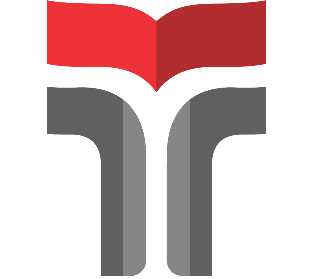
**LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA**

**MODUL VII**

**ALGORITMA SEARCHING**

# Disusun Oleh :

NAMA : Aulia Radix Putra Winarko  
NIM : 2311102056

# Dosen

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA   
FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

1. **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:

1. **Operasi pada Queue**

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

1. **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

1. **Pencarian Hashing**

Algoritma ini menggunakan tabel hash untuk memetakan elemen ke dalam indeks tertentu. Pencarian hashing sangat cepat untuk menemukan elemen, dengan kompleksitas waktu rata-rata O(1). Namun, algoritma ini membutuhkan memori tambahan untuk menyimpan tabel hash.

1. **Pencarian Pohon (Tree Search)**

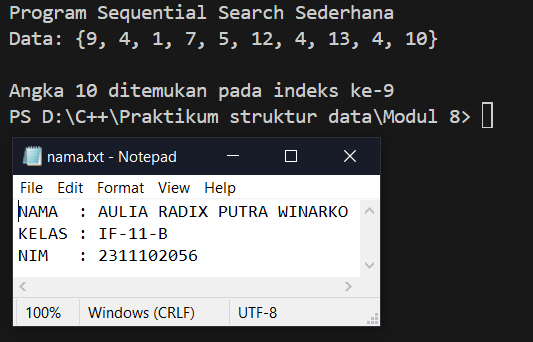
Algoritma ini menggunakan struktur data pohon untuk mencari elemen. Algoritma ini dapat digunakan untuk mencari elemen dalam struktur data .

1. **Guided** [

Guided 1

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {      const int n = 10;      int \*data = new int[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 << "Program Sequential Search Sederhana" << endl;      cout << "Data: {9, 4, 1, 7, 5, 12, 4, 13, 4, 10}" << endl;      if (ketemu)      {          cout << "\nAngka " << cari << " ditemukan pada indeks ke-" << i << endl;      }      else      {          cout << cari << "tidak dapat ditemukan pada data." << endl;      }      return 0;  } |

Screenshots Output



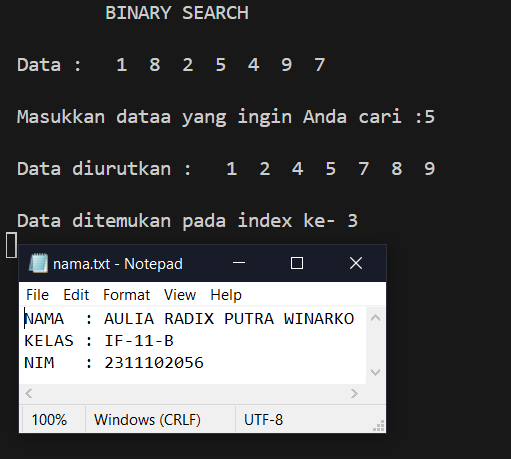
Deskripsi:

Kode diatas menggunakan binary search untuk mencari apakah ada salah satu huruf didalam kalimat yang diinputkan sebelumnya. Untuk kalimat dan kata akan diinputkan oleh pengguna dan diakhir akan ada aksi dimana kondisi bernilai benar dan salah

Guided 2

|  |
| --- |
| #include <iostream>  using namespace std;  #include <conio.h>  #include <iomanip>  int dataa[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 (dataa[j] < dataa[min])              {                  min = j;              }          }          temp = dataa[i];          dataa[i] = dataa[min];          dataa[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 (dataa[tengah] == cari)          {              b\_flag = 1;              break;          }          else if (dataa[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 dataa awal      for (int x = 0; x < 7; x++)          cout << setw(3) << dataa[x];      cout << endl;      cout << "\n Masukkan dataa yang ingin Anda cari :";      cin >> cari;      cout << "\n Data diurutkan : ";      // urutkan dataa dengan selection sort      selection\_sort();      // tampilkan dataa setelah diurutkan      for (int x = 0; x < 7; x++)          cout << setw(3) << dataa[x];      cout << endl;      binarysearch();      \_getche();      return EXIT\_SUCCESS;  } |

Screenshots Output



Deskripsi:

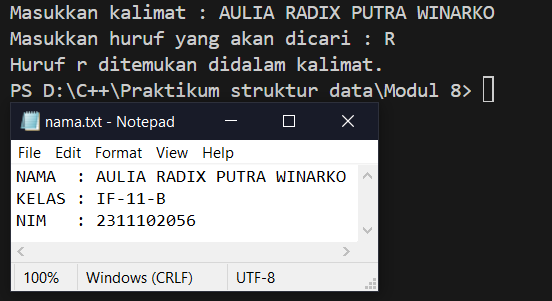
Kode di atas menggunakan binary search untuk mencari angka yang berada pada array yang sudah dibuat dan diberikan nilai sebelumnya, untuk angka yang dicari nantinya pengguna akan diminta memasukkan angka yang sudah tertera diatas untuk mencari angka tersebut ada diposisi indeks berapa

