MINGGU 13
Searching

DESKRIPSI TEMA

Searching merupakan suatu fungsi pencari yang dirancang untuk mencari suatu data dari sekumpulan data. Pencarian data dapat dicari dalam kelompok data yang terurut (sorted) maupun tidak terurut (unsorted). Pada pertemuan ini, kita akan mempelajari beberapa algoritma untuk searching, yaitu Sequential Search, Bianry Search, Interpolation Search, dan Jump Search.

Sequential Search atau disebut juga sebagai Linear Search merupakan algoritma untuk pencarian data yang dilakukan secara beruntun dari depan (awal) hingga ke belakang (akhir). Pada Sequential Search, ddata **tidak perlu** diurutkan terlebih dahulu.

Binary Search merupakan algoritma untuk pencarian data yang memerlukan data yang **telah** di-sort terlebih dahulu, dilakukan secara berulang-ulang dengan membagi search interval menjadi 2 bagian satu per satu. Langkah – langkah Binary Search adalah sebagai berikut:

1. Urutkan data terlebih dahulu secara ascending (dapat diurutkan secara descending, namun pada tutorial ini hanya menggunakan cara sorting ascending saja)
2. Membuat variabel seperti low, mid, dan high sebagai penanda / pembatas suatu data dan melakukan input data. Input yang dimasukkan akan dijadikan key dalam pencarian.
3. Melakukan pengulangan yang terdiri dari beberapa proses:
    a. Menghitung nilai mid → mid = (low + high) / 2
    b. Membandingkan key dengan data yang ada di indeks mid. Proses ini menghasilkan 3 kondisi, yaitu:
        a) Jika array[mid] == key → proses selesai
        b) Jika array[mid] < key → high = mid – 1
        c) Jika array[mid] > key → low = mid + 1
4. Kembali ke proses ketiga hingga nilai low > high atau array[mid] == key

Interpolation Search merupakan algoritma untuk pencarian data yang dilakukan pada sekumpulan data dan merupakan pengembangan dari Binary Search. Apabila pada Binary Search selalu memulai pengecekan pada indeks mid, Interpolation Search akan memulai pengecekan di lokasi indeks yang berbeda, tergantung dengan nilai key yang akan dicari. Adapun rumus yang digunakan dalam tiap iterasi adalah sebagai berikut.

$$Posisi\ Relatif = \left\{\frac{kunci - data[low]}{data[high] - data[low]}\ x\ (high - low)\right\} + low$$

Jump Search merupakan algoritma untuk pencarian data tanpa harus mengecek semua elemen yang ada pada array. Indeks pencarian yang digunakan adalah akar kuadrat dari sebuah ukuran array. Oleh karena itu, indeksnya bergantung pada ukuran dari sebuah array. Langkah – langkah Jump Search adalah sebagai berikut.

1. Urutkan data terlebih dahulu secara ascending (dapat diurutkan secara descending, namun pada tutorial ini hanya menggunakan cara sorting ascending saja)
2. Jump (loncat) dari indeks awal.
   Variabel jump = akar kuadrat dari ukuran array
3. Cek kondisi dengan menggunakan perulangan
   Jika nilai array[jump] <= key **dan** jump < n
   Ubah variabel start menjadi jump
   Tambahkan variabel jump dengan nilai akar kuadrat ukuran indeks yang digunakan
   Jika jump > n-1 → jump = n
4. Jika perulangan selesai pada tahap 3, lakukan Linear Search dari array[start] hingga nilai jump terakhir.

CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

1. Mahasiswa mampu menerapkan Sequential (Linear) Search menggunakan Bahasa C
2. Mahasiswa mampu menerapkan Binary Search menggunakan Bahasa C
3. Mahasiswa mampu menerapkan Interpolation Search menggunakan Bahasa C
4. Mahasiswa mampu menerapkan Jump Search menggunakan Bahasa C

PENUNJANG PRAKTIKUM

1. CodeBlocks
2. Dev-C++ (alternatif)

LANGKAH-LANGKAH PRAKTIKUM

A. Tutorial 1.1 – Sequential Search
   - Buatlah sebuah file dengan nama W13_SequentialSearch.c
   - Salinlah *code* berikut ke dalam *file* yang telah dibuat

```c
#include <stdio.h>

int main(){
    int numbers[] = {2, 4, 3, 123, 23, 56, 7895, 456, 342, 67, 2345};
    int size = sizeof(numbers) / 4; // Size dari array numbers.
    int searchKey; // Key yang akan dicari
    int found = 0;
    int i;

//  printf("Sizeof numbers = %d\n", size);
    printf("Masukkan angka yang ingin dicari: "); scanf("%d", &searchKey);

    // Linear Search
    for(i=0; i<size; i++){
        if(searchKey == numbers[i]) {
            found = 1;
            break;
        }
    }

    // Print hasil
    if(found) {
        printf("Angka %d ditemukan pada index %d", searchKey, i);
    } else {
        printf("Angka %d tidak ditemukan pada array numbers", searchKey);
    }

    return 0;
}
```

