

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL VIII
ALGORITMA SEARCHING**



Disusun Oleh :

NAMA : FAHRI RAMADHAN

NIM : 2311102024

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

**BAB I TUJUAN
PRAKTIKUM**

1. Menunjukkan beberapa algoritma dalam Pencarian.
2. Menunjukkan bahwa pencarian merupakan suatu persoalan yang bisa diselesaikan dengan beberapa algoritma yang berbeda.
3. Dapat memilih algoritma yang paling sesuai untuk menyelesaikan suatu permasalahan pemrograman.

BAB II DASAR TEORI

Pencarian (Searching) yaitu proses menemukan suatu nilai tertentu pada kumpulan data. Hasil pencarian adalah salah satu dari tiga keadaan ini: data ditemukan, data ditemukan lebih dari satu, atau data tidak ditemukan. Searching juga dapat dianggap sebagai proses pencarian suatu data di dalam sebuah array dengan cara mengecek satu persatu pada setiap index baris atau setiap index kolomnya dengan menggunakan teknik perulangan untuk melakukan pencarian data. Terdapat 2 metode pada algoritma Searching, yaitu:

a. Sequential Search

Sequential Search merupakan salah satu algoritma pencarian data yang biasa digunakan untuk data yang berpola acak atau belum terurut. Sequential search juga merupakan teknik pencarian data dari array yang paling mudah, dimana data dalam array dibaca satu demi satu dan diurutkan dari index terkecil ke index terbesar, maupun sebaliknya. Konsep Sequential Search yaitu:

- Membandingkan setiap elemen pada array satu per satu secara berurut.
- Proses pencarian dimulai dari indeks pertama hingga indeks terakhir.

- Proses pencarian akan berhenti apabila data ditemukan. Jika hingga akhir array data masih juga tidak ditemukan, maka proses pencarian tetap akan dihentikan.
- Proses perulangan pada pencarian akan terjadi sebanyak jumlah N elemen pada array.

Algoritma pencarian berurutan dapat dituliskan sebagai berikut :

1. $i \leftarrow 0$
2. $ketemu \leftarrow false$
3. Selama (tidak $ketemu$) dan ($i \leq N$) kerjakan baris 4
4. Jika ($Data[i] = x$) maka $ketemu \leftarrow true$, jika tidak $i \leftarrow i + 1$
5. Jika ($ketemu$) maka i adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

b. Binary Search

Binary Search termasuk ke dalam interval search, dimana algoritma ini merupakan algoritma pencarian pada array/list dengan elemen terurut. Pada metode ini, data harus diurutkan terlebih dahulu dengan cara data dibagi menjadi dua bagian (secara logika), untuk setiap tahap pencarian. Dalam penerapannya algoritma ini sering digabungkan dengan algoritma sorting karena data yang akan digunakan harus sudah terurut terlebih dahulu. Konsep Binary Search:

- Data diambil dari posisi 1 sampai posisi akhir N.
- Kemudian data akan dibagi menjadi dua untuk mendapatkan posisi data tengah.
- Selanjutnya data yang dicari akan dibandingkan dengan data yang berada di posisi tengah, apakah lebih besar atau lebih kecil.
- Apabila data yang dicari lebih besar dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kanan dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada bagian kanan dengan acuan posisi data tengah akan menjadi posisi awal untuk pembagian tersebut.
- Apabila data yang dicari lebih kecil dari data tengah, maka dapat dipastikan bahwa data yang dicari kemungkinan berada di sebelah kiri dari data tengah. Proses pencarian selanjutnya akan dilakukan pembagian data menjadi dua bagian pada bagian kiri. Dengan acuan posisi data tengah akan menjadi posisi akhir untuk pembagian selanjutnya.
- Apabila data belum ditemukan, maka pencarian akan dilanjutkan dengan kembali membagi data menjadi dua.

- Namun apabila data bernilai sama, maka data yang dicari langsung ditemukan dan pencarian dihentikan.

