktu yang cepat.

Dalam *sorting* biasanya mempunyai sebuah aturan, berikut adalah aturan dari *heap sort* :

a. Untuk mengisikan *heap* dimulai dari level 1 sampai ke level di bawahnya, jika dalam level yang sama semua kunci *heap* belum terisi, maka tidak boleh mengisi di bawahnya.
b. *Heap* dalam kondisi terurut apabila *left child* <> *paarent*.
c. Penambahan kunci diletakkan pada posisi terakhir dari level dan di sebelah kanan *child* yang terakhir, kemudian diurutkan dengan cara *upheap*.
d. Bila menghapus *heap* dengan mengambil kunci pada *parent* di level 1 kemudian digantikan posisi kunci terakhir, selanjutnya di-*sort* kembali metode *downheap*.