Modul 12

Sorting

Tujuan

1. Praktikan dapat memahami metode metode sorting
2. Praktikan dapat menerapkan metode sorting yang telah dipelajari

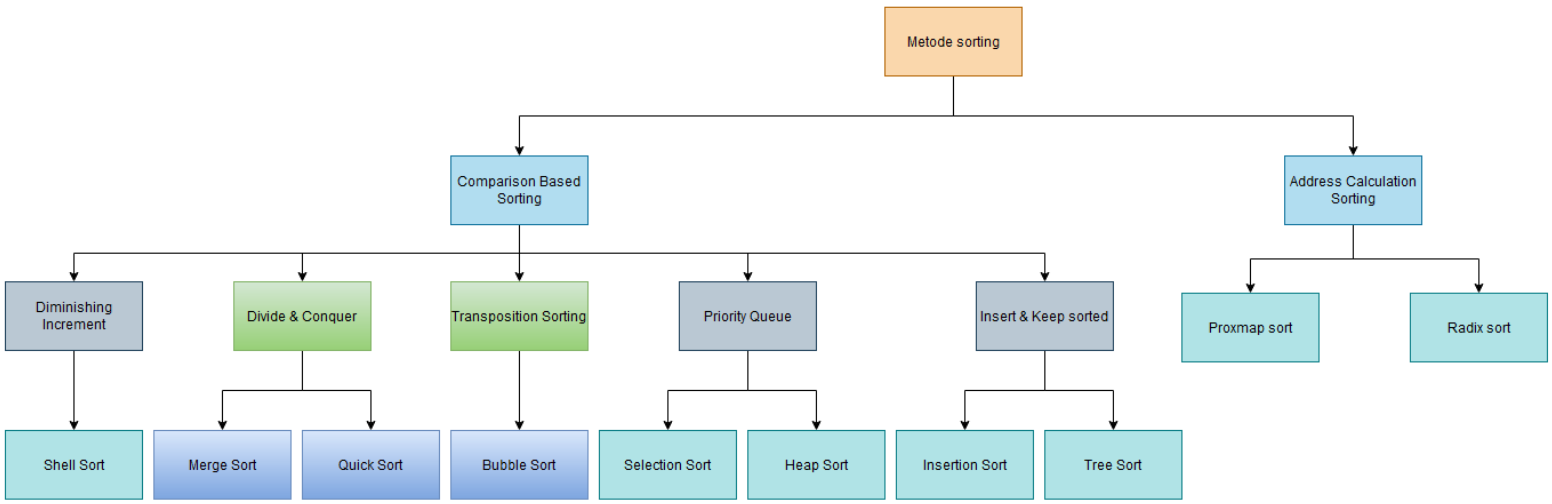
Dasar Teori

1. Pengertian
   Sorting merupakan metode untuk melakukan penyusunan data, dengan tujuan membuat data tersebut urut secara ascending (naik / kecil ke besar) atau descending (turun / besar ke kecil).

2. Syarat melakukan sorting
   a. Dalam kumpulan item atau objek yang ingin disorting harus memiliki kunci seperti contoh array of integer / list integer.
   b. Kunci tersebut dapat dibandingkan dengan kunci yang lain
   c. Dalam array kunci dapat berupa elemen dari array itu sendiri , jika dalam list berupa info elemen list
   d. Pengurutan dapat dilakukan secara ascending atau descending

Metode Sorting

1. Comparison based sorting
   a. Transposition Sorting (pertukaran)
      - Bubble sort
   b. Insert and Keep Sorted (penyisipan)
      - Insertion sort
      - Tree sort
   c. Priority Queue (antrian prioritas)
      - Selection sort
      - Heap sort
   d. Divide and Conquer (bagi dan urutkan)
      - Quick Sort
      - Merge Sort
   e. Diminishing Increment (penambahan menurun)
      - Shell Sort
2. Address Calculation Sorting
   - ProxmapSort
   - RadixSort


Metode sorting
Comparison Based Sorting
Address Calculation Sorting
Diminishing Increment
Divide & Conquer
Transposition Sorting
Priority Queue
Insert & Keep sorted
Proxmap sort
Radix sort
Shell Sort
Merge Sort
Quick Sort
Bubble Sort
Selection Sort
Heap Sort
Insertion Sort
Tree Sort

Guided

Guided hanya akan membahas 3 jenis sorting saja yaitu bubble, merge dan quick sort, teman teman dapat mencari jenis sorting yang lain pada sumber lainnya.

