LAPORAN PRATIKUM STRUKTUR DATA DAN ALGORITMA

“BINARY SEARCH”

NAMA : TASYA ULFA RAHMAZANI

NIM : 2023573010061

KELAS : TI.1E

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER
POLITEKNIK NEGERI LHOKSEUMAWE

2024

LEMBAR PENILAIAN

Kode Mata Kuliah	: TI2018
No Praktikum	: 02/TIK/TI.1E/2024
Judul Praktikum	: Binary Search
Tanggal Pelaksana Praktikum	: 13 Maret 2024
Tanggal Pengumpulan Laporan	: 20 Maret 2024
Dosen Pengampuh	: Hendrawaty, ST., MT.
Nilai	:

DAFTAR ISI

C. Percobaan

1. Buatlah program binary search berdasarkan langkah-langkah yang telah dijelaskan diatas, dimana data pada larik/array terurut menaik.

   Algoritma

   Deklarasi:

   - Deklarasikan array L berisi data yang ingin dicari.
   - Deklarasikan variabel X untuk menyimpan nilai yang ingin dicari.
   - Deklarasikan variabel i dan j untuk menandai batas awal dan akhir pencarian.

   Melakukan pencarian:

   - Lakukan loop while selama batas awal (i) masih lebih kecil atau sama dengan batas akhir (j).

   Menghitung nilai tengah:

   - Di dalam loop, hitung nilai tengah (k) dengan rumus k = i + (j - i) / 2.

   Membandingkan nilai di tengah:

   - Bandingkan nilai di tengah (L[k]) dengan nilai yang dicari (X).
   - Jika sama, nilai ditemukan dan indeksnya ditampilkan.
   - Jika lebih kecil, geser batas kiri (i) ke kanan (i = k + 1).
   - Jika lebih besar, geser batas kanan (j) ke kiri (j = k - 1).

   Keluar dari loop:

   - Jika loop selesai dan batas awal (i) lebih besar dari batas akhir (j), nilai tidak ditemukan.

   Menampilkan hasil:

   - Jika nilai ditemukan, tampilkan indeksnya.
   - Jika nilai tidak ditemukan, tampilkan pesan bahwa nilai tidak ditemukan.

Source Code:

```cpp
#include <iostream>

using namespace std;

int main() {
  //Deklarasi
  int L[10] = {12, 14, 15, 17, 23, 25, 45, 67, 68, 70};
  int X = 14, i = 0, j = 10 - 1;

  //Melakukan pencarian dengan algoritma binary search
  while (i<=j) {
    // 4. Menghitung nilai tengah
    int k = i + (j - i) / 2;

    //Membandingkan nilai di tengah dengan nilai yang dicari
    if (L[k] == X) {
      cout << "Nilai " << X << " ditemukan pada indeks " << k << endl;
      return 0;
    } else if (L[k] < X) {
      // Jika nilai di tengah lebih kecil, geser batas kiri ke kanan
      i = k + 1;
    } else {
      // Jika nilai di tengah lebih besar, geser batas kanan ke kiri
      j = k - 1;
    }
  }

  // Nilai tidak ditemukan
  cout << "Nilai " << X << " tidak ditemukan" << endl;

  return 0;
}

```

Output:

```
Nilai 14 ditemukan pada indeks 1
```

2. Buatlah program binary search berdasarkan langkah-langkah yang telah dijelaskan diatas, dimana data pada larik/array terurut menurun.

   Algoritma

   Deklarasi:

   - Deklarasikan array L berisi data yang ingin dicari.
   - Deklarasikan variabel X untuk menyimpan nilai yang ingin dicari.
   - Deklarasikan variabel i dan j untuk menandai batas awal dan akhir pencarian.

   Melakukan pencarian:

   - Lakukan loop while selama batas awal (i) masih lebih kecil atau sama dengan batas akhir (j).

   Menghitung nilai tengah:

   - Di dalam loop, hitung nilai tengah (k) dengan rumus k = i + (j - i) / 2.

   Membandingkan nilai di tengah:

   - Bandingkan nilai di tengah (L[k]) dengan nilai yang dicari (X).
   - Jika sama, nilai ditemukan dan indeksnya ditampilkan.
   - Jika lebih besar, geser batas kiri (i) ke kanan (i = k + 1).

- Jika lebih kecil, geser batas kanan (j) ke kiri (j = k - 1).

Keluar dari loop:

- Jika loop selesai dan batas awal (i) lebih besar dari batas akhir (j), nilai tidak ditemukan.

Menampilkan hasil:

- Jika nilai ditemukan, tampilkan indeksnya.
- Jika nilai tidak ditemukan, tampilkan pesan bahwa nilai tidak ditemukan.

