Algorithm Analysis

- Kita perlu memproses jumlah data yang sangat besar.
- Harus diyakinkan bahwa program berhenti dalam batas waktu yang wajar (reasonable)
- Tidak terikat pada programming language atau bahkan metolodologi (mis. Procedural vs OO)

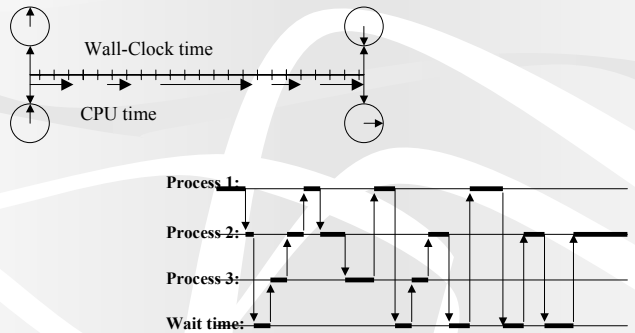
Program Attributes

- Reliable code
- Easy modification
- Faster code
- Ease of use
- Total development time
- Total development cost
- Mean space and time taken

Algoritma

- Suatu set instruksi yang harus diikuti oleh computer untuk memecahkan suatu masalah.
- Dengan algorithm yang benar, perlu diketahui berapa resource yang terpakai spt: time & space yang diperlukan oleh algorithm.

Time Consumption



Algorithm's Functions

- Jumlah waktu yg dibutuhkan hampir selalu tergantung pada jumlah input yang diproses.
- Exact value depends on:
  - speed of host machine
  - machine architecture
  - quality of compilers

Algorithm Analysis

- Langkah perhitungan resource yang diperlukan oleh sebuah algorithm.
- Tujuan sesi ini:
  - bagaimana memperkirakan waktu yang dibutuhkan oleh sebuah algoritma
  - teknik untuk menurunkan running time
  - pendekatan matematis untuk menggambarkan running time

Contoh Algoritma

Mencari nilai terbesar dalam sebuah array yang berisi integer positif

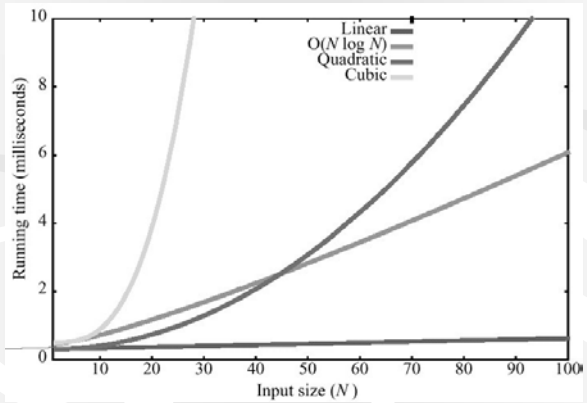
```
int biggest ( int a[] ) {
    int temp = 0, n = size_of_array;
    for (i=0; i<n; i++)
        if (a[i] > temp) temp = a[i];
    return temp;
}
```

running time linear thd jumlah integer dalam array

Algorithm's Functions

- For a given program on a given computer, we can plot the graph that represents the running time function.
- Four common functions:
  - linear
  - logarithmic
  - quadratic
  - cubic

Function Curve



Linear Sample

- Download time N kilobytes
- 2-sec delay for connection setup
- speed of transfer: 1.6 Kb/sec
- T(N) = N/1.6 + 2
- N = 80 K $\rightarrow$ T(N) = 52 sec
- N = 160K $\rightarrow$ T(N) = 102 sec

Big-O Performance Measure

- O(*expr*)
- Big-Oh T(N) is O(F(N)) if there are positive constant c and N_0 such that $T(N) \leq cF(N)$ when $N \geq N_0$.

Rule of Big-O

- reduce all constant factors and terms to 1
  $10n^2 + 9 + 120n + 400 \rightarrow n^2 + 1 + n + 1$
- throw away all constant terms except one
  $n^2 + 1 + n + 1 \rightarrow n^2 + n + 1$
- throw away all terms but the dominant one
  $n^2 + n + 1 \rightarrow n^2$

Big Oh

- Big-Oh notation suatu algoritma dinyatakan oleh suku yang dominan (dominant term).

