

# **LAPORAN PRAKTIKUM 4**

## **ANALISIS ALGORITMA**



**Dibuat oleh:**  
AHMAD IRFAN FADHOLI  
140810180034

**UNIVERSITAS PADJADJARAN**  
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**  
**2020**

### Studi Kasus 1: MERGE SORT

Setelah Anda mengetahui Algoritma Merge-Sort mengadopsi paradigma divide & conquer, lakukan Hal berikut:

1. Buat program Merge-Sort dengan bahasa C++
2. Kompleksitas waktu algoritma merge sort adalah  $O(n \lg n)$ . Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

```
1. #include <iostream>
2. #include <chrono>
3. #include <unistd.h>
4. using namespace std;
5. void merge(int *,int, int , int );
6. void merge_sort(int *arr, int low, int high){
7.
8.     int mid;
9.     if (low < high){
10.
11.         mid=(low+high)/2;
12.         merge_sort(arr,low,mid);
13.         merge_sort(arr,mid+1,high);
14.         merge(arr,low,high,mid);
15.     }
16. }
17.
18. void merge(int *arr, int low, int high, int mid){
19.     int i, j, k, c[50];
20.     i = low;
21.     k = low;
22.     j = mid + 1;
23.     while (i <= mid && j <= high) {
24.         if (arr[i] < arr[j]) {
25.             c[k] = arr[i];
26.             k++;
27.             i++;
28.         }
29.         else {
30.             c[k] = arr[j];
31.             k++;
32.             j++;
33.         }
34.     }
35.     while (i <= mid) {
36.         c[k] = arr[i];
37.         k++;
38.         i++;
39.     }
40.     while (j <= high) {
41.         c[k] = arr[j];
42.         k++;
43.         j++;
44.     }
45.     for (i = low; i < k; i++) {
46.         arr[i] = c[i];
47.     }
```

```

48. }
49.
50.
51. int main(){
52.     int arr[20], num;
53.     cout<<"Masukkan banyak data : ";
54.     cin>>num;
55.     cout<<"Masukkan Data : ";
56.     for (int i = 0; i < num; i++) {
57.         cin>>arr[i];
58.     }
59.     auto start = std::chrono::steady_clock::now();
60.     merge_sort(arr, 0, num-1);
61.     auto end = std::chrono::steady_clock::now();
62.     auto elapsed =
63.         std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);
64.
65.     cout<<"Data setelah di sorting\n";
66.     for (int i = 0; i < num; i++){
67.         cout<<arr[i]<<" ";
68.     }
69.     cout << "\nElapsed time in nanoseconds : " << elapsed.count()<< " ns" << endl;
70.
71. }

```

Kompleksitas waktu :

```

Masukkan banyak data : 10
Masukkan Data : 10 9 8 7 6 5 4 3 2 1
Data setelah di sorting
1 2 3 4 5 6 7 8 9 10
Elapsed time in nanoseconds : 700 ns

```

### Studi Kasus 2: SELECTION SORT

Selection sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma selection sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma selection sort
- Tentukan  $T(n)$  dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi  $T(n)$  dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++

