Studi Kasus 1 : Merge Sort

Running time Merge sort dengan jumlah data 20, di komputer saya.

```
PS D:\BELAJAR\CODE\AnAlgo\W4> c++ -g -Wall -o .\merge.exe merge.cpp
PS D:\BELAJAR\CODE\AnAlgo\W4> .\merge.exe
3 1 4 21 43 25 4 124 5 42 8 97 5 10 11 90 24 76 21 58
1 3 4 4 5 5 8 10 11 21 21 24 25 42 43 58 76 90 97 124
runtime in nanoseconds : 0 ns
PS D:\BELAJAR\CODE\AnAlgo\W4>
```

Tetapi jika mengacu ke kompleksitas waktu O maka $T(20 \log_{10} 20) = 26$

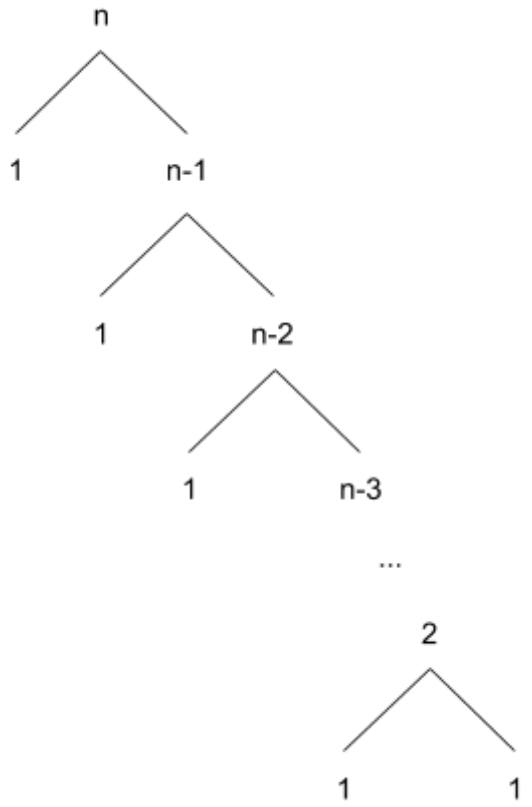
Kode :

```
#include <iostream>
#include <chrono>

using namespace std;

void merge(int list[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = list[l + i];
    for (j = 0; j < n2; j++)
        R[j] = list[m + 1 + j];

    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            list[k] = L[i];
            i++;
        }
        else
        {
            list[k] = R[j];
            j++;
        }
        k++;
    }
```

```
    while (i < n1)
    {
        list[k] = L[i];
        i++;
        k++;
    }

    while (j < n2)
    {
        list[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int list[], int l, int r)
{
    if (l < r)
    {
        int m = l + (r - l) / 2;

        mergeSort(list, l, m);
        mergeSort(list, m + 1, r);

        merge(list, l, m, r);
    }
}

void print(int list[], int n)
{
    for (int i = 0; i < n; i++)
        cout << list[i] << " ";
    cout << "\n";
}

int main()
{
    int list[20]{3, 1, 4, 21, 43, 25, 4, 124, 5, 42, 8, 97, 5, 10, 11, 90, 24, 76, 21,
58};
    int size = sizeof(list) / sizeof(list[0]);

    print(list, size);

    auto start = chrono::high_resolution_clock::now();
    mergeSort(list, 0, size - 1);
    auto stop = chrono::high_resolution_clock::now();

    print(list, size);

    auto duration = chrono::duration_cast<chrono::nanoseconds>(stop - start);

    cout << "runtime in nanoseconds : " << duration.count() << " ns\n";
}
```

Studi Kasus 2 : Selection Sort

Mencari kompleksitas waktu asimptotik dengan metode recursion-tree
Subproblem = 1
Masalah setiap subproblem = n-1
Waktu proses pembagian = n
Waktu proses penggabungan = n
$T(n) = \Theta(1)\, T(n-1) + \Theta(n)$



$$
\begin{aligned}
T(n) &= cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + cn \\
&= c((n\text{-}1)(n\text{-}2)/2) + cn \\
&= c((n^2\text{-}3n+2)/2) + cn \\
&= c(n^2/2)\text{-}(3n/2)+1 + cn \\
&= O(n^2)
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + cn \\
&= c((n\text{-}1)(n\text{-}2)/2) + cn \\
&= c((n^2\text{-}3n+2)/2) + cn \\
&= c(n^2/2)\text{-}(3n/2)+1 + cn \\
&= \Omega(n^2)
\end{aligned}
$$

