

PRAKTIKUM STRUKTUR DATA
MODUL 4
“ALGORITMA PENCAIRAN (SEARCHING)”



Disusun oleh :

Diana Eka Permata Sari

TRPL 2C

(362458302090)

PROGAM STUDI TEKNOLOGI REKAYASA PERANKAT LUNAK
JURUSAN BISNIS DAN INFORMATIKA

2025

Jalan Raya Jember No.KM13, Kawang, Labanasem, Kec. Kabat,
Kabupaten Banyuwangi, Jawa Timur 68461

Tujuan :

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

Dasar Teori :

Algoritma pencarian (searching algorithm) adalah algoritma yang menerima sebuah argument kunci dan dengan Langkah-langkah tertentu akan mencari rekaman dengan kunci tertentu. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (successful) atau tidak ditemukan (unsuccessful). Metode pencarian atau penelusuran data dapat dilakukan dengan dua cara yaitu penelusuran internal (internal searching) dan penelusuran eksternal (external searching). Pada penelusuran internal, semua rekaman yang diketahui berada dalam pengingat komputer, sedangkan pada penelusuran eksternal, tidak semua rekaman yang diketahui berada dalam pengingat computer, tetapi ada sejumlah rekaman yang tersimpan dalam penyimpanan luar misalnya pita atau cakram magnetis. Selain itu metode pencarian data juga dapat dikelompokkan menjadi pencarian statis (static searching) dan pencarian dinamis (dynamic searching). Pada pencarian statis, banyaknya rekaman yang diketahui dianggap tetap, pada pencarian dinamis, banyaknya rekaman yang diketahui bisa berubah-ubah yang disebabkan oleh penambahan atau penghapusan suatu rekaman. Ada dua macam teknik pencarian yaitu pencarian sekuensial/linier (Linear/sequential search) dan pencarian biner (binary search). Perbedaan dari dua teknik ini terletak pada keadaan data. Pencarian sekuensial digunakan apabila data dalam keadaan acak/tidak terurut. Sebaliknya, pencarian biner digunakan pada data yang sudah dalam keadaan urut.

1. Pencarian Berurutan (Linear/Sequential Search)

Pencarian berurutan sering disebut dengan pencarian linear, merupakan metode pencarian yang paling sederhana. Proses algoritma dimulai dari data paling awal, kemudian ditelusuri dengan menaikkan indeks data, apabila data sama dengan kunci pencarian dibentikan dan diberikan nilai pengembalian true, apabila sampai indeks terakhir data tidak ditemukan, maka diberikan nilai pengembalian false. Pencarian berurutan sering disebut dengan pencarian linear, merupakan metode pencarian yang paling sederhana. Proses algoritma dimulai dari data paling awal, kemudian ditelusuri dengan menaikkan indeks data, apabila data sama dengan kunci pencarian dibentikan dan diberikan nilai pengembalian true, apabila sampai indeks terakhir data tidak ditemukan, maka diberikan nilai pengembalian false.

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. Dalam kehidupan sehari-hari, sebenarnya kita juga sering menggunakan pencarian biner. Misalnya saat ingin mencari suatu kata dalam kamus. Prinsip dari pencarian biner adalah: Mula-mula diambil posisi awal 0 dan posisi akhir

= $N - 1$, kemudian dicari posisi data tengah dengan rumus (posisi awal + posisi akhir) / 2. Kemudian data yang dicari dibandingkan dengan data tengah. Jika lebih kecil, proses dilakukan Kembali tetapi posisi akhir dianggap sama dengan posisi tengah - 1. Jika lebih besar, proses dilakukan Kembali tetapi posisi awal dianggap sama dengan posisi tengah + 1. Demikian seterusnya sampai data tengah sama dengan yang dicari.

