

**LAPORAN PRAKTIKUM
ALGORITMA & STRUKTUR DATA
MODUL 6**



SEARCHING

Oleh:

Rika Fauliana Rahmi NIM. 2410817120017

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025**

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA
MODUL 6

Laporan Praktikum Algoritma & Struktur Data Modul 6: Searching ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Rika Fauliana Rahmi
NIM : 2410817120017

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Fauzan Ahsani
NIM. 2310817310009

Muti'a Maulida, S.Kom., M.TI.
NIP. 198810272019032013

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI.....	3
DAFTAR GAMBAR	4
DAFTAR TABEL	5
SOAL PRAKTIKUM	6
PEMBAHASAN	8
A. Source Code	8
B. Output Program.....	15
C. Pembahasan.....	18
GITHUB.....	27

DAFTAR GAMBAR

Gambar 1 Screenshot tampilan awal program	15
Gambar 2 Screenshot tampilan ketika memilih opsi 1 (data ditemukan)	16
Gambar 3 Screenshot tampilan ketika memilih opsi 1 (data tidak ditemukan)	16
Gambar 4 Screenshot tampilan ketika memilih opsi 2 (data ditemukan)	16
Gambar 5 Screenshot tampilan ketika memilih opsi 2 (data tidak ditemukan)	17
Gambar 6 Screenshot tampilan ketika memilih opsi 3.....	17
Gambar 7 Screenshot tampilan ketika memilih opsi 4.....	17

DAFTAR TABEL

Tabel 1 Source Code Praktikum	8
-------------------------------------	---

SOAL PRAKTIKUM

Ketikkan source code berikut pada program IDE bahasa pemrograman C++

(Gabungkan 2 code berikut menjadi 1 file (Menu):

- Sequential Searching

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <time.h>
4
5  using namespace std;
6
7  int random(int bil)
8  {
9      int jumlah = rand() % bil;
10     return jumlah;
11 }
12
13 void randomize()
14 {
15     srand(time(NULL));
16 }
17
18 void clrscr()
19 {
20     system("cls");
21 }
22
23 int main()
24 {
25     clrscr();
26     int data[100];
27     int cari = 20;
28     int counter = 0;
29     int flag = 0;
30     int save;
31     randomize();
32     printf("generating 100 number . . .\n");
33     for (int i = 0; i < 100; i++)
34     {
35         data[i] = random(100) + 1;
36         printf("%d ", data[i]);
37     }
38     printf("\ndone.\n");
39
40     for (int i = 0; i < 100; i++)
41     {
42         if (data[i] == cari)
43         {
44             counter++;
45             flag = 1;
46             save = i;
47         }
48     }
49
50     if (flag == 1)
51     {
52         printf("Data ada, sebanyak %d!\n", counter);
53         printf("pada indeks ke-%d", save);
54     }
55     else
56     {
57         printf("Data tidak ada!\n");
58     }
59 }
60
```

- Binary searching

```

1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int n, kiri, kanan, tengah, temp, key;
7      bool ketemu = false;
8
9      cout << "Masukan jumlah data? ";
10     cin >> n;
11     int angka[n];
12     for (int i = 0; i < n; i++)
13     {
14         cout << "Angka ke - [" << i << "] : ";
15         cin >> angka[i];
16     }
17
18     for (int i = 0; i < n; i++)
19     {
20         for (int j = 0; j < n - 1; j++)
21         {
22             if (angka[j] > angka[j + 1])
23             {
24                 temp = angka[j];
25                 angka[j] = angka[j + 1];
26                 angka[j + 1] = temp;
27             }
28         }
29     }
30     cout << "-----\n";
31     cout << "Data yang telah diurutkan adalah:\n";
32     for (int i = 0; i < n; i++)
33     {
34         cout << angka[i] << " ";
35     }
36     cout << "\n-----\n";
37     cout << "Masukan angka yang dicari: ";
38     cin >> key;
39
40     kiri = 0;
41     kanan = n - 1;
42     while (kiri <= kanan)
43     {
44         tengah = (kiri + kanan) / 2;
45         if (key == angka[tengah])
46         {
47             ketemu = true;
48             break;
49         }
50         else if (key < angka[tengah])
51         {
52             kanan = tengah - 1;
53         }
54         else
55         {
56             kiri = tengah + 1;
57         }
58     }
59     if (ketemu == true)
60     {
61         cout << "Angka ditemukan! ";
62     }
63     else
64     {
65         cout << "Angka tidak ditemukan!";
66     }
67     return 0;
68 }

```

- Tampilan Menu Program :