1. **Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)**

Unguided 1

|  |
| --- |
| #include <iostream>  #include <string>  #include <algorithm>  using namespace std;  bool binarySearch(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 true;          }          if (sentence[mid] < target)          {              left = mid + 1;          }          else          {              right = mid - 1;          }      }      return false;  }  int main()  {      string sentence;      char target;      cout << "Masukkan kalimat : ";      getline(cin, sentence);      cout << "Masukkan huruf yang akan dicari : ";      cin >> target;      // Mengubah seluruh huruf menjadi lowercase untuk pencarian yang tidak case-sensitive      transform(sentence.begin(), sentence.end(), sentence.begin(), ::tolower);      target = tolower(target);      if (binarySearch(sentence, target))      {          cout << "Huruf " << target << " ditemukan didalam kalimat." << endl;      }      else      {          cout << "Huruf tidak ditemukan didalam kalimat." << endl;      }      return 0;  } |

Screenshots Output



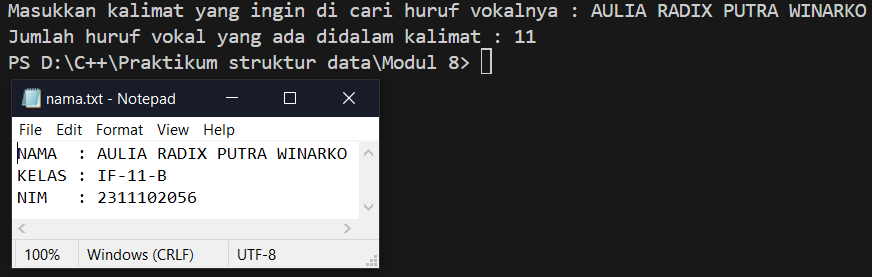
Deskripsi:

Kode diatas menggunakan binary search untuk mencari apakah ada salah satu huruf didalam kalimat yang diinputkan sebelumnya. Untuk kalimat dan kata akan diinputkan oleh pengguna dan diakhir akan ada aksi dimana kondisi bernilai benar dan salah.

Unguided 2

|  |
| --- |
| #include <iostream>  #include <string>  #include <algorithm>  using namespace std;  int countVowels(string sentence)  {      int count = 0;      transform(sentence.begin(), sentence.end(), sentence.begin(), ::tolower);      for (char c : sentence)      {          if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')          {              count++;          }      }      return count;  }  int main()  {      string sentence;      cout << "Masukkan kalimat yang ingin di cari huruf vokalnya : ";      getline(cin, sentence);      int vowelCount = countVowels(sentence);      cout << "Jumlah huruf vokal yang ada didalam kalimat : " << vowelCount << endl;      return 0;  } |

Screenshots Output



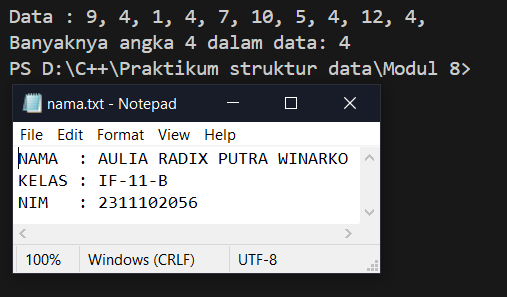
Deskripsi:

Kode di atas adalah implementasi dari antrian (queue) menggunakan struktur data linked list untuk menyimpan data mahasiswa dengan atribut Nama dan NIM. Program ini menggunakan MENU yang memungkinkan user untuk melakukan operasi pada antrian melalui menu interaktif.

Unguided 3

|  |
| --- |
| #include <iostream>  #include <vector>  // Fungsi untuk menghitung berapa banyak angka 4 dalam data  int countNumber(const std::vector<int> &data, int target)  {      int count = 0;      for (int num : data)      {          if (num == target)          {              count++;          }      }      return count;  }  int main()  {      std::vector<int> data = {9, 4, 1, 4, 7, 10, 5, 4, 12, 4};      int target = 4;      // int ukuran = sizeof(data) / sizeof(data[0]);      int ukuran = 10;      // Menghitung berapa banyak angka 4 dalam data      int count = countNumber(data, target);      std::cout << "\nData : ";      for (int i = 0; i < ukuran; i++)      {          std::cout << data[i] << ", ";      }      std::cout << "\nBanyaknya angka " << target << " dalam data: " << count << std::endl;      return 0;  } |

Screenshots Output



Deskripsi:

Untuk program diatas menggunakan sequential search untuk menghitung berapa kali angka 4 muncul dalam sebuah data. Data yang diberikan adalah 9, 4, 1, 4, 7, 10, 5, 4, 12, 4. Program akan melakukan perulangan pada setiap elemen data dan memeriksa apakah angka tersebut sama dengan 4. Jika iya, maka counter akan bertambah satu. Setelah perulangan selesai, program akan mengembalikan jumlah angka 4 yang ditemukan dan menampilkannya. Dengan demikian, program ini dapat digunakan untuk menghitung kemunculan angka 4 dalam data menggunakan algoritma Sequential Search.

1. Kesimpulan

Kesimpulan yang dapat diambil dari laporan praktikum dengan “Modul 8 : ALGORITMA SEARCHING” adalah bahwa praktikum ini bertujuan untuk mempelajari fungsi dari penggunaan Sequential Searching dan Binary Searching dalam pemrogramman. Praktisi jadi bisa memahami penggunaan dari konsep tersebut dan bagaimana kegunaan dari konsep tersebut dalam program yang dibuat, serta setelah melakukan pratikum, mahasiswa mampu menerapkan konsep tersebut didalam program yang mereka buat.

Karumanchi, N. (2016). Data Structures and algorithms made easy: Concepts, problems, Interview Questions. CareerMonk Publications.