Tugas Pendahuluan :

1. Buatlah flowchart untuk operasi pencarian bilangan bulat dengan metode sequential search dan binary search

Jawab :

a. Sequential search

1. mulai : titik awal flowchart
2. input data : meminta pengguna untuk memasukkan daftar bilangan bulat dan nilai kunci yang ingin dicari
3. inisialisasi : set indeks 'I' ke 0
4. periksa indeks: apakah 'I' kurang dari daftar ?
 - jika ya, lanjut ke Langkah berikutnya
 - jika tidak, pergi ke Langkah 8
5. periksa elemen : apakah elemen pada indeks 'I' sama dengan kunci ?
 - jika ya, pergi ke langkah 6
 - jika tidak, lanjut ke Langkah 7
6. output ditemukan : cetak bahwa kunci ditemukan pada indeks 'I'
7. increment indeks : tambal ke 'I' dan Kembali ke Langkah 4
8. output tidak ditemukan : cetak bahwa kunci tidak ditemukan
9. selesai

b. Binary search :

1. Mulai
2. Inisialisasi left = 0, right = panjang data - 1
3. Apakah left <= right?
 - Jika tidak: Pergi ke langkah 6
4. Hitung mid = (left + right) / 2
5. Apakah data[mid] == target?
 - Jika ya: Pergi ke langkah 7
 - Jika tidak:
 - Jika data[mid] < target: left = mid + 1
 - Jika data[mid] > target: right = mid - 1

- Kembali ke langkah 3
 - 6. Selesai dengan hasil: target tidak ditemukan
 - 7. Selesai dengan hasil: target ditemukan di indeks mid
2. Buatlah flowchart untuk operasi penyisipan sebelum dan sesudah serta operasi penghapusan data kunci

Jawab :

a. Penyisipan sebelum

1. Mulai
2. Input data baru dan elemen target
3. Temukan indeks elemen target:
 - Apakah elemen target ditemukan?
 - Jika ya: Pergi ke langkah 4
 - Jika tidak: Pergi ke langkah 7
4. Pindahkan elemen setelah target ke kanan
5. Sisipkan data baru di indeks target
6. Selesai
7. Selesai dengan hasil: elemen target tidak ditemukan

b. penyisipan sesudah

1. Mulai
2. Input data baru dan elemen target
3. Temukan indeks elemen target:
 - Apakah elemen target ditemukan?
 - Jika ya: Pergi ke langkah 4
 - Jika tidak: Pergi ke langkah 7
4. Pindahkan elemen setelah target ke kanan
5. Sisipkan data baru di indeks target + 1
6. Selesai
7. Selesai dengan hasil: elemen target tidak ditemukan

c. penghapusan data kunci

1. Mulai
2. Input elemen yang ingin dihapus (kunci)
3. Temukan indeks elemen kunci:
 - Apakah elemen kunci ditemukan?

- Jika ya: Pergi ke langkah 4
 - Jika tidak: Pergi ke langkah 6
 - 4. Hapus elemen kunci dari indeks
 - 5. Pindahkan elemen setelah kunci ke kiri
 - 6. Selesai dengan hasil: elemen kunci tidak ditemukan
 - 7. Selesai
3. Buatlah deklarasi data pegawai dan flowchart fungsi untuk pencarian data dengan metode sequential search dan binary search sebagai berikut
- Data bertipe rekaman Bernama Pegawai yang mempunya 4 data, yaitu:
 - NIP, bertipe bulat
 - Nama, bertipe string
 - Alamat, bertipe string
 - Golongan, bertipe char
 - Kunci yang digunakan untuk pencarian data berdasarkan NIP dan Nama
- Jawab :
- a. Sequential search
1. Mulai
 2. Input NIP atau Nama target
 3. Inisialisasi $i = 0$
 4. Apakah $i < \text{panjang data pegawai}$?
 - Jika tidak: Pergi ke langkah 8
 5. Apakah $\text{data}[i].\text{nip} == \text{target NIP}$ atau $\text{data}[i].\text{nama} == \text{target Nama}$?
 - Jika ya: Pergi ke langkah 7
 - Jika tidak: $i = i + 1$, kembali ke langkah 4
 6. Selesai dengan hasil: data pegawai ditemukan di indeks i
 7. Selesai dengan hasil: data pegawai tidak ditemukan
- b. Binary search
1. Mulai
 2. Input NIP atau Nama target
 3. Inisialisasi $\text{left} = 0$, $\text{right} = \text{panjang data pegawai} - 1$
 4. Apakah $\text{left} \leq \text{right}$?
 - Jika tidak: Pergi ke langkah 7
 5. Hitung $\text{mid} = (\text{left} + \text{right}) / 2$
 6. Apakah $\text{data}[\text{mid}].\text{nip} == \text{target NIP}$ atau $\text{data}[\text{mid}].\text{nama} == \text{target Nama}$?
 - Jika ya: Pergi ke langkah 8

