

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



SEARCHING

Oleh:

Indra Suryadilaga NIM. 2410817310014

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

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA
MODUL 5

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 : Indra Suryadilaga
NIM : 2410817310014

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Zulfa Auliya Akbar
NIM. 2210817210010

Andreyan Rizky Baskara, S.Kom., M.Kom
NIP. 199307032019031011

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI.....	3
DAFTAR GAMBAR	4
SOAL 1	5
A. Source Code	8
B. Output Program.....	13
C. Pembahasan.....	14
GITHUB.....	19

DAFTAR GAMBAR

Gambar 1. Soal No. 1 Source Code Sequential Searching	6
Gambar 2. Soal No. 1 Source Code Binary Searching	7
Gambar 3. Output Sequential Searching.....	13
Gambar 4. Output Binary Searching.....	13
Gambar 5. Output Penjelasan Perbedaan Sequential dan Binary Searching	14

DAFTAR TABEL

Table 1. Source Code Algoritma Searching.....	8
---	---

SOAL 1

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
```

Gambar 1. Soal No. 1 Source Code Sequential Searching

- 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 }

```

Gambar 2. Soal No. 1 Source Code Binary Searching

A. Source Code

Table 1. Source Code Algoritma Searching

1	#include <iostream>
2	#include <conio.h>
3	#include <random>
4	#include <vector>
5	#include <algorithm>
6	#include <iomanip>
7	using namespace std;
8	
9	// Sequential Search
10	void sequentialSearch(const vector<int>& nums, int target) {
11	vector<int> foundIndices;
12	
13	cout << "\n100 angka acak telah digenerate.\nNums:\n";
14	for (int i = 0; i < nums.size(); i++) {
15	cout << "[" << i << "]: " << nums[i] << " ";
16	}
17	cout << "\n\nMasukkan angka yang ingin dicari: ";
18	cin >> target;
19	
20	for (int i = 0; i < nums.size(); i++) {
21	if (nums[i] == target) {
22	foundIndices.push_back(i);
23	}
24	}
25	
26	if (!foundIndices.empty()) {
27	cout << "\nTarget " << target << " ditemukan sebanyak
	" << foundIndices.size() << " kali di indeks: ";
28	for (int i : foundIndices) {
29	cout << i << " ";
30	}
31	cout << "\n";
32	} else {

33	cout << "Target " << target << " tidak ditemukan dalam data.\n";
34	}
35	}
36	
37	// Binary Search
38	void binarySearch(const vector<int>& numsUnsorted, int target)
	{
39	vector<int> nums = numsUnsorted;
40	
41	sort(nums.begin(), nums.end());
42	
43	cout << "\n" << nums.size() << " angka telah digenerate dan diurutkan.\nNums:\n";
44	for (int i = 0; i < nums.size(); i++) {
45	cout << "[" << i << "]: " << nums[i] << " ";
46	}
47	
48	cout << "\n\nMasukkan angka yang ingin dicari: ";
49	cin >> target;
50	
51	int kiri = 0, kanan = nums.size() - 1;
52	int posisi = -1;
53	
54	while (kiri <= kanan) {
55	int tengah = (kiri + kanan) / 2;
56	if (nums[tengah] == target) {
57	posisi = tengah;
58	break;
59	} else if (target < nums[tengah]) {
60	kanan = tengah - 1;
61	} else {
62	kiri = tengah + 1;
63	}
64	}
65	

```

66     if (posisi != -1)
67         cout << "\nTarget " << target << " ditemukan di index
= " << posisi << "\n";
68     else
69         cout << "\nTarget tidak ditemukan dalam data.\n";
70 }
71
72 void clearScreen() {
73     system("cls");
74 }
75
76 // Penjelasan
77 void explain() {
78     cout << "\n=== PERBEDAAN SEQUENTIAL DAN BINARY SEARCHING
===\n";
79     cout << "Sequential Search:\n";
80     cout << " - Tidak perlu data terurut.\n";
81     cout << " - Mencari satu-satu dari awal hingga akhir.\n";
82     cout << " - Waktu pencarian: O(n).\n\n";
83     cout << "Binary Search:\n";
84     cout << " - Hanya bisa dilakukan pada data yang sudah
terurut.\n";
85     cout << " - Membagi dua bagian data hingga ketemu.\n";
86     cout << " - Waktu pencarian: O(log n).\n";
87 }
88
89 int main() {
90     int opt, target;
91
92     do {
93         cout << "\nPilih menu\n";
94         cout << "1. Sequential Searching\n";
95         cout << "2. Binary Searching\n";
96         cout << "3. Jelaskan Perbedaan Sequential Searching
dan Binary Searching!\n";
97         cout << "4. Exit\n";

