Analisis Kompleksitas Algoritma Studi Kasus "Non-Overlapping Schedules"

Permasalahan

Jadwal adalah struktur data berisi Aktivitas dengan atribut "Hari", "Waktu Mulai", dan "Waktu Selesai", tanpa tumpang tindih antar Aktivitas. List Aktivitas mirip Jadwal tetapi dapat berisi Aktivitas tumpang tindih.
List Viable Aktivitas adalah subset List Aktivitas yang tidak tumpang tindih dengan Jadwal, di mana Aktivitas dianggap tumpang tindih hanya jika terjadi pada hari yang sama dan waktu yang saling beririsan.
Tujuan: Identifikasi Aktivitas dari List Aktivitas yang tidak tumpang tindih dengan Jadwal.

Perhitungan

Rekursif

Best Case:

Terjadi ketika:

- Perbandingan `AktL.Hari == AktJ.Hari` selalu bernilai True di setiap iterasi Aktivitas pada List Aktivitas.
- Perbandingan `not(AktL_End <= AktJ_Start or AktL_Start >= AktJ_End)` di setiap iterasi Aktivitas pada List Aktivitas.

Perhitungan:

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

Sehingga:

$$T_{\text{best}}(nL) = nL = \Theta(nL)$$

Berkelas Linier

Worst Case:

Terjadi ketika:

- Perbandingan `AktL.Hari == AktJ.Hari` bernilai True dan perbandingan `not(AktL_End <= AktJ_Start or AktL_Start >= AktJ_End)` bernilai False
- Perbandingan `AktL.Hari == AktJ.Hari` bernilai False di setiap iterasi Aktivitas pada List Aktivitas.

Perhitungan:

$$T(nL) = \begin{cases} nJ & , nL = 1 \\ T(nL-1)+nJ & , nL > 1 \end{cases}$$
$$T(nL) = T(nL-1)+nJ$$
$$T(nL) = T(nL-(nL-1))+(nL-1)(nJ)$$
$$T(nL) = T(1)+(nL-1)\cdot nJ$$
$$T(nL) = nL\cdot nJ - nJ + nJ$$

Sehingga:

$$T_{\text{worst}}(nL) = nL\cdot nJ \in \Theta(nL\cdot nJ)$$

Berkelas Kuadratik

Average Case:

Terjadi ketika:

- Perbandingan `not(AktL_End <= AktJ_Start or AktL_Start >= AktJ_End)` tidak selalu bernilai True ataupun False pada List Aktivitas

Perhitungan:

$$T(nL) = \frac{\Sigma_{i=1}^{T_{\text{worst}}(n)}(i)}{T_{\text{worst}}(n)} = \frac{\frac{(T_{\text{worst}}(n))(T_{\text{worst}}(n)+1)}{2}}{T_{\text{worst}}(n)} = \frac{T_{\text{worst}}(n)+1}{2} = \frac{nL\cdot nJ+1}{2}$$

Sehingga:

$$T_{\text{Average}}(n) = \frac{nL\cdot nJ+1}{2} \in \Theta(nL\cdot nJ)$$

Berkelas kuadratik

Iteratif

Best Case:

Terjadi ketika:

- Perbandingan `AktL.Hari == AktJ.Hari` dan perbandingan `not(AktL_End <= AktJ_Start or AktL_Start >= AktJ_End)` selalu bernilai True di setiap iterasi Aktivitas pada List Aktivitas.

Perhitungan:

$$T(n) = \Sigma_{i=1}^{nL}(1) = nL - 1 + 1 = nL$$

Sehingga:

$$T_{\text{best}}(n) = nL \in \Theta(nL)$$

Berkelas Linier

Worst Case:

Terjadi ketika:

- Perbandingan `AktL.Hari == AktJ.Hari` bernilai True dan perbandingan `not(AktL_End <= AktJ_Start or AktL_Start >= AktJ_End)` bernilai False
- Perbandingan `AktL.Hari == AktJ.Hari` bernilai False di setiap iterasi Aktivitas pada List Aktivitas.

