PENGANTAR

ANALISIS ALGORITMA DAN KOMPLEKSITAS

APAKAH ALGORITMA?

- **Asal kata: ilmuwan/astronomer/matematikawan Persia, Abdullah Muhammad bin Musa al-Khwarizmi**
- **Algoritma adalah:**
  - Prosedur dengan sejumlah langkah untuk mendapatkan hasil yang diperlukan.
  - Sederetan langkah komputasi yang mentransformasi input menjadi output.
- **Algoritma dalam ilmu komputer: prosedur komputasional yang didefinisikan dengan jelas yang mengambil nilai sebagai input dan menghasilkan nilai sebagai output.**

- **Contoh-contoh algoritma:**
  - Langkah-langkah yang rinci untuk membuat kue disertai dengan bahan-bahan yang diperlukan.
  - Menentukan rata-rata dari sejumlah bilangan.
- **Yang bukan algoritma:**
  - Daftar bahan yang diperlukan untuk membuat kue tanpa instruksi bagaimana membuatnya.
  - Contoh kasus menghitung rata-rata.
  - Suatu struktur data (stack atau queue)
    - Operasi-operasi push/pop dan queue/dequeue merupakan algoritma.

MASALAH-MASALAH YANG DISELESAIKAN ALGORITMA

- **Human Genome Project:**
  - Identifikasi 100 ribu gen pada DNA manusia.
  - Contoh penggunaan algoritma: menentukan similarity dari 2 sekuens pasangan basa menggunakan permasalahan longest common subsequence.
- **Internet:**
  - Akses informasi dengan cepat.
  - Contoh penggunaan algoritma: menentukan rute yang akan dipakai untuk data, menemukan laman yang dicari secara cepat.
- **E-commerce:**
  - Barang dan jasa bisa diperjualbelikan secara elektronik
  - Contoh penggunaan algoritma: penggunaan kriptografi kunci publik dan digital signatures.

- **Mengalokasikan sumber daya dengan cara yang paling menguntungkan: pemrograman linier**
  - Perusahaan minyak menempatkan sumurnya.
  - Calon anggota legislatif membeli alat kampanye.
  - Perusahaan penerbangan menugasi krunya.
  - ISP menempatkan *resource* tambahan.

BEBERAPA ISTILAH

- **Contoh definisi formal masalah pengurutan (*sorting*):**
  - Input: barisan n bilangan bulat positif $[a_1, a_2, \ldots, a_n]$.
  - Output: permutasi $[a_1', a_2', \ldots, a_n']$ dari $[a_1, a_2, \ldots, a_n]$ sedemikian hingga $a_1' \leq a_2' \leq \ldots \leq a_n'$.
- **Instance (dari permasalahan):**
  - Terdiri dari input yang diperlukan untuk menghitung penyelesaian masalah.
  - Untuk masalah sorting, contoh instance: [32, 64, 12, 15, 17].
  - Ukuran instance: jumlah item dalam input.
- **Domain definisi masalah:**
  - Adalah himpunan instances yang harus diperhitungkan pada saat masalah didefinisikan.
  - Contoh pada masalah sorting: himpunan bilangan bulat positif.
  - Bisa mempermudah pembuktian kebenaran algoritma.

- **Algoritma yang BENAR:**
  - Untuk setiap instance, dihasilkan output yang benar.
  - Menyelesaikan permasalahan komputasi yang diberikan.
  - Pembuktian: induksi matematika, kontradiksi
- **Bagaimana menentukan bahwa suatu algoritma salah?**