```

98	cout << "Pilih: ";
99	cin >> opt;
100	
101	switch (opt) {
102	case 1: {
103	vector<int> nums(100);
104	mt19937_64 rng(random_device{}());
105	uniform_int_distribution<int> dist(1, 50);
106	
107	for (auto& val : nums) {
108	val = dist(rng);
109	}
110	
111	sequentialSearch(nums, target);
112	break;
113	}
114	
115	case 2: {
116	int size;
117	cout << "Masukkan ukuran vector: ";
118	cin >> size;
119	
120	vector<int> nums(size);
121	mt19937_64 rng(random_device{}());
122	uniform_int_distribution<int> dist(1, 100);
123	
124	for (auto& val : nums) {
125	val = dist(rng);
126	}
127	
128	binarySearch(nums, target);
129	break;
130	}
131	
132	case 3:
133	explain();

134	break;
135	
136	case 4:
137	cout << "\nTERIMA KASIH\n";
138	cout << "Program was made by Daniel Noprianto (2410817110010)\n";
139	break;
140	
141	default:
142	cout << "Opsi tidak valid. Coba lagi.\n";
143	}
144	
145	if (opt != 4) {
146	cout << "\nTekan sembarang tombol untuk melanjutkan...";
147	getch();
148	clearScreen();
149	}
150	
151	} while (opt != 4);
152	
153	return 0;
154	}

B. Output Program

```
D:\Github\Algoritma-Dan-Stri x + v

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

100 angka acak telah digenerate.
Nums:
[0]:17 [1]:2 [2]:42 [3]:4 [4]:27 [5]:40 [6]:50 [7]:32 [8]:46 [9]:16 [10]:23 [11]:43
[12]:49 [13]:11 [14]:24 [15]:24 [16]:45 [17]:1 [18]:39 [19]:24 [20]:14 [21]:29 [22]
]:28 [23]:26 [24]:6 [25]:7 [26]:24 [27]:18 [28]:46 [29]:17 [30]:38 [31]:9 [32]:43 [
33]:30 [34]:12 [35]:25 [36]:48 [37]:36 [38]:48 [39]:3 [40]:7 [41]:3 [42]:8 [43]:44
[44]:42 [45]:7 [46]:47 [47]:10 [48]:45 [49]:14 [50]:33 [51]:39 [52]:23 [53]:43 [54]
:26 [55]:26 [56]:16 [57]:28 [58]:12 [59]:8 [60]:50 [61]:32 [62]:38 [63]:38 [64]:29
[65]:13 [66]:9 [67]:26 [68]:31 [69]:30 [70]:33 [71]:40 [72]:46 [73]:35 [74]:45 [75]
:13 [76]:50 [77]:27 [78]:37 [79]:50 [80]:2 [81]:6 [82]:19 [83]:43 [84]:21 [85]:11 [
86]:7 [87]:14 [88]:41 [89]:10 [90]:29 [91]:48 [92]:28 [93]:25 [94]:25 [95]:34 [96]:
7 [97]:7 [98]:29 [99]:12

Masukkan angka yang ingin dicari: 13

Target 13 ditemukan sebanyak 2 kali di indeks: 65 75

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 3. Output Sequential Searching

```
D:\Github\Algoritma-Dan-Stri x + v

Pilih menu
1. Sequential Searching
2. Binary Searching
3. Jelaskan Perbedaan Sequential Searching dan Binary Searching!
4. Exit
Pilih: 2
Masukkan ukuran vector: 6

