

Tugas Praktikum Analisis Algoritma
ALGORITMA DIVIDE & CONQUER



Dibuat Oleh :

Adryan Luthfi Faiz

140810160049

UNIVERSITAS PADJADJARAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI S1 TEKNIK INFORMATIKA
2019

STUDI KASUS

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?

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++

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++

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++

PROGRAM C++

```
/******
```

Semua Sorting (Merge , Select , Insert , Bubble)

Disatuin jadi 1 program "MergeSort.cpp"

Inputnya random sekitar 1 - 100 jadi gak perlu diinput satu-satu

```
*****/
```

```
/******
```

Adryan Luthfi Faiz

140810160049

IYON AdryanLF

```
*****/
```

```
#include <iostream>
```

```
#include <chrono>
```

```
#include <ctime>    // For time()
```

```
#include <cstdlib>  // For srand() and rand()
```

```
#include <stdlib.h> // for system cls clear screen
```

```
using namespace std;
```

```
using namespace std::chrono;
```

```
// --- Merge Sorting --- //
```

```

void merge (int *a, int low, int high, int mid) {
    int i, j, k, temp[high-low+1];
    i = low;
    k = 0;
    j = mid + 1;

    while (i <= mid && j <= high) {
        if (a[i] < a[j]) {
            temp[k] = a[i];
            k++;
            i++;
        }
        else {
            temp[k] = a[j];
            k++;
            j++;
        }
    }

    while (i <= mid) {
        temp[k] = a[i];
        k++;
        i++;
    }

    while (j <= high) {
        temp[k] = a[j];
        k++;
        j++;
    }

    for (i = low; i <= high; i++) {
        a[i] = temp[i-low];
    }
}

```

```

void mergeSort(int *a, int low, int high) {
    int mid;
    if (low < high) {
        mid=(low+high)/2;
        mergeSort(a, low, mid);
        mergeSort(a, mid+1, high);
    }
}

```

```

        merge(a, low, high, mid);
    }
}

// --- Selection Sorting --- //
void selectionSort (int arr[], int n){
    int i, j;
    for (i = 0; i < n; ++i){
        for (j = i+1; j < n; ++j){
            if (arr[i] > arr[j]){
                arr[i] = arr[i]+arr[j];
                arr[j] = arr[i]-arr[j];
                arr[i] = arr[i]-arr[j];
            }
        }
    }
}

```

```

// --- Insertion Sorting --- //
struct list {
    int data;
    list *next;
};

list* InsertinList(list *head, int n){
    list *newnode = new list;
    list *temp = new list;

    newnode->data = n;
    newnode->next = NULL;

    if(head == NULL){
        head = newnode;
        return head;
    }
    else {
        temp = head;

        if(newnode->data < head->data){

```

```

        newnode->next = head;
        head = newnode;
        return head;
    }

    while(temp->next != NULL) {
        if(newnode->data < (temp->next)->data)
            break;

        temp=temp->next;
    }

    newnode->next = temp->next;
    temp->next = newnode;
    return head;
}
}

```

```

// --- Bubble Sorting --- //
void bubbleSort (int arr[], int n) {
    int i, j;
    for (i = 0; i < n; ++i) {
        for (j = 0; j < n-i-1; ++j) {
            if (arr[j] > arr[j+1]) {
                arr[j] = arr[j]+arr[j+1];
                arr[j+1] = arr[j]-arr[j + 1];
                arr[j] = arr[j]-arr[j + 1];
            }
        }
    }
}

```

```

void listSort () {
    cout<<" --- List Sorting --- "<<endl<<endl;
    cout<<"1. Merge Sorting"<<endl;
    cout<<"2. Selection Sorting"<<endl;
    cout<<"3. Insertion Sorting"<<endl;
    cout<<"4. Bubble Sorting"<<endl<<endl;
    cout<<"Pilih Sorting yang ingin digunakan : ";
}

```

```

int main() {
    int n, i , num , pilih ;
    int temprand;
    char coba;
    list *head = new list;
    head = NULL;
    srand(time(0)); // Initialize random number generator.

    start:
    pilih = 0;

    listSort();
    while(pilih < 1 || pilih > 4) {
        cin>>pilih;
        if (pilih < 1 || pilih > 4) cout<<"ERROR , Masukkan angka diantara 1 - 4 : ";
    }

    cout<<endl<<"Masukkan jumlah elemen data yang ingin diurutkan: ";
    cin>>n;

    int arr[n];

    high_resolution_clock::time_point t1 = high_resolution_clock::now();

    switch (pilih) {
        case 1 :
            cout <<"Berikut elemen random :";
            for(i = 0; i < n; i++) {
                temprand = (rand() % 100) + 1;
                arr[i] = temprand;
                cout<<temprand<<" ";
            }
            mergeSort(arr, 0, n-1);
            break;
        case 2 :
            cout <<"Berikut elemen random :";
            for(i = 0; i < n; i++) {
                temprand = (rand() % 100) + 1;
                arr[i] = temprand;
                cout<<temprand<<" ";
            }
    }
}