```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih :

```

Jelaskan perbedaan Sequential Searching dan Binary Searching beserta kelebihan dan kekurangan masing-masing?

PEMBAHASAN

A. Source Code

Tabel 1 Source Code Praktikum

1	#include <iostream>
2	#include <conio.h>
3	#include <random>
4	#include <vector>
5	#include <algorithm>
6	
7	using namespace std;
8	
9	void sequentialSearch(vector<int> &nums, int target)
	{
10	int counter = 0;
11	int lastIndex = -1;
12	
13	for (int i = 0; i < nums.size(); i++) {
14	if (nums[i] == target) {
15	counter++;
16	lastIndex = i;
17	}
18	}
19	
20	if (counter > 0) {
21	cout << "Data ditemukan sebanyak " <<
	counter << " kali.\n";
22	cout << "Kemunculan terakhir pada indeks ke-
	" << lastIndex << ".\n";
23	} else {

24	cout << "Data tidak ditemukan!\n";
25	}
26	}
27	
28	void binarySearch(vector<int> &nums, int target) {
29	sort(nums.begin(), nums.end());
30	
31	cout << "Data setelah diurutkan:\n";
32	cout << "Data: ";
33	for (int val : nums) {
34	cout << val << " ";
35	}
36	cout << endl;
37	
38	int kiri = 0, kanan = nums.size() - 1;
39	bool ketemu = false;
40	
41	while (kiri <= kanan) {
42	int tengah = (kiri + kanan) / 2;
43	if (nums[tengah] == target) {
44	ketemu = true;
45	break;
46	} else if (target < nums[tengah]) {
47	kanan = tengah - 1;
48	} else {
49	kiri = tengah + 1;
50	}
51	}
52	
53	if (ketemu) {

54	cout << "Angka ditemukan!" << endl;
55	} else {
56	cout << "Angka tidak ditemukan!" << endl;
57	}
58	}
59	
60	void clearScreen() {
61	system("cls");
62	}
63	
64	void explain() {
65	cout << "\n--- PENJELASAN SEQUENTIAL DAN BINARY SEARCHING ---\n\n";
66	
67	cout << "[1] SEQUENTIAL SEARCHING:\n";
68	cout << "- Juga dikenal sebagai linear search.\n";
69	cout << "- Teknik pencarian yang dilakukan dengan memeriksa satu per satu elemen dalam struktur data,\n";
70	cout << " dimulai dari indeks pertama hingga indeks terakhir.\n";
71	cout << "- Jika data ditemukan, pencarian bisa dihentikan (jika hanya mencari satu data),\n";
72	cout << " atau dilanjutkan untuk menghitung jumlah kemunculan data tersebut.\n";
73	cout << "- Kelebihan:\n";
74	cout << " * Tidak memerlukan data yang sudah diurutkan.\n";

75	cout << " * Sederhana dan mudah diimplementasikan.\n";
76	cout << "- Kekurangan:\n";
77	cout << " * Tidak efisien untuk dataset yang sangat besar karena membutuhkan waktu $O(n)$,\n";
78	cout << " di mana n adalah jumlah elemen.\n\n";
79	
80	cout << "[2] BINARY SEARCHING:\n";
81	cout << "- Merupakan metode pencarian yang jauh lebih efisien dibanding sequential,\n";
82	cout << " namun hanya dapat digunakan pada data yang telah terurut.\n";
83	cout << "- Prinsip kerjanya adalah membagi dua (divide and conquer):\n";
