KELOMPOK AKA 12

ANALISIS KOMPLEKSITAS ALGORITMA
PENCARIAN REKOMENDASI FILM DENGAN ID
MENGGUNAKAN ALGORITMA
REKURSIF JUMP SEARCH DAN REKRUSIF SEQUENTIAL SEARCH

Kita ingin mengatasi masalah dalam mencari rekomendasi film di dalam sistem. Setiap film memiliki ID unik, dan setiap bulan pasangan film baru muncul. Dengan pertumbuhan ini, kita ingin mengimplementasikan dua algoritma pencarian rekursif, yaitu Rekursif Jump Search dan Rekursif Sequential Search, untuk memudahkan pengguna menemukan film yang sesuai dengan preferensi mereka

JUMP SEARCH

JUMP SEARCH ADALAH ALGORITMA PENCARIAN YANG DILAKUKAN DENGAN CARA MELOMPATI SEJUMLAH ELEMEN TERTENTU DALAM PROSES PENCARIAN.

REKURSIF

```
int jumpSearchRecursive(int arr[], int x, int n, int step) {
    if (step >= n) return -1;

    int nextStep = step + sqrt(n);
    int endIndex = (nextStep < n) ? nextStep : n - 1;

    if (arr[endIndex] >= x) {
        for (int i = step; i < endIndex; ++i) {
            if (arr[i] == x) return i;
        }
        return -1;
    }

    return jumpSearchRecursive(arr, x, n, nextStep);
}
```

Input size : n
Basic Operation : Perbandingan
Basic Operation on Rekursif :
Basis -> ketika step ≥ n, maka operasi dasar o
Recruens -> ketika step < n, maka ada operasi perbandingan untuk menentukan nextStep, lalu dilakukan perulangan sebanyak nextStep untuk mencari elemen x. Jumlah perbandingan di rekurens adalah 1 (untuk menentukan nextStep) + nextStep (untuk perulangan).

ANALISIS

Kompleksitas Waktu:
Untuk kasus basis, tidak ada operasi perbandingan -> o
Untuk kasus recurens, kompleksitas di ukur dari jumlah perbandingan (1) ditambah kompleksitas waktu untuk JumpSeacrhRecursive(n-1)
Sehingga relasi rekursinya dapat dibentuk menjadi:

$$T(n) = \begin{cases} o, & n \leq o \\ T(n-\sqrt{n})+1+\sqrt{n}, & n > o \end{cases}$$

SOLUSI

T(n) = T(n-sqrt(n)) + 1 + sqrt(n)
T(n-2*sqrt(n)) + 1 + sqrt(n) + 1 + sqrt(n-sqrt(n))
T(n-k*sqrt(n)) + k * (1 + sqrt(n))
.....
T(o-sqrt(o)) + 1 + sqrt(o)
(o-sqrt(o)) + 1 + sqrt(o)
sehingga didapatkan T(n):
$T(n) = n \in o(\sqrt{n})$

SEQUENTIAL SEARCH

SEQUENTIAL SEARCH ADALAH ALGORITMA PENCARIAN DENGAN MEMERIKSA SETIAP ELEMEN SATU PER SATU DARI AWAL HINGGA TARGET DITEMUKAN.

REKURSIF

```
int sequentialSearchRecursive(int x, int Data[], int i) {
    if (i == Max) {
        return -1;
    }

    if (Data[i] == x) {
        return i; // Basis: elemen ditemukan
    } else {
        return sequentialSearchRecursive(x, Data, i + 1);
    }
}
```

Input size : n
Basic Operation : Perbandingan
Basic Operation on Recursive :
Basis -> tidak ada operasi perbandingan untuk i == Max -> o
Recruens -> i < Max

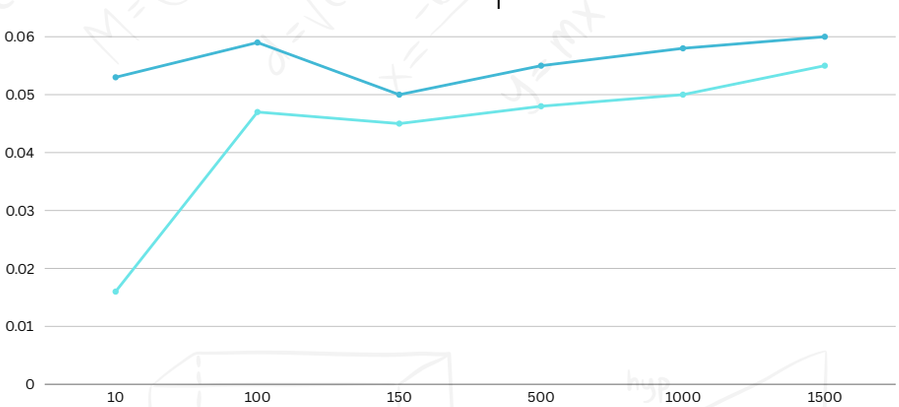
ANALISIS

Kompleksitas Waktu:
Untuk kasus basis, tidak ada operasi perbandingan -> o
Untuk kasus recurens, kompleksitas di ukur dari jumlah perbandingan (1) ditambah kompleksitas waktu untuk JumpSeacrhRecursive(x, Data, i+1)
Sehingga relasi rekursinya dapat dibentuk menjadi:

$$T(n) = \begin{cases} o, & n \leq o \\ T(n-1)+1, & n > o \end{cases}$$

SOLUSI

T(n) = T(n-1)+1
= T(n-2)+1+1
= T(n-3)+1+1+1
.....
T(o)+n
o + n
sehingga didapatkan T(n):
$T(n) = n \in o(n)$



Berdasarkan pengujian, Jump Search cenderung memberikan kinerja yang lebih baik daripada Sequential Search, terutama pada jumlah data yang lebih besar. Sequential Search cenderung menunjukkan pertumbuhan waktu secara linier, sementara Jump Search memiliki kompleksitas waktu yang lebih baik dan efisien, terutama ketika data sudah diurutkan

Referensi :
1. https://www.geeksforgeeks.org/jump-search/
2. Materi Mata Kuliah Analisis Kompleksitas algoritma

Muhammad Hashfi Hadyan(1302220079) Muhammad Ghaziveda Belvanaufal(1302220011) Reihan Ramadhana Anwari(1302220048)