$10n^3 + n^2 + 40n + 80 \rightarrow \mathbf{O(n^3)}$

$0.5n^2 + \log n + 100000 \rightarrow \mathbf{O(n^2)}$

$15n + 4n\log n + 10^3 \rightarrow \mathbf{O(n\log n)}$

Dominant Term

- Mengapa hanya suku dominant saja yang diperhitungkan?
- Contoh: $n^3 + 350n^2 + n \rightarrow O(n^3)$
- Untuk n = 10000
  - Nilai aktual 1,003,500,010,000
  - Nilai estimasi 1,000,000,000,000
  - Kesalahan dalam estimasi 0.35%, dapat diabaikan

Dominant Term

- Untuk jumlah input (n) yang besar, suku dominan lebih mengindikasikan perilaku dari algoritma.
- Untuk n yang kecil, umumnya berjalan sangat cepat sehingga kita tidak perlu perhatikan.

Dominant Term

- Contoh:
  - Algoritma A = $0.5n^3 + 20n + 10$
  - Algoritma B = $1000n + 250$
- Execution time untuk n = 10
  - Algoritma A = 710
  - Algortima B = 10250
- Execution time untuk n = 100
  - Algoritma A = 502,010
  - Algortima B = 100250

Dominant Term

- Mengapa nilai konstan/koefisien pada setiap suku tidak diperhatikan?
  - Nilai konstan/koefisien tidak memiliki arti pada mesin yang berbeda
- Big-Oh digunakan untuk merepresentasikan laju pertumbuhan (*growth rate*)

Rule of Big-O

- Jika ada lebih dari satu parameter, maka aturan tersebut berlaku untuk setiap parameter.

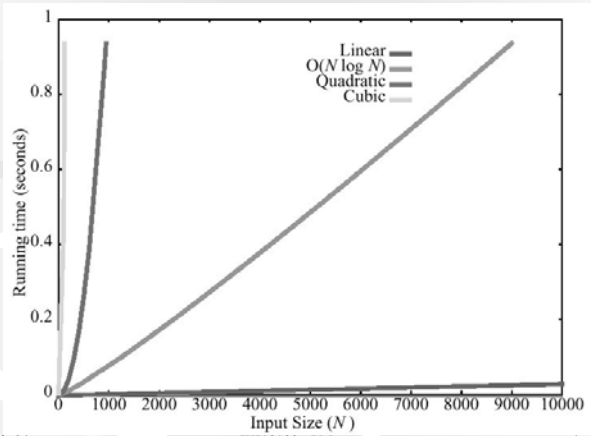
$4n\log(m) + 50n^2 + 500m + 1853$

$O(n \log(m) + n^2 + m)$

$4\ m \log(m) + 50n^2 + 500m + 853$

$O(m \log(m) + n^2)$

Function Curve



Best, Average, Worst Case

- Big-Oh running time dari sequential search: best, average and worst?
- Mencari index dari suatu elemen dalam sebuah array size n.

```
int SeqSearch ( int k, Array a ) {
    int i;
    for (i = 0; i<n; i++)
        if (k == a[ i ]) return i;
    return ( -1 );
}
```

Best and Worst Case

```
int SeqSearch ( int k, Array a ) {
    int i;
    for (i = 0; i<n; i++)
        if (k == a[ i ]) return i;
    return ( -1 );
}
```

- Best case: O(1)
- Worst case: O(n)

Average Case

```
int SeqSearch ( int k, int a[ ] ) {
    int i;
    for (i = 0; i<n; i++)
        if (k == a[ i ]) return i;
    return ( -1 );
}
```

a

b

Total computation time untuk semua kemungkinan posisi k ditemukan dibagi oleh jumlah kemungkinan.

$$\sum_{i=1}^{n} (ai + b) = a \sum_{i=1}^{n} i + \sum_{i=1}^{n} b$$

Average Case

Total computation time untuk semua kemungkinan posisi k ditemukan dibagi oleh jumlah kemungkinan.

$$\text{Total} = \sum_{i=1}^{n} (ai + b) = a \sum_{i=1}^{n} i + \sum_{i=1}^{n} b$$

$$= a\, n(n+1)/2 + bn$$

$$\text{Average} = \text{Total} / n = \frac{a(n+1)}{2}$$

$$= \frac{a}{2}n + \left(\frac{a}{2} + b\right) \rightarrow \mathbf{O(n)}$$

Examples of Algorithm Running Times

Jarak Terdekat dari Dua Titik

- Algortima untuk menemukan sepasang dari N titik dalam suatu bidang (2 dimensi) yang memiliki jarak terpendek dibanding pasangan titik yang lain.
- N (N - 1) / 2 $\longrightarrow$ **$O(n^2)$**

Tiga Titik Segaris

- Algortima untuk menemukan tiga yang segaris dari N titik dalam suatu bidang (2 dimensi).
- N (N - 1) (N - 2) / 6 $\longrightarrow$ **$O(n^3)$**

Tugas Baca

4 versi (cubic, quadratic, linear, logarithmic) dari Maximum Contiguous Subsequence Sum Problem