



Struktur Data
Pendidikan Teknik Informatika
Universitas Negeri Padang
TIM DOSEN ©2022

JOBSHEET (JS-12)

Searching

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa mampu mengaplikasikan konsep Linear Search dalam menyelesaikan kasus.
2. Mahasiswa mampu mengaplikasikan konsep Binary Search dalam menyelesaikan kasus.

HARDWARE & SOFTWARE

1. Personal Computer
2. IDE: Dev C++

TEORI SINGKAT

A. Linear Search

Pencarian adalah proses menemukan beberapa elemen tertentu dalam daftar. Jika elemen ada dalam daftar, maka prosesnya disebut berhasil, dan proses mengembalikan lokasi elemen itu; jika tidak, pencarian disebut tidak berhasil.

Pencarian linier juga disebut sebagai algoritma pencarian sekuensial. Ini adalah algoritma pencarian yang paling sederhana. Dalam pencarian Linear, kita cukup menelusuri daftar sepenuhnya dan mencocokkan setiap elemen daftar dengan item yang lokasinya dapat ditemukan. Jika kecocokan ditemukan, maka lokasi item dikembalikan; jika tidak, algoritme mengembalikan NULL.

Langkah-langkah yang digunakan dalam implementasi Linear Search adalah sebagai berikut :

1. Pertama, kita harus melintasi elemen array menggunakan for loop.
2. Dalam setiap iterasi for loop, bandingkan elemen pencarian dengan elemen array saat ini, dan jika elemen cocok, maka kembalikan indeks elemen array yang sesuai.
3. Jika elemen tidak cocok, maka pindah ke elemen berikutnya.
4. Jika tidak ada kecocokan atau elemen pencarian tidak ada dalam larik yang diberikan, kembalikan -1.

Untuk memahami cara kerja algoritma pencarian linier, mari kita ambil array yang tidak disortir. Akan mudah untuk memahami cara kerja pencarian linier dengan sebuah contoh.

Misalkan elemen array adalah :

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	52

Misalkan elemen yang akan dicari adalah $K = 41$

Sekarang, mulai dari elemen pertama dan bandingkan K dengan setiap elemen array.

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	52

↑
 $K \neq 70$

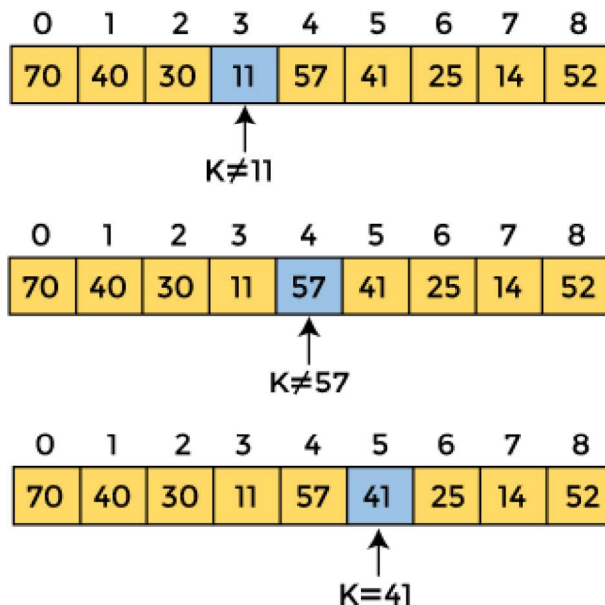
Nilai K , yaitu, 41, tidak cocok dengan elemen pertama larik. Jadi, pindah ke elemen berikutnya. Dan ikuti proses yang sama sampai elemen masing-masing ditemukan.

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	52

↑
 $K \neq 40$

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	52

↑
 $K \neq 30$



Sekarang, elemen yang akan dicari telah ditemukan. Jadi algoritma akan mengembalikan indeks elemen yang cocok.

B. Binary Search

Pencarian biner adalah teknik pencarian yang bekerja secara efisien pada daftar yang diurutkan. Oleh karena itu, untuk mencari elemen ke dalam beberapa daftar menggunakan teknik pencarian biner, kita harus memastikan bahwa daftar tersebut diurutkan.

