**TUGAS PRAKTIKUM ANALISIS ALGORITMA**

**Modul Praktikum 4**



**Muhammad Afif**

**140810170045**

**Kelas: A**

**JURUSAN S-1 TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA dan ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

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?

Jawab:

1. Source Code

#include <iostream>

using namespace std;

void merge(int a[], int p, int q, int r) { //p=low,q=mid,r=high

// subarray1 = a[p..q], subarray2 = a[q+1..r]

int N = r-p+1;

int b[N];

int left = p, right = q+1, bIdx = 0;

while (left <= q && right <= r) // the merging

b[bIdx++] = (a[left] <= a[right]) ? a[left++] : a[right++];

while (left <= q) b[bIdx++] = a[left++]; // leftover, if any

while (right <= r) b[bIdx++] = a[right++]; // leftover, if any

for (int k = 0; k < N; k++) a[p+k] = b[k]; // copy back

}

void mergeSort(int a[], int p, int r) {

// the array to be sorted is a[p..r]

if (p < r) { // base case: p >= r (0 or 1 item)

int q = (p+r) / 2;

mergeSort(a, p , q ); // divide into two halves

mergeSort(a, q+1, r); // then recursively sort them

merge(a, p, q, r); // conquer: the merge subroutine

}

}

int main()

{

int n = 8;

int a[]={1, 5, 19, 20, 2, 11, 15, 17};//array of input

int p=1;

int r=n;

cout<<"input: ";

for (int i = 0; i < n; i++) { //print array

cout<<a[i]<<" ";

}

mergeSort(a,p,r);

cout<<"\nsorted output: ";

for (int i = 0; i < n; i++) {

cout<<a[i]<<" ";

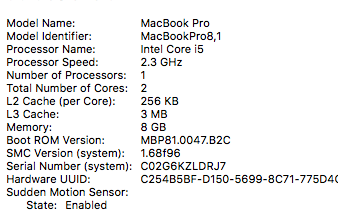
}

cout<<endl;

return 0;

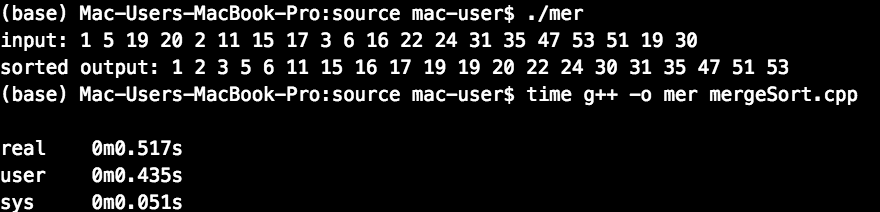
}

2. Kecepatan komputer:



Input -> a={1, 5, 19, 20, 2, 11, 15, 17 ,3, 6, 16, 22, 24, 31, 35, 47,53,51,19,30}

Running Time :



**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:

• 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++ .

Jawab:

Source Code :

#include <iostream>

using namespace std;

// Return minimum index

int minIndex(int a[], int i, int j)

{

if (i == j)

return i;

// Find minimum of remaining elements

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

// Return minimum of current and remaining.

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

}

// Recursive selection sort. n is size of a[] and index

// is index of starting element.

void recurSelectionSort(int a[], int n, int index = 0)

{

// Return when starting and size are same

if (index == n)

return;

// calling minimum index function for minimum index

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

// Swapping when index nd minimum index are not same

if (k != index)

swap(a[k], a[index]);

// Recursively calling selection sort function

recurSelectionSort(a, n, index + 1);

}

int main()

{

int arr[] = {3, 1, 5, 2, 7, 0};

int n = sizeof(arr)/sizeof(arr[0]);

// Calling function

recurSelectionSort(arr, n);

//printing sorted array

for (int i = 0; i<n ; i++)

cout << arr[i] << " ";

cout << endl;

return 0;

}

**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:

• 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++

Jawab:

Source Code :

#include <iostream>

using namespace std;

void insertionSortRecursive(int arr[], int n)

{

// Base case

if (n <= 1)

return;

// Sort first n-1 elements

insertionSortRecursive( arr, n-1 );

// Insert last element at its correct position

// in sorted array.

int last = arr[n-1];

int j = n-2;

/\* Move elements of arr[0..i-1], that are

greater than key, to one position ahead

of their current position \*/

while (j >= 0 && arr[j] > last)

{

arr[j+1] = arr[j];

j--;

}

arr[j+1] = last;

}

// A utility function to print an array of size n

void printArray(int arr[], int n)

{

for (int i=0; i < n; i++)

cout << arr[i] <<" ";

}

int main()

{

int arr[] = {12, 11, 13, 5, 6};

int n = sizeof(arr)/sizeof(arr[0]);

insertionSortRecursive(arr, n);

printArray(arr, n);

return 0;

}

**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:

• 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++

Jawab:

Source Code :

#include <bits/stdc++.h>

using namespace std;

void bubbleSort(int arr[], int n)

{

// Base case

if (n == 1)

return;

// One pass of bubble sort. After

// this pass, the largest element

// is moved (or bubbled) to end.

for (int i=0; i<n-1; i++)

if (arr[i] > arr[i+1])

swap(arr[i], arr[i+1]);

// Largest element is fixed,

// recur for remaining array

bubbleSort(arr, n-1);

}

/\* Function to print an array \*/

void printArray(int arr[], int n)

{

for (int i=0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

}

int main()

{

int arr[] = {64, 34, 25, 12, 22, 11, 90};

int n = sizeof(arr)/sizeof(arr[0]);

bubbleSort(arr, n);

printf("Sorted array : \n");

printArray(arr, n);

return 0;

}

Sumber :

<https://www.geeksforgeeks.org/recursive-insertion-sort/>

[https://www.geeksforgeeks.org/recursive-selection-sort/](https://www.geeksforgeeks.org/recursive-insertion-sort/)

[https://www.geeksforgeeks.org/recursive-bubble-sort/](https://www.geeksforgeeks.org/recursive-insertion-sort/)

<https://visualgo.net/en/sorting>