*18 / 09 / 2023*

*Day – 15*

**Simple Search & Binary Search**

* **Pengenalan**
* Algoritma adalah sekumpulan instruksi untuk menyelesaikan tugas tertentu.
* Buku ini menjelaskan tentang algoritma yang sengaja dipilih karena beberapa alasan, alasan pertama adalah karena algoritma tersebut cepat, karena algoritma tersebut juga bisa menyelesaikan masalah-masalah yang cukup menarik atau kombinasi dari keduanya.
* Kabar baik nya adalah karena algoritma yang disertakan oleh textbook ini bisa dibilang sebagai algoritma yang cukup populer maka hampir bisa dipastikan implementasi dari algoritma tersebut juga sudah tersedia dalam bahasa pemrograman yang menjadi favorit teman-teman semuanya.
* Tetapi, implementasi algoritma ini terbilang tidak berguna, kalua kita tidak memahami *Trade-Offs* nya.
* Atau kita tidak bisa membandingkan kelebihan dan kekurangan antar algoritma
* **Binary Search**
* Disini kita dihadapkan dengan studi kasus untuk mencari kata dalam suatu kamus, kata tersebut diawali dengan huruf “K”.
* Langkah yang kita bisa lakukan disini adalah kita bisa saja melakukan pencarian dari halaman pertama dari kamus tersebut dimana kita akan terus membalik setiap halaman nya satu persatu mulai halaman paling awal sampai kita menemukan kata yang kita cari.
* Tetapi untuk kasus kita kali ini, disini kita memahami bahwa kamus itukan akan menyimpan sekumpulan kata secara terurut dalam urutan alphabet dan kita tau huruf “K” itu atau kata yang diawali dengan huruf “K” tersebut tidak mungkin akan berada dibagian awal dari kamus.
* Kemungkinan besar kita akan memulai pencarian dibagian tengah kamus, karena kita tahu bahwa huruf “K” itu berada dibagian tengah dari rangkaian huruf alphabet.
* Kasus pencarian kata pada kamus yang kita hadapi ini bisa dikenal dengan *search problem* atau masalah pencarian.
* Salah satu solusi yang bisa gunakan disini adalah dengan menggunakan *Binary Search.*
* *Binary Search* ini sendiri adalah algoritma dimana inputan algoritma ini adalah suatu list dari sejumlah elemen yang terurut, untuk kasus kita pada kali ini elemen terurut nya berupa kata-kata terurut secara alphabet.
* Ketika elemen yang mau kita cari yang berada list yang berurut tadi, maka *binary search* akan mengembalikan posisi dari kata tadi, tetapi kata yang dicari tidak ditemukan, maka *binary search* akan mengembalikan nilai *null*.
* **Contoh Kasus**
* Disini kita dihadapkan pada kasus pencarian lebih sederhana.
* Semisal saja disini saya sedang memikirkan suatu angka antara 1 – 100.
* Dan kalian mencoba untuk menebak angka dalam pikiran saya dengan jumlah tebakan sesuai sedikit mungkin.
* Untuk setiap tebakan yang kalian berikan saya akan memberitahu apakah tebakan tadi terlalu kecil, terlalu tinggi atau tepat.
* **Cara Simpler Search Way**
* Semisal saja kalian akan menebak semacam ini: 1, 2, 3, 4, …..
* Semisal saja kalian menjawab “1”, lalu saya menjawab “terlalu rendah”.
* Berarti nilai “1” nya dieliminasi.
* Dan disini kita masih memiliki 99 kemungkinan angka yang lain.
* Lalu, kalian menjawab lagi “2”, dan disini saya Kembali menjawab “terlalu rendah”.
* Berarti nilai “2” dieliminasi.
* Kita sudah mengeliminasi 2 kemungkinan angka dan masih tersisa 98 kemungkinan angka.
* Dan proses ini akan terus berlangsung secara *sequential* mulai dari angka paling kecil hingga angka yang dicari itu memang ditemukan.
* Algoritma pencarian semacam ini dikenal dengan istilah *Simple Search* (pencarian bodoh karena cara pencarian nya sangat bodoh sekali).
* Untuk setiap tebakan, kalian hanya mengeliminasi satu angka saja.
* Semisal kalau angka yang ada didalam pikiran saya adalah 99, maka kalian membutuhkan 99 kali tebakan bila menggunakan *simple search* ini.
* **Cara Binary Search**
* Ini adalah cara yang paling baik dari permasalahan yang kita hadapi.
* Mulai dari “50”.
* Dimana angka “50” ini adalah nilai tengahnya karena total jangkauan nya mulai dari 1 – 100 maka kita memilih “50”.
* Ketika kita memilih angka “50” dan tebakan dibilang sebagai “terlalu rendah”, maka kita sudah mengeliminasi 50 angka.
* Mulai bisa dipahami yah bagaimana algoritma ini jauh lebih efisien bahkan pada tebakan pertama walaupun tebakan nya salah kita sudah berhasil mengeliminasi setidak nya setengah dari *search space* nya.
* Disini disebutkan, “terlalu rendah” tetapi kau sudah mengeliminasi setengah dari angka yang disebutkan.
* Sekarang kamu tau bahwa angka 1-50 adalah “terlalu rendah”. Tebakan selanjutnya adalah “75”.
* Berikut nya tebakan kedua kita akan pilih nilai tengah dari ruas yang sebelah kanan dimana tebakan berikut nya adalah “75”.
* Hal berikutnya, kita jawab “75” dan respon nya adalah “terlalu tinggi”.
* Walaupun tebakan kedua kita masih salah, tetapi kita sudah berhasil mengeliminasi setengah dari sisa angka yang ada.
