TUGAS
ANALISIS ALGORITMA

Disusun Oleh:

Rafa Azka Ulinnuha 140810200033

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2022

Soal Tower of Hanoi

Tabel hasil running program dengan jumlah data yang berbeda

No	Jumlah piring	Jumlah langkah	Durasi hasil running
1	1	$2^1 - 1 = 2$	0.002002s
2	5	$2^5 - 1 = 31$	0.0580093s
3	10	$2^{10} - 1 = 1{,}023$	1.26047s
4	15	$2^{15} - 1 = 32{,}767$	22.9965s
5	20	$2^{20} - 1 = 1{,}048{,}575$	620.59s

Jumlah piring = 1

```
Piring 1 pindah dari A ke C
Durasi hasi running     : 0.002002s
```

Jumlah piring = 5

```
Piring 1 pindah dari B ke A
Piring 2 pindah dari B ke C
Piring 1 pindah dari A ke C
Durasi hasi running     : 0.0580093s
```

Jumlah piring = 10

```
Piring 1 pindah dari A ke B
Piring 2 pindah dari A ke C
Piring 1 pindah dari B ke C
Durasi hasi running     : 1.26047s
```

Jumlah piring = 15

```
Piring 1 pindah dari B ke A
Piring 2 pindah dari B ke C
Piring 1 pindah dari A ke C
Durasi hasi running     : 22.9965s
```

Jumlah piring = 20

```
Piring 1 pindah dari A ke B
Piring 2 pindah dari A ke C
Piring 1 pindah dari B ke C
Durasi hasi running     : 620.59s
```

Soal Sorting

1. Maks Sort

```
PROGRAM MAXIMUM SELECTION SORT
{memilih elemen maksimum kemudian mempertukarkan elemen maksimum
tersebut dengan elemen paling akhir}

DEKLARASI
int arr[n];
int temp;
int U;
int Imaks;

DEFINISI
U ← n - 1;
for (int i = 0; i < U; i++)
      Imaks ← 0;
      for (int j = 1; j <= U; j++)
            if (arr[j] > arr[Imaks]) then
                  Imaks ← j;
            endif
      endfor
      temp ← arr[U];
      arr[U] ← arr[Imaks];
      arr[Imaks] ← temp;
      U ← U-1;
endfor
```

Analisis Kompleksitas Waktu

1	`U ← n - 1;`	1
2	`for (int i = 0; i < U; i++)`	1 + n + n - 1
3	`    Imaks ← 0;`	n – 1
4	`    for (int j = 1; j <= U; j++)`	(n – 1) + (x + 1) + x
5	`        if (arr[j] > arr[Imaks]) then`	3x
6	`            Imaks ← j;`	x
7	`        endif`	
8	`    endfor`	
9	`    temp ← arr[U];`	2(n – 1)
10	`    arr[U] ← arr[Imaks];`	2(n – 1)
11	`    arr[Imaks] ← temp;`	2(n – 1)
12	`    U ← U-1;`	n – 1
13	`endfor`	

x $= 1 + 2 + 3 + \ldots + (n - 1)$

$= \frac{n-1\,(n-1+1)}{2}$

$= \frac{n\,(n-1)}{2}$

T(n) $= 1 + 1 + n + n - 1 + n - 1 + (n - 1) + (x + 1) + x + 3x + x + 2(n - 1) + 2(n - 1) + 2(n - 1) + n - 7$

$= 6x + 11n - 7$

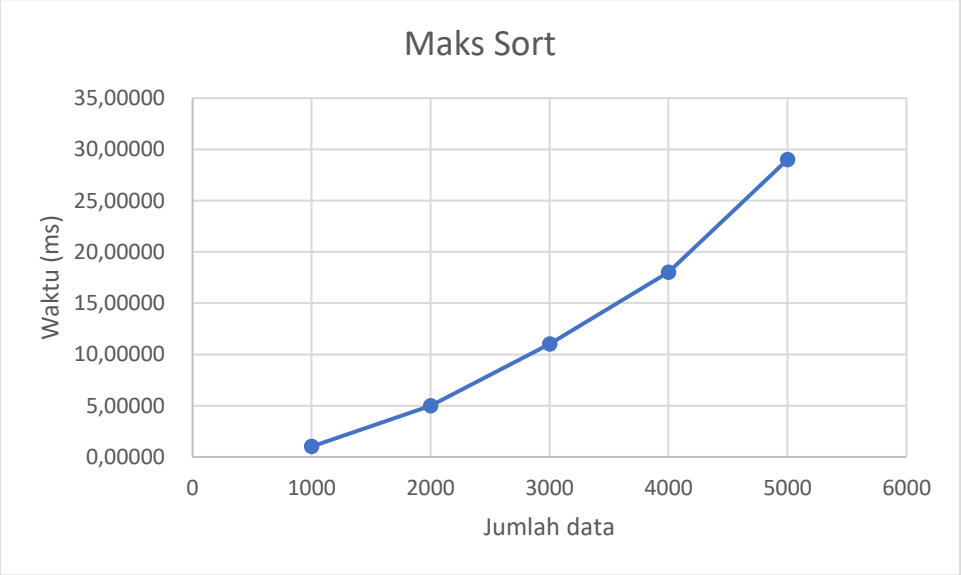
$= 6(\frac{n\,(n-1)}{2}) + 11n - 7$
$= 3n^2 - 3n + 11n - 7$
$= 3n^2 + 8n - 5$
$= O(n^2)$

