Algoritma Searching

Dosen Pengampu: Qonitatul Hasanah, S.S.T., M.Tr.T.

6.1. Definisi Searching

Searching Algorithm adalah algoritma untuk menemukan item dengan sifat tertentu diantara koleksi item. Item dapat disimpan secara individual sebagai catatan dalam database; atau mungkin elemen tersebut didefinisikan dalam bentuk rumus atau prosedur matematika, seperti akar sebuah persamaan dengan variabel integer. Dalam kehidupan sehari-hari sering kali kita melakukan pencarian, misal:

1. Pencarian nomor telepon di daftar Buku Telepon, mencari buku di perpustakaan dan lain-lain.
2. Komputer dapat mencari berbagai informasi di *World wide web* berdasarkan *keyword* yang kita masukan pada mesin pencarian seperti *yahoo.com*, *ask.co.uk*, atau *google.com.*
3. Di aplikasi *Spreadsheet* yang mengandung daftar (nama, alamat atau apapun), sering diterapkan mekanisme searching untuk menemukan data yang dicari.

6.2. Sequential Seacrhing

Dalam ilmu komputer, teknik pencarian linear juga dikenal sebagai teknik pencarian Sequential. Pencarian linear adalah algoritma yang paling sederhana dan termasuk dalam pencarian *brute-force*. Pencarian ini akan menelusuri satu-persatu setiap item dalam kumpulan/daftar (selanjutnya akan disebut **deret**) data secara berurutan hingga ditemukan atau telah menelusuri semua data dalam deret tersebut. Pencarian dengan teknik ini sebenarnya dapat dimulai dari data indeks [0] hingga akhir, ataupun sebaliknya. Deret data yang dimaksud dapat berupa *Array* ataupun *linked list*. Pencarian dilakukan berdasarkan perbandingan item kunci yang dicari dengan setiap data. Oleh karenanya kecepatan pencarian tergantung pada posisi data yang sedang dicari. Jika target pencarian berada pada elemen pertama dalam deret maka hanya membutuhkan satu perbandingan, jika target berada pada elemen kedua maka membutuhkan dua kali perbandingan. Jadi jika target berada pada elemen ke-*i* dalam deret maka membutuhkan sejumlah *i* perbandingan. Pencarian dengan metode ini akan mencapai kondisi terbaik apabila data yang dicari ada di awal deret dan mencapai kondisi terburuk jika data ada di deretan terakhir. Kelebihan menggunakan sequential searching antara lain :

1. Metode Sequential ini adalah metode yang sederhana dan konvensional dalam pencarian data.
2. Dapat diterapkan pada deret data yang belum terurut.

Namun sudah tentu ada kerugian diantaranya :

1. Metode ini tidak cukup baik ketika elemen dalam deret berjumlah banyak (Misal lebih dari100 elemen).
2. Mengkonsumsi lebih banyak waktu jika dibanding metode yang lain untuk jumlah data yang sama.

Berikut pada Gambar 6.1 disajikan ilustrasi pencarian dengan metode sequential (Gambar 6.1). Dicari data 7 dalam kumpulan data dalam array sebagai berikut : 5, 3, 2, 4, 7, 1, 0 dan 6.

1. Telusuri satu persatu setiap elemen dalam array, ketika sampai di elemen array index [0] yaitu data 5, dilakukan pengecekan apakah data [0] == 7 ?, jika tidak lanjutkan ke elemen berikutnya.
2. Ketika sampai di elemen array index [1] yaitu data 3, dilakukan pengecekan apakah data [1] == 7 ?, jika tidak lanjutkan ke elemen berikutnya.
3. Ketika sampai di elemen array index [2] yaitu data 2, dilakukan pengecekan apakah data [2] == 7 ?, jika tidak lanjutkan ke elemen berikutnya.
4. Ketika sampai di elemen array index [3] yaitu data 4, dilakukan pengecekan apakah data [3] == 7 ?, jika tidak lanjutkan ke elemen berikutnya.
5. Ketika sampai