Algoritma pencarian biner dapat dituliskan sebagai berikut :

1. $L = 0$
2. $R = N - 1$
3. ketemu false
4. Selama $(L \leq R)$ dan (tidak ketemu) kerjakan baris
5. sampai dengan 8 5) $m = (L + R) / 2$
6. Jika $(Data[m] = x)$ maka ketemu true
7. Jika $(x < Data[m])$ maka $R = m - 1$
8. Jika $(x > Data[m])$ maka $L = m + 1$
9. Jika (ketemu) maka m adalah indeks dari data yang dicari, jika tidak data tidak ditemukan

BAB III GUIDED

Guided 1

```

#include <iostream> using
namespace std;

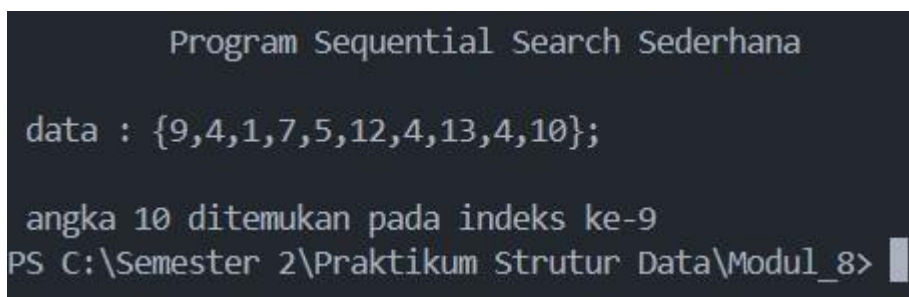
int main() {
int n = 10;
    int data[n] =
{9,4,1,7,5,12,4,13,4,10};    int cari =
10;        bool ketemu = false;    int
i;

    // algoritma Sequential
Search    for (i = 0; i < n;
i++) {        if (data[i] ==
cari) {            ketemu =
true;                break;
        }
    }
    cout << "\t Program Sequential Search Sederhana\n"
<< endl;
    cout << " data : {9,4,1,7,5,12,4,13,4,10};" << endl;

    if (ketemu ) {
        cout << "\n angka " << cari << " ditemukan
pada indeks ke-" << i << endl;
    } else {
        cout << cari << " tidak dapat ditemukan pada
data." << endl;
    }
}

```

Screenshots Output



```

Program Sequential Search Sederhana

data : {9,4,1,7,5,12,4,13,4,10};

angka 10 ditemukan pada indeks ke-9
PS C:\Semester 2\Praktikum Strutur Data\Modul_8>

```

Deskripsi:

Program diatas merupakan implementasi dari sequential search. Program ini mencari angka cari dalam array data menggunakan algoritma pencarian sekuensial. Algoritma ini memeriksa setiap elemen dalam array satu per satu hingga menemukan nilai yang dicari atau mencapai akhir array. Jika nilai yang dicari ditemukan, program akan memberitahu

pada indeks berapa nilai tersebut ditemukan. Jika tidak ditemukan, program akan memberitahu bahwa nilai tersebut tidak ada dalam array.

