ALGORITMA DAN STRUKTUR DATA
SEARCHING

Dr.Arna Fariza,S.Kom,M.Kom

Disusun Oleh :

Nama : Ghaly Abrarian Putra

Kelas : D3 IT A

NRP : 3123500018

Mei 2024

Praktikum 13

Algoritma Pencarian (Searching)

A. TUJUAN PEMBELAJARAN

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

1. Memahami konsep pencarian dengan metode sequential search dan binary search.
2. Mengimplementasikan algoritma sequential search dan binary search dalam bentuk flowchart.
3. Membuat diagram alir dan mengimplementasikan algoritma pada suatu permasalahan.

B. DASAR TEORI

Algoritma pencarian (*searching algorithm*) adalah algoritma yang menerima sebuah argumen kunci dan dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (*successful*) atau tidak ditemukan (*unsuccessful*).

Ada dua macam teknik pencarian yaitu pencarian sekuensial (*sequential search*) dan pencarian biner (*binary search*). Perbedaan dari dua teknik ini terletak pada keadaan data. Pencarian sekuensial digunakan apabila data dalam keadaan acak atau tidak terurut. Sebaliknya, pencarian biner digunakan pada data yang sudah dalam keadaan urut.

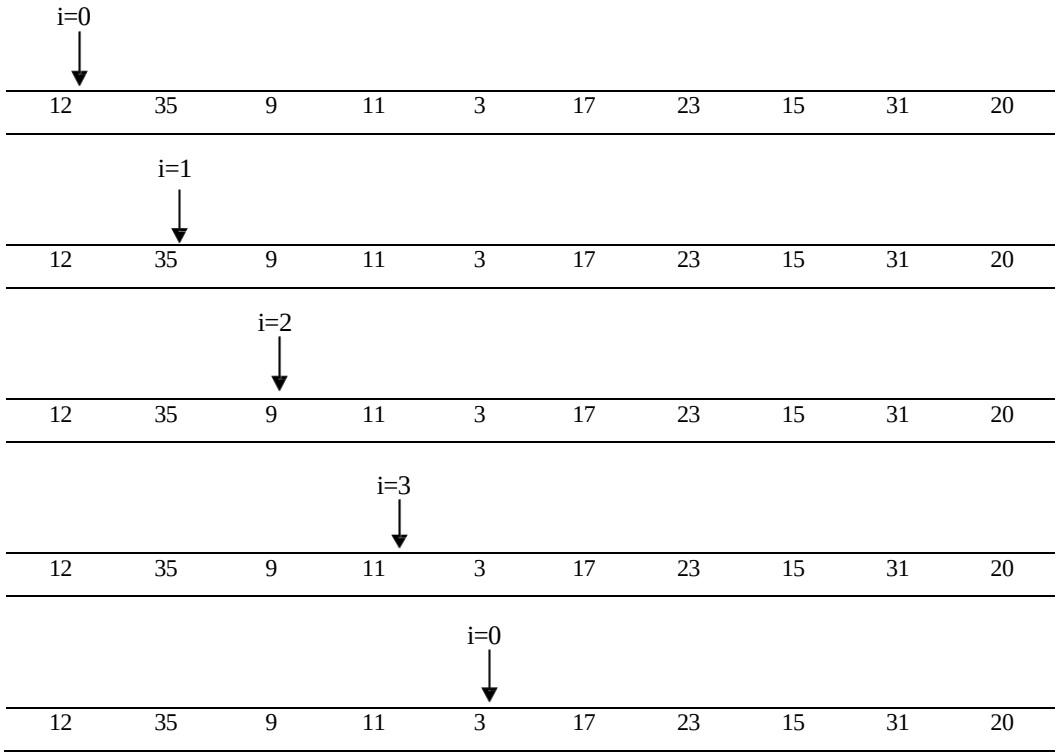
B.1 Pencarian Berurutan (*Sequential Search*)

Algoritma pencarian dapat dijelaskan sebagai berikut : pencarian dimulai dari data paling awal, kemudian ditelusuri dengan menaikkan indeks data, apabila data sama dengan kunci pencarian dihentikan dan diberikan nilai pengembalian true,

apabila sampai indeks terakhir data tidak ditemukan maka diberikan nilai pengembalian false.

Ilustrasi dari algoritma pencarian biner adalah sebagai berikut :

Kunci=3



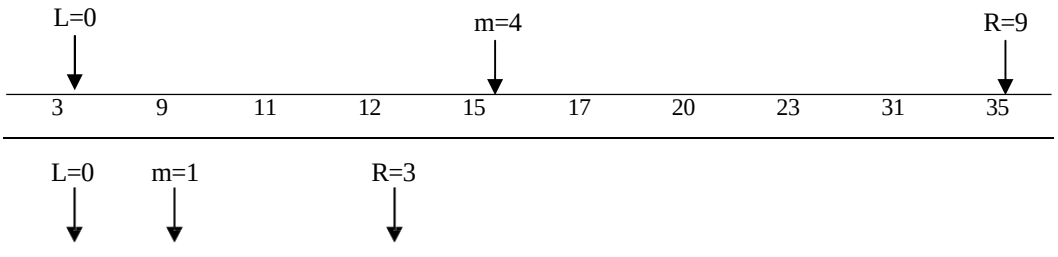
`Data[4]`=3 sama dengan kunci=3 maka data ditemukan dan diberikan nilai pengembalian i (posisi) dan proses dihentikan. Apabila data tidak ditemukan, maka fungsi akan mengembalikan nilai -1

B.2 Pencarian Biner (*Binary Search*)

Salah satu syarat agar pencarian biner dapat dilakukan adalah data sudah dalam keadaan urut. Dengan kata lain, apabila data belum dalam keadaan urut, pencarian biner tidak dapat dilakukan.