Source Code:

```cpp
#include <iostream>

using namespace std;

int main() {
  //Deklarasi
  int L[10] = {70, 68, 67, 45, 25, 23, 17, 15, 14, 12};
  int X = 14;
  int i = 0;
  int j = 10 - 1;

  //Melakukan pencarian dengan algoritma binary search
  while (i <= j) {
    //Menghitung nilai tengah
    int k = i + (j - i) / 2;

    //Membandingkan nilai di tengah dengan nilai yang dicari
    if (L[k] == X) {
      cout << "Nilai " << X << " ditemukan pada indeks " << k << endl;
      return 0;
    } else if (L[k] > X) {
      //Jika nilai di tengah lebih besar, geser batas kiri ke kanan
      i = k + 1;
    } else {
      //Jika nilai di tengah lebih kecil, geser batas kanan ke kiri
      j = k - 1;
    }
  }

  // Nilai tidak ditemukan
  cout << "Nilai " << X << " tidak ditemukan" << endl;

  return 0;
}
```

Output:

```
Nilai 14 ditemukan pada indeks 8
```

ANALISIS HASIL PERCOBAAN

1. **Deklarasi:**
   - **int L[10] = {12, 14, 15, 17, 23, 25, 45, 67, 68, 70};**: Mendeklarasikan array L berisi 10 elemen dengan nilai awal yang sudah terurut naik.
   - **int X = 14;**: Mendeklarasikan variabel X untuk menyimpan nilai yang ingin dicari (14).
   - **int i = 0;**: Mendeklarasikan variabel i untuk menandai batas awal pencarian, diinisialisasi dengan 0.
   - **int j = 10 - 1;**: Mendeklarasikan variabel j untuk menandai batas akhir pencarian, diinisialisasi dengan 9 (indeks terakhir array).

   Melakukan Pencarian:
   - **while (i <= j)**: Loop while berulang selama batas awal (i) masih lebih kecil atau sama dengan batas akhir (j).
   - **k = i + (j - i) / 2;**: Menghitung nilai tengah (k) dengan rumus k = (batas awal + batas akhir) / 2.
   - **if (L[k] == X)**: Membandingkan nilai di tengah (L[k]) dengan nilai yang dicari (X).
   - Jika sama (L[k] == X), nilai ditemukan dan indeksnya ditampilkan (cout << "Nilai " << X << " ditemukan pada indeks " << k << endl;) dan program keluar (return 0;).
   - **else if (L[k] < X)**: Jika nilai di tengah lebih kecil (L[k] < X), geser batas kiri (i) ke kanan (i = k + 1).
   - else: Jika nilai di tengah lebih besar (L[k] > X), geser batas kanan (j) ke kiri (j = k - 1).

   Nilai Tidak Ditemukan:
   - Jika loop while selesai dan i lebih besar dari j, nilai tidak ditemukan (cout << "Nilai " << X << " tidak ditemukan" << endl;).

2. **Deklarasi:**
   - **int L[10] = {70, 68, 67, 45, 25, 23, 17, 15, 14, 12};**: Mendeklarasikan array L berisi 10 elemen dengan nilai awal yang sudah terurut menurun.
   - **int X = 14;**: Mendeklarasikan variabel X untuk menyimpan nilai yang ingin dicari (14).
   - **int i = 0;**: Mendeklarasikan variabel i untuk menandai batas awal pencarian, diinisialisasi dengan 0.
   - **int j = 10 - 1;**: Mendeklarasikan variabel j untuk menandai batas akhir pencarian, diinisialisasi dengan 9 (indeks terakhir array).

   Melakukan Pencarian:
   - **while (i <= j)**: Loop while berulang selama batas awal (i) masih lebih kecil atau sama dengan batas akhir (j).
   - **int k = i + (j - i) / 2;**: Menghitung nilai tengah (k) dengan rumus k = (batas awal + batas akhir) / 2.
   - **if (L[k] == X)**: Membandingkan nilai di tengah (L[k]) dengan nilai yang dicari (X).
   - Jika sama (L[k] == X), nilai ditemukan dan indeksnya ditampilkan (cout << "Nilai " << X << " ditemukan pada indeks " << k << endl;) dan program keluar (return 0;).
   - **else if (L[k] > X)**: Jika nilai di tengah lebih besar (L[k] > X), geser batas kiri (i)

ke kanan (i = k + 1). Perhatikan arah perbandingan dan pembaruan batas ini berbeda dengan array terurut naik.

- else: Jika nilai di tengah lebih kecil (L[k] < X), geser batas kanan (j) ke kiri (j = k - 1). Perhatikan arah perbandingan dan pembaruan batas ini berbeda dengan array terurut naik.

Nilai Tidak Ditemukan:

- Jika loop while selesai dan i lebih besar dari j, nilai tidak ditemukan (cout << "Nilai " << X << " tidak ditemukan" << endl;).

KESIMPULAN

Dari hasil percobaan yang dilakukan, adapun beberapa kesimpulan yaitu:

1. Program ini menerapkan algoritma binary search untuk mencari nilai dalam array terurut naik.
    - Deklarasi array terurut: Array L berisi data yang sudah terurut naik.
    - Perhitungan nilai tengah: Nilai tengah dihitung dengan rumus k = (batas awal + batas akhir) / 2.
    - Perbandingan dengan nilai yang dicari: Nilai di tengah dibandingkan dengan nilai yang dicari.
    - Pembaruan batas pencarian: Batas pencarian diperbarui berdasarkan hasil perbandingan.
    - Penghentian loop: Loop berhenti ketika nilai ditemukan atau batas awal lebih besar dari batas akhir.

2. Program ini menerapkan algoritma binary search untuk mencari nilai dalam array terurut menurun.
    - Deklarasi array terurut: Array L berisi data yang sudah terurut menurun.
    - Perhitungan nilai tengah: Nilai tengah dihitung dengan rumus k = (batas awal + batas akhir) / 2.
    - Perbandingan dengan nilai yang dicari: Nilai di tengah dibandingkan dengan nilai yang dicari.
    - Pembaruan batas pencarian: Batas pencarian diperbarui berdasarkan arah perbandingan dan nilai yang dicari.
    - Penghentian loop: Loop berhenti ketika nilai ditemukan atau batas awal lebih besar dari batas akhir.