Maka, waktu kompleksitas Maks Sort adalah $O(n^2)$.

Tabel waktu eksekusi algoritma Maks Sort dapat dilihat pada tabel dibawah ini:

Jumlah data	Waktu (ms)
1000	1,01400
2000	4,98950
3000	11,01580
4000	18,01120
5000	29,00340

Kompleksitas algoritma Maks Sort dapat digambarkan seperti grafik dibawah ini:



2. Bubble Sort

```
PROGRAM BUBBLE SORT
{ mengulang proses pembandingan antara tiap-tiap elemen array dan
menukarnya apabila urutannya salah}

DEKLARASI
int arr[n];
int temp;

DEFINISI
for (i = 0; i < n; i++)
      for (j = i+1; j < n; j++)
            if (arr[j] < arr[i]) then
                  temp ← arr[i];
```

```
                arr[i] ← arr[j];
                arr[j] ← temp;
            endif
        endfor
endfor
```

Analisis Kompleksitas Waktu

1	`for (i = 0; i < n; i++)`	1 + n + n - 1
2	`    for (j = 1; j < n; j++)`	(n – 1) + (x + 1) + x
3	`        if (arr[j] < arr[i]) then`	3x
4	`            temp ← arr[i];`	2x
5	`            arr[i] ← arr[j];`	2x
6	`            arr[j] ← temp;`	2x
7	`        endif`	
8	`    endfor`	
9	`endfor`	

x $= 1 + 2 + 3 + \ldots + (n - 1)$

$= \frac{n - 1\,(n - 1 + 1)}{2}$

$= \frac{n\,(n - 1)}{2}$

T(n) $= 1 + n + n - 1 + (n - 1) + (x + 1) + x + 3x + 2x + 2x + 2x$

$= 10x + 3n - 2$

$= 10(\frac{n\,(n - 1)}{2}) + 3n - 2$

$= 5n^2 - 5n + 3n - 2$

$= 5n^2 - 2n - 2$

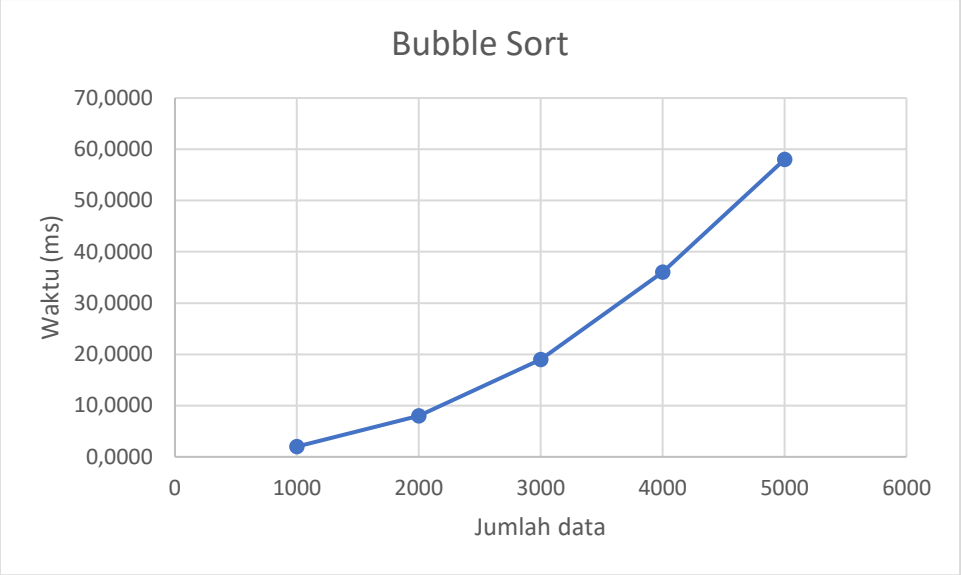
$= O(n^2)$

Maka, waktu kompleksitas Bubble Sort adalah $O(n^2)$.