Ilustrasi dari algoritma pencarian biner adalah sebagai berikut :

Kunci=3



3	9	11	12	15	17	20	23	31	35

L=0;m=0;R=0

3	9	11	12	15	17	20	23	31	35

`Data[1]`=3 sama dengan kunci=3 maka data ditemukan dan diberikan nilai pengembalian i (posisi) dan proses dihentikan. Apabila data tidak ditemukan, maka fungsi akan mengembalikan nilai -1.

C. PERCOBAAN

1. Buatlah workspace menggunakan Visual C++.
2. Buatlah project baru SEARCHING yang berisi file *C source* untuk metode pencarian *sequential search* dan *binary search.*
3. Cobalah untuk masing-masing percobaan di bawah dengan menambahkan menu pilihan metode pencarian pada program utama.

Percobaan 1 : Implementasi pencarian dengan metode *sequential search*

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

#define     MAX   10

int Data[MAX];

int SequentialSearch(int x)
{
      int i = 0;
      bool ketemu = false;

      while ((!ketemu) && (i < MAX)){
            if(Data[i] == x)
                  ketemu = true;
            else
                  i++;
      }
      if(ketemu)
            return i;
      else
            return -1;
}

void main()
{
   int i;

   //pembangkit bilangan random
   srand(0);

   //membangkitkan bilangan integer random
   printf("\nDATA : ");
   for (i = 0; i < MAX; i++)
   {
       Data[i] = rand()/1000+1;
         printf("%d ", Data[i]);
   }

   int Kunci;

   printf("\nKunci : ");
   scanf("%d", &Kunci);

   int ketemu = SequentialSearch(Kunci);
   if(ketemu>0)
      printf("Data ditemukan pada posisi %d", ketemu);
   else
      printf("Data tidak ditemukan");
}
```

Percobaan 2 : Implementasi pencarian dengan metode *binary seach*

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

#define     MAX   10
```

```
int Data[MAX];

// Prosedur menukar data

void Tukar (int *a, int *b)
{
      int temp;
      temp = *a;
      *a = *b;
      *b = temp;
}
// Prosedur pengurutan metode Quick Sort

void QuickSort(int L, int R)
{
      int i, j, x;
      x = Data[(L+R)/2];
      i = L;
      j = R;
      while (i <= j){
            while(Data[i] < x)
                  i++;
            while(Data[j] > x)
                  j--;
            if(i <= j){
                  Tukar(&Data[i], &Data[j]);
                  i++;
                  j--;
            }
      }
      if(L < j)
            QuickSort(L, j);
      if(i < R)
            QuickSort(i, R);
}

// Fungsi pencarian biner

int BinarySearch(int x)
{
      int L = 0, R = MAX-1, m;
      bool ketemu = false;

      while((L <= R) && (!ketemu))
      {
            m = (L + R) / 2;
            if(Data[m] == x)
                  ketemu = true;
            else if (x < Data[m])
                  R = m – 1;
            else
                  L = m + 1;
      }
      if(ketemu)
            return m;
      else
            return -1;
}
```

```
void main()
{
   int i;

   //pembangkit bilangan random
   srand(0);

   //membangkitkan bilangan integer random
   printf("\nDATA : ");
   for (i = 0; i < MAX; i++)
   {
       Data[i] = rand()/1000+1;
         printf("%d ", Data[i]);
   }

   //mengurutkan data
   QuickSort(0, MAX-1);

   int Kunci;
   printf("\nKunci : ");
   scanf("%d", &Kunci);

   int ketemu = BinarySearch(Kunci);
   if(ketemu>0)
      printf("Data ditemukan pada posisi %d", ketemu);
   else
      printf("Data tidak ditemukan");
}
```

D. LAPORAN RESMI

1. Tugas pada praktikum ini tambahkan pada program yang sudah dibuat fungsi pencarian sequential sorted, fungsi rekursi pencarian sequential, dan fungsi rekursi pencarian binary. Tuliskan source code dan hasil running program, lalu buat analisis. Dan berikan kesimpulan hasil praktikum

   Source Code :

