SOFIA UMAROH

Algoritma Pencarian (Searching)

ISA 105 Algoritma dan Pemrograman

Sofia Umaroh, S.Pd., M.T

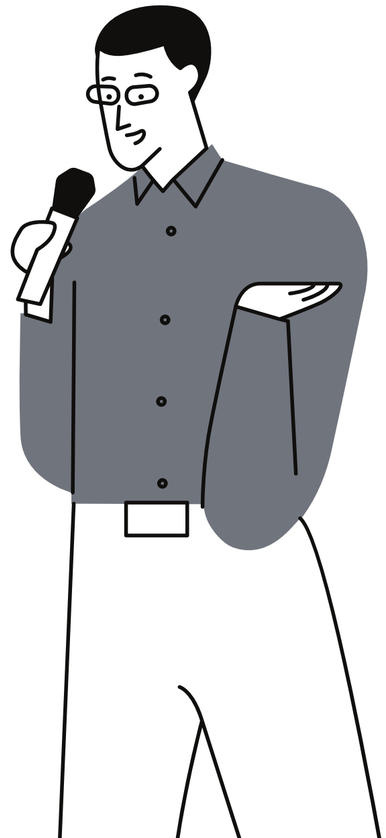
Agenda Hari Ini

1. Definisi Pencarian
2. Metode Pencarian
3. Contoh pencarian dalam Array
4. Latihan

SOFIA **UMAROH**

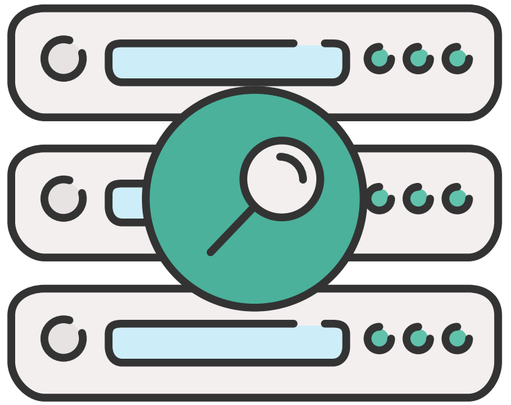
Memahami

Definisi Pencarian



Pencarian (Search)

- Pencarian pada sekumpulan data merupakan proses yang penting pada suatu program
- Pertimbangan memilih algoritme pencarian adalah **“Seberapa cepat pencarian harus dilakukan?”**
- Decara umum, semakin cepat proses pencarian, maka semakin kompleks algoritmanya.



Metode Pencarian:

1. **Sequential (linear) search**
2. **Binary search**

Sequential (Linear) Search

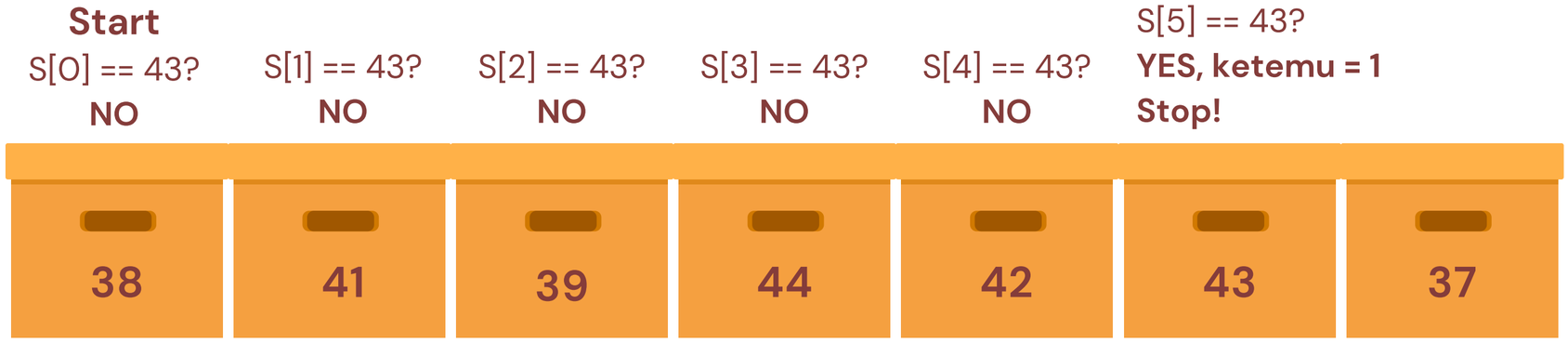
- **Memeriksa setiap elemen** dari daftar array data sampai ditemukan kecocokan
- Dapat digunakan untuk mencari daftar **yang tidak berurutan**
- Bagaimana jika data didistribusikan secara acak?

