

Analisis Kompleksitas Algoritma Selection Sort dan Insertion Sort

Dosen Pengampu: Dr. Ema Rachmawati, S.T., M.T.



Disusun Oleh Kelompok 4:

Aidil Arifin Nizar (1301204180)
Muhammad Fauzan (1301204251)

SELECTION SORT

1) Definisi dan pseudocode

```
Procedure SelectionSort(in/out arr[0..n-1] of integer, in n int)
  Kamus
    i, j, min_idx, temp : integer
  Algoritma
    for i ← 0 to n - 1 do
      min_idx ← i
      for j ← i + 1 to n do
        if arr[j] < arr[min_idx] then
          min_idx = j
        endif
      endfor
      temp ← arr[min_idx]
      arr[min_idx] ← arr[i]
      arr[i] ← temp
    endfor
  endprocedure
```

Figure 1. Pseudocode Selection Sort Algorithm (Ascending)

Selection sort merupakan sebuah teknik pengurutan dengan cara mencari nilai tertinggi / terendah di dalam array kemudian menempatkan nilai tersebut di tempat semestinya.

2) Hitung Kompleksitas Algoritma

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=i+1}^n = \sum_{i=0}^{n-1} n - (i + 1) + 1 = \sum_{i=0}^{n-1} n - i \quad (1)$$

$$= n \sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i = n \sum_{i=0}^{n-1} 1 - \frac{(n-1)n}{2} \quad (2)$$

$$= n^2 - \frac{(n-1)n}{2} = \frac{n(n+1)}{2} \quad (3)$$

$$= \frac{n^2 + n}{2} = \frac{1}{2}n^2 + n \in O(n^2) \quad (4)$$

Figure 1.1. Kelas Efisiensi Selection Sort Algorithm

3) Ilustrasi proses **pengurutan** dengan Insertion Sort

Array A

Pass	1	3	0	1	2	0	4	1	8	0
Pass = 1	0 ---- i = 0 idx = 2	3	1	1	2	0	4	1	8	0
Pass = 2	0	0 ---- i = 1 idx = 5	1	1	2	3	4	1	8	0
Pass = 3	0	0	0 ---- i = 2 idx = 9	1	2	3	4	1	8	1
Pass = 4	0	0	0	1 ---- i = 3 idx = 3	2	3	4	1	8	1
Pass = 5	0	0	0	1	1 ---- i = 4 idx = 7	3	4	2	8	1
Pass = 6	0	0	0	1	1	1 ----- i = 5 idx = 9	4	2	8	3
Pass = 7	0	0	0	1	1	1	2 ---- i = 6 idx = 7	4	8	3
Pass = 8	0	0	0	1	1	1	2	3 --- i = 7 idx = 9	8	4
Pass = 9	0	0	0	1	1	1	2	3	4 - i = 8 idx = 9	8

INSERTION SORT

1) Definisi dan pseudocode

```
procedure insertionSort(in/out arr[n] of integer)
Kamus
    i : integer
    key : integer
    j : integer
Algoritma |
    for i ← 2 to n do
        key ← arr[i]
        j ← i - 1
        while j >= 0 and key < arr[j] do
            arr[j+1] ← arr[j]
            j ← j - 1
        endwhile
        arr[j + 1] ← key
    endfor
endprocedure
```

Figure 2.1 Pseudocode Insertion Sort Algorithm (Ascending)

Insertion Sort merupakan sebuah teknik pengurutan dengan cara membandingkan dan mengurutkan dua data pertama pada array, kemudian membandingkan data pada array berikutnya apakah sudah berada di tempat semestinya. Algoritma insertion sort seperti proses pengurutan kartu yang berada di tangan kita.

2) Hitung Kompleksitas Algoritma

$$\sum_{i=2}^n \sum_{j=1}^i 1 = \sum_{i=2}^n (i - 1) + 1 = \sum_{i=2}^n \quad (1)$$

$$= \left(\sum_{i=1}^n i \right) - 1 = \left(\frac{n(n+1)}{2} \right) - 1 \quad (2)$$

$$= \frac{n^2 + 2n - 2}{2} = \frac{(n+2)(n-1)}{2} = O(n^2) \quad (3)$$

Figure 2.2 Notasi Asimtotik Insertion Sort Algorithm

3) Ilustrasi Proses Pengurutan dengan Insertion Sort

Array B

Pass	1	3	0	1	2	0	4	2	5	1
p1	1	3	0	1	2	0	4	2	5	1
p2	0	1	3	1	2	0	4	2	5	1
p3	0	1	1	3	2	0	4	2	5	1
p4	0	1	1	2	3	0	4	2	5	1
p5	0	0	1	1	2	3	4	2	5	1
p6	0	0	1	1	2	3	4	2	5	1
p7	0	0	1	1	2	2	3	4	5	1
p8	0	0	1	1	2	2	3	4	5	1
p9	0	0	1	1	1	2	2	3	4	5
p10	0	0	1	1	1	2	2	3	4	5

4) **Tabel dan grafik** yang menggambarkan perubahan *running time*

<u>N</u>	<u>Selection Sort (s)</u>	<u>Insertion Sort (s)</u>
10	0.048	0.045
100	0.068	0.059
1000	0.200	0.198
10000	2.159	2.146
100000	35.427	31.808



Figure 3. Grafik Perbandingan Running Time

5) Kesimpulan

Dari hasil analisis yang sudah kita implementasikan, dapat diketahui bahwa algoritma iterative **Insertion Sort** lebih cepat daripada **Selection Sort** dan kelas efisiensi dari kedua algoritma yang telah kita uji adalah **quadratic** .