LAPORAN PRAKTIKUM 4
ANALISIS ALGORITMA

Dibuat oleh:
AHMAD IRFAN FADHOLI
140810180034

UNIVERSITAS PADJADJARAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
2020

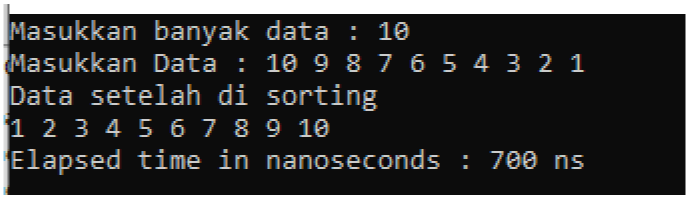
Studi Kasus 1: MERGE SORT

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?

```cpp
#include <iostream>
#include <chrono>
#include <unistd.h>
using namespace std;
void merge(int *,int, int , int );
void merge_sort(int *arr, int low, int high){

    int mid;
    if (low < high){

        mid=(low+high)/2;
        merge_sort(arr,low,mid);
        merge_sort(arr,mid+1,high);
        merge(arr,low,high,mid);
    }
}

void merge(int *arr, int low, int high, int mid){
    int i, j, k, c[50];
    i = low;
    k = low;
    j = mid + 1;
    while (i <= mid && j <= high) {
        if (arr[i] < arr[j]) {
            c[k] = arr[i];
            k++;
            i++;
        }
        else  {
            c[k] = arr[j];
            k++;
            j++;
        }
    }
    while (i <= mid) {
        c[k] = arr[i];
        k++;
        i++;
    }
    while (j <= high) {
        c[k] = arr[j];
        k++;
        j++;
    }
    for (i = low; i < k; i++)  {
        arr[i] = c[i];
    }
}


int main(){
    int arr[20], num;
    cout<<"Masukkan banyak data : ";
    cin>>num;
    cout<<"Masukkan Data : ";
    for (int i = 0; i < num; i++) {
        cin>>arr[i];
    }
    auto start = std::chrono::steady_clock::now();
    merge_sort(arr, 0, num-1);
    auto end = std::chrono::steady_clock::now();
    auto elapsed =
    std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);

    cout<<"Data setelah di sorting\n";
    for (int i = 0; i < num; i++){
        cout<<arr[i]<<" ";
    }
    cout << "\nElapsed time in nanoseconds : " << elapsed.count()<< " ns" << endl;

}
```

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

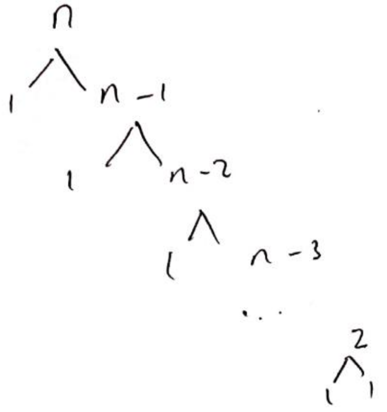
Sub problem = 1

Masalah setiap subproblem = n-1

Waktu pembagian = n

-11- penggabungan = n

$T(n) = \{ \Theta(1)\ T(n-1) + \Theta(n)$



Worst case

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

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

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

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

$O(n^2)$

Best case =

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

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

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

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

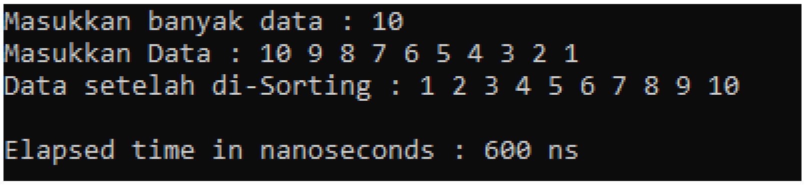
$= c\left(\frac{n^3}{2}\right) - \frac{3n}{2} + 1 + Cn$