```

```

    }
    selectionSort(arr, n);
    break;
case 3 :
    cout <<"Berikut elemen random :";
    for(i = 0; i < n; i++) {
        num = (rand() % 100) + 1;
        head = InsertinList(head, num);
        cout<<num<<" ";
    }
    cout<<endl<<endl;
    cout<<"¥nArray yang telah diurutkan: ";
    while(head != NULL) {
        cout<<" "<<head->data;
        head = head->next;
    }
    break;
case 4 :
    cout <<"Berikut elemen random :";
    for(i = 0; i < n; i++) {
        temprand = (rand() % 100) + 1;
        arr[i] = temprand;
        cout<<temprand<<" ";
    }
    bubbleSort(arr, n);
    break;
}
cout<<endl<<endl;

if (pilih !=3){ // Selain insertion karena insertion make list
    cout<<"¥nArray yang telah diurutkan: ";
    for (i = 0; i < n; i++) cout<<" "<<arr[i];
}

high_resolution_clock::time_point t2 = high_resolution_clock::now();
auto duration = duration_cast<microseconds>( t2 - t1 ).count();
cout<<endl<<duration<<" microseconds"<<endl;

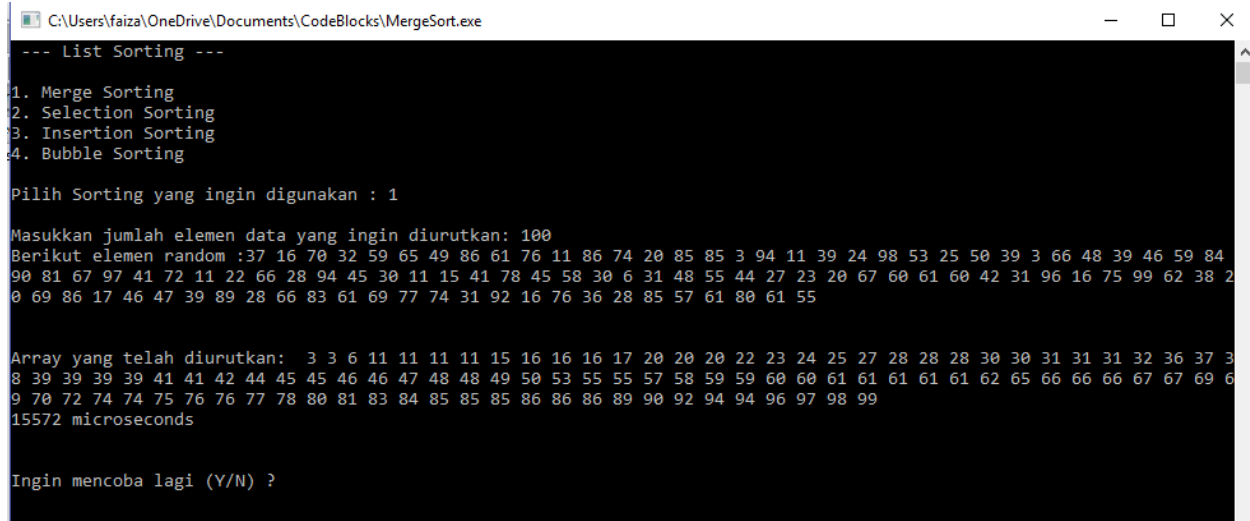
cout<<endl<<endl<<"Ingin mencoba lagi (Y/N) ? ";
cin>>coba;
if (coba == 'y' || coba == 'Y') {
    system("CLS");
}

```



```
    goto start:
}
}
```

MERGE SORT



```
C:\Users\faiza\OneDrive\Documents\CodeBlocks\MergeSort.exe
--- List Sorting ---
1. Merge Sorting
2. Selection Sorting
3. Insertion Sorting
4. Bubble Sorting

Pilih Sorting yang ingin digunakan : 1

Masukkan jumlah elemen data yang ingin diurutkan: 100
Berikut elemen random :37 16 70 32 59 65 49 86 61 76 11 86 74 20 85 85 3 94 11 39 24 98 53 25 50 39 3 66 48 39 46 59 84
90 81 67 97 41 72 11 22 66 28 94 45 30 11 15 41 78 45 58 30 6 31 48 55 44 27 23 20 67 60 61 60 42 31 96 16 75 99 62 38 2
0 69 86 17 46 47 39 89 28 66 83 61 69 77 74 31 92 16 76 36 28 85 57 61 80 61 55

Array yang telah diurutkan:  3 3 6 11 11 11 11 15 16 16 16 17 20 20 20 22 23 24 25 27 28 28 28 30 30 31 31 31 32 36 37 3
8 39 39 39 39 41 41 42 44 45 45 46 46 47 48 48 49 50 53 55 55 57 58 59 59 60 60 61 61 61 61 62 65 66 66 66 67 67 69 6
9 70 72 74 74 75 76 76 77 78 80 81 83 84 85 85 85 86 86 86 89 90 92 94 94 96 97 98 99
15572 microseconds

Ingin mencoba lagi (Y/N) ?
```