* Dengan *binary search,* kita selalu menebak angka diposisi tengah dan kita selalu mengeliminasi setengah dari jumlah angka yang tersisa setiap kali kita melakukan tebakan.
* Angka berikutnya adalah “63” karena “63” ini nilai tengah antara “50” dan “75” dan “terlalu tinggi” ucap yang ditebak.
* Lalu terus kita menebak angka “57”.
* Dan akhir nya ketemu.
* Dan disini angka yang ada didalam pikiran saya adalah “57”.
* **Penjelasan Binary Search Secara Performa**
* Apa yang kita pelajari sebelumnya adalah *binary search.* Kita baru saja mempelajari algoritma pertama kita.
* Dan ini merupakan detail dari berapa banyak angka yang bisa kita eliminasi setiap kali kita melakukan tebakan.
* Karena jumlah tebakan ada 100 maka pada tebakan pertama kita sudah mengeliminasi sejumlah 50 angka, pada tebakan kedua mengeliminasi sejumlah 25 angka.
* Tebakan berikutnya kita sudah mengeliminasi 13 angka.
* Tebakan selanjutnya lagi, kita sudah mengeliminasi sejumlah 7 angka.
* Tebakan berikutnya kita sudah menebak sejumlah 4 angka.
* Tebakan berikutnya kita sudah menebak sejumlah 2 angka.
* Hingga pada tebakan terakhir pasti kita menemukan hasilnya.
* Dan kita bisa lihat disini, maka jumlah tebakan maksimal atau jumlah tebakan terburuk yang bisa kita lakukan adalah 7 tebakan atau 7 tahapan.
* Kita sudah mengeliminasi setengah dari angka yang dicari setiap kali kita menebak dengan menggunakan *binary search.*
* Mau angka berapapun yang kita pikirkan, kalian hanya bisa menebak maksimal 7 tebakan atau 7 kali karena ketika kita mengeliminasi sebuah angka, itu adalah jumlah yang sangat banyak setiap kali kita menebak sebuah angka.
* Disini kita dihadapkan suatu kasus untuk mencari suatu kata dalam suatu kamus dan didalam kamus tersebut terdapat 240,000 kata.
* Untuk kondisi terburuk, berapa banyak tahapan / langkah yang perlu dilakukan ketika menggunakan *simple search* dan ketika kita menggunakan *binary search*?
* Kalau kita menggunakan *simple search* maka jumlah tahapan terbanyak yang mungkin kita lakukan adalah 240,000 tahapan ini dengan kasus ketika kata yang mau kita cari itu ada dibagian paling akhir dari kamus nya.
* Kalo kita membandingkan dengan *binary search* maka perbandingan nya sebagai berikut.
* Pada tebakan pertama kita sudah mengeliminasi sejumlah 120,000 kata.
* Tebakan berikut nya 60,000 kata.
* Tebakan berikutnya lagi adalah 30,000 dan seterusnya.
* Untuk kondisi terburuknya sekalipun *binary search* akan membutuhkan total jumlah tahapan nya sejumlah 18 tahap saja.
* Karena disetiap kali tebakan atau disetiap kali langkah *binary search* sudah mengeliminasi sejumlah setengah dari total jumlah koleksi kata pada setiap kamusnya.
* Kalo kita menggunakan *binary search* total jumlah langkah terburuk nya adalah 18 langkah dan tentu nya ini sangat berbeda jauh bila dibandingkan dengan 240,000 langkah ketika menggunakan *simple search*.
* Secara umum, untuk elemen sejumlah n, kalo kita menggunakan *binary search* maka akan membutuhkan log2 n tahapan dan ini merupakan kondisi terburuk nya, sedangkan kalo kita menggunakan *simple search* itu akan membutuhkan n tahapan untuk kondisi terburuk nya.
* **Logaritma**
* Logaritma itu adalah kebalikan dari perpangkatan / exponensial.
* 102 = 100 <=> log10100 = 2
* 103 = 1000 <=> log101000 = 3
* 23 = 8 <=> log28 = 3
* 24 = 16 <=> log216 = 4
* 25 = 32 <=> log232 = 5
* Dalam buku ini, Ketika kita membahas seputar *Big O Notation* (kita akan bahas setalah materi ini), log selalu mengacu pada log2.
* Ketika kita melakukan pencarian dengan menggunakan *simple search,* maka dalam kondisi terburuk kita harus melakukan pengecekan pada setiap elemen.
* Dengan kata lain, kalo kita memiliki 8 buah elemen maka kita harus melakukan pengecekan sejumlah 8 kali untuk kondisi terburuknya.
* Tetapi kalo kita menggunakan *binary search*, pada kondisi terburuknya kita hanya cukup melakukan pencarian sebanyak log n kali.
* Sebagai contoh, kalo kita memiliki elemen sebanyak 8 elemen dan kalo kita ingin melakukan pencarian dengan *binary search*, maka jumlah pencarian terburuknya adalah log 8 = 3, karena 23 = 8. Jadi, untuk jumlah 8 elemen kalian hanya melakukan pengecekan sebanyak 3 kali untuk kondisi terburuknya.
* Kalo kita memiliki 1,024 elemen, maka jumlah terburuknya adalah log 1,024 = 10, karena 210 == 1,024. Oleh karena itu, untuk jumlah 1,024 elemen kalian hanya melakukan pengecekan sebanyak 10 kali untuk kondisi terburuknya.
* *Binary Search* hanya bisa digunakan ketika list yang kita miliki sudah terurut *(sorted list)*. Sebagai contoh, nama-nama didalam buku telepon yang sudah terurut secara alphabet, jadi kalian bisa menggunakan *binary search* untuk mencari sebuah nama dibuku telepon.