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

Avg case = $T(n) = \frac{n^2 + n^2}{2} = n^2$

$= \Theta\ (n^2)$

```cpp
#include<iostream>
#include <chrono>
#include <unistd.h>
using namespace std;
void swap(int &a, int &b) {
   int temp;
   temp = a;
   a = b;
   b = temp;
}
void printData(int *array, int length) {
   for(int i = 0; i<length; i++)
      cout << array[i] << " ";
   cout << endl;
}
void selectionSort(int *array, int length) {
   int i, j, imin;
   for(i = 0; i<length-1; i++) {
      imin = i;
      for(j = i+1; j<length; j++)
         if(array[j] < array[imin])
            imin = j;
         swap(array[i], array[imin]);
   }
}
int main() {
   int n,arr[30];
   cout << "Masukkan banyak data : ";
   cin >> n;
   cout << "Masukkan Data : ";
   for(int i = 0; i<n; i++) {
      cin >> arr[i];
   }
   auto start = std::chrono::steady_clock::now();
   selectionSort(arr, n);
   auto end = std::chrono::steady_clock::now();
   auto elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);
   cout << "Data setelah di-Sorting : ";
   printData(arr, n);
   cout << "\nElapsed time in nanoseconds : " << elapsed.count()<< " ns" << endl;
}
```



```
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:

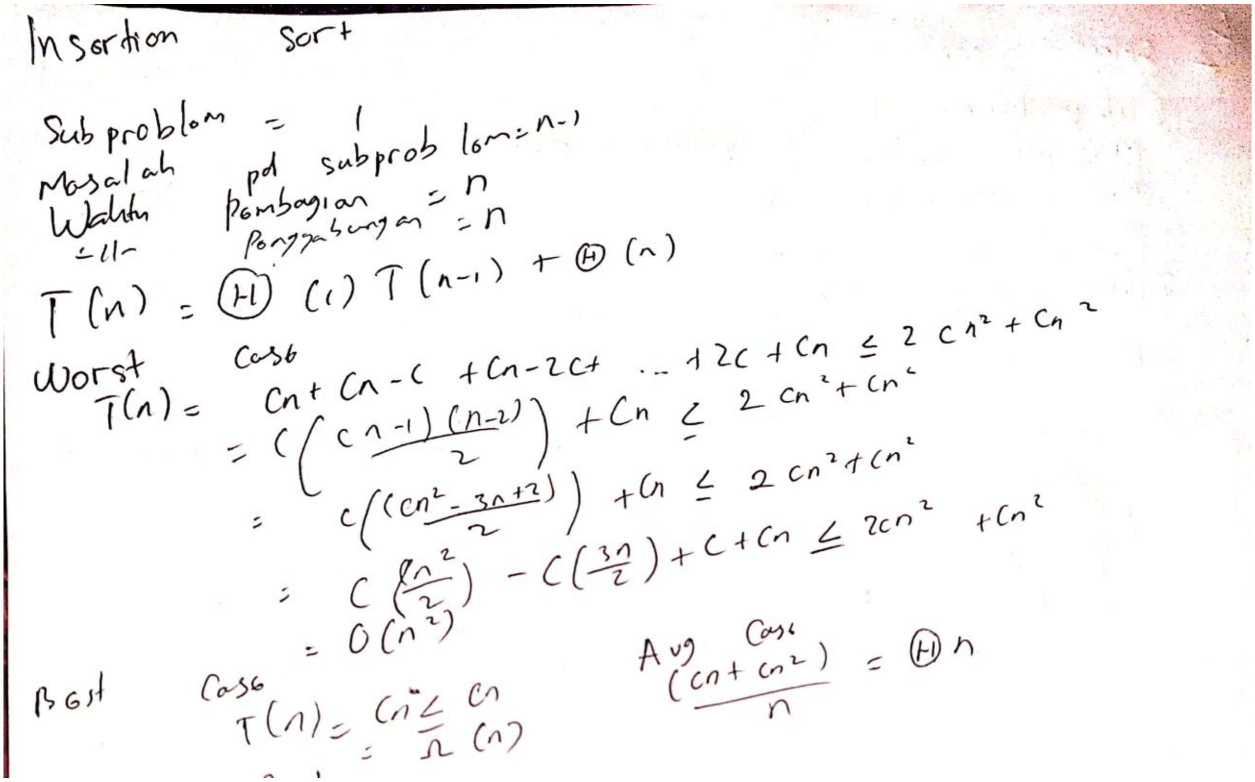
- Pelajari cara kerja algoritma insertion sort
- Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

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

- Selesaikan persamaan rekurensi T(n) dengan **metode subtitusi** 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++



Insortion sort

Subproblom = 1

Masalah pd subprob lom = n-1

Waktu pembagian = n

-11- Penggabungan = n

$T(n) = \Theta(1)\, T(n-1) + \Theta(n)$

Worst Case

$T(n) = Cn + Cn - C + Cn - 2C + \ldots + 2C + Cn \leq 2Cn^2 + Cn^2$

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

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

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

$= O(n^2)$

Avg Case

$\frac{(Cn + Cn^2)}{n} = \Theta n$

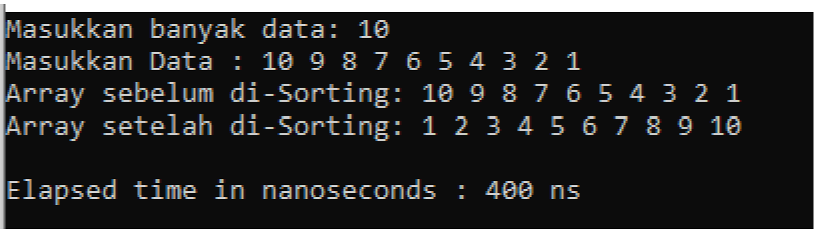
Best Case

$T(n) = Cn \leq Cn$

$= \Omega(n)$

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