Pencarian biner mengikuti pendekatan membagi dan menaklukkan di mana daftar dibagi menjadi dua bagian, dan item dibandingkan dengan elemen tengah daftar. Jika kecocokan ditemukan, lokasi elemen tengah dikembalikan. Jika tidak, kami mencari ke salah satu bagian tergantung pada hasil yang dihasilkan melalui pertandingan.

CATATAN: Pencarian biner dapat diimplementasikan pada elemen array yang diurutkan. Jika elemen daftar tidak diatur dengan cara yang diurutkan, kita harus mengurutkannya terlebih dahulu.

Untuk memahami cara kerja algoritma pencarian Biner, mari kita ambil array yang diurutkan. Akan mudah untuk memahami cara kerja pencarian Biner dengan sebuah

contoh.

Misalkan elemen array adalah :

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

Misalkan elemen yang dicari adalah, $K = 56$

Kita harus menggunakan rumus di bawah ini untuk menghitung pertengahan array :

$$\text{pertengahan (mid)} = (\text{awal} + \text{akhir}) / 2$$

Jadi, dalam array yang diberikan :

$$\text{awal} = 0$$

$$\text{akhir} = 8$$

$$\text{mid} = (0 + 8) / 2 = 4. \text{ Jadi, 4 adalah pertengahan array.}$$

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69



$$A[\text{mid}] = 39$$

$$A[\text{mid}] < K \text{ (or, } 39 < 56)$$

$$\text{So, beg} = \text{mid} + 1 = 5, \text{ end} = 8$$

$$\text{Now, mid} = (\text{beg} + \text{end}) / 2 = 13 / 2 = 6$$

0	1	2	3	4	5	6	7	8
10	12	24	29	39	40	51	56	69

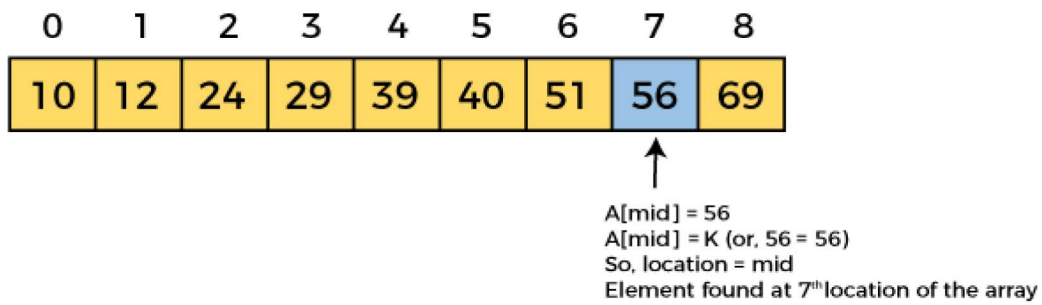


$$A[\text{mid}] = 51$$

$$A[\text{mid}] < K \text{ (or, } 51 < 56)$$

$$\text{So, beg} = \text{mid} + 1 = 7, \text{ end} = 8$$

$$\text{Now, mid} = (\text{beg} + \text{end}) / 2 = 15 / 2 = 7$$



Sekarang, elemen untuk mencari ditemukan. Jadi algoritma akan mengembalikan indeks elemen yang cocok.

LATIHAN

1. Linear Search

```

/* Nama File : linear search 1
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

// linear search

#include <stdio.h>

int linearSearch(int a[], int n, int val) {
    // Going through array sequentially
    for (int i = 0; i < n; i++)
    {
        if (a[i] == val)
            return i+1;
    }
    return -1;
}

int main() {
    int a[] = {70, 40, 30, 11, 57, 41, 25, 14, 52}; // given
    array
    int val = 41; // value to be searched
    int n = sizeof(a) / sizeof(a[0]); // size of array
  
```

```
int res = linearSearch(a, n, val); // Store result

printf("The elements of the array are - ");
for (int i = 0; i < n; i++)
    printf("%d ", a[i]);
printf("\nElement to be searched is - %d", val);
if (res == -1)
    printf("\nElement is not present in the array");
else
    printf("\nElement is present at %d position of array",
res);
return 0;
}
```

2. Linear Search

```
/* Nama File   : linear search 2
Pembuat       : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

// linear search

#include <stdio.h>

int search(int array[], int n, int x) {

    // Going through array sequentially
    for (int i = 0; i < n; i++)
        if (array[i] == x)
            return i;
    return -1;
}

int main() {
    int array[] = {2, 4, 0, 1, 9};
    int x = 1;
    int n = sizeof(array) / sizeof(array[0]);
```



```

    int result = search(array, n, x);

    (result == -1) ? printf("Element not found") :
printf("Element found at index: %d", result);
}