- Jika tidak:
 - Jika $\text{data}[\text{mid}].\text{nip} < \text{target NIP}$: $\text{left} = \text{mid} + 1$
 - Jika $\text{data}[\text{mid}].\text{nip} > \text{target NIP}$: $\text{right} = \text{mid} - 1$
 - Kembali ke langkah 4
7. Selesai dengan hasil: data pegawai tidak ditemukan
 8. Selesai dengan hasil: data pegawai ditemukan di indeks mid

Percobaan :

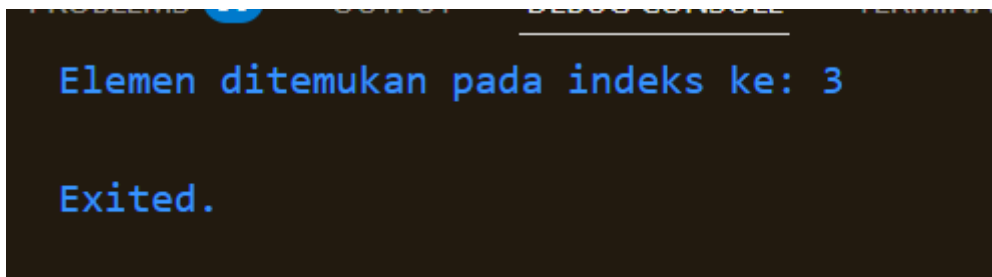
1. Implementasi pencarian dengan metode Linear search dengan Teknik iterative

```

2. int sequentialSearch(List<int> arr, int x) {
3.     int N = arr.length;
4.     for (int i = 0; i < N; i++) {
5.         if (arr[i] == x) return i;
6.     }
7.     return -1;
8. }
9.
10. void main() {
11.     List<int> arr = [2, 3, 4, 10, 40];
12.     int x = 10;
13.
14.     int result = sequentialSearch(arr, x);
15.     if (result == -1) {
16.         print("Elemen tidak ada dalam array");
17.     } else {
18.         print("Elemen ditemukan pada indeks ke: $result");
19.     }
20. }

```

Hasil Run :



```

Elemen ditemukan pada indeks ke: 3

Exited.

```

2. Implementasi pencarian dengan metode Linear search dengan Teknik rekursif

```

3. int linearSearch(List<int> arr, int size, int key) {
4.   if (size == 0) {
5.     return -1;
6.   } else if (arr[size - 1] == key) {
7.     // Return the index of found key.
8.     return size - 1;
9.   } else {
10.    return linearSearch(arr, size - 1, key);
11.  }
12.}
13.void main() {
14.  List<int> arr = [2, 3, 4, 10, 40];
15.  int x = 10;
16.  int result = linearSearch(arr, arr.length, x);
17.  if (result == -1) {
18.    print("Elemen tidak ada dalam array");
19.  } else {
20.    print("Elemen ditemukan pada indeks ke: $result");
21.  }
22.}

```

Hasil Run :

```

Elemen ditemukan pada indeks ke: 3

Exited.