84	cout << " * Bandingkan nilai tengah dengan target.\n";
85	cout << " * Jika target < nilai tengah, lanjut ke separuh kiri.\n";
86	cout << " * Jika target > nilai tengah, lanjut ke separuh kanan.\n";
87	cout << "- Proses ini terus diulang hingga data ditemukan atau rentang pencarian habis.\n";
88	cout << "- Kelebihan:\n";
89	cout << " * Sangat cepat pada data besar dengan waktu pencarian $O(\log n)$.\n";
90	cout << "- Kekurangan:\n";
91	cout << " * Hanya bisa digunakan jika data sudah diurutkan sebelumnya.\n";

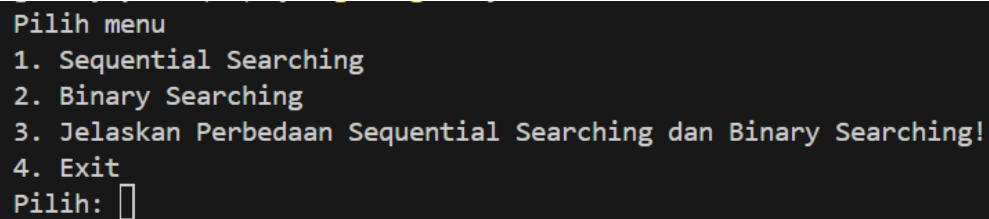
92	cout << " * Proses pengurutan (sorting) dapat memakan waktu tambahan.\n\n";
93	
94	cout << "Kesimpulan:\n";
95	cout << "- Gunakan Sequential Search jika data belum diurutkan dan pencarian sederhana.\n";
96	cout << "- Gunakan Binary Search jika data sudah diurutkan dan efisiensi sangat dibutuhkan.\n";
97	}
98	
99	int main() {
100	int opt, target;
101	do {
102	cout << "Pilih menu" << endl;
103	cout << "1. Sequential Searching" << endl;
104	cout << "2. Binary Searching" << endl;
105	cout << "3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!" << endl;
106	cout << "4. Exit" << endl;
107	cout << "Pilih: ";
108	cin >> opt;
109	
110	switch (opt) {
111	case 1: {
112	vector<int> nums(100);
113	mt19937_64 rng(random_device{}());
114	uniform_int_distribution<int>
	dist(1, 50);
115	
116	for (auto &val: nums) {

117	val = dist(rng);
118	}
119	
120	cout << "100 data acak telah dibuat:\n";
121	cout << "Data: ";
122	for (int val : nums) {
123	cout << val << " ";
124	}
125	cout << "\n";
126	
127	cout << "Masukkan angka yang ingin dicari: ";
128	cin >> target;
129	
130	sequentialSearch(nums, target);
131	break;
132	}
133	
134	case 2: {
135	int size;
136	cout << "Masukkan ukuran vector: ";
137	cin >> size;
138	
139	vector<int> nums(size);
140	mt19937_64 rng(random_device{}());
141	uniform_int_distribution<int> dist(1, 100);
142	
143	for (auto &val: nums) {

144	val = dist(rng);
145	}
146	
147	cout << size << " data acak telah dibuat:\n";
148	cout << "Data: ";
149	for (int val : nums) {
150	cout << val << " ";
151	}
152	cout << "\n";
153	
154	cout << "Masukkan angka yang ingin dicari: ";
155	cin >> target;
156	
157	binarySearch(nums, target);
158	break;
159	}
160	
161	case 3:
162	explain();
163	break;
164	
165	case 4:
166	cout << "\nTERIMA KASIH\n";
167	cout << "Programme was made by Rika Fauliana Rahmi (2410817120017)" << endl;
168	break;
169	
170	default:

171	cout << "Opsi tidak terdefinisi, mohon masukkan ulang opsi" << endl;
172	break;
173	}
174	
175	if (opt != 4) {
176	cout << "\nTekan sembarang tombol untuk melanjutkan...";
177	getch();
178	clearScreen();
179	}
180	
181	} while (opt != 4);
182	
183	return 0;
184	}

B. Output Program