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

Kode :

```
#include <iostream>
using namespace std;

int minIndex(int a[], int i, int j)
{
    if (i == j)
        return i;

    int k = minIndex(a, i + 1, j);

    return (a[i] < a[k])? i : k;
}

void selection(int a[], int n, int index = 0)
{
    if (index == n)
       return;

    int k = minIndex(a, index, n-1);

    if (k != index)
       swap(a[k], a[index]);

    selection(a, n, index + 1);
}

void print(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << "\n";
}

int main()
{
    int list[10] = {12,121,51,52,6,51,4,5,37,34};
    int size = sizeof(list)/sizeof(list[0]);

    print(list, size);
    selection(list, size);
    print(list, size);

}
```

Studi Kasus 3 : Insertion Sort

Mencari kompleksitas waktu asimptotik dengan metode substitusi

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

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

$$\begin{aligned} T(n) &= cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + cn \Leftarrow 2cn^2 + cn^2 \\ &= c((n\text{-}1)(n\text{-}2)/2) + cn \quad \Leftarrow 2cn^2 + cn^2 \\ &= c((n^2\text{-}3n+2)/2) + cn \quad \Leftarrow 2cn^2 + cn^2 \\ &= c(n^2/2)\text{-}c(3n/2)+c+cn \Leftarrow 2cn^2 + cn^2 \\ &= O(n^2) \end{aligned}$$

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

$$\begin{aligned} T(n) &= (cn+cn^2)/n \\ &= \Theta(n) \end{aligned}$$

Kode :

```
#include <iostream>

using namespace std;

void insertion(int list[], int n)
{
    if (n <= 1)
        return;

    insertion(list, n - 1);

    int last = list[n - 1];
    int j = n - 2;

    while (j >= 0 && list[j] > last)
    {
        list[j + 1] = list[j];
        j--;
    }
    list[j + 1] = last;
}

void print(int list[], int n)
```

```
{
    for (int i = 0; i < n; i++)
        cout << list[i] << " ";
    cout << "\n";
}

int main()
{
    int list[10] = {12, 123, 11, 12, 3, 1, 411, 41, 3, 5};
    int size = sizeof(list) / sizeof(list[0]);

    print(list, size);
    insertion(list, size);
    print(list, size);
```

Studi Kasus 4 : Bubble Sort

Mencari kompleksitas waktu asimptotik dengan metode master

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

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

$T(n) = cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + c \Leftarrow 2cn^2 + cn^2$
$= c((n\text{-}1)(n\text{-}2)/2) + c \Leftarrow 2cn^2 + cn^2$
$= c((n^2\text{-}3n+2)/2) + c \Leftarrow 2cn^2 + cn^2$
$= c(n^2/2)\text{-}c(3n/2)+2c \Leftarrow 2cn^2 + cn^2$
$= O(n^2)$

$T(n) = cn + cn\text{-}c + cn\text{-}2c + \ldots\ldots + 2c + c \Leftarrow 2cn^2 + cn^2$
$= c((n\text{-}1)(n\text{-}2)/2)+c \Leftarrow 2cn^2 + cn^2$
$= c((n^2\text{-}3n+2)/2)+c \Leftarrow 2cn^2 + cn^2$
$= c(n^2/2)\text{-}c(3n/2)+2c \Leftarrow 2cn^2 + cn^2$
$= \Omega(n^2)$

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

Kode :

```
#include <iostream>

using namespace std;

void bubbleSort(int list[], int n)
{
    if (n == 1)
        return;

    for (int i = 0; i < n - 1; i++)
        if (list[i] > list[i + 1])
            swap(list[i], list[i + 1]);

    bubbleSort(list, n - 1);
}

void print(int list[], int n)
{
    for (int i = 0; i < n; i++)
        cout << list[i] << " ";
```

```
    cout << "\n";
}

int main()
{
    int list[10] = {2, 31, 44, 1, 51, 5, 3, 2, 42, 6};
    int size = sizeof(list) / sizeof(list[0]);

    print(list, size);
    bubbleSort(list, size);
    print(list, size);
}
```