```

3.Implementasi pencarian dengan metode binary search dengan Iterative

```

4. void binarySearch(List<int> arr, int l, int r, int x) {
5.   while (r - l > 1) {
6.     int mid = (l + r) ~/ 2;
7.     if (arr[mid] < x) {
8.       l = mid + 1;
9.     } else {
10.      r = mid;
11.    }
12.  }
13.  if (arr[l] == x) {
14.    print("Found At Index $l");

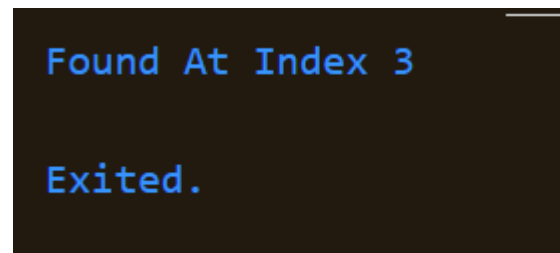
```

```

15. } else if (arr[r] == x) {
16.     print("Found At Index $r");
17. } else {
18.     print("Not Found");
19. }
20.}
21.void main() {
22.    List<int> arr = [2, 3, 4, 10, 40];
23.    int x = 10;
24.    int size = arr.length;
25.    binarySearch(arr, 0, size - 1, x);
26.}

```

Hasil Run :



```

Found At Index 3

Exited.

```

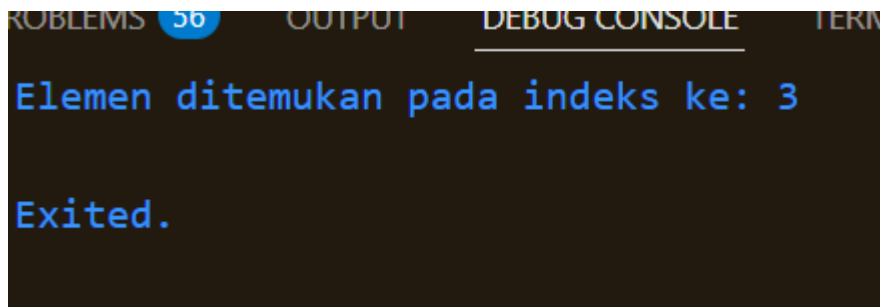
4. Implementasi pencarian dengan metode binary search dengan Rekursif

```

1. int binarySearch(List<int> arr, int l, int r, int x) {
2.     if (r >= l) {
3.         int mid = l + (r - l) ~/ 2;
4.         if (arr[mid] == x) return mid;
5.         if (arr[mid] > x) return binarySearch(arr, l, mid - 1, x);
6.         return binarySearch(arr, mid + 1, r, x);
7.     }
8.     return -1;
9. }
10. void main() {
11.     List<int> arr = [2, 3, 4, 10, 40];
12.     int x = 10;
13.     int size = arr.length;
14.     int result = binarySearch(arr, 0, size - 1, x);
15.     if (result == -1) {
16.         print("Elemen tidak ada dalam array");
17.     } else {
18.         print("Elemen ditemukan pada indeks ke: $result");
19.     }
20. }

```

Hasil Run :



Latihan :

1. (soal no 4) Implementasikan prosedur untuk menyisipkan data sebelum dan sesudah data kunci serta prosedur untuk menghapus data kunci pada tugas pendahuluan

```
2. void main() {
3.     List<int> data = [10, 20, 30, 40, 50];
4.     int key = 30;
5.     int newDataBefore = 25;
6.     int newDataAfter = 35;
7.
8.     print('Data awal: $data');
9.
10.    // untuk menyisipkan data sebelum kunci
11.    insertBefore(data, key, newDataBefore);
12.    print('Setelah menyisipkan $newDataBefore sebelum $key: $data');
13.
14.    // untuk menyisipkan data setelah kunci
15.    insertAfter(data, key, newDataAfter);
16.    print('Setelah menyisipkan $newDataAfter setelah $key: $data');
17.
18.    // untuk menghapus data kunci
19.    removeKey(data, key);
20.    print('Setelah menghapus $key: $data');
21.}
22.
23.void insertBefore(List<int> data, int key, int newData) {
24.    int index = data.indexOf(key);
25.    if (index != -1) {
26.        data.insert(index, newData);
27.    } else {
28.        print('Kunci $key tidak ditemukan.');
```

```

38. }
39.}
40.
41.void removeKey(List<int> data, int key) {
42.    if (data.remove(key) == false) {
43.        print('Kunci $key tidak ditemukan untuk dihapus.');
```

