Catatan Tambahan Materi Analisis Algoritma

Kompetensi Yang diharapkan

1. Memahami bahwa pengukuran berdasarkan wall-time (stopwatch) tidak dapat cukup merepresentasikan kinerja algoritma sebenarnya akibat faktor-faktor hardware (CPU, memory, dlsb), software (compiler, OS, multitasking), implementasi (bahasa pemrograman, library), dlsb.
2. Mampu menganalisis algoritma secara intuitif dengan mengestimasi fungsi waktu eksekusi, dan mendapatkan Notasi Big-Oh dari fungsi waktu sebagai kelas kompleksitas dari algoritma.
3. Berdasarkan notasi Big-Oh dapat membandingkan kelas kompleksitas dari algoritma-algoritma yang terkait.
4. Memahami bahwa pengukuran berdasarkan analisis algoritma lebih independent dari hal-hal tersebut di atas namun tetap masih menyembunyikan beberapa aspek.
   a) Notasi Big-Oh tetrkait paradigma komputasinya, jika algoritma sikuensial maka hanya komparabel dengan algoritma sikuensial juga
   b) Notasi Big-Oh lebih menunjukkan *growth-rate* dari waktu komputasi terkait peningkatan jumlah data, sehingga actual time dari algoritma-algoritma di kelas notasi yang sama tidak berarti waktu eksekusinya sama, tetapi growth-ratenya saja yang sama.
   Contoh: dua algoritma A dan B sama-sama memiliki kompleksitas $O(N^2)$, pengukuran waktu eksekusi dari sejumlah data diamati sebagai dalam tabel berikut:

N	Algoritma A	Algoritma B
1000	0.05s	0.2s
100000	500s	2000s

   Growth-rate kedua algoritma adalah kuadratis, yaitu setiap peningkatan N sebanyak x kali, maka waktu meningkat sebanyak x^2 kali, tetapi waktu eksekusinya amatlah berbeda.

   c) Notasi Big-Oh lebih menunjukkan kecenderungan dalam kondisi data yang berjumlah umum/besar, sehingga untuk data yang kecil dapat terjadi anomali.

5. Memahami bahwa bekerja dengan lebih keras menemukan sejumlah pengamatan (Lemma) dari problem komputasi yang sedang dihadapi dan dengan bantuan alat ukur notasi Big-Oh, mungkin kita dapat memperbaiki solusi algoritma yang lebih cepat (Contoh problem Maximum Subsequence Sum).

Mengestimasi waktu eksekusi secara

Dalam kuliah SDA fungsi waktu diperoleh secara intuitif sebagai berikut:

- operasi di luar loop (for atau while atau do-while) dilakukan secara waktu konstan (kecuali jika terkait dengan harga n)
- operasi di dalam loop dengan iterator yang berubah dengan deret hitung dilakukan secara waktu linear.
- operasi di dalam loop dengan iterator yang berubah secara geometrik dilakukan secara waktu logaritmis
- operasi di dalam pemanggilan rekursif dianalisis sebagai fungsi rekursif dari data.

Guideline dalam menentukan kelas kompleksitas

Untuk menentukan kelas kompleksitas, jika fungsi waktu $T(N)$ dari banyaknya data N terdiri dari beberapa suku, maka suku yang paling signifikan yang diambil serta koefisiennya diabaikan:

- di antara semua N^k , maka ambil k terbesar (dengan koefisien tidak nol!), contoh: $T(N) = 6N^3 + 3N^2 \in O(N^3)$
- faktor log n, maka $1 << \log N << N$, dengan demikian $N^{k-1} << N^{k-1} \log N << N^k$,
  contoh: $T(N) = N^3 + N^2 \log N \in O(N^3)$
- setiap logaritma basis selain 2 menjadi berbasis 2,
  contoh $T(N) = 5 \log_{10} N = 5 / (\log 10) \log N \in O(\log N)$
- setiap eksponensial dari bilangan selain e=2.71... menjadi eksponensial dari bilangan e dan $n^k << e^n$.

Secara umum, dalam membandingkan dua signifikansi dari suku $g(N)$ dan $h(N)$, jadi $g(N) >> h(N)$ hanya jika $\lim_{N\to\infty} h(N)/g(N) = 0$, dan $g(N)$ sama dengan $h(N)$ hanya jika limit tsb adalah konstanta bukan nol.

Jika fungsi waktu $T(N,M)$ bergantung dari beberapa data berbeda misalnya masing-masing sebanyak N dan M, maka signifikansi setiap suku diperiksa dengan memperhatikan kedua faktor tersebut.

- satu suku lebih signifikan dari suku yang lain jika keduanya lebih atau sama signifikansinya,
  contoh: $f(N,M) = N^3M + N^2 \log M \in O(N^3M)$
- dua suku dengan signifikan kedua faktor saling mengalahkan, maka kedua suku tetap diperhitungkan,
  contoh: $f(N,M) = N^3M + N^4 \log M + N^2M \in O(N^3M + N^4 \log M)$