

GUIDED 12

SORTING

Sorting merupakan **metode pengurutan data**, dengan tujuan membuat data tersebut urut secara **ascending** (kecil ke besar) atau **descending** (besar ke kecil). Terdapat beberapa syarat untuk melakukan sorting sebagai berikut:

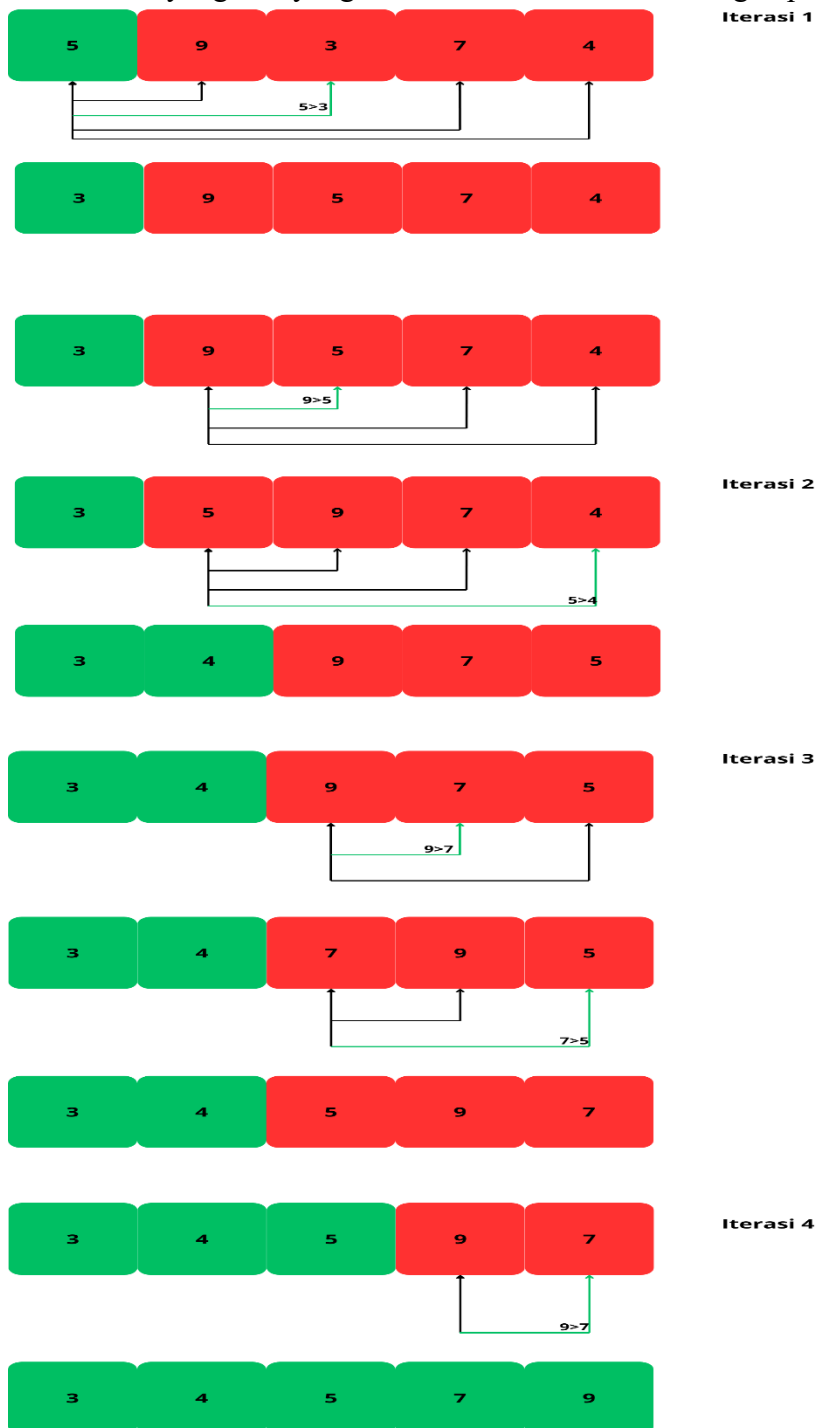
1. Dalam kumpulan item atau objek yang ingin disorting harus memiliki kunci seperti contoh array of integer / list integer
2. Kunci-kunci itu harus bisa diperbandingkan satu sama lain.
3. Pada array, kunci bisa berupa nilai elemen array itu sendiri dan pada List, kunci biasanya berupa info elemen list
4. Pengurutan dapat dilakukan secara ascending atau descending.

Metode Sorting

1. Comparison Based Sorting
 - a. Transposition Sorting (Pertukaran)
 - Bubble Sort
 - b. Insert and Keep Sorting (Penyisipan)
 - Insertion Sort
 - Tree Sort
 - c. Priority Queue (antrian prioritas)
 - Quick Sort
 - Merge Sort
 - d. Divide and Conquer (bagi dan urutkan)
 - Shell Sort
 - e. Diminishing Increment (penambahan menurun)
2. Address Calculation Sorting
 - a. ProxmapSort
 - b. RadixSort