```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 

```

Gambar 1 Screenshot tampilan awal program

```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 1
100 data acak telah dibuat:
Data: 9 37 28 8 37 31 41 31 38 26 9 41 8 49 5 26 32 48 3 28 15 46 9 20 20 3 4 4 28 16 34 26 18 47 27 20 16 21 2 22 26 30 49 13 32 34 46 32 4
13 35 13 1 28 33 24 29 7 25 37 24 28 14 36 47 41 49 4 25 16 11 4 2 9 12 19 27 48 37 32 35 29 47 8 29 21 49 23 50 34 16 6 21 17 4 8 44 10 8
19
Masukkan angka yang ingin dicari: 9
Data ditemukan sebanyak 4 kali.
Kemunculan terakhir pada indeks ke-73.

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 2 Screenshot tampilan ketika memilih opsi 1 (data ditemukan)

```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 1
100 data acak telah dibuat:
Data: 49 26 38 5 21 37 32 14 40 40 38 15 36 45 42 32 42 27 10 15 8 36 49 13 42 48 2 35 41 1 13 28 20 4 12 19 28 27 27 46 14 9 16 24 5 22 44
19 10 43 4 27 20 36 5 24 16 28 7 6 22 32 22 1 46 39 15 44 14 19 40 32 9 20 41 39 9 42 50 9 7 40 19 7 31 16 50 18 9 33 11 28 23 28 5 4 18 34
7 10
Masukkan angka yang ingin dicari: 25
Data tidak ditemukan!

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 3 Screenshot tampilan ketika memilih opsi 1 (data tidak ditemukan)

```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 2
Masukkan ukuran vector: 30
30 data acak telah dibuat:
Data: 63 65 53 29 90 88 16 95 75 37 14 95 39 57 27 96 40 71 83 52 79 45 16 85 11 55 41 55 65 13
Masukkan angka yang ingin dicari: 14
Data setelah diurutkan:
Data: 11 13 14 16 16 27 29 37 39 40 41 45 52 53 55 55 57 63 65 65 71 75 79 83 85 88 90 95 95 96
Angka ditemukan!

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 4 Screenshot tampilan ketika memilih opsi 2 (data ditemukan)


```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 2
Masukkan ukuran vector: 30
30 data acak telah dibuat:
Data: 48 33 46 29 70 98 5 8 83 65 68 55 77 38 67 2 60 89 75 39 68 97 65 87 19 51 63 57 99 97
Masukkan angka yang ingin dicari: 12
Data setelah diurutkan:
Data: 2 5 8 19 29 33 38 39 46 48 51 55 57 60 63 65 65 67 68 68 70 75 77 83 87 89 97 97 98 99
Angka tidak ditemukan!

Tekan sembarang tombol untuk melanjutkan...

```

Gambar 5 Screenshot tampilan ketika memilih opsi 2 (data tidak ditemukan)

```

--- PENJELASAN SEQUENTIAL DAN BINARY SEARCHING ---

[1] SEQUENTIAL SEARCHING:
- Juga dikenal sebagai linear search.
- Teknik pencarian yang dilakukan dengan memeriksa satu per satu elemen dalam struktur data, dimulai dari indeks pertama hingga indeks terakhir.
- Jika data ditemukan, pencarian bisa dihentikan (jika hanya mencari satu data), atau dilanjutkan untuk menghitung jumlah kemunculan data tersebut.
- Kelebihan:
  * Tidak memerlukan data yang sudah diurutkan.
  * Sederhana dan mudah diimplementasikan.
- Kekurangan:
  * Tidak efisien untuk dataset yang sangat besar karena membutuhkan waktu  $O(n)$ , di mana  $n$  adalah jumlah elemen.

[2] BINARY SEARCHING:
- Merupakan metode pencarian yang jauh lebih efisien dibanding sequential, namun hanya dapat digunakan pada data yang telah terurut.
- Prinsip kerjanya adalah membagi dua (divide and conquer):
  * Bandingkan nilai tengah dengan target.
  * Jika target < nilai tengah, lanjut ke separuh kiri.
  * Jika target > nilai tengah, lanjut ke separuh kanan.
- Proses ini terus diulang hingga data ditemukan atau rentang pencarian habis.
- Kelebihan:
  * Sangat cepat pada data besar dengan waktu pencarian  $O(\log n)$ .
- Kekurangan:
  * Hanya bisa digunakan jika data sudah diurutkan sebelumnya.
  * Proses pengurutan (sorting) dapat memakan waktu tambahan.

Kesimpulan:
- Gunakan Sequential Search jika data belum diurutkan dan pencarian sederhana.
- Gunakan Binary Search jika data sudah diurutkan dan efisiensi sangat dibutuhkan.

```

Gambar 6 Screenshot tampilan ketika memilih opsi 3

```

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 4

TERIMA KASIH
Programme was made by Rika Fauliana Rahmi (2410817120017)

```

Gambar 7 Screenshot tampilan ketika memilih opsi 4

C. Pembahasan

```
#include <iostream>
```

input/output standar (cin, cout, dll).

```
#include <conio.h>
```

fungsi `getch()` yang digunakan untuk menahan layar sampai user menekan tombol apa saja.

```
#include <random>
```

menghasilkan bilangan acak yang lebih modern dan aman dibanding `rand()`.

```
#include <vector>
```

digunakan untuk membuat array dinamis (vector) yang fleksibel.

```
#include <algorithm>
```

mengandung fungsi `sort()` untuk pengurutan.

