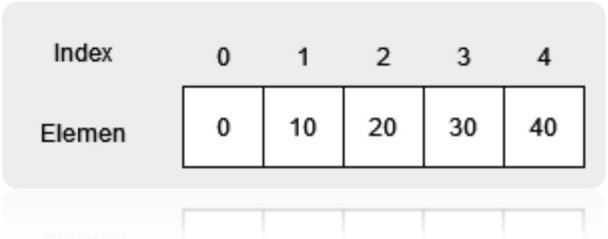
Modul 14 Searching

A. Pengertian

a. Searching merupakan proses pencarian data dalam sekumpulan data dengan menggunakan kunci data yang dapat diperbandingkan. Berikut adalah beberapa hal yang harus diperhatikan dalam proses searching.
   i. Data mempunyai kunci atau sesuatu yang dapat diperbandingkan dengan data yang lain.
   ii. Data tersimpan dalam array, baik yang sudah terurut maupun belum terurut.
   iii. Data yang dicari pada array dapat dicari atau tidak.

B. Sequential Search

Sebenarnya metode sequential search sudah sering kita lakukan dalam praktikum sebelumnya. Metode pencarian sequential adalah metode pencarian dimana kita akan memeriksa tiap-tiap elemen. Pemeriksaan ini akan terus dilakukan hingga kita menemukan elemen yang sama seperti yang kita cari atau hingga tidak ada elemen lagi yang bisa kita periksa.



Pada gambar diatas, jika kita ingin melakukan pencarian elemen dengan nilai 20 dalam array tersebut. Kita akan menggunakan perulangan untuk secara urut memeriksa nilai pada array dimulai dari index terkecil apakah sama dengan 20 atau tidak.

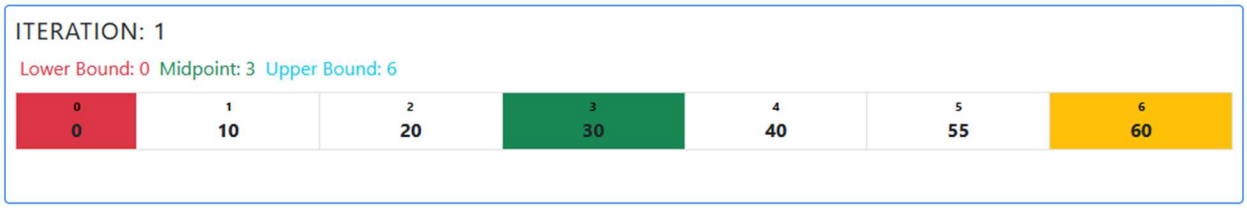
Metode pencarian sequential search lebih mudah diterapkan daripada metode pencarian binary seach dan sequential search tidak memerlukan data yang sudah terurut. Dibalik keuntungan yang ditawarkan oleh sequential search, metode ini memiliki kekurangan jika data yang perlu diperiksa adalah data yang sangat besar, karena metode ini harus melakukan kunjungan pada setiap elemen.

C. Binary Search

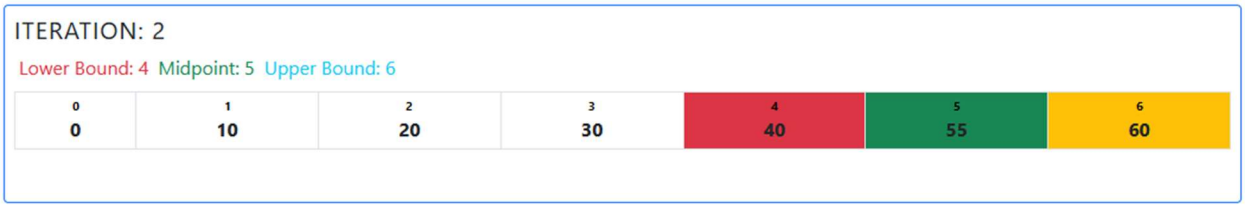
Metode pencarian binary search lebih cepat jika dibandingkan dengan sequential search. Tetapi dengan menggunakan metode ini kita diwajibkan untuk menerapkannya pada data yang sudah terurut. Metode pencarian binary seach bekerja dengan terus membagi jangkauan pencarian menjadi 2 bagian secara terus menerus hingga data ditemukan atau tidak ada lagi elemen untuk diperiksa.

Untuk memudahkan pemahaman perhatikan penggambaran binary seach berikut ini

1. Kita memiliki array dengan elemen seperti diatas, dan kita ingin mencari elemen istimewa dengan nilai **55**.



2. Pada iterasi pertama kita menetapkan index kiri(lower) dengan nilai 0, index kanan (upper) dengan 6, dan index tengah(mid) dengan 3. Penetapan ini bertujuan untuk mengurangi jangkauan pada tiap iterasinya. Kemudian dilakukan pemeriksaan apakah nilai pada index tengah sama dengan 55?



3. Karena nilai index tengah < dari 55, maka elemen 55 seharusnya berada pada sisi kanan mid. Pada iterasi kedua kita akan menetapkan index kiri dengan nilai mid +1 = 4, index kanan dengan nilai 6(tetap), dan index tengah dengan nilai (kanan + kiri)/2 . Kemudian dilakukan pemeriksaan apakah nilai pada index tengah sama dengan 55.

GUIDED

Format Penamaan GD13_X_YYYYY (X = Kelas Y = NPM).