```cpp
      while(j > 0 && array[j-1]>temp) {
         array[j] = array[j-1];
         j--;
      }
      array[j] = temp;
   }
}
void printData(int *array, int length) {
   for(int i = 0; i<length; i++)
      cout << array[i] << " ";
   cout << endl;
}

int main() {
   int n,arr[30];
   cout << "Masukkan banyak data: ";
   cin >> n;

   cout << "Masukkan Data : ";
   for(int i = 0; i<n; i++) {
      cin >> arr[i];
   }
   cout << "Array sebelum di-Sorting: ";
   printData(arr, n);
   auto start = std::chrono::steady_clock::now();
   insertionSort(arr, n);
   auto end = std::chrono::steady_clock::now();
   auto elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);
   cout << "Array setelah di-Sorting: ";
   printData(arr, n);
   cout << "\nElapsed time in nanoseconds : " << elapsed.count()<< " ns" << endl;
}
```



```
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
```

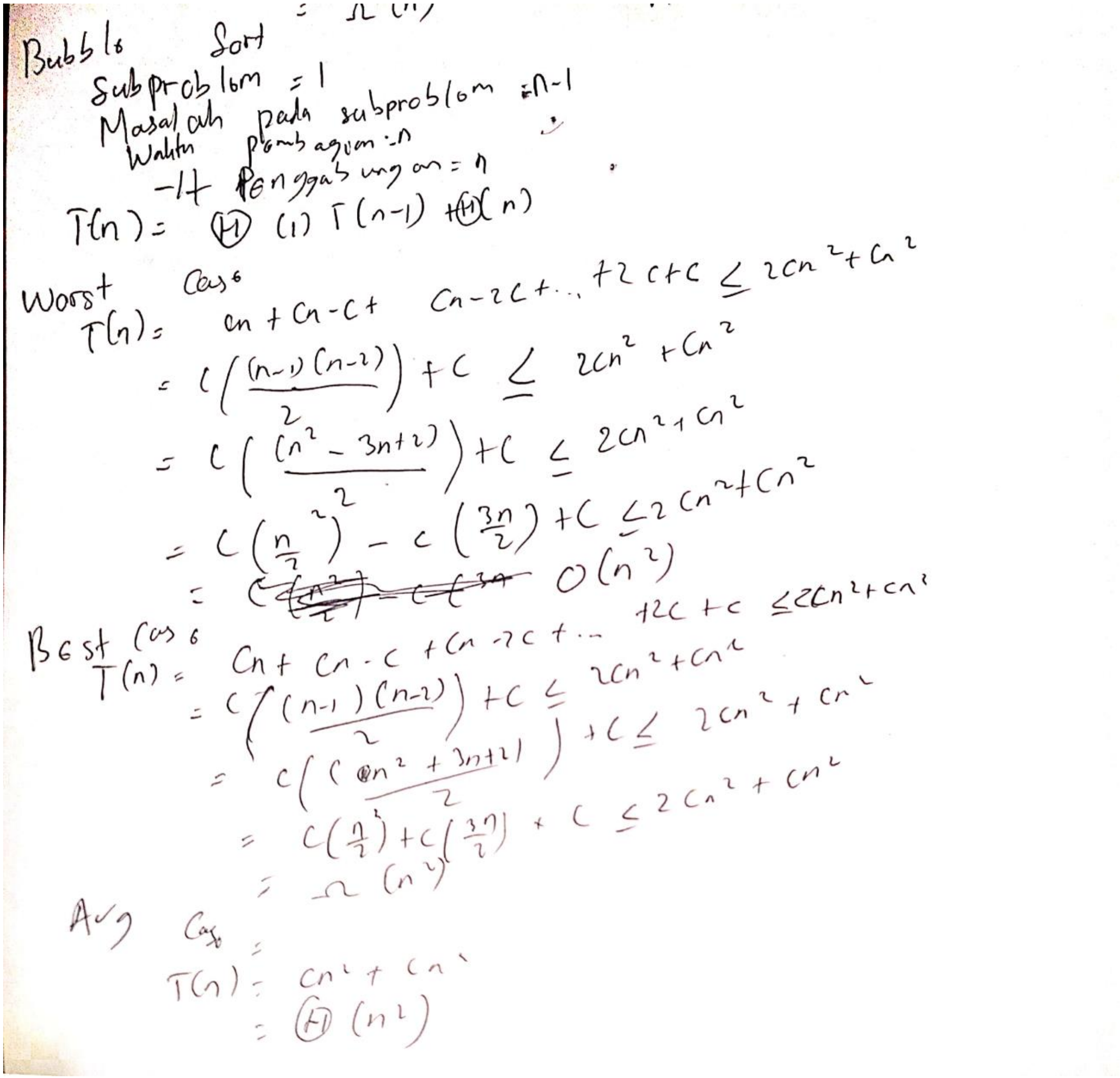
Studi Kasus 4: BUBBLE SORT

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:

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

- 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++



$= \Omega(n)$

Bubble Sort

Subproblem = 1

Masalah pada subproblem = n-1

Waktu pembagian = n

-H Penggabungan = n

$T(n) = \Theta(1)\,T(n-1) + \Theta(n)$

Worst Case

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

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

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

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

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

Best Case