B. Tutorial 2.1 – Binary Search

- Buatlah sebuah file dengan nama W13_BinarySearch.c
- Salinlah *code* berikut ke dalam *file* yang telah dibuat

```c
#include <stdio.h>

int binarySearch(int *arr, int size, int key){
  int low, mid, high;

  low = 0;
  high = size - 1;

  while (low <= high){
    mid = (low + high) / 2;

    if(arr[mid] == key) return mid;
    if(key < arr[mid]) high = mid - 1;
    else low = mid + 1;
  }

  return -1;
}

void shellSort(int *bil, int n){···

void printArray(int *arr, int size){
  int i=0;

  for(i=0; i<size; i++) printf("%d ", arr[i]);
  printf("\n");
}
```

```c
int main(){
    int numbers[] = {2, 4, 3, 123, 23, 56, 7895, 456, 342, 67, 2345};
    int size = sizeof(numbers) / 4; // Size dari array numbers.
    int searchKey; // Key yang akan dicari
    int found = 0;

    printf("Array before sorting: "); printArray(numbers, size);
    shellSort(numbers, size);
    printf("Array after sorting: "); printArray(numbers, size);

    printf("Masukkan angka yang ingin dicari: "); scanf("%d", &searchKey);

    found = binarySearch(numbers, size, searchKey);

    // Print hasil
    if(found != -1) {
        printf("Angka %d ditemukan pada index %d", searchKey, found);
    } else {
        printf("Angka %d tidak ditemukan pada array numbers", searchKey);
    }

    return 0;
}
```

C. Tutorial 3.1 – Interpolation Search

- Buatlah sebuah file dengan nama W13_InterpolationSearch.c
- Salinlah *code* berikut ke dalam *file* yang telah dibuat

```c
#include <stdio.h>

int interpolationSearch(int *arr, int size, int key){
  int low, pos, high;

  low = 0;
  high = size - 1;
  printf("sizeof arr = %d\n", high);

  do{
    pos = ((key - arr[low]) * (high - low) / (arr[high] - arr[low])) + low;

    if(arr[pos] == key) return pos;
    if(arr[pos] > key) high = pos - 1;
    else low = pos + 1;
  }while(key >= arr[low] && key <= arr[high]);

  return -1;
}

void shellSort(int *bil, int n){ ···

void printArray(int *arr, int size){
  int i=0;

  for(i=0; i<size; i++) printf("%d ", arr[i]);
  printf("\n");
}
```

```c
int main(){
    int numbers[] = {2, 4, 3, 123, 23, 56, 7895, 456, 342, 67, 2345};
    int size = sizeof(numbers) / 4; // Size dari array numbers.
    int searchKey; // Key yang akan dicari
    int found = 0;

    printf("Array before sorting: "); printArray(numbers, size);
    shellSort(numbers, size);
    printf("Array after sorting: "); printArray(numbers, size);

    printf("Masukkan angka yang ingin dicari: "); scanf("%d", &searchKey);

    found = interpolationSearch(numbers, size, searchKey);

    // Print hasil
    if(found != -1) {
        printf("Angka %d ditemukan pada index %d", searchKey, found);
    } else {
        printf("Angka %d tidak ditemukan pada array numbers", searchKey);
    }

    return 0;
}
```

D. Tutorial 4.1 – Jump Search

- Buatlah sebuah file dengan nama W13_JumpSearch.c
- Salinlah *code* berikut ke dalam *file* yang telah dibuat

```c
#include <stdio.h>
#include <math.h>

int jumpSearch(int *arr, int size, int key){
  int start, jump, i;

  start = 0;
  jump = sqrt(size);
  printf("\nAkar kuadrat terdekat dari %d = %d\n", size, jump);

  while (arr[jump] <= key && jump < size){
    start = jump;
    jump += sqrt(size);
    if (jump > size - 1) jump = size;
  }

  for (i = start; i<jump; i++){
    if (arr[i] == key) return i;
  }

  return -1;
}

void shellSort(int *bil, int n){···

void printArray(int *arr, int size){
  int i=0;

  for(i=0; i<size; i++) printf("%d ", arr[i]);
  printf("\n");
}
```

```c
int main(){
    int numbers[] = {2, 4, 3, 123, 23, 56, 7895, 456, 342, 67, 2345};
    int size = sizeof(numbers) / 4; // Size dari array numbers.
    int searchKey; // Key yang akan dicari
    int found = 0;

    printf("Array before sorting: "); printArray(numbers, size);
    shellSort(numbers, size);
    printf("Array after sorting: "); printArray(numbers, size);

    printf("\nMasukkan angka yang ingin dicari: "); scanf("%d", &searchKey);

    found = jumpSearch(numbers, size, searchKey);

    // Print hasil
    if(found != -1) {
        printf("Angka %d ditemukan pada index %d\n", searchKey, found);
    } else {
        printf("Angka %d tidak ditemukan pada array numbers\n", searchKey);
    }

    return 0;
}
```

E. Tugas

Gunakanlah tugas pertemuan 3 kalian / menggunakan file dari file pendukung sebagai base code tugas pertemuan ini. Tambahkan menu nomor 4 dengan nama "Show student detail". Saat memasuki menu tersebut, aplikasi akan menunjukkan list mahasiswa yang telah diurutkan secara ascending dan aplikasi akan meminta input di bawah tabel mahasiswa. Input yang diminta adalah NIM dari mahasiswa. Jika NIM tidak ditemukan, maka tidak ada detail yang akan ditampilkan. Jika NIM ditemukan, tampilkan data mahasiswa tersebut beserta dengan nilainya. Tampilan aplikasi mengikuti tugas pertemuan 3 atau file pendukung.

REFERENSI