- **Rata-rata kasus** diperlukan pembandingan sebanyak N/2
- **Kasus terbaik** nilainya sama dengan elemen pertama yang diuji
- **Kasus terburuk** nilai **tidak ada** dalam daftar N pada setiap pembandingan

Sequential (Linear) Search

Misalnya Anda diminta untuk mencari sepatu dengan ukuran 43 di dalam kotak yang tertutup

Set var **ketemu = 0** (Asumsi data belum ditemukan)
Pencarian berhenti apabila **ketemu = 1**



Ingat! Anda tidak pernah tau isi setiap lokernya

Sepatu ukuran 43 ditemukan di indeks ke-5

Sequential (Linear) Search

- Misalnya Anda diminta untuk mencari apakah ada **Anton** pada barisan mahasiswa? ada di urutan berapa? Bagaimana caranya?
- Cara paling sederhana adalah dengan melihat wajah mahasiswa satu persatu secara berurutan

Pencarian di Bahasa C

Siapkan variabel status pencarian, set dengan 0, artinya data belum ditemukan. Misalnya:

```
int ketemu = 0
```

Ketemu akan diubah nilainya **menjadi 1** apabila data yang dibandingan **Sama**

Algortima:

```
i <- 1
ketemu <- 0

while (ketemu == 0) and (i < N) do
   if (Data[i] == key) then
      ketemu <- 1
   else
      i <- i + 1
endwhile

if (ketemu) then
   output ( i ," is index of data")
else
   output ("data not found")
endif
```

Linear Search

Langkah 1: Deklarasi var ketemu, array data, dan counter

```
int main(){
    int ketemu,i;
    int sepatu[]={38,41,39,44,42,43,37};
 ...
}
```

Langkah 2: Asumsi data belum ditemukan, maka inisialisasi ketemu = 0

```
int main(){
    ketemu = 0;
}
```

```
> Sepatu ukuran 43 ditemukan di indeks ke-4
> Program ended with exit code: 0
```

Langkah 3: lakukan pengulangan dengan syarat ketemu masih 0, dan jumlah pengulangan tidak kurang dari N

```
int main() {
    ...
    i=0;   // Pengulangan
    while (ketemu==0 && i<7) {
        if(sepatu[i] == 43){
            ketemu = 1;
        }else{
            i++;
        }
    }
    if (ketemu==1) {
        printf("Sepatu ukuran 43
ditemukan di indeks ke-%d\n",i-1);
    }else{
        printf("Tidak ditemukan");
    }
  return 0;
}
```

Fungsi Linear Search menggunakan For

```
#include <stdio.h>
#define SIZE 100

/* function prototype */
int linearSearch( const int array[], int key, int size );

int main(){
    int a[ SIZE ]; /* membuat array a */
    int x; /* counter */
    int searchKey; /* kunci pencarian */
    int elemen; /* variable untuk menyimpan lokasi searchKey */

    /* mengisi data array */
    for ( x = 0; x < SIZE; x++ ) {
        a[x] = 2 * x;
    }
    printf( "Masukan angka yang dicari:\n" );
    scanf( "%d", &searchKey );
    ...
```

Fungsi Linear Search menggunakan For

```
...
int main(){
    ...
    printf( "Masukan angka yang dicari: " );
    scanf( "%d", &searchKey );

    /* melakukan pencarian melalui fungsi linearSearch */
    element = linearSearch( a, searchKey, SIZE )

    /* Menampilkan hasil */
    if ( element != -1 ) {
        printf( "Angka ditemukan di elemen ke-%d\n", elemen );
    } else {
        printf( "Nilai tidak ditemukan\n" );
    } /* endif */

    return 0;

}/* end main */
```

