



Sorting Algorithms

1. Insertion
2. Selection
3. Bubble
4. Shell
5. Quick
6. Merge



Insertion Sort

- Metode penyisipan (Insertion sort) bertujuan untuk menjadikan bagian sisi kiri array terurutkan sampai dengan seluruh array berhasil diurutkan.
- Metode ini mengurutkan bilangan-bilangan yang telah dibaca; dan berikutnya secara berulang akan menyisipkan bilangan-bilangan dalam array yang belum terbaca ke sisi kiri array yang telah terurut.



Insertion Sort

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Bilangan paling kiri (3) bisa dikatakan telah terurut secara relatif thd dirinya sendiri. Thus, we don't need to do anything.



3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Cek, untuk melihat apakah bilangan kedua (10) lebih kecil dari pada yang pertama (3). Jika ya, tukarkan kedua bilangan ini. Namun, kali ini kita tidak perlu melakukan penukaran.



3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Bagian biru/abu-abu (dua bilangan pertama) sekarang dalam keadaan terurut secara relatif.

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Berikutnya, kita perlu menyisipkan bilangan ketiga (4) ke dalam bagian biru/abu-abu sehingga setelah penyisipan tersebut, bagian biru/abu-abu tetap dalam keadaan terurut secara relatif;

CARANYA...



Pertama : Ambil bilangan ketiga (4).

4

3	10		6	8	9	7	2	1	5
---	----	--	---	---	---	---	---	---	---

Kedua : Geser bilangan kedua (10) shg ada ruang untuk disisipi.

4

3		10	6	8	9	7	2	1	5
---	--	----	---	---	---	---	---	---	---

Ketiga : Sisipkan bilangan 4 ke posisi yang tepat

3	4	10	6	8	9	7	2	1	5
---	---	----	---	---	---	---	---	---	---



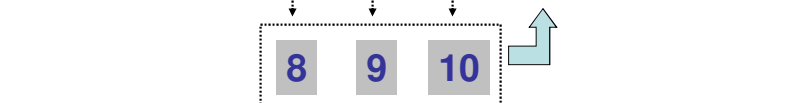
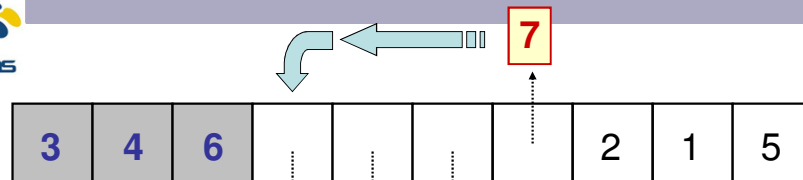
Sekarang, tiga bilangan pertama sudah terurut secara relatif dan kita sisipkan bilangan keempat kepada tiga bilangan pertama tsb. Setelah penyisipan, empat bilangan pertama haruslah dalam keadaan terurut secara relatif.

3	4	6	10	8	9	7	2	1	5
---	---	---	----	---	---	---	---	---	---

Ulangi proses tsb sampai bilangan terakhir disisipkan.

3	4	6	8	10	9	7	2	1	5
---	---	---	---	----	---	---	---	---	---

3	4	6	8	9	10	7	2	1	5
---	---	---	---	---	----	---	---	---	---



3	4	6	7	8	9	10	2	1	5
---	---	---	---	---	---	----	---	---	---

2	3	4	6	7	8	9	10	1	5
---	---	---	---	---	---	---	----	---	---

1	2	3	4	6	7	8	9	10	5
---	---	---	---	---	---	---	---	----	---

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----



Algoritma Metode Penyisipan

1. $i \leftarrow 1$
2. selama ($i < n$) kerjakan baris 3 sampai dgn 9
3. $key \leftarrow A[i]$
4. $j \leftarrow i - 1$
5. selama $j \geq 0$ dan ($A[j] > key$) kerjakan baris 6 dan 7
6. $A[j + 1] \leftarrow A[j]$
7. $j \leftarrow j - 1$
8. $A[j+1] \leftarrow key$
9. $i \leftarrow i + 1$



Pseudo Code

```
InsertionSort(A, n) {
  for i = 1 to n {
    key = A[i]           //Assign elemen array indeks i ke key
    j = i - 1           //Inialisasi j utk perbandingan
    //bandingkan elemen array pd indeks j dgn key
    //if j >= 0 dan elemen indeks j > key
    while (j >= 0) and (A[j] > key) {
      A[j+1] = A[j]     //pindahkan elemen tsb ke 1 posisi berikutnya
      j = j - 1         //go to next lower element
    }
    A[j+1] = key        //Lanjutkan sampai A[j] not > key
                        //assign temp kembali ke array
  }
}
```



Insertion Sort → Analysis

- Running time bukan hanya bergantung pada ukuran array, namun juga pada susunan isinya
- BEST CASE:
 - Array sudah dalam keadaan terurut naik
 - Loop terdalam tidak pernah dieksekusi
 - Jumlah pergeseran : $2(n-1)$
 - Jumlah perbandingan key (C) : $(n-1)$



Insertion Sort → Analysis

- WORST CASE
 - Array dalam urutan kebalikannya
 - Loop terdalam dieksekusi sebanyak $p-1$ kali, untuk $p = 2, 3, \dots, n$
 - Jumlah pergeseran :
$$2(n-1) + (1 + 2 + \dots + n-1) = 2(n-1) + n * (n-1) / 2$$
 - Jumlah perbandingan key :
$$(1 + 2 + \dots + n-1) = n * (n-1) / 2$$