NON-OVERLAPPING SCHEDULE

ANALISIS KOMPLEKSITAS ALGORITMA

Deskripsi

Algoritma nonOverlapping-Schedule merupakan algoritma yang dapat mencari seluruh Aktivitas pada suatu List Aktivitas yang tidak tumpang tindih dengan seluruh Aktivitas pada suatu Jadwal

Input:

- List Aktivitas: dapat berisi Aktivitas yang saling bertumpuk waktunya
- Jadwal: berisi Aktivitas yang tidak boleh bertumpuk waktunya

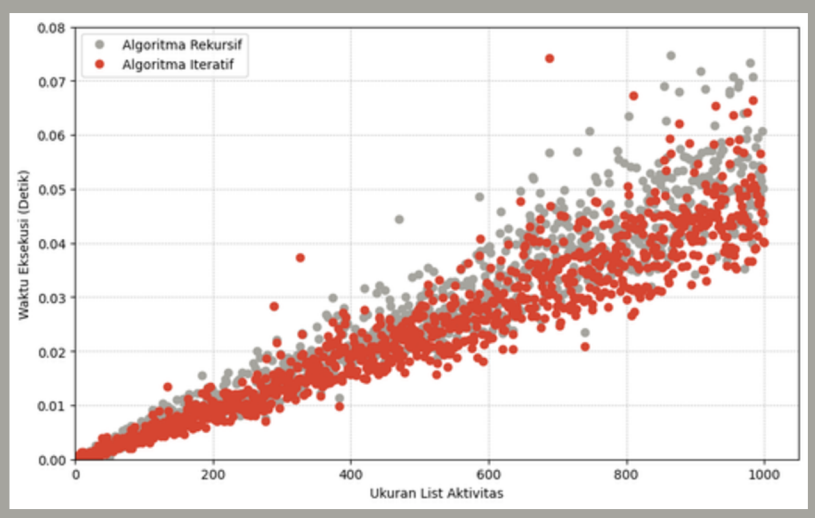
Output:

- List Viable: berisi Aktivitas yang tidak bertumpuk waktunya dengan seluruh Aktivitas di Jadwal

ASIMTOTIK KOMPLEKSITAS

ALGORITMA ITERATIF

```
function nonOverlappingSchedule_Iteratif
(
    List : array[1..maxL] of Aktivitas,
    Jadwal : array[1..maxJ] of Aktivitas,
    nL, nJ : integer
) -> array[1..maxL] of Aktivitas
kamus
    AktL : Aktivitas
    AktL_Start, AktL_End, i, nV : integer
    List_Viable : array[1..maxL] of Aktivitas
algoritma
    List_Viable ← emptyArrayAktivitas
    i ← 1
    nV ← 0
    while i ≤ nL do
        AktL ← List[i]
        AktL_Start ← AktL.WMulai.Jam * 60 + AktL.WMulai.Menit
        AktL_End ← AktL.WSelesai.Jam * 60 + AktL.WSelesai.Menit
        if is_viable_Aktivitas_Iteratif(Jadwal, nJ, AktL, AktL_End, AktL_Start) then
            nV ← nV + 1
            List_Viable[nV] ← AktL
        end if
        i ← i + 1
    end while
    return List_Viable
end function
```



Hasil perhitungan kompleksitas waktu algoritma Iteratif dan algoritma rekursif adalah $\theta(nL \cdot nJ)$ keduanya merupakan kompleksitas kelas kuadratik.

ALGORITMA REKURSIF

```
function nonOverlappingSchedule_Rekursif
(
    List, List_Viable : array[1..maxL] of Aktivitas,
    Jadwal : array[1..maxJ] of Aktivitas,
    i, nV, nL, nJ : integer
) -> array[1..nL] of Aktivitas
kamus
    AktL : Aktivitas
    AktL_Start, AktL_End : integer
algoritma
    if nV == 0 then
        List_Viable ← emptyArrayAktivitas
    end if
    if i ≥ nL then
        return List_Viable
    end if
    AktL ← List[i]
    AktL_Start ← AktL.WMulai.Jam * 60 + AktL.WMulai.Menit
    AktL_End ← AktL.WSelesai.Jam * 60 + AktL.WSelesai.Menit
    if is_viable_Aktivitas_Rekursif(Jadwal, nJ, AktL, AktL_End, AktL_Start, 1) then
        nV ← nV + 1
        List_Viable[nV] ← AktL
    end if
    return nonOverlappingSchedule_Rekursif(List, List_Viable, Jadwal, i + 1, nV, nL,
nJ)
end function
```


oleh:
M. Rifqi Dzaky Azhad
103012330009
IF-47-08