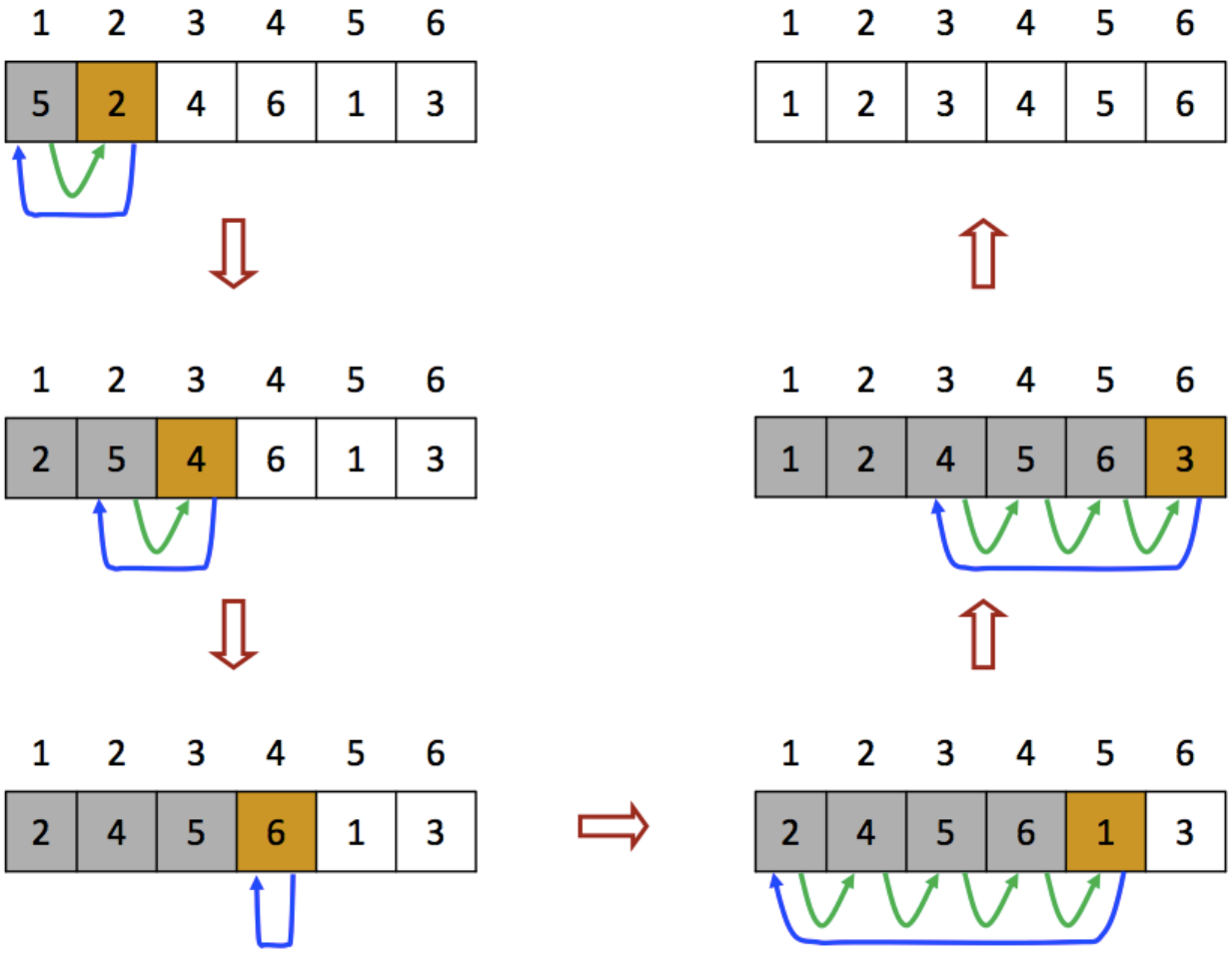
ANALISIS DAN DESAIN

- **Algorithmic adalah cabang ilmu komputer yang terdiri dari desain dan analisis algoritma komputer.**
- **Desain berhubungan dengan:**
  - Gambaran dari algoritma pada level yang abstrak menggunakan pseudo language, dan
  - Pembuktian dari kebenaran algoritma, artinya algoritma dapat menyelesaikan permasalahan pada semua kasus.
- **Analisis berhubungan dengan evaluasi kinerja (analisis kompleksitas)**
- **Perhatian utama pada algoritma:**
  - Correctness (kebenaran)
  - Kompleksitas

CONTOH: INSERTION SORT

- **Definisi masalah pengurutan:**
  - Input: deretan n bilangan bulat positif $[a_1, a_2, \ldots, a_n]$.
  - Output: permutasi $[a_1', a_2', \ldots, a_n']$ dari $[a_1, a_2, \ldots, a_n]$ sedemikian hingga $a_1' \leq a_2' \leq \ldots \leq a_n'$.
  - Diberikan deretan input 31, 41, 59, 26, 41, 58, algoritma sorting mengeluarkan output deretan 26, 31, 41, 41, 58, 59.
- **Insertion sort: algoritma yang efisien untuk mengurutkan sejumlah kecil elemen.**

- **Ilustrasi insertion sort:**



- **Algoritma insertion sort :**
  - INSERTION-SORT(A)

```
1.  for j ← 2 to length[A] do
2.        key ← A[j]
3.        /* Insert A[j] into the sorted sequence A[1... j – 1]*/
4.        i ← j –1
5.        while i > 0 and A[i] > key do
6.              A[i + 1] ← A[i]
7.               i ← i –1
8.        A[i + 1] ← key
```

LOOP INVARIANT UNTUK MEMBUKTIKAN CORRECTNESS

- **Kita dapat menggunakan loop invariant untuk membuktikan correctness.**
  - Initialization: algoritma benar/berlaku sebelum iterasi pertama pada loop.
  - Maintenance: jika algoritma benar sebelum iterasi dari loop, algoritma akan tetap benar sebelum iterasi berikutnya.
  - Termination: pada saat loop berhenti, terdapat properti yang menunjukkan bahwa algoritma tersebut benar.
- **Menggunakan loop invariants mirip seperti induksi matematika.**

CORRECTNESS DARI INSERTION-SORT

- **Loop invariant: pada setiap awal dari iterasi *for* baris 1-8, sub-array *A*[1..*j* − 1] memuat elemen-elemen yang awalnya berada di *A*[1..*j* − 1] tetapi sudah terurut.**
  - Initialization: sebelum iterasi pertama, $j = 2$. $A[1]$ secara otomatis terurut.
  - Maintenance: perhatikan bahwa *while* loop menggerakkan $A[j-1]$, $A[j-2]$, $A[j-3]$, ..., dan seterusnya ke satu posisi sebelah kanan sampai posisi yang benar untuk $A[j]$ ditemukan. Dengan demikian, A[1..j] memuat elmen yang terurut.
  - Termination: *for* loop berakhir ketika j melebihi n, yaitu ketika $j = n + 1$. Maka, $A[1...n]$ terurut.

LATIHAN

- **Perhatikan masalah pencarian berikut :**
  - Input: deretan n angka A = [a_1, a_2, …, a_n] dan nilai v.
  - Output: indeks i sedemikian hingga v = A[i] atau nilai NIL jika v tidak ada di A.

a. Tuliskan pseudo-code untuk linear search (yang menyelesaikan masalah di atas) dengan cara menelusuri array untuk menemukan v.

b. Menggunakan suatu loop invariant, buktikan bahwa algoritma Anda benar.