## 2. Selection sort

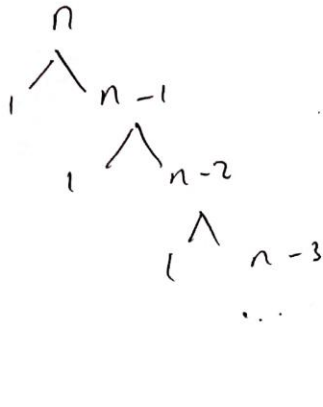
Sub problem = 1

Masalah kecil subproblem =  $n-1$

Waktu pembagian =  $n$

-11- penggalangan = n

$$T(n) = 1 + T(n-1) + O(n)$$



Worst case

Worst Case  
 $T(n) = Cn + Cn - C + (n-2)C + \dots + 2C + Cn$

$$= C \left( \frac{(n-1)(n-2)}{2} \right) + Cn$$

$$= C \left( \frac{n^2 - 3n + 2}{2} \right) + Cn$$

$$= c\left(\frac{n^3}{2}\right) - \frac{3n}{2} + 1 + cn$$

next case  $= O(n^2)^2$

Best case

$$T(n) = Cn + Cn - c + Cn - 2c + \dots + 2c + Cn$$

$$= C\left(\frac{n-1}{2}, \frac{n-2}{2}\right) + C_n$$

$$= C\left(\frac{n^2 - 3n + 2}{2}\right) + (n$$

$$= C \left( \frac{cn^2 - 3n + 2}{2} \right) + Cn$$

$$= C\left(\frac{n^2}{2}\right) - \frac{3n}{2} + 1 + 6n$$

$$= -2(n^2)^2$$

Aug Case:  $T(n) = n^2 + n^2 = n^2$

$$= \overline{(H)}^2 (n^4)$$

```

1. #include<iostream>
2. #include <chrono>
3. #include <unistd.h>
4. using namespace std;
5. void swap(int &a, int &b) {
6.     int temp;
7.     temp = a;
8.     a = b;
9.     b = temp;
10. }
11. void printData(int *array, int length) {
12.     for(int i = 0; i<length; i++)
13.         cout << array[i] << " ";
14.     cout << endl;
15. }
16. void selectionSort(int *array, int length) {
17.     int i, j, imin;
18.     for(i = 0; i<length-1; i++) {
19.         imin = i;
20.         for(j = i+1; j<length; j++)
21.             if(array[j] < array[imin])
22.                 imin = j;
23.         swap(array[i], array[imin]);
24.     }
25. }
26. int main() {
27.     int n, arr[30];
28.     cout << "Masukkan banyak data : ";
29.     cin >> n;
30.     cout << "Masukkan Data : ";
31.     for(int i = 0; i<n; i++) {
32.         cin >> arr[i];
33.     }
34.     auto start = std::chrono::steady_clock::now();
35.     selectionSort(arr, n);
36.     auto end = std::chrono::steady_clock::now();
37.     auto elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);
38.     cout << "Data setelah di-Sorting : ";
39.     printData(arr, n);
40.     cout << "\nElapsed time in nanoseconds : " << elapsed.count() << " ns" << endl;
41. }

```

```

Masukkan banyak data : 10
Masukkan Data : 10 9 8 7 6 5 4 3 2 1
Data setelah di-Sorting : 1 2 3 4 5 6 7 8 9 10

Elapsed time in nanoseconds : 600 ns

```

### Studi Kasus 3: INSERTION SORT

Insertion sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma insertion sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma insertion sort
- Tentukan  $T(n)$  dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi  $T(n)$  dengan **metode substitusi** untuk mendapatkan

kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ

- Lakukan implementasi koding program untuk algoritma insertion sort dengan menggunakan bahasa C++

### Insertion Sort

Sub problem = 1  
Masalah pd subprob ( $m = n-1$ )  
Waktu pembagian =  $n$   
Pengulangan =  $n$

$$T(n) = \textcircled{H} (1) T(n-1) + \textcircled{H} (n)$$

Worst Case

$$\begin{aligned} T(n) &= cn + cn - c + cn - 2c + \dots + 2c + cn \leq 2cn^2 + cn^2 \\ &= \left( \frac{cn^2 - 1}{2} \right) + cn \leq 2cn^2 + cn^2 \\ &= c \left( \frac{cn^2 - 3n + 2}{2} \right) + cn \leq 2cn^2 + cn^2 \\ &= c \left( \frac{n^2}{2} \right) - c \left( \frac{3n}{2} \right) + c + cn \leq 2cn^2 + cn^2 \\ &= O(n^2) \end{aligned}$$

Best Case

$$\begin{aligned} T(n) &= cn \leq cn \\ &= \Omega(n) \end{aligned}$$

Avg Case

$$\frac{(cn + cn^2)}{n} = \textcircled{H} n$$

```
1. #include<iostream>
2. #include <chrono>
3. #include <unistd.h>
4. using namespace std;
5. void insertionSort(int *array, int length) {
6.     int temp, j;
7.     for(int i = 1; i<length; i++) {
8.         temp = array[i];
9.         j = i;
```

```

10.     while(j > 0 && array[j-1]>temp) {
11.         array[j] = array[j-1];
12.         j--;
13.     }
14.     array[j] = temp;
15. }
16. }
17. void printData(int *array, int length) {
18.     for(int i = 0; i<length; i++)
19.         cout << array[i] << " ";
20.     cout << endl;
21. }
22.
23. int main() {
24.     int n,arr[30];
25.     cout << "Masukkan banyak data: ";
26.     cin >> n;
27.
28.     cout << "Masukkan Data : ";
29.     for(int i = 0; i<n; i++) {
30.         cin >> arr[i];
31.     }
32.     cout << "Array sebelum di-Sorting: ";
33.     printData(arr, n);
34.     auto start = std::chrono::steady_clock::now();
35.     insertionSort(arr, n);
36.     auto end = std::chrono::steady_clock::now();
37.     auto elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);
38.     cout << "Array setelah di-Sorting: ";
39.     printData(arr, n);
40.     cout << "\nElapsed time in nanoseconds : " << elapsed.count()<< " ns" << endl;
41. }