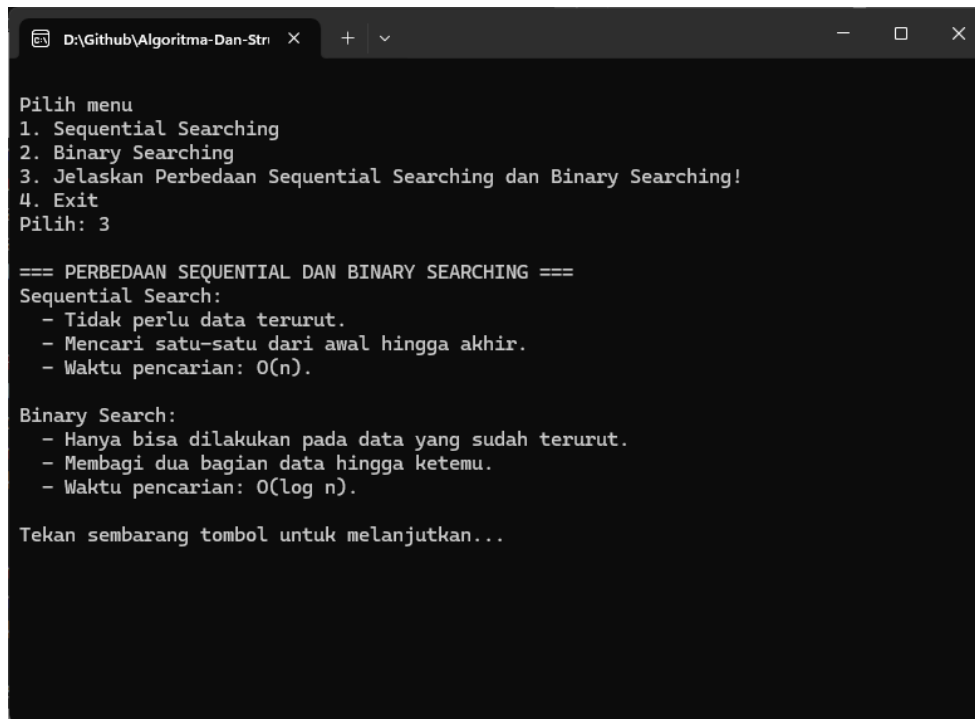
6 angka telah digenerate dan diurutkan.
Nums:
[0]:4 [1]:7 [2]:34 [3]:54 [4]:80 [5]:83

Masukkan angka yang ingin dicari: 54

Target 54 ditemukan di index = 3

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 4. Output Binary Searching



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

=== PERBEDAAN SEQUENTIAL DAN BINARY SEARCHING ===
Sequential Search:
- Tidak perlu data terurut.
- Mencari satu-satu dari awal hingga akhir.
- Waktu pencarian: O(n).

Binary Search:
- Hanya bisa dilakukan pada data yang sudah terurut.
- Membagi dua bagian data hingga ketemu.
- Waktu pencarian: O(log n).

