Vega Savera Yuana

140810160053

1. Merge Sort

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

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

   /* create temp arrays */
   int L[n1], R[n2];

   /* Copy data to temp arrays L[] and R[] */
   for (i = 0; i < n1; i++)
      L[i] = arr[l + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[m + 1+ j];

   i = 0; // Initial index of first subarray
   j = 0; // Initial index of second subarray
   k = l; // Initial index of merged subarray
   while (i < n1 && j < n2)
   {
      if (L[i] <= R[j])
      {
         arr[k] = L[i];
         i++;
      }
      else
      {
         arr[k] = R[j];
         j++;
      }
      k++;
   }


   while (i < n1)
```

```
    {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        int m = l+(r-l)/2;
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);
        merge(arr, l, m, r);
    }
}

void createArray(int A[] , int size){
    for(int i = 0; i < size ; i++){
        A[i] = size - i;
    }
}

int main()
{
    int arr[10000];
    int n = sizeof(arr)/sizeof(arr[0]);
    createArray(arr,n);

    high_resolution_clock::time_point t1 = high_resolution_clock::now();
    mergeSort(arr, 0, n - 1);
    high_resolution_clock::time_point t2 = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>( t2 - t1 ).count();
    cout<<endl <<duration <<" microseconds" <<endl;

    return 0;
}
```

Running Time

Banyak data yang di uji : 1000, 10.000 , 20.000 , 50.000

- 1000 Data

```
1000 Data
0 microseconds

Process returned 0 (0x0)   execution time : 0.139 s
Press any key to continue.
```

- 10.000 Data

```
10000 Data
1998 microseconds

Process returned 0 (0x0)   execution time : 0.029 s
Press any key to continue.
```

- 20000 Data

```
20000 Data
3997 microseconds

Process returned 0 (0x0)   execution time : 0.030 s
Press any key to continue.
```

- 50000 Data

```
50000 Data
12993 microseconds

Process returned 0 (0x0)   execution time : 0.045 s
Press any key to continue.
```

Kompleksitas Waktu

Big-O = Big-Ω = Big-θ = n * log n

2. Insertion Sort

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

void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++) {
      key = arr[i];
```

```
        j = i - 1;


        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
void createArray(int A[] , int size){
    for(int i = 0; i < size ; i++){
        A[i] = size - i;
    }
}

int main()
{
    int arr[500];
    int n = sizeof(arr) / sizeof(arr[0]);
    createArray(arr,n);

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

    return 0;
}
```

Running Time
Banyak data yang di uji : 10.000 , 20.000 , 50.000 , 100.000

- 1000 Data

```
1000 Data
1999 microseconds

Process returned 0 (0x0)   execution time : 0.109 s
Press any key to continue.
```

- 10.000 Data

```
10000 Data
156258 microseconds

Process returned 0 (0x0)   execution time : 0.203 s
Press any key to continue.
```

- 20.000 Data

```
20000 Data
640680 microseconds

Process returned 0 (0x0)   execution time : 0.750 s
Press any key to continue.
```

- 50.000 Data

```
50000 Data
3484131 microseconds

Process returned 0 (0x0)   execution time : 3.594 s
Press any key to continue.
```

Kompleksitas Waktu

Big-O = n

Big-Ω = Big-θ = n^2

3. Selection Sort

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


void swap(int *xp,int *yp){
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}
void selectionSort(int arr[],int n)
{

   int i , j ,min_idx;

   //One by one move boudary of unsorted subarray
   for(i = 0;i<n-1;i++)
```

```
    {

        //Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
          if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
        swap(&arr[min_idx], &arr[i]);
    }
}
void createArray(int A[] , int size){
    for(int i = 0; i < size ; i++){
        A[i] = size - i;
    }
}

void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main()
{   cout<<"50000 Data";
    int arr[50000] ;
    int n = sizeof(arr)/sizeof(arr[0]);
    createArray(arr,n);

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

    return 0;
}
```

Running Time
Banyak data yang di uji : 1000, 10.000 , 20.000 , 50.000 ,

- 1000 Data

```
1000 Data
1997 microseconds

Process returned 0 (0x0)   execution time : 0.130 s
Press any key to continue.
```

- 10000 Data

```
10000 Data
152935 microseconds

Process returned 0 (0x0)   execution time : 0.273 s
Press any key to continue.
```

- 20000 Data

```
20000 Data
560786 microseconds

Process returned 0 (0x0)   execution time : 0.669 s
Press any key to continue.
```

- 50000 Data

```
50000 Data
3265395 microseconds

Process returned 0 (0x0)   execution time : 3.359 s
Press any key to continue.
```

Big-O = Big-Ω = Big-θ = n^2

4. Bubble Sort

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

void swap(int *xp, int *yp){
   int temp = *xp;
   *yp = *yp;
   *yp = temp;
}
void bubbleSort(int arr[], int n)
{
   int i , j;
   for(i = 0;i<n-1;i++){
      for (j = 0; j < n-i-1; j++){
         if (arr[j]>arr[j+1])
```

```
            swap(&arr[j], &arr[j+1]);
        }
    }
}
void createArray(int A[] , int size){
    for(int i = 0; i < size ; i++){
        A[i] = size - i;
    }
}
int main()
{
    cout<<"50000 Data";
    int arr[50000];
    int n = sizeof(arr)/sizeof(arr[0]);
    createtArray(arr,n);

    high_resolution_clock::time_point t1 = high_resolution_clock::now();
    bubbleSort(arr, n);
    high_resolution_clock::time_point t2 = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>( t2 - t1 ).count();
    cout<<endl <<duration <<" microseconds" <<endl;
    return 0;
}
```

- 1000 Data

```
1000 Data
2002 microseconds

Process returned 0 (0x0)   execution time : 0.364 s
Press any key to continue.
```

- 10000 Data

```
10000 Data
164903 microseconds

Process returned 0 (0x0)   execution time : 0.242 s
Press any key to continue.
```

- 20000 Data

```
20000 Data
663127 microseconds

Process returned 0 (0x0)   execution time : 0.689 s
Press any key to continue.
```

- 50000 Data

```
50000 Data
3414249 microseconds

Process returned 0 (0x0)   execution time : 3.439 s
Press any key to continue.
```

Big-O = n

Big-Ω = Big-θ = n^2