

LAPORAN PRAKTIKUM
MODUL 8
ALGORITMA SEARCHING



Disusun oleh:
Aji Tri Prasetyo
NIM : 2311102064

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI 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

Searching (pencarian) adalah proses menemukan lokasi dari suatu elemen dalam struktur data tertentu, seperti array atau list. Algoritma pencarian berperan penting dalam pemrograman karena banyak aplikasi memerlukan pencarian data secara cepat dan efisien. Berikut ini adalah beberapa konsep dasar dan algoritma pencarian yang umum digunakan dalam C++. Sequential search adalah metode pencarian paling sederhana di mana elemen dicari satu per satu dari awal sampai akhir struktur data. Algoritma ini tidak memerlukan data yang diurutkan. Binary search adalah metode pencarian yang lebih efisien dibandingkan sequential search tetapi memerlukan data yang diurutkan. Algoritma ini bekerja dengan membagi data menjadi dua bagian dan terus melakukan pembagian hingga elemen ditemukan atau ruang pencarian habis.

Ukuran Data: Untuk data kecil, perbedaan waktu antara algoritma mungkin tidak signifikan. Namun, untuk data besar, algoritma dengan kompleksitas waktu yang lebih rendah akan lebih efisien.

Keterurutan Data: Binary search memerlukan data yang diurutkan, sehingga jika data belum diurutkan, diperlukan langkah tambahan untuk mengurutkan data terlebih dahulu. Algoritma pencarian adalah komponen fundamental dalam pemrograman yang digunakan untuk menemukan elemen tertentu dalam struktur data. Sequential search dan binary search adalah dua algoritma pencarian dasar dengan karakteristik dan penggunaan yang berbeda. Memilih algoritma yang tepat bergantung pada ukuran data, keterurutan data, dan kebutuhan performa dari aplikasi yang dikembangkan.

BAB III

GUIDED

1. GUIDED 1

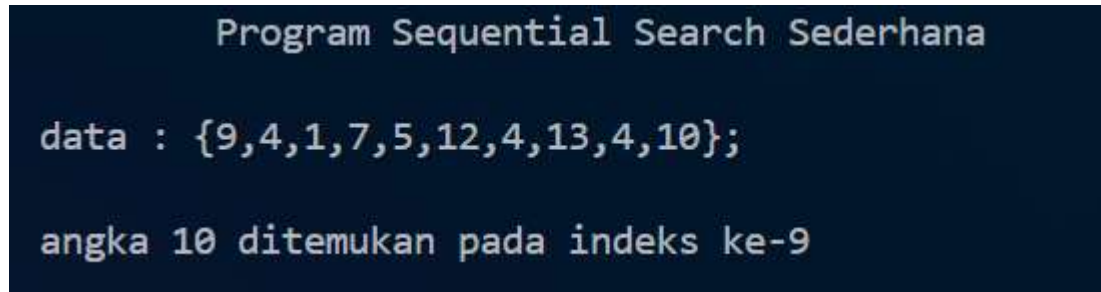
SOURCE CODE

```
#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;
    }
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Program menampilkan array data. Program mencari nilai 10 dalam array. Karena nilai 10 ditemukan pada indeks ke-9, program menampilkan pesan bahwa angka 10 ditemukan pada indeks ke-9.

2. GUIDED 2

SOURCE CODE

```
#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;
    }
}
```

```

    }

    }

    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()
{

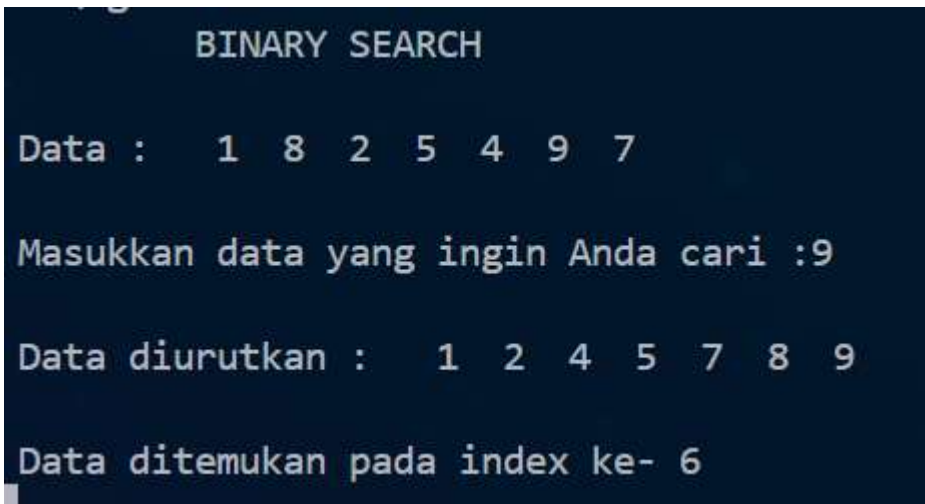
```

```

    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;
}

```

SCREENSHOOT PROGRAM



The screenshot shows the output of a C++ program titled "BINARY SEARCH". It displays the initial data array, the user input for the search value, the sorted array, and the final search result.