Tabel waktu eksekusi algoritma Bubble Sort, yaitu sebagai berikut:

Jumlah data	Waktu (ms)
1000	2,0031
2000	8,0140
3000	19,0100
4000	36,0108
5000	58,0101

Kompleksitas algoritma Bubble Sort dapat di gambarkan seperti grafik dibawah ini:



3. Insertion Sort

```
PROGRAM INSERTION SORT
{ membandingkan dan mengurutkan dua data pertama pada array, kemudian
membandingkan data para array berikutnya apakah sudah berada di
tempat semestinya}

DEKLARASI
int arr[n];
int key;

DEFINISI
for (i = 1; i < n; i++)
      key ← arr[i];
      j ← i - 1;
      while (j >= 0 && arr[j] > key)
            arr[j + 1] ← arr[j];
            j ← j - 1;
      endwhile
      arr[j + 1] ← key;
endfor
```

Analisis Kompleksitas Waktu

1	`for (i = 0; i < n; i++)`	1 + n + n - 1
2	`key ← arr[i];`	n – 1
3	`j ← i - 1;`	n – 1
4	`while (j >= 0 && arr[j] > key)`	(x + 1) + 2x
5	`arr[j + 1] ← arr[j];`	3x
6	`j ← j - 1;`	x
7	`endwhile`	
8	`arr[j + 1] ← key;`	2(n – 1)
9	`endfor`	

x = 1 + 2 + 3 + … + (n – 1)

$= \frac{n-1\,(n-1+1)}{2}$

$= \frac{n\,(n-1)}{2}$

T(n) $= 1 + n + n - 1 + (n - 1) + (n - 1) + (x + 1) + 2x + 3x + x + 2(n - 1)$

$= 7x + 6n - 3$

$= 7(\frac{n\,(n-1)}{2}) + 6n - 3$

$= 3{,}5n^2 - 3{,}5n + 6n - 3$

$= 3{,}5n^2 + 2{,}5n - 3$

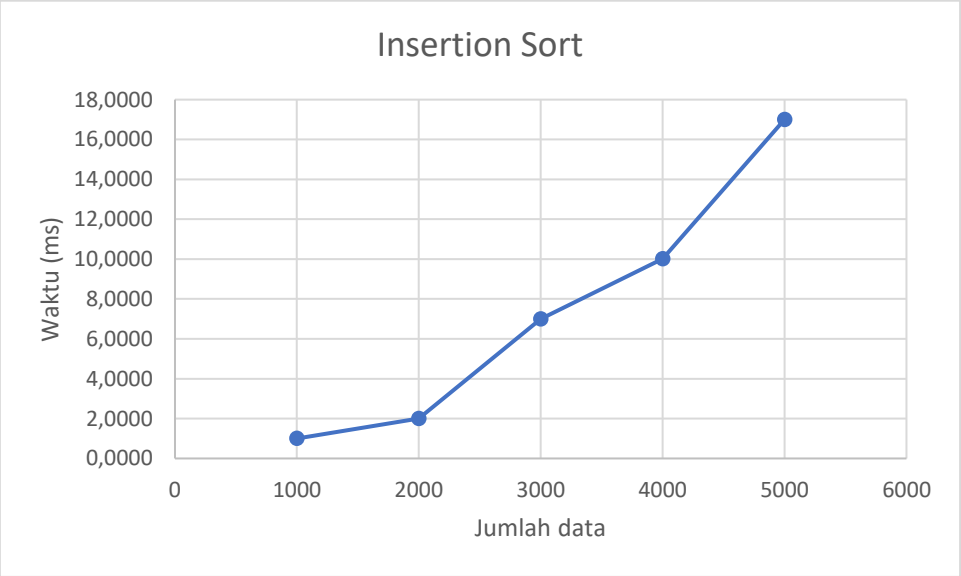
$= O(n^2)$

Maka, waktu kompleksitas Insertion Sort adalah $O(n^2)$.