Penjelasan :

- Untuk mengumpulkan data awal yang disimpan ke dalam list 'data'
- 'insertbefore' yaitu untuk mencari indeks dari kunci dan menyisipkan data 'newdata' sebelum kunci jika ditemukan
- 'removekey' yaitu untuk menghapus kunci dari list. Jika kunci tidak ditemukan, akan mencetak pesan 'kunci tidak ditemukan'

Hasil runnya :

```

Data awal: [10, 20, 30, 40, 50]
Setelah menyisipkan 25 sebelum 30: [10, 20, 25, 30, 40, 50]
Setelah menyisipkan 35 setelah 30: [10, 20, 25, 30, 35, 40, 50]
Setelah menghapus 30: [10, 20, 25, 35, 40, 50]

Exited.
```

1. (soal no 3) Pada percobaan menggunakan Linier atau sequential Search, proses penelusuran akan berhenti pada saat data telah ditemukan. Namun apabila data yang sama dengan kata kunci yang dimasukkan itu tersedia lebih satu, maka data selanjutnya tidak akan di cek. Buatlah program untuk melakukan penelusuran dengan algoritma linier atau sequential search yang mencetak seluruh data yang memiliki nilai sama dengan kata kunci yang diinputkan!

```

2. void main() {
3.     List<int> data = [10, 20, 30, 20, 40, 50, 20, 60];
4.     int key = 20;
5.
6.     print('Data: $data');
7.     print('Mencari nilai: $key');
```

```

16.}
17.
18.List<int> linearSearch(List<int> data, int key) {
19.    List<int> indices = [];
20.
21.    for (int i = 0; i < data.length; i++) {
22.        if (data[i] == key) {
23.            indices.add(i); // Menyimpan indeks jika nilai sama dengan kunci
24.        }
25.    }
26.
27.    return indices; // Mengembalikan daftar indeks yang ditemukan
28.}

```

Penjelasan :

- Mengumpulkan data yang disimpan dalam list 'data', yang berisi beberapa nilai, termasuk nilai yang sama
- Menginput kunci 'key' menyimpan nilai yang akan dicari
- Fungsi dari 'linearsearch' yaitu :
 - Menerima list 'data' dan 'key' sebagai parameter
 - Menggunakan loop untuk memeriksa setiap elemen dalam list
 - Jika elemen sama dengan 'key', indeks elemen tersebut ditambahkan ke list 'indices'
 - Setelah loop selesai, fungsi mengembalikan list 'indices' yang berisi semua indeks dimana nilai 'key' ditemukan

Hasil Run :

```

Data: [10, 20, 30, 20, 40, 50, 20, 60]
Mencari nilai: 20
Nilai 20 ditemukan pada indeks: [1, 3, 6]

Exited.

```

Kesimpulan :

1. Didalam metode sequential search itu metode sederhana tetapi kurang efisien untuk data besar karena memeriksa elemen satu per satu
2. Didalam metode binary search lebih efisien terutama untuk list yang terurut karena membagi data menjadi 2 bagian

Referensi :

<https://hutomosaktikartiko.medium.com/solusi-binary-search-tree-dengan-bahasa-pemrograman-dart-52ffef5370b>

<https://www.kodeco.com/books/data-structures-algorithms-in-dart/v1.0/chapters/12-binary-search>