Guided 2

```
#include <iostream> using namespace
std; #include <conio.h> #include
<iomanip> int data[7] = {1, 8, 2, 5,
4, 9, 7}; int cari; void
selection_sort()
{      int temp, min, i, j;
for (i = 0; i < 7; i++)
    {          min = i;          for
(j = i + 1; j < 7; j++)
        {          if (data[j] <
data[min])
            {
min = j;
            }          }
temp = data[i];
data[i] = data[min];
data[min] = temp;
    } } void
binarysearch()
```

```

{
    // searching    int awal, akhir,
    tengah, b_flag = 0;    awal = 0;
    akhir = 7;    while (b_flag == 0 &&
    awal <= akhir)
    {
        tengah = (awal +
    akhir) / 2;    if (data[tengah]
    == cari)
    {
        b_flag = 1;
        break;
    }    else if
    (data[tengah] < cari)
    awal = tengah + 1;    else
    akhir = tengah - 1;
    }    if (b_flag
    == 1)
        cout << "\n Data ditemukan pada index ke-
    "<<tengah<<endl;
    else cout
        << "\n Data tidak ditemukan\n";
    } int
    main()
    {
        cout << "\t BINARY SEARCH " << endl;    cout <<
        "\n Data : ";    // tampilkan data awal    for (int
        x = 0; x < 7; x++)    cout << setw(3) << data[x];
        cout << endl;    cout << "\n Masukkan data yang ingin
        Anda cari :";
    }
}

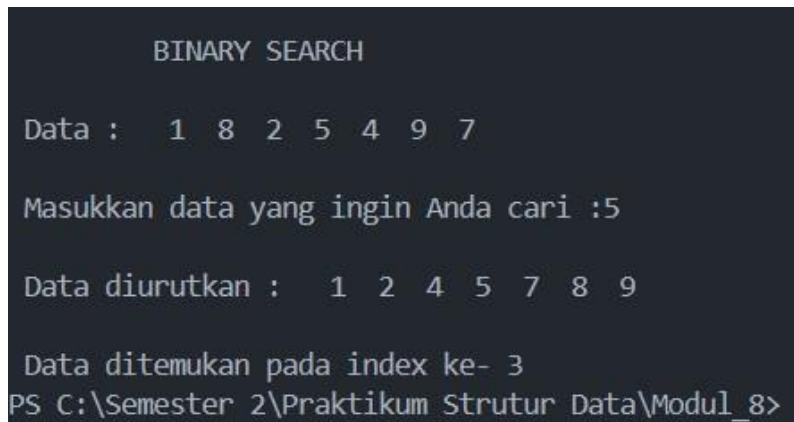
```

```

        cin >> cari;      cout << "\n Data
diurutkan : ";          // urutkan data
dengan selection sort
selection_sort();
        // tampilkan data setelah
diurutkan      for (int x = 0; x < 7;
x++)          cout << setw(3) <<
data[x];      cout << endl;
binarysearch();    _getche();
return EXIT_SUCCESS;
}

```

Screenshots Output:



```

BINARY SEARCH

Data :  1  8  2  5  4  9  7

Masukkan data yang ingin Anda cari :5

Data diurutkan :  1  2  4  5  7  8  9

Data ditemukan pada index ke- 3
PS C:\Semester 2\Praktikum Strukur Data\Modul_8>

```

Deskripsi :

Program di atas merupakan implementasi dari binary search. Program ini menggabungkan dua algoritma yaitu selection sort untuk mengurutkan array dan binary search untuk mencari elemen dalam array yang telah diurutkan. Pengguna diminta untuk memasukkan elemen yang ingin dicari, dan program akan menampilkan apakah elemen tersebut ditemukan dan pada indeks berapa.

UNGUIDED

Unguided 1

```
#include <iostream>
#include <algorithm>
#include <string>

using namespace std;

void selection_sort(string &str) {
    int n = str.length();
    for (int i = 0; i < n - 1; i++) {
        int min_idx = i;
        for (int j = i + 1; j < n; j++) {
            if (str[j] < str[min_idx]) {
                min_idx = j;
            }
        }
        swap(str[i], str[min_idx]);
    }
}

int binary_search(const string &str, char target) {
    int left = 0, right = str.length() - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (str[mid] == target)
            return mid;
        if (str[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}

int main() {
    string sentence_2311102024;
    char target_2311102024;

    cout << "Masukkan kalimat: ";
    getline(cin, sentence_2311102024);

    cout << "Masukkan huruf yang ingin dicari: ";
    cin >> target_2311102024;

    selection_sort(sentence_2311102024);

    int index = binary_search(sentence_2311102024,
target_2311102024);
```

```

        if (index != -1)
            cout << "Huruf " << target_2311102024 << " ditemukan
pada indeks ke-" << index << endl;
        else
            cout << "Huruf " << target_2311102024 << " tidak
ditemukan dalam kalimat." << endl;

        return 0;
    }

```

Screenshots Output:

```

Masukkan kalimat: fahri
Masukkan huruf yang ingin dicari: f
Huruf f ditemukan pada indeks ke-1
PS C:\Users\fahri\OneDrive\Documents\Modul_8> 

```

Deskripsi:

Program di atas merupakan program untuk mengurutkan huruf-huruf dalam kalimat yang diinput menggunakan selection sort dan kemudian mencari huruf tertentu dalam kalimat yang sudah diurutkan menggunakan binary search. Program ini menggabungkan dua algoritma yaitu selection sort untuk mengurutkan huruf-huruf dalam kalimat dan binary search untuk mencari huruf tertentu dalam kalimat yang sudah diurutkan. Program ini membantu pengguna menemukan posisi huruf tertentu dalam kalimat.