Header.h

```h
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

#define max 10

int isEmpty(int array[max]);
int isFull(int array[max]);
void createEmpty(int array[max]);
void tukar(int *a, int *b);
void bubbleSort(int array[max]);
int binarySearch(int array[max], int kiri, int kanan, int cari);
int sequentialSearch(int array[max], int cari);
void print(int array[max]);
void copyArray(int a[max], int b[max]);
int cekUnik(int array[max], int cek);
void insertRandom(int array[max]);
void invalid();
```

Source.c

```c
#include "header.h"

int isEmpty(int array[max]){
    int i = 0;
    for(i=0;i<max;i++){
        if(array[i]!=-1){
            return 0;
        }
    }
    return 1;
}
int isFull(int array[max]){
    int i = 0;
    for(i=0;i<max;i++){
        if(array[i]==-1){
            return 0;
        }
    }
    return 1;
}
void createEmpty(int array[max]){
    int i = 0;
    for(i=0;i<max;i++){
        array[i]=-1;
    }
}
void tukar(int *a, int *b){
    int temp = (*a);
    (*a) = (*b);
    (*b) = temp;

}
void bubbleSort(int array[max]){
    int i = 0,j =0;
    for(i=0;i<max;i++){
        for(j=i+1;j<max;j++){
            if(array[i]>array[j]){
                tukar(&array[i],&array[j]);
            }
        }
    }
}
```

```c
int binarySearch(int array[max], int kiri, int kanan, int cari){
    if(kanan >= kiri){
        int mid = (kiri+kanan)/2;
        printf("\nkiri  mid kanan");
        printf("\n%d(%d)    %d(%d)  %d(%d)\n",kiri,array[kiri],mid,array[mid],kanan,array[kanan]);
        if(cari==array[mid]){
            printf("\n data ditemukan\n");
            return mid;
        }else if(cari < array[mid]){
            printf("\nmencari disebelah kiri mid\n");
            return binarySearch(array,kiri,mid-1,cari);
        }else{
            printf("\nmencari disebelah kanan mid\n");
            return binarySearch(array,mid+1,kanan,cari);
        }

    }
    return -1;
}
int sequentialSearch(int array[max], int cari){
    int i = 0;
    for(i=0;i<max;i++){
        if(array[i]==cari){
            return i;
        }
    }
    return -1;
}
void print(int array[max]){
    int i = 0;
    printf("\n Posisi   Isi");
    for(i=0;i<max;i++){
        printf("\n %d   %d",i,array[i]);
    }
}
void copyArray(int a[max], int b[max]){
    int i = 0;
    for(i=0;i<max;i++){
        b[i] = a[i];
    }
}
int cekUnik(int array[max], int cek){
    int i =0;
    for(i=0;i<max;i++){
        if(array[i]==cek){
            return 0;
        }
    }
    return 1;
}
void insertRandom(int array[max]){
    int i =0, num =0;
    srand((unsigned)time(NULL));
    for(i=0;i<max;i++){
        num = rand()% 70 + 1;
        if(cekUnik(array,num)){
            array[i]=num;
        }
        else i--;
    }
}
void invalid(){
    printf("\n[!] INPUT NGUWAWOR    [!]");
}
```

Main.c

```c
#include "header.h"
/* run this program using the console pauser or add your own getch, system("pause") or input loop */

int main(int argc, char *argv[]) {
    int array[10],sortedArray[10];
    int menu = 0;
    int cari,found;
    //printf("\n");
    createEmpty(array);
    do{
        system("cls");
        printf("\t\t GUIDED Searching \n");
        printf("\n[1] Generate Number");
        printf("\n[2] Sequential Search");
        printf("\n[3] Binary Search");
        printf("\n[0] End Program");
        printf("\n>>>");scanf("%d",&menu);

        switch(menu){
            case 1:{
                if(isFull(array)){
                    createEmpty(array);
                }

                insertRandom(array);
                printf("\nHasil Generate Aray :");
                print(array);
                break;
            }
            case 2:{
                if(isEmpty(array)){
                    printf("\n[!] Nguwawor Array Masih Kosong   [!]");
                    break;
                }

                printf("\nData Array :");
                print(array);

                printf("\nData yang ingin dicari :");scanf("%d",&cari);
                found = sequentialSearch(array,cari);
                if(found==-1){
                    printf("\n[!] Nguwawor Data Tidak Ada   [!]");
                }else{
                    printf("\nData (%d) ada pada index %d", array[found],found);
                }
                break;
            }
            case 3:{
                if(isEmpty(array)){
                    printf("\n[!] Nguwawor Array Masih Kosong   [!]");
                    break;
                }

                printf("\n Array Sebelum Sorting :");print(array);
                copyArray(array, sortedArray);

                bubbleSort(sortedArray);
                printf("\n Array Sesudah Sorting :");print(sortedArray);

                printf("\nData yang ingin dicari :");scanf("%d",&cari);
                found = binarySearch(sortedArray, 0, max - 1, cari);

                if(found==-1){
                    printf("\n[!] Nguwawor Data Tidak Ada   [!]");
                }else{
                    printf("\nData (%d) ada pada index %d", sortedArray[found],found);
                }
                break;
            }
            case 0:{
                printf(" Nama | NPM | Kelas");
                break;
            }
            default:{
                invalid();
                break;
            }
        }

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