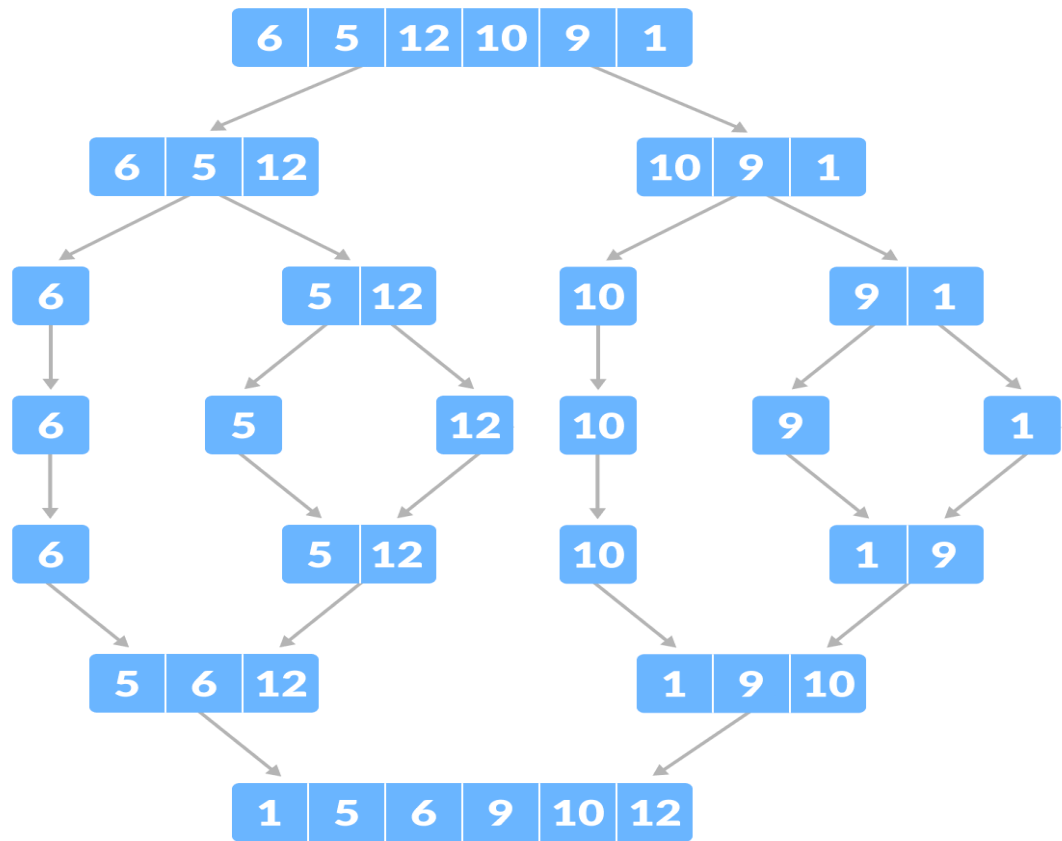
a. Bubble Sort

Bubble Sort merupakan pengurutan yang dilakukan dengan cara membandingkan data satu ke data yang lain yang setelah itu ditukar sesuai dengan pengurutan yang diminta.



b. Merge Sort

Merge sort merupakan metode yang menggunakan cara divide and conquer, dimana metode ini akan memecah data kemudian mengurutkan nya perbagian lalu digabung kembali.



c. Quick Sort

Quick Sort merupakan metode sorting dengan pendekatan rekursif. Dimana, proses pengurutan dilakukan dengan memecah data menjadi dua (sub array kiri dan sub array kanan) berdasarkan nilai pivot yang dipilih. Dimana nantinya sub array kiri merupakan data yang lebih kecil dari nilai pivot dan data kanan merupakan data dengan nilai yang lebih besar dari pivot.

1 12 5 26 7 14 3 7 2 unsorted

Diagram illustrating the initial state of the array [1, 12, 5, 26, 7, 14, 3, 7, 2] for the partitioning step. The pivot value is 7. The index i is 0, and the pivot value is 7.

1 12 5 26 7 14 3 7 2 $12 \geq 7 \geq 2$, swap 12 and 2

1 2 5 7 7 14 3 26 12

$7 \geq 7 \geq 3$, swap 7 and 3

1 2 5 7 3 14 7 26 12 $i > j$, stop partition

1 2 5 7 3 14 7 26 12 run quick sort recursively

■ ■ ■

1 2 3 5 7 7 12 14 26 sorted

Guided

Guided hanya akan membahas 3 jenis sorting yaitu bubble, merge dan quick sort yang diterapkan pada array. Untuk metode lainnya praktikan dapat mencari pada sumber lainnya.

Header.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#define MAX 5
typedef int address;
typedef struct{
    address akhir;
    int t[MAX];
}array;

//==Fungsi Array==//
void createEmpty(array *a);
int isEmpty(array a);
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 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 menu,i,bantu;
    createEmpty(&a);
    do{
        system("cls");
        printf("\n[1]. Input Array");
        printf("\n[2]. Print Array Unsorted");
        printf("\n[3]. Bubble Sort");
        printf("\n[4]. Merge Sort");
        printf("\n[5]. Quick Sort");
        printf("\n>> ");scanf("%d",&menu);
        switch(menu){
            case 1:
                for(i=0;i<MAX;i++){
                    printf("Masukkan data ke %d : ",i+1);scanf("%d",&bantu);
                    add(&a,bantu);
                }
                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_XXXXX.zip

X= Kelas

Y= 5 Digit NPM Akhir Praktikan

Hint: untuk UGD akan menggunakan srand, jadi dipersiapkan. Terima kasih, semangat teman-teman praktikan.