Fungsi Linear Search menggunakan For

```
...
/*  bandingkan  kunci  untuk  setiap  elemen  array  sampai  lokasi
ditemukan atau sampai akhir larik tercapai; return subskrip elemen
jika ditemukan, -1 jika tidak ditemukan */

int linearSearch( const int array[], int key, int size ) {
    int n; /* counter */

    /* pengulangan dalam array */
    for ( n = 0; n < size; ++n ) {
       if ( array[ n ] == key ) {
           return n; /* return lokasi kunci */
       } /* end if */
    } /* end for */
    return -1; /* key not found */
} /* end function linearSearch */
```

```
> Masukan angka yang dicari: 36
> Angka ditemukan di elemen ke-18
> Program ended with exit code: 0
```

Binary Search

- Dapat dilakukan pada **Array yang diurutkan**
- Pencarian linier bekerja dengan baik untuk **array kecil atau dengan data yang tidak terurut**
- Pencarian biner mencari elemen array dengan cara terus membagi array **menjadi dua bagian**.
- **Kasus terbaik**: Nilai yang dicari berada di elemen tengah

- **Urutkan** elemen array terlebih dahulu
- **Membandingkan** nilainya dengan elemen tengah
- **Jika tidak sama dan lebih kecil,** maka pencarian lanjut ke elemen bagian kanan, **abaikan elemen kiri**
- **Jika tidak sama dan lebih besar,** maka pencarian lanjut ke elemen bagian kiri, **abaikan elemen kanan**

Algoritma Binary Search

- **Urutkan** elemen array terlebih dahulu
- **Data** diambil dari posisi **low = 0** dan posisi **high = N–1**
- Kemudian cari **posisi data tengah** dengan rumus: (low + high) / 2
- Kemudian data yang dicari **dibandingkan** dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
- **Jika lebih besar**, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1
- **Jika lebih kecil,** maka proses pencarian dicari dengan posisi akhir adalah posisi tengah – 1
- **Jika data sama**, berarti ketemu. Break; Selesai

Binary Search

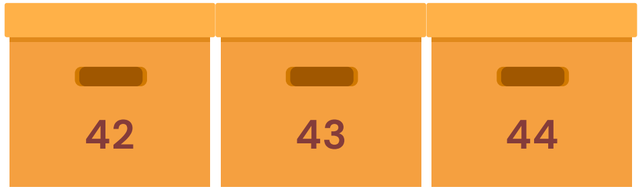
Langkah 1: Urutkan elemen array Sepatu (Asumsikan data sudah terurut atau gunakan algoritma pengurutan

Ingat! Pencarian biner selalu membagi 2 setiap kali belum ditemukan

low = 0 high = SIZE - 1

mid = (low+high)/2
= (0+6)/2 = 3
mid = indeks ke-3

key > sepatu[mid] ? YES
Lanjut pencarian bagian kanan



low = mid+1 = 4 high = SIZE - 1

mid = (low+high)/2
= (4+6)/2 = 5
mid = indeks ke-5

key = sepatu[mid] ? YES
STOP

Sepatu ukuran 43 ditemukan di indeks ke-mid, yaitu ke-5

Binary Search

Langkah 1: inisialisasi variabel low = 0, high = SIZE – 1

```
int main(){
    int i,low,high,ketemu;
    int sepatu[]={38,41,39,44,42,43,37};
    n = sizeof(sepatu)/sizeof(int);
    low = 0; high= n; ketemu = -1;
    key = 43;
}
```

Langkah 2: lakukan pengulangan selama low <= high, dan Tentukan nilai mid = (low + high)/ 2

```
int main() {
    while (low <= high) {
        mid=(low+high)/2;
        ...
```

Langkah 3: lakukan 3 pengecekan

```
int main() {
        ...
        if(key == sepatu[mid]){
            ketemu = mid;
            break;
        }else if(key < sepatu[mid]){
            high = mid-1;
        }else{
            low = mid+1
        }
    }
    if (ketemu== -1) {
        printf("Tidak ditemukan");
    }else{
        printf("Sepatu ukuran 43
ditemukan di indeks ke-%d\n",ketemu");
    }
  return 0;
}
```

How fast Binary Search?

- Klik link berikut, carilah angka dari kumpulan data pada permainan tersebut
- Coba kedua link pencarian ya!
- Terapkan cara pencarian binary search
- Bandingkan dengan cara pencarian linier, mana yang lebih cepat?

Find the number!

- Search Game 1
- Search Game 2

Terima kasih

TUGAS Mandiri @ Elearning

Jangan lupa kerjakan Latihan dan belajar mandiri di elearning ya!

Next:
Algoritma Pencarian