Tekan sembarang tombol untuk melanjutkan...
```

Gambar 5. Output Penjelasan Perbedaan Sequential dan Binary Searching

C. Pembahasan

Alur Program Sorting

Ketika program dijalankan, sistem akan menampilkan menu utama dengan 4 pilihan yang tersedia menggunakan do-while loop yang terus berjalan hingga user memilih opsi 4 (Exit). Setiap kali user memilih opsi 1-3, program akan melakukan proses sesuai dengan pilihan yang dipilih.

Untuk opsi 1 (Sequential Searching), program akan menggenerate 100 angka acak dengan rentang 1-50 menggunakan random number generator *mt19937_64* dan *uniform_int_distribution*. Data yang dihasilkan disimpan ke vector dan ditampilkan beserta indeksinya. User kemudian diminta untuk memasukkan angka yang ingin dicari, dan program akan mencari semua angka tersebut dalam data.

Untuk opsi 2 (Binary Searching), program akan meminta user untuk memasukkan ukuran vector terlebih dahulu, kemudian menggenerate angka acak

sesuai ukuran yang ditentukan dengan rentang 1-100. Data yang dihasilkan akan diurutkan terlebih dahulu menggunakan fungsi *sort()* dari *STL*, kemudian ditampilkan. Setelah itu, user diminta memasukkan angka yang ingin dicari, dan program akan melakukan pencarian menggunakan algoritma binary search.

Untuk opsi 3, program akan menampilkan penjelasan mengenai perbedaan antara Sequential Search dan Binary Search, termasuk karakteristik dan kompleksitas waktu masing-masing algoritma.

Pada akhir setiap operasi (kecuali exit), program akan menampilkan pesan "*Tekan sembarang tombol untuk melanjutkan...*" dan menunggu input menggunakan *getch()*. Setelah user menekan tombol apapun, layar akan dibersihkan menggunakan *system("cls")* dan menu utama akan ditampilkan kembali. Loop ini akan terus berlanjut hingga user memilih opsi 4 untuk keluar dari program.

Analisis Algoritma Sequential Search

Algoritma Sequential Search (Linear Search) bekerja dengan cara memeriksa setiap elemen dalam array secara berurutan dari awal hingga akhir untuk mencari target yang diinginkan. Implementasi ini menggunakan vector *foundIndices* untuk menyimpan semua indeks di mana target ditemukan, sehingga dapat menangani kasus di mana target muncul lebih dari sekali.

Proses searching dimulai dengan loop *for* yang iterasi dari indeks 0 hingga ukuran vector dikurangi satu. Pada setiap iterasi, program membandingkan elemen pada posisi *i* dengan target yang dicari. Jika elemen tersebut sama dengan target, indeks *i* akan disimpan dalam vector *foundIndices*. Proses ini berlanjut hingga seluruh elemen dalam array telah diperiksa.

Setelah proses pencarian selesai, program akan memeriksa apakah vector *foundIndices* kosong atau tidak. Jika tidak kosong, program akan menampilkan informasi bahwa target ditemukan beserta jumlah kemunculan dan semua indeks di

mana target tersebut ditemukan. Jika kosong, program akan menampilkan pesan bahwa target tidak ditemukan dalam data.

Kelebihan dari Sequential Search adalah algoritma ini dapat bekerja pada data yang tidak terurut dan dapat menemukan semua kemunculan target dalam satu kali pencarian. Algoritma ini juga mudah diimplementasikan dan dipahami, serta memiliki space complexity yang rendah yaitu $O(1)$ jika hanya mencari satu kemunculan. Sequential Search juga tidak memerlukan preprocessing data seperti pengurutan.

Kekurangan utama Sequential Search terletak pada time complexity yang kurang efisien untuk data berukuran besar. Pada worst case (target berada di akhir array atau tidak ada), algoritma ini memiliki time complexity $O(n)$ karena harus memeriksa semua elemen. Average case juga memiliki time complexity $O(n/2)$ yang masih tergolong $O(n)$, sedangkan best case (target berada di awal array) memiliki time complexity $O(1)$.

Analisis Algoritma Binary Search

Algoritma Binary Search menggunakan strategi divide and conquer dengan cara membagi array menjadi dua bagian secara berulang hingga target ditemukan atau tidak ada lagi elemen yang dapat dibagi. Algoritma ini hanya dapat bekerja pada data yang sudah terurut, sehingga implementasi ini menggunakan fungsi `sort()` terlebih dahulu untuk mengurutkan data.

Proses searching dimulai dengan menentukan batas kiri ($kiri = 0$) dan batas kanan ($kanan = ukuran\ array - 1$). Kemudian program menghitung posisi tengah menggunakan formula $tengah = (kiri + kanan) / 2$. Elemen pada posisi tengah akan dibandingkan dengan target yang dicari.

Jika elemen tengah sama dengan target, maka target telah ditemukan dan posisinya disimpan. Jika target lebih kecil dari elemen tengah, pencarian akan dilanjutkan pada bagian kiri array dengan mengubah $kanan = tengah - 1$. Jika target

lebih besar dari elemen tengah, pencarian akan dilanjutkan pada bagian kanan array dengan mengubah $kiri = tengah + 1$. Proses ini berlanjut hingga target ditemukan atau $kiri > kanan$ (target tidak ada).

Implementasi ini menggunakan variabel posisi untuk menyimpan indeks di mana target ditemukan. Jika posisi masih bernilai -1 setelah loop selesai, berarti target tidak ditemukan dalam data. Program kemudian akan menampilkan hasil pencarian kepada user.

Kelebihan utama Binary Search adalah time complexity yang sangat efisien yaitu $O(\log n)$ pada semua kasus (best, average, dan worst case). Algoritma ini sangat cocok untuk data berukuran besar dan memiliki performa yang predictable. Binary Search juga memiliki space complexity $O(1)$ pada implementasi iterative dan memiliki efisiensi yang tinggi dalam praktik.

Kekurangan Binary Search terletak pada requirement bahwa data harus sudah terurut terlebih dahulu. Jika data belum terurut, diperlukan proses sorting yang memiliki time complexity minimal $O(n \log n)$, sehingga untuk pencarian sekali saja mungkin tidak efisien. Algoritma ini juga hanya dapat menemukan satu kemunculan target (biasanya yang pertama ditemukan) dan lebih kompleks untuk diimplementasikan dibanding Sequential Search.

Perbandingan Sequential Search dan Binary Search

Dari segi kompleksitas waktu, Binary Search memiliki keunggulan yang signifikan dengan $O(\log n)$ dibandingkan Sequential Search yang memiliki $O(n)$. Namun, Binary Search memerlukan data yang sudah terurut, sedangkan Sequential Search dapat bekerja pada data apa adanya.

Dari segi penggunaan praktis, Sequential Search lebih cocok untuk dataset kecil atau ketika data sering berubah dan tidak selalu terurut. Binary Search lebih cocok

untuk dataset besar yang sudah terurut atau ketika pencarian dilakukan berulang kali pada data yang sama.

Sequential Search juga memiliki keunggulan dalam menemukan semua kemunculan target dalam satu kali pencarian, sedangkan Binary Search biasanya hanya menemukan satu kemunculan. Untuk aplikasi yang memerlukan pencarian multiple occurrences, Sequential Search atau modifikasi Binary Search mungkin lebih sesuai.

GITHUB

<https://github.com/IndraSuryadilaga/Algoritma-Dan-Struktur-Data>