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

$$\le 2cn^2 + cn^2$$

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

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

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

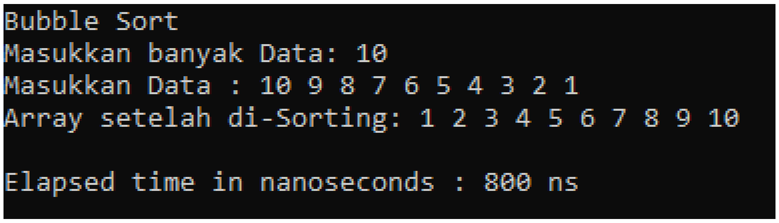
$$= \Omega(n^2)$$

Avg Case =

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

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

```cpp
#include<iostream>
#include <chrono>
#include <unistd.h>
using namespace std;

void swap(int &a, int &b) {
   int temp;
   temp = a;
   a = b;
   b = temp;
}
void printData(int *array, int length) {
   for(int i = 0; i<length; i++)
      cout << array[i] << " ";
   cout << endl;
}
void bubbleSort(int *array, int length) {
   for(int i = 0; i<length; i++) {
      int isSwap = 0;
      for(int j = 0; j<length-i-1; j++) {
         if(array[j] > array[j+1]) {
            swap(array[j], array[j+1]);
            isSwap = 1;
         }
      }
      if(!isSwap)
         break;
   }
}
int main() {
   int n,arr[30];
   cout<< "Bubble Sort\n";
   cout << "Masukkan banyak Data: ";
   cin >> n;
   cout << "Masukkan Data : ";
   for(int i = 0; i<n; i++) {
      cin >> arr[i];
   }
   auto start = std::chrono::steady_clock::now();
   bubbleSort(arr, n);
   auto end = std::chrono::steady_clock::now();
   auto elapsed = std::chrono::duration_cast<std::chrono::nanoseconds>(end - start);
   cout << "Array setelah di-Sorting: ";
   printData(arr, n);
   cout << "\nElapsed time in nanoseconds : " << elapsed.count()<< " ns" << endl;
}
```



```
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
```