Tabel waktu eksekusi algoritma Insertion Sort, yaitu sebagai berikut:

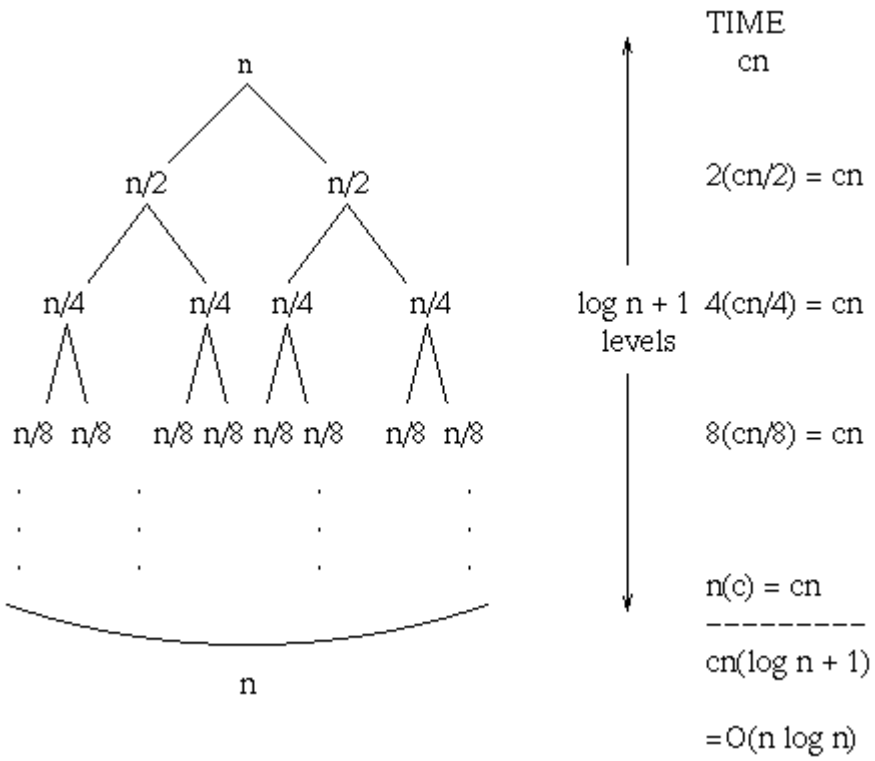
Jumlah data	Waktu (ms)
1000	1,0004
2000	2,0009
3000	7,0022
4000	10,0089
5000	17,0025

Kompleksitas algoritma Insertion Sort dapat di gambarkan seperti grafik dibawah ini:



4. Quick Sort

Analisis Kompleksitas Waktu

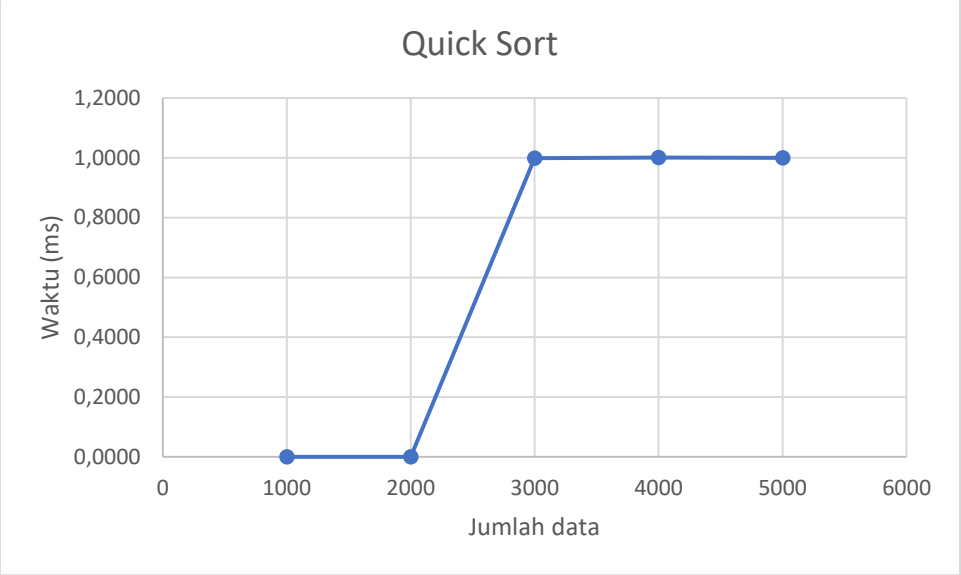


Maka, waktu kompleksitas Quick Sort adalah O(n log n).