Fungsi Sequential Search

```
void sequentialSearch(vector<int> &nums, int target) {
```

fungsi untuk melakukan pencarian linear/sequential pada vector `nums` terhadap nilai `target`.

```
int counter = 0;  
int lastIndex = -1;
```

- `counter`: menghitung berapa kali `target` ditemukan.
- `lastIndex`: menyimpan indeks kemunculan terakhir.

```
for (int i = 0; i < nums.size(); i++) {  
    if (nums[i] == target) {  
        counter++;  
        lastIndex = i;
```

melakukan iterasi dan pengecekan apakah elemen sama dengan `target`.

```
    if (counter > 0) {  
        cout << "Data ditemukan sebanyak " << counter <<  
        " kali.\n";
```

```

        cout << "Kemunculan terakhir pada indeks ke-" <<
lastIndex << ".\n";
    } else {
        cout << "Data tidak ditemukan!\n";
    }
}

```

menampilkan hasil pencarian: jumlah dan posisi terakhir atau pesan tidak ditemukan.

Fungsi Binary Search

```

void binarySearch(vector<int> &nums, int target) {
    sort(nums.begin(), nums.end());
}

```

mengurutkan data sebelum pencarian biner dilakukan (syarat Binary Search).

```

cout << "Data setelah diurutkan:\n";
cout << "Data: ";
for (int val : nums) {
    cout << val << " ";
}
cout << endl;

```

menampilkan data setelah diurutkan.

```

int kiri = 0, kanan = nums.size() - 1;
bool ketemu = false;

```

- kiri dan kanan sebagai batas awal dan akhir pencarian.
- ketemu menyimpan status pencarian.

```

while (kiri <= kanan) {
    int tengah = (kiri + kanan) / 2;
    if (nums[tengah] == target) {
        ketemu = true;
        break;
    } else if (target < nums[tengah]) {
        kanan = tengah - 1;
    } else {
        kiri = tengah + 1;
    }
}

```

implementasi algoritma binary search:

- Cari nilai tengah.
- Sesuaikan batas kiri/kanan berdasarkan perbandingan.

```
if (ketemu) {
    cout << "Angka ditemukan!" << endl;
} else {
    cout << "Angka tidak ditemukan!" << endl;
```

menampilkan hasil dari binary search.

Fungsi Clear Layar :

```
void clearScreen() {
    system("cls");
```

Penjelasan Algoritma :

```
void explain() {
```

Fungsi main() :

```
int main() {
    int opt, target;
```

- opt: pilihan menu dari user.
- target: nilai yang ingin dicari.

```
do {
    cout << "Pilih menu" << endl;
    ...
    cin >> opt;
```

menampilkan menu dan meminta pilihan user.

Case 1 : Sequential Search

```
switch (opt) {
    case 1: {
        vector<int> nums(100);
        mt19937_64 rng(random_device{}());
```

```
uniform_int_distribution<int> dist(1,
50);
```

- Membuat vector `nums` berisi 100 elemen.
- `mt19937_64` dan `random_device` digunakan untuk RNG yang berkualitas.
- Angka acak dari 1–50.

```
for (auto &val: nums) {
    val = dist(rng);
    cout << "100 data acak telah dibuat:\n";
    cout << "Data: ";
    for (int val : nums) {
        cout << val << " ";
    }
    cout << "\n";

    cout << "Masukkan angka yang ingin
dicari: ";

    cin >> target;

    sequentialSearch(nums, target);
    break;
```

- Data acak ditampilkan.
- Input `target` dari user.
- Fungsi `sequentialSearch` dipanggil untuk mencari.

Case 2 : Binary Search

```
case 2: {

    int size;
```

```

        cout << "Masukkan ukuran vector: ";

        cin >> size;

        vector<int> nums(size);

        mt19937_64 rng(random_device{}());

        uniform_int_distribution<int> dist(1,
100);

```

- Ukuran vector ditentukan oleh user.
- Angka acak dari 1–100.

```

        for (auto &val: nums) {

            val = dist(rng);

        }

        cout << size << " data acak telah
dibuat:\n";

        cout << "Data: ";

        for (int val : nums) {

            cout << val << " ";

        }

        cout << "\n";

```

```
        cout << "Masukkan angka yang ingin  
dicari: ";  
  
        cin >> target;  
  
        binarySearch(nums, target);  
  
        break;
```

- Data acak ditampilkan.
- Fungsi `binarySearch` mencari target dengan data yang telah diurutkan.