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 15572 micro s = 0.015572 s

Big-O = Big-Ω = Big-Θ = $n * \log n$

SELECTION SORT

```
C:\Users\faiza\OneDrive\Documents\CodeBlocks\MergeSort.exe
--- List Sorting ---
1. Merge Sorting
2. Selection Sorting
3. Insertion Sorting
4. Bubble Sorting

Pilih Sorting yang ingin digunakan : 2

Masukkan jumlah elemen data yang ingin diurutkan: 100
Berikut elemen random :1 57 54 98 49 56 99 68 38 40 9 59 22 27 81 100 38 40 57 77 63 71 75 95 28 96 30 66 8 17 9 51 30 4
9 57 58 87 83 50 66 76 60 33 66 1 68 33 73 65 90 80 16 94 29 67 62 75 49 28 93 33 27 26 11 47 42 46 75 70 62 49 17 51 90
82 100 38 16 50 30 31 47 37 57 14 100 74 80 25 85 50 43 41 4 41 92 9 87 17 22

Array yang telah diurutkan: 1 1 4 8 9 9 9 11 14 16 16 17 17 17 22 22 25 26 27 27 28 28 29 30 30 30 31 33 33 33 37 38 38
38 40 40 41 41 42 43 46 47 47 49 49 49 49 50 50 50 51 51 54 56 57 57 57 57 58 59 60 62 62 63 65 66 66 66 67 68 68 70 71
73 74 75 75 75 76 77 80 80 81 82 83 85 87 87 90 90 92 93 94 95 96 98 99 100 100 100
31199 microseconds

Ingin mencoba lagi (Y/N) ?
```

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 31199 microseconds = 0.031199 s

Big-O = Big-Ω = Big-Θ = n^2

INSERTION SORT

```
C:\Users\faiza\OneDrive\Documents\CodeBlocks\MergeSort.exe
--- List Sorting ---
1. Merge Sorting
2. Selection Sorting
3. Insertion Sorting
4. Bubble Sorting

Pilih Sorting yang ingin digunakan : 3

Masukkan jumlah elemen data yang ingin diurutkan: 100
Berikut elemen random :69 64 56 7 73 34 39 28 22 96 76 64 17 22 100 55 84 98 85 4 60 40 36 52 19 37 9 8 26 12 11 69 34 1
1 21 35 68 65 37 84 45 28 20 81 63 28 14 30 97 24 52 42 53 97 75 17 28 37 86 25 58 91 11 21 75 83 94 49 17 3 51 35 35 51
45 71 57 56 2 69 56 73 85 81 28 73 62 4 33 77 53 25 59 93 81 47 13 22 50 45

Array yang telah diurutkan: 2 3 4 4 7 8 9 11 11 11 12 13 14 17 17 17 19 20 21 21 22 22 22 24 25 25 26 28 28 28 28 28 30
33 34 34 35 35 35 36 37 37 37 39 40 42 45 45 45 47 49 50 51 51 52 52 53 53 55 56 56 56 57 58 59 60 62 63 64 64 65 68 69
69 69 71 73 73 73 75 75 76 77 81 81 81 83 84 84 85 85 86 91 93 94 96 97 97 98 100

31235 microseconds

Ingin mencoba lagi (Y/N) ?
```

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 31235 microseconds = 0.031235 s

Big-O = n

Big-Ω = Big-θ = n^2

BUBBLE SORT

```
C:\Users\faiza\OneDrive\Documents\CodeBlocks\MergeSort.exe
--- List Sorting ---
1. Merge Sorting
2. Selection Sorting
3. Insertion Sorting
4. Bubble Sorting
Pilih Sorting yang ingin digunakan : 4
Masukkan jumlah elemen data yang ingin diurutkan: 100
Berikut elemen random :1 89 47 12 43 6 72 3 45 62 93 62 93 11 77 19 49 12 73 78 5 1 72 54 68 25 37 88 93 98 27 1 20 53 3
0 100 16 27 23 61 69 19 68 65 9 61 43 49 90 94 14 88 32 45 61 41 41 9 52 50 24 4 61 43 22 92 48 10 38 60 48 25 36 57 13
88 92 61 98 13 8 48 60 89 27 90 55 85 40 80 94 76 50 53 6 91 17 3 60 26

Array yang telah diurutkan: 1 1 1 3 3 4 5 6 6 8 9 9 10 11 12 12 13 13 14 16 17 19 19 20 22 23 24 25 25 26 27 27 27 30 3
2 36 37 38 40 41 41 43 43 43 45 45 47 48 48 48 49 49 50 50 52 53 53 54 55 57 60 60 60 61 61 61 61 61 62 62 65 68 68 69 7
2 72 73 76 77 78 80 85 88 88 88 89 89 90 90 91 92 92 93 93 93 94 94 98 98 100
31196 microseconds

Ingin mencoba lagi (Y/N) ?
```

Kompleksitas waktu:

Durasi waktu yang dibutuhkan untuk 100 input: 31196 microseconds = 0.031196 s

Big-O = n

Big-Ω = Big-θ = n^2