```

```

Masukkan banyak data: 10
Masukkan Data : 10 9 8 7 6 5 4 3 2 1
Array sebelum di-Sorting: 10 9 8 7 6 5 4 3 2 1
Array setelah di-Sorting: 1 2 3 4 5 6 7 8 9 10

Elapsed time in nanoseconds : 400 ns

```

#### Studi Kasus 4: BUBBLE SORT

Bubble sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma bubble sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma bubble sort
- Tentukan  $T(n)$  dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi  $T(n)$  dengan **metode master** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma bubble sort dengan menggunakan bahasa C++



Bubble Sort =  $O(n^2)$   
 Subproblem = 1  
 Masalah pada subproblem  $n-1$   
 Waktu pembagian:  $n$   
 -1 + Penggabungan =  $n$   
 $T(n) = \Theta(1) T(n-1) + \Theta(n)$

Worst Case  

$$T(n) = cn + cn - c + cn - 2c + \dots + 2c + c \leq 2cn^2 + cn^2$$

$$= c \left( \frac{(n-1)(n-2)}{2} \right) + c \leq 2cn^2 + cn^2$$

$$= c \left( \frac{n^2 - 3n + 2}{2} \right) + c \leq 2cn^2 + cn^2$$

$$= c \left( \frac{n^2}{2} \right) - c \left( \frac{3n}{2} \right) + c \leq 2cn^2 + cn^2$$

$$= \cancel{c \left( \frac{n^2}{2} \right)} - \cancel{c \left( \frac{3n}{2} \right)} + c = O(n^2)$$

Best Case  

$$T(n) = cn + cn - c + cn - 2c + \dots + 2c + c \leq 2cn^2 + cn^2$$

$$= c \left( \frac{(n-1)(n-2)}{2} \right) + c \leq 2cn^2 + cn^2$$

$$= c \left( \frac{n^2 - 3n + 2}{2} \right) + c \leq 2cn^2 + cn^2$$

$$= c \left( \frac{n^2}{2} \right) + c \left( \frac{3n}{2} \right) + c \leq 2cn^2 + cn^2$$

$$= \Omega(n^2)$$

Avg Case  

$$T(n) = cn^2 + cn^2$$

$$= \Theta(n^2)$$

```

1. #include<iostream>
2. #include <chrono>
3. #include <unistd.h>
4. using namespace std;
5.
6. void swap(int &a, int &b) {
7.     int temp;
8.     temp = a;
9.     a = b;
10.    b = temp;
11. }
12. void printData(int *array, int length) {
13.     for(int i = 0; i<length; i++)
14.         cout << array[i] << " ";
15.     cout << endl;
16. }
17. void bubbleSort(int *array, int length) {

```

```

18.     for(int i = 0; i<length; i++) {
19.         int isSwap = 0;
20.         for(int j = 0; j<length-i-1; j++) {
21.             if(array[j] > array[j+1]) {
22.                 swap(array[j], array[j+1]);
23.                 isSwap = 1;
24.             }
25.         }
26.         if(!isSwap)
27.             break;
28.     }
29. }
30. int main() {
31.     int n,arr[30];
32.     cout<< "Bubble Sort\n";
33.     cout << "Masukkan banyak Data: ";
34.     cin >> n;
35.     cout << "Masukkan Data : ";
36.     for(int i = 0; i<n; i++) {
37.         cin >> arr[i];
38.     }
39.     auto start = std::chrono::steady_clock::now();
40.     bubbleSort(arr, n);
41.     auto end = std::chrono::steady_clock::now();
42.     auto elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);
43.     cout << "Array setelah di-Sorting: ";
44.     printData(arr, n);
45.     cout << "\nElapsed time in nanoseconds : " << elapsed.count()<< " ns" << endl;
46. }

```

```

Bubble Sort
Masukkan banyak Data: 10
Masukkan Data : 10 9 8 7 6 5 4 3 2 1
Array setelah di-Sorting: 1 2 3 4 5 6 7 8 9 10

Elapsed time in nanoseconds : 800 ns

```