Header.h

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
typedef int infotype;
typedef int address;

typedef struct{
        address akhir;
        infotype t[MAX];
}array;

//==Fungsi Array==//
void createEmpty(array *a);
int isEmpty(array a);
void setArray(array *a, int n);
void printInfo(array a);
//===//

//==Bubble Sort==//
void bubbleSort(array *a);
void swap(int *a, int *b);
//===//

//==Merge Sort==//
void mergeSort(array *a);
void add(array *a, int x);
int length(array a);
array cloneArray(array a);
void partInto2(array *a, array *a2);
void merge(array *a, array t);
//===//

//==Quick Sort==//
void quickSort (array *a, int awal, int akhir);
//===//
```

Source.c

```
#include "header.h"


void createEmpty(array *a){
	a->akhir = -1;
}


int isEmpty(array a){
	return  a.akhir == -1;
}


void setArray(array *a, int n)
{
	int i;
	createEmpty(&(*a));
	for(i=0;i<n;i++)
	{
		printf("Masukkan data ke %d : ",i+1);
		scanf("%d",&(*a).t[i]);
	}
	(*a).akhir=n-1;
}


void printInfo(array a)
{
	int i;
	for(i=0;i<=a.akhir;i++)
	{
		printf("%d ",a.t[i]);
	}
}
```

```
void bubbleSort(array *a)
{
	int i,j;
	for(i=0;i<=(*a).akhir-1;i++)
	{
		for(j=i+1;j<=(*a).akhir;j++)
		{
			if((*a).t[i] > (*a).t[j])
			{
				swap(&(*a).t[i], &(*a).t[j]);
			}
		}
	}
}
void swap(int *a, int *b){
	int temp = *a;
	*a = *b;
	*b = temp;
}
void mergeSort(array *a)
{
	array t;
	if(length(*a)>1)
	{
		partInto2(&(*a),&t);
		mergeSort(&(*a));
		mergeSort(&t);
		merge(&(*a),t);
	}
}

void add(array *a, int x)
{
	(*a).akhir++;
	(*a).t [(*a).akhir] = x;
}
```

```
int length(array a)
{
	return(a.akhir+1);
}

array cloneArray(array a)
{
	int i;
	array temp;
	for(i=0;i<=a.akhir;i++){
		temp.t[i] = a.t[i];
	}
	temp.akhir = a.akhir;
	return temp;
}

void partInto2(array *a, array *a2)
{
	int i, len = length(*a);
	(*a).akhir = len / 2 + len % 2 - 1;
	(*a2).akhir = len - length(*a)-1;

	for(i=0;i<=(*a2).akhir;i++)
	{
		(*a2).t[i] = (*a).t[i+length(*a)];
	}
}

void merge(array *a, array t)
{
	array temp;
	createEmpty(&temp);

	int i=0,j=0,k;

	while(i<=(*a).akhir && j<=t.akhir)
	{

		if((*a).t[i] < t.t[j])
```

```
                {
                        add(&temp,(*a).t[i]);
                        i++;
                }
                else
                {
                        add(&temp,t.t[j]);
                        j++;
                }

        }
        if(i > (*a).akhir)
        {

                for(k=j;k<=t.akhir;k++)
                {
                        add(&temp,t.t[k]);
                }

        }
        else
        {

                for(k=i;k<=(*a).akhir;k++)
                {
                        add(&temp,(*a).t[k]);
                }

        }

        (*a)=cloneArray(temp);
}
```

```
void quickSort (array *a, int awal, int akhir)
{
        int i = awal, j = akhir;
        int pivot = (*a).t [(awal + akhir) / 2];

        while (i <= j)
        {
                while ((*a).t[i] < pivot)
                        i++;
                while ((*a).t[j] > pivot)
                        j--;
                if (i <= j)
                {
                        swap(&(*a).t[i], &(*a).t[j]);
                        i++; j--;
                }

        }

        if (awal < j)
                quickSort(&(*a), awal, j);
        if (i < akhir)
                quickSort(&(*a), i, akhir);

}
```

Main.c

```
#include "header.h"

int main(int argc, char *argv[]) {
	array a,temp;
	int jmlElemen, menu;
	do{
		system("cls");
		printf("\n[1]. Set Array");
		printf("\n[2]. Print Array");
		printf("\n[3]. Bubble Sort");
		printf("\n[4]. Merge Sort");
		printf("\n[5]. Quick Sort");
		printf("\n>> ");scanf("%d",&menu);

		switch(menu){
			case 1:
				printf("\n Jumlah Elemen Yang Ingin Ditampung :
");scanf("%d",&jmlElemen);
				setArray(&a, jmlElemen);
				break;

			case 2:
				printf("\nTampil Array Unsorted : ");
				printInfo(a);
				break;

			case 3:
				temp = cloneArray(a);
				printf("\nTampil Array Sebelum Sorting (Bubble Sort) : ");
				printInfo(temp);

				printf("\nTampil Array Sesudah Sorting (Bubble Sort) : ");
				bubbleSort(&temp);
				printInfo(temp);

				break;
```

```
                        case 4:
                                temp = cloneArray(a);
                                printf("\nTampil Array Sebelum Sorting (Merge Sort): ");
                                printInfo(temp);

                                printf("\nTampil Array Sesudah Sorting (Merge Sort) : ");
                                mergeSort(&temp);
                                printInfo(temp);
                                break;

                        case 5:
                                temp = cloneArray(a);
                                printf("\nTampil Array Sebelum Sorting (Quick Sort): ");
                                printInfo(temp);

                                printf("\nTampil Array Sesudah Sorting (Quick Sort) : ");
                                quickSort(&temp,0,temp.akhir);
                                printInfo(temp);
                                break;

                }getch();

        }while(menu!=0);
        return 0;
}
```

Format penamaan :

GD12_X_YYYY.zip

X = Kelas

Y = 4 digit NPM akhir