```

BINARY SEARCH

Data :   1   8   2   5   4   9   7

Masukkan data yang ingin Anda cari :9

Data diurutkan :   1   2   4   5   7   8   9

Data ditemukan pada index ke- 6

```

DESKRIPSI PROGRAM

Program menampilkan array data. Program mencari nilai 10 dalam array. Karena nilai 10 ditemukan pada indeks ke-9, program menampilkan pesan bahwa angka 10 ditemukan pada indeks ke-9.

UNGUIDED

1. UNGUIDED 1

Buatlah sebuah program untuk mencari sebuah huruf pada sebuah kalimat yang sudah di input dengan menggunakan Binary Search!

SOURCE CODE

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;

int binarySearch(const string& sentence, char target) {
    int left = 0;
    int right = sentence.length() - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (sentence[mid] == target) {
            return mid;
        }

        if (sentence[mid] < target) {
            left = mid + 1;
        }
    }
}
```



```

        } else {
            right = mid - 1;
        }
    }

    return -1;
}

int main() {
    string sentence;
    char target;

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

    sort(sentence.begin(), sentence.end());

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

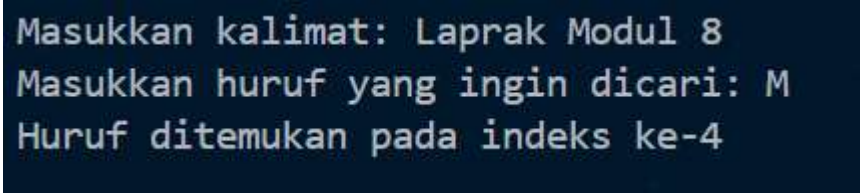
    int index = binarySearch(sentence, target);

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

    return 0;
}

```

SCREENSHOOT PROGRAM



```
Masukkan kalimat: Laprak Modul 8
Masukkan huruf yang ingin dicari: M
Huruf ditemukan pada indeks ke-4
```

DESKRIPSI PROGRAM

Fungsi `binarySearch` Fungsi ini melakukan pencarian biner (binary search) pada string yang sudah diurutkan. Parameter `const string& sentence` merujuk pada kalimat yang akan dicari. Parameter `char target` adalah karakter yang akan dicari dalam kalimat. Menggunakan dua indeks, `left` dan `right`, untuk menandai batas pencarian.

2. UNGUIDED 2

Buatlah sebuah program yang dapat menghitung banyaknya huruf vocal dalam sebuah kalimat!

SOURCE CODE

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string kalimat;
    int jumlahVokal = 0;

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

    for (char huruf : kalimat) {
        if (huruf == 'a' || huruf == 'i' || huruf == 'u' || huruf
            == 'e' || huruf == 'o' ||
```

```

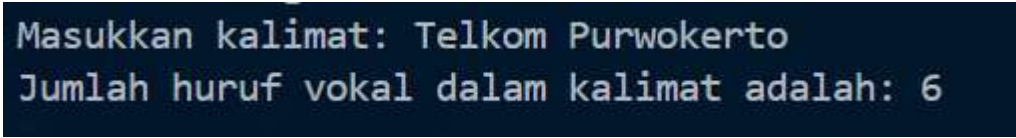
        huruf == 'A' || huruf == 'I' || huruf == 'U' || huruf
== 'E' || huruf == 'O') {
            jumlahVokal++;
        }
    }

    cout << "Jumlah huruf vokal dalam kalimat adalah: " <<
jumlahVokal << endl;

    return 0;
}

```

SCREENSHOOT PROGRAM



```

Masukkan kalimat: Telkom Purwokerto
Jumlah huruf vokal dalam kalimat adalah: 6

```

DESKRIPSI PROGRAM

Loop for (char huruf : kalimat): Menggunakan loop for berbasis rentang untuk iterasi langsung melalui setiap karakter dalam string kalimat.

3. UNGUIDED 3

Diketahui data = 9, 4, 1, 4, 7, 10, 5, 4, 12, 4. Hitunglah berapa banyak angka 4 dengan menggunakan algoritma Sequential Search!

SOURCE CODE

```

#include <iostream>
using namespace std;

int sequentialSearch(int arr[], int n, int key) {
    int count = 0;
    for (int i = 0; i < n; i++) {

```

```
        if (arr[i] == key) {
            count++;
        }
    }
    return count;
}

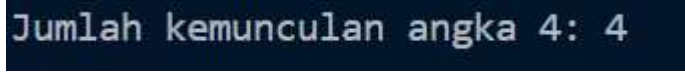
int main() {
    int data[] = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};
    int key = 4;
    int n = sizeof(data) / sizeof(data[0]);

    int occurrences = sequentialSearch(data, n, key);

    cout << "Jumlah kemunculan angka 4: " << occurrences << endl;

    return 0;
}
```

SCREENSHOOT PROGRAM

A screenshot of a terminal window with a dark background. The text "Jumlah kemunculan angka 4: 4" is displayed in a light blue, monospaced font.

DESKRIPSI PROGRAM

Fungsi `sequentialSearch` mengembalikan jumlah kemunculan elemen `key` dalam array setelah iterasi selesai.

BAB IV

KESIMPULAN

Setelah melakukan pembelajaran mengenai Queue di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Untuk data kecil atau tidak diurutkan, sequential search adalah pilihan yang mudah dan cepat diimplementasikan.
2. Untuk data besar dan terurut, binary search menawarkan kecepatan dan efisiensi yang jauh lebih tinggi.

DAFTAR PUSTAKA

Adi Santoso. (19 July, 2017) Pengertian Searching Beserta Jenis dan Contoh Program Searching pada C++. diakses pada 02 Juni 2024 dari <https://onlyvista.blogspot.com/2017/07/pengertian-searching-jenis-jenis.html>