LAPORAN PRAKTIKUM 4
ANALISIS ALGORITMA

Rio Sapta Samudera

140810180030

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?

```
#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:

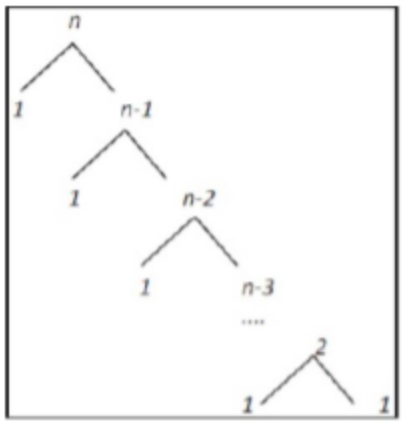
- Pelajari cara kerja algoritma selection sort
- Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer:

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

- 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++

T(n) selection sort
for I ← n downto 2 do {*pass sebanyak n-1 kali*}
imaks ← 1
for j ← 2 to I do
if $x_j > x_{imaks}$ then
imaks ← j
endif
endfor
{pertukaran x_{imaks} dengan x_i}
temp ← x_i
x_i ← x_{imaks}
x_{imaks} ← temp
endfor

Subproblem = 1
Masalah setiap subproblem = n-1
Waktu proses pembagian = n
Waktu proses penggabungan = n
$T(n) = \{\Theta(1)\, T(n-1) + \Theta(n)$



T(n) = cn + cn-c + cn-2c + ... + 2c + cn
= c((n-1)(n-2)/2) + cn
= c((n^2-3n+2)/2) + cn
= c(n^2/2)-(3n/2)+1 +cn
= O(n^2)

T(n) = cn + cn-c + cn-2c + ... + 2c + cn
= c((n-1)(n-2)/2) + cn
= c((n^2-3n+2)/2) + cn
= c(n^2/2)-(3n/2)+1 +cn
= Ω (n^2)

T(n) = cn^2
= Θ(n^2)

```
#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;
      cout << "Masukkan banyak data : ";
      cin >> n;
      int arr[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++

```
for I ← 2 to n do
        insert ← x_i
        j ← i
        while (j < i) and (x[j-i] > insert) do
                x[j] ← x[j-1]
                j ← j-1
        endwhile
        x[j] = insert
endfor
```

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

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

T(n) = cn + cn-c +cn-2c + ... + 2c +cn <= $2cn^2 + cn^2$

= c((n-1)(n-2)/2) + cn<= $2cn^2 + cn^2$

= c((n^2-3n+2)/2) + cn<= $2cn^2 + cn^2$

= c(n^2/2)-c(3n/2)+c+cn <= $2cn^2 + cn^2$

=O(n^2)

T(n) = cn <= cn

= Ω (n)

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

= Θ(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;
      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:

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

- 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++

Subproblem = 1
Masalah setiap subproblem = n-1
Waktu proses pembagian = n
Waktu proses penggabungan

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

T(n) = cn + cn-c +cn-2c + ... + 2c +c <= $2cn^2 + cn^2$
= c((n-1)(n-2)/2) + c<= $2cn^2 + cn^2$
= c((n^2-3n+2)/2) + c<= $2cn^2 + cn^2$
= c(n^2/2)-c(3n/2)+2c <= $2cn^2 + cn^2$
=O(n^2)

T(n) = cn + cn-c +cn-2c + ... + 2c +c <= $2cn^2 + cn^2$
= c((n-1)(n-2)/2) + c<= $2cn^2 + cn^2$
= c((n^2-3n+2)/2) + c<= $2cn^2 + cn^2$
= c(n^2/2)-c(3n/2)+2c <= $2cn^2 + cn^2$
= Ω (n^2)

T(n) = $cn^2 + cn^2$
= Θ(n^2)

```
#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
```