```

3. Binary Search

```

/* Nama File   : binary search 1
Pembuat       : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

// binary search

#include <stdio.h>
int binarySearch(int a[], int beg, int end, int val)
{
    int mid;
    if(end >= beg)
    {
        mid = (beg + end)/2;
/* if the item to be searched is present at middle */
        if(a[mid] == val)
        {
            return mid+1;
        }

        /* if the item to be searched is smaller than
middle, then it can only be in left subarray */
        else if(a[mid] < val)
        {
            return binarySearch(a, mid+1, end, val);
        }

        /* if the item to be searched is greater than
middle, then it can only be in right subarray */
        else
        {
            return binarySearch(a, beg, mid-1, val);
        }
    }
}

```

```

        }
    }
    return -1;
}

int main() {
    int a[] = {11, 14, 25, 30, 40, 41, 52, 57, 70}; // given
array
    int val = 40; // value to be searched
    int n = sizeof(a) / sizeof(a[0]); // size of array
    int res = binarySearch(a, 0, n-1, val); // Store result

    printf("The elements of the array are - ");
    for (int i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\nElement to be searched is - %d", val);
    if (res == -1)
        printf("\nElement is not present in the array");
    else
        printf("\nElement is present at %d position of array",
res);
    return 0;
}

```

4. Binary Search

```

/* Nama File   : binary search 2
Pembuat       : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

// binary search

#include <stdio.h>

int binarySearch(int array[], int x, int low, int high) {
    // Repeat until the pointers low and high meet each other
    while (low <= high) {
        int mid = low + (high - low) / 2;

```



```

        if (array[mid] == x)
            return mid;

        if (array[mid] < x)
            low = mid + 1;

        else
            high = mid - 1;
    }

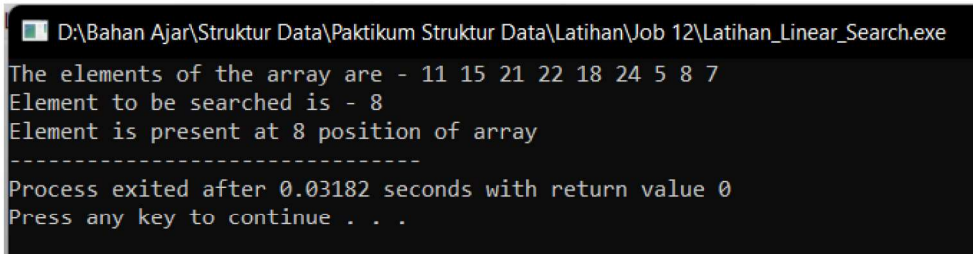
    return -1;
}

int main(void) {
    int array[] = {3, 4, 5, 6, 7, 8, 9};
    int n = sizeof(array) / sizeof(array[0]);
    int x = 4;
    int result = binarySearch(array, x, 0, n - 1);
    if (result == -1)
        printf("Not found");
    else
        printf("Element is found at index %d", result);
    return 0;
}

```

TUGAS

1. Pahami dan pelajari program Linear Search pada Latihan, setelah itu buatlah program baru yang outputnya seperti berikut :



```

D:\Bahan Ajar\Struktur Data\Paktikum Struktur Data\Latihan\Job 12\Latihan_Linear_Search.exe
The elements of the array are - 11 15 21 22 18 24 5 8 7
Element to be searched is - 8
Element is present at 8 position of array
-----
Process exited after 0.03182 seconds with return value 0
Press any key to continue . . .

```

2. Pahami dan pelajari program Binary Search pada Latihan, setelah itu buatlah program baru yang outputnya seperti berikut :

```
D:\Bahan Ajar\Struktur Data\Paktikum Struktur Data\Latihan\Job 12\Latihan_Binary_Search.exe
The elements of the array are - 5 7 8 16 21 80 87 89 100
Element to be searched is - 100
Element is present at 9 position of array
-----
Process exited after 0.02206 seconds with return value 0
Press any key to continue . . .
```

DAFTAR PUSTAKA

- Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Weasley.
- Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*. Bandung : Unikom.
- Liem, Inggriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.
- Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.
- Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com