Perhitungan:

$$T(n) = \Sigma_{i=1}^{nL}\left(\Sigma_{i=1}^{nJ}(1)\right) = \Sigma_{i=1}^{nL}(nJ) = nL\cdot nJ$$

Berkelas kuadratik

$$T_{\text{worst}} = nL\cdot nJ \in \Theta(nL\cdot nJ)$$

Average Case:

Terjadi ketika:

- Perbandingan `not(AktL_End <= AktJ_Start or AktL_Start >= AktJ_End)` tidak selalu bernilai True ataupun False pada List Aktivitas

Perhitungan:

$$T(n) = \frac{\Sigma_{i=1}^{T_{\text{worst}}(n)}(i)}{T_{\text{worst}}(n)} = \frac{\frac{(T_{\text{worst}}(n))(T_{\text{worst}}(n)+1)}{2}}{T_{\text{worst}}(n)} = \frac{T_{\text{worst}}(n)+1}{2} = \frac{nL\cdot nJ+1}{2}$$

Sehingga:

$$T_{\text{Average}}(n) = \frac{nL\cdot nJ+1}{2} \in \Theta(nL\cdot nJ)$$

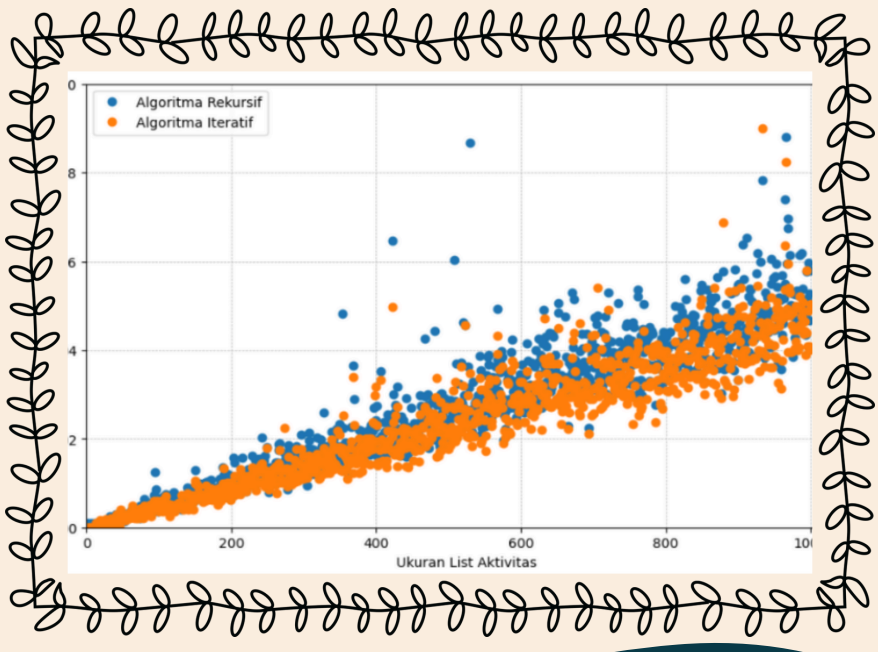
Berkelas kuadratik

Didapatkan kelas asimtotiknya θ(nL . nJ)

Contoh Jadwal

Hari	Nama Aktivitas	Waktu Mulai	Waktu Selesai
Wednesday	STRUKTUR DATA	15:30	18:30
Saturday	STRUKTUR DATA	13:30	15:30
Wednesday	SISTEM BASIS DATA	12:30	15:30
Tuesday	ORGANISASI DAN ARSITEKTUR KOMPUTER	09:30	12:30
Thursday	ANALISIS KOMPLEKSITAS ALGORITMA	09:30	11:30
Friday	TEORI BAHASA DAN AUTOMATA	09:30	11:30
Monday	TEORI PELUANG	07:30	10:30
Friday	STRUKTUR DATA	07:30	09:30

Perbandingan Running Time



Kesimpulan

Hasil perhitungan kompleksitas waktu algoritma iteratif adalah θ(nL . nJ) dan algoritma rekursif adalah θ(nL . nJ), kedua hasilnya sama dan terlihat dari perbandingan running time bahwasannya kedua algoritma memiliki waktu eksekusi yang serupa, akan tetapi semakin besar inputan, semakin terlihat bahwa algoritma iteratif lebih efisien dibandingkan dengan algoritma rekursif.