Unguided 2

```

#include <iostream>

#include <string>

using namespace std;

bool is_vowel(char c) {
    char vowels_2311102024[] = {'a', 'e', 'i', 'o', 'u', 'A',
'E', 'I', 'O', 'U'};

    for (char vowel_2311102024 : vowels_2311102024) {
        if (c == vowel_2311102024) return true;
    }
}

```

```

        return false;
    }

    int count_vowels(const string &sentence_2311102024) {
        int count = 0;
        for (char c : sentence_2311102024) {
            if (is_vowel(c)) {
                count++;
            }
        }
        return count;
    }

    int main() {
        string sentence_2311102024;

        cout << "Masukkan kalimat: ";
        getline(cin, sentence_2311102024);

        int vowel_count = count_vowels(sentence_2311102024);

        cout << "Jumlah huruf vokal dalam kalimat: " << vowel_count
        << endl;

        return 0;
    }

```

Screenshots Output:

```

Masukkan kalimat: shibal sekiya
Jumlah huruf vokal dalam kalimat: 5
PS C:\Users\fahri\OneDrive\Documents\Modul_8> 

```

Deskripsi:

Program di atas merupakan program untuk menghitung berapa huruf vokal dalam kalimat. Program ini menghitung jumlah huruf vokal dalam sebuah kalimat dengan menggunakan fungsi `is_vowel` untuk memeriksa setiap karakter apakah merupakan huruf vokal dan fungsi `count_vowels` untuk melakukan perhitungan tersebut. Hasilnya adalah jumlah huruf vokal dalam kalimat yang dimasukkan oleh pengguna.

Unguided 3

```
#include <iostream>

using namespace std;

int countNum(const int arr[], int size, int target) {
    int count = 0;
    for (int i = 0; i < size; ++i) {
        if (arr[i] == target) {
            count++;
        }
    }
    return count;
}

int main() {
    int data_2311102024[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int size_2311102024 = sizeof(data_2311102024) /
sizeof(data_2311102024[0]);
    int target_2311102024 = 4;

    int count = countNum(data_2311102024, size_2311102024,
target_2311102024);

    cout << "Jumlah angka " << target_2311102024 << " terdapat
sebanyak : " << count << endl;

    return 0;
}
```

Screenshots Output:

```
Jumlah angka 4 terdapat sebanyak : 4  
PS C:\Users\fahri\OneDrive\Documents\Modul_8>
```

Deskripsi:

Program di atas merupakan program untuk menghitung berapa banyak angka 4 didalam array. Program ini menggunakan loop for untuk mengiterasi array. Program ini menggunakan variabel count untuk menyimpan jumlah kemunculan angka target.

BAB V

KESIMPULAN

Kesimpulan yang dapat saya ambil, Searching adalah proses menemukan suatu nilai tertentu pada kumpulan data. Ada dua metode utama dalam algoritma searching, yaitu sequential search dan binary search. Sequential search digunakan data acak atau tidak terurut. Membandingkan elemen array satu per satu secara berurutan. Kelebihannya lebih sederhana. Kekurangannya lambat untuk data besar. Binary search digunakan untuk data terurut. Membagi data menjadi dua bagian secara berulang untuk menemukan elemen yang dicari. Kelebihannya cepat untuk data besar. Kekurangannya membutuhkan data terurut terlebih dahulu.

BAB VI DAFTAR PUSTAKA

- [1] Asisten Praktikum. 2024. Modul 8 “Searching”. Diakses 03 Juni 2024, 19:00 WIB. <https://lms.ittelkom-pwt.ac.id/>
- [2] Karumanchi, N. (2016). Data Structures and algorithms made easy: Concepts, problems, Interview Questions. CareerMonk Publications.