Tabel waktu eksekusi algoritma Quick Sort, yaitu sebagai berikut:

Jumlah data	Waktu (ms)
1000	0,0000
2000	0,0000
3000	0,9985
4000	1,0009
5000	0,9998

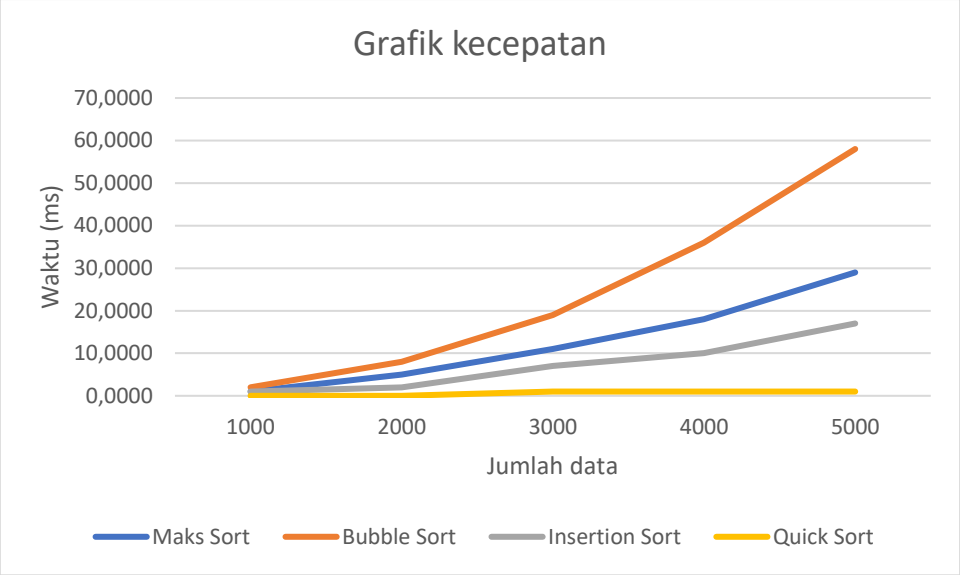
Kompleksitas algoritma Quick Sort dapat di gambarkan seperti grafik dibawah ini:



Tabel waktu eksekusi program dengan jumlah data yang berbeda

No	Jumlah Data	Maks Sort	Bubble Sort	Insertion Sort	Quick Sort
		Waktu Eksekusi Program (s)			
1	1000	0.0010114	0.0020031	0.0010004	0
2	2000	0.0049895	0.008014	0.0020009	0
3	3000	0.0110158	0.01901	0.0070022	0.0009985
4	4000	0.0180112	0.0360108	0.0100089	0.0010009
5	5000	0.0290034	0.0580101	0.0170025	0.0009998

No	Jumlah Data	Maks Sort	Bubble Sort	Insertion Sort	Quick Sort
		Waktu Eksekusi Program (ms)			
1	1000	1,0140	2,0031	1,0004	0,0000
2	2000	4,9895	8,0140	2,0009	0,0000
3	3000	11,0158	19,0100	7,0022	0,9985
4	4000	18,0112	36,0108	10,0089	1,0009
5	5000	29,0034	58,0101	17,0025	0,9998



Dari grafik diatas, terlihat jelas bahwa waktu eksekusi setiap algoritma sorting dengan menggunakan data yang sama berbeda-beda. Hal ini disebabkan oleh kompleksitas waktu yang dimiliki oleh setiap algoritma sorting. Tingkat kompleksitas suatu algoritma memiliki urutan sebagai berikut:

$$O(1)<O(\log n)<O(n)<O(n \log n)<O(n^2)<\ldots<O(2^n)<O(n!)$$

Jika kompleksitasnya semakin kecil, maka waktu runningnya akan semakin cepat. Algoritma quick sort memiliki kompleksitas O(n) = (n log n), sedangkan bubble sort memiliki kompleksitas O(n) = (n^2). Oleh karena itu, dapat dilihat bahwa quick sort menjadi algoritma tercepat dan bubble sort menjadi algoritma terlambat dari keempat algoritma sorting.

Output Program

```
Maks Sort
Durasi hasil running    : 0.0010114s
Durasi hasil running    : 0.0049895s
Durasi hasil running    : 0.0110158s
Durasi hasil running    : 0.0180112s
Durasi hasil running    : 0.0290034s

Bubble Sort
Durasi hasil running    : 0.0020031s
Durasi hasil running    : 0.008014s
Durasi hasil running    : 0.01901s
Durasi hasil running    : 0.0360108s
Durasi hasil running    : 0.0580101s
```

```
Insertion Sort
Durasi hasil running    : 0.0010004s
Durasi hasil running    : 0.0020009s
Durasi hasil running    : 0.0070022s
Durasi hasil running    : 0.0100089s
Durasi hasil running    : 0.0170025s

Quick Sort
Durasi hasil running    : 0s
Durasi hasil running    : 0s
Durasi hasil running    : 0.0009985s
Durasi hasil running    : 0.0010009s
Durasi hasil running    : 0.0009998s
```