Case 3 : explain

```
case 3:  
  
    explain();  
  
    break;
```

Case 4 : exit

```
case 4:  
  
    cout << "\nTERIMA KASIH\n";  
  
    cout << "Programme was made by Rika  
Fauliana Rahmi (2410817120017)" << endl;  
  
    break;
```

keluar dari program dan menampilkan ucapan terima kasih dan identitas pembuat.

```
default:
```

```
        cout << "Opsi tidak terdefinisi, mohon  
masukkan ulang opsi" << endl;  
  
        break;
```

Menangani input selain case 1-4.

```
        if (opt != 4) {  
  
            cout << "\nTekan sembarang tombol untuk  
melanjutkan...";  
  
            getch();  
  
            clearScreen();
```

memberi jeda setelah operasi selesai dan membersihkan layar sebelum kembali ke menu.

```
    } while (opt != 4);  
  
    return 0;
```

mengulangi menu hingga pengguna memilih 4 (exit).

1. Sequential Searching

Sequential Searching adalah metode pencarian data yang paling sederhana. Cara kerjanya adalah dengan memeriksa setiap elemen data satu per satu, mulai dari awal hingga akhir, sampai elemen yang dicari ditemukan atau seluruh elemen telah diperiksa.

Cara Kerja :

1. Mulai dari elemen pertama dalam array.

2. Bandingkan elemen saat ini dengan data yang dicari.
3. Jika cocok, data ditemukan.
4. Jika tidak cocok, pindah ke elemen berikutnya dan ulangi langkah 2.
5. Proses berhenti jika data ditemukan atau jika semua elemen sudah diperiksa dan data tidak ditemukan.

Kelebihan Sequential Searching :

1. Sederhana dan mudah diimplementasikan
2. Tidak memerlukan data terurut.
3. Efektif untuk data berukuran kecil
4. Cocok untuk data dinamis.

Kekurangan Sequential Searching :

1. Tidak efisien untuk data berukuran besar
2. Kompleksitas waktu $O(n)$: dalam kasus terburuk (data yang dicari berada di akhir atau tidak ada), algoritma ini harus memeriksa semua n elemen. Ini membuatnya sangat tidak efisien untuk dataset besar.
3. Bukan pilihan optimal untuk pencarian berulang.

1. Binary Searching

Binary Searching adalah metode pencarian yang jauh lebih efisien daripada Sequential Searching, tetapi memiliki syarat mutlak yaitu, data harus dalam keadaan terurut (ascending atau descending). Algoritma ini bekerja dengan membagi ruang pencarian menjadi dua bagian secara berulang.

Cara Kerja :

1. Pastikan data terurut, jika data tidak terurut, binary search tidak akan bekerja dengan benar.
2. Tentukan titik tengah (median) dari daftar data.
3. Bandingkan data yang dicari dengan elemen di titik tengah.
4. Ada tiga kemungkinan :

- Jika kata kunci sama dengan elemen tengah, data ditemukan.
- Jika kata kunci lebih kecil dari elemen tengah, abaikan setengah bagian kanan (termasuk elemen tengah) dan lanjutkan pencarian di setengah bagian kiri.
- Jika kata kunci lebih besar dari elemen tengah, abaikan setengah bagian kiri (termasuk elemen tengah) dan lanjutkan pencarian di setengah bagian kanan.

Kelebihan Binary Searching :

1. Sangat efisien untuk data berukuran besar.
2. Kompleksitas waktu $O(\log n)$, yang jauh lebih baik daripada $O(n)$ pada sequential search. Ini berarti waktu pencarian hanya bertambah sedikit bahkan jika jumlah data berlipat ganda.
3. Cocok untuk pencarian berulang.

Kekurangan Binary Searching :

1. Membutuhkan data terurut.
2. Implementasi lebih rumit, algoritmanya sedikit lebih kompleks daripada sequential search, terutama jika diimplementasikan secara rekursif.
3. Kurang efisien untuk data kecil atau data dinamis, untuk data yang sangat sedikit, overhead dari binary search mungkin membuatnya tidak jauh lebih cepat daripada sequential search. Jika data sering berubah, menjaga data tetap terurut akan menjadi tantangan dan membutuhkan waktu ekstra untuk pengurutan ulang.

GITHUB

<https://github.com/DSA25-ULM/task-6-searching-rikafaulianarahmi>