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

#define MAX 10

int Data[MAX];

void Tukar(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

void SequentialSort()
{
    int i, j;
    for (i = 0; i < MAX - 1; i++)
    {
        for (j = i + 1; j < MAX; j++)
        {
            if (Data[i] > Data[j])
            {
                Tukar(&Data[i], &Data[j]);
            }
        }
    }
}

int SequentialSearchSorted(int x)
{
    int i = 0;
    while (i < MAX && Data[i] <= x)
    {
        if (Data[i] == x)
            return i;
```

```
        i++;
    }
    return -1;
}

int SequentialSearch(int x, int index)
{
    if (index == MAX)
        return -1;
    if (Data[index] == x)
        return index;
    return SequentialSearch(x, index + 1);
}

int BinarySearch(int x, int L, int R)
{
    if (L > R)
        return -1;
    int m = (L + R) / 2;
    if (Data[m] == x)
        return m;
    else if (x < Data[m])
        return BinarySearch(x, L, m - 1);
    else
        return BinarySearch(x, m + 1, R);
}

int main()
{
    int Kunci, n, ketemu, i;
    char lagi = 'y';
     while (lagi == 'y')
    {
    srand(0);
    printf("\nDATA SEBELUM DIURUTKAN: ");
    for (i = 0; i < MAX; i++)
    {
        Data[i] = rand() % 100 + 1;
        printf("%d ", Data[i]);
    }

    SequentialSort();
    printf("\nDATA SETELAH DIURUTKAN SECARA SEKUENSIAL: ");
    for (i = 0; i < MAX; i++)
    {
        printf("%d ", Data[i]);
    }
```

```
    printf("\nKunci : ");
    scanf("%d", &Kunci);

       printf("Pilih Algoritma:\n");
        printf("1. Sequential Search\n");
        printf("2. Binary Search\n");
        printf("3. Sequential Search Sorted\n");
        printf("Masukan pilihan: ");
        scanf("%d", &n);

        switch (n)
        {
        case 1:
            ketemu = SequentialSearch(Kunci, 0);
            if (ketemu >= 0)
                printf("Data ditemukan pada index %d\n", ketemu);
            else
                printf("Data tidak ditemukan\n");
            break;
        case 2:
            ketemu = BinarySearch(Kunci, 0, MAX - 1);
            if (ketemu >= 0)
                printf("Data ditemukan pada posisi %d", ketemu);
            else
                printf("Data tidak ditemukan");
            break;
        case 3:
            ketemu = SequentialSearchSorted(Kunci);
            if (ketemu >= 0)
                printf("Data ditemukan pada posisi %d", ketemu);
            else
                printf("Data tidak ditemukan");
            break;

        default:
            printf("Pilihan tidak valid\n");
            continue;
        }
        printf("\nlagi ta? ");
        getchar();
        scanf("%c", &lagi);
    }
    printf("Anda Keluar dari program");

    return 0;
}
```

Output program :

```
DATA SEBELUM DIURUTKAN: 31 92 16 73 62 42 11 38 99 42
DATA SETELAH DIURUTKAN SECARA SEKUENSIAL: 11 16 31 38 42 42 62 73 92 99
Kunci : 31
Pilih Algoritma:
1. Sequential Search
2. Binary Search
3. Sequential Search Sorted
Masukan pilihan: 1
Data ditemukan pada index 2

lagi ta? y

DATA SEBELUM DIURUTKAN: 31 92 16 73 62 42 11 38 99 42
DATA SETELAH DIURUTKAN SECARA SEKUENSIAL: 11 16 31 38 42 42 62 73 92 99
Kunci : 31
Pilih Algoritma:
1. Sequential Search
2. Binary Search
3. Sequential Search Sorted
Masukan pilihan: 2
Data ditemukan pada posisi 2
lagi ta? y

DATA SEBELUM DIURUTKAN: 31 92 16 73 62 42 11 38 99 42
DATA SETELAH DIURUTKAN SECARA SEKUENSIAL: 11 16 31 38 42 42 62 73 92 99
Kunci : 31
Pilih Algoritma:
1. Sequential Search
2. Binary Search
3. Sequential Search Sorted
Masukan pilihan: 3
Data ditemukan pada posisi 2
lagi ta? t
Anda Keluar dari program%
```

Analisa :

Program ini mengimplementasikan tiga algoritma pencarian, yaitu pencarian sekuensial, pencarian biner, dan pencarian sekuensial pada data yang sudah diurutkan. Selain itu, terdapat juga algoritma pengurutan sekuensial untuk mengurutkan data sebelum dilakukan pencarian. Setelah mengurutkan data, user diminta untuk memasukkan kunci pencari (anangka yang ingin dicari), lalu memilih algoritma pencarian yang diinginkan dimana
1. Sequential Search,
2. Binary Search,
3. Sequential Search Sorted.

Seluruh proses tersebut terjadi di dalam sebuah loop, yang memungkinkan pengguna untuk melakukan pencarian berulang jika diinginkan.

E. KESIMPULAN

Ada dua jenis utama algoritma pencarian: sekuensial dan biner.

Sekuensial cocok untuk data yang tidak terurut, sementara biner cocok untuk data yang urut. Pilihan algoritma tergantung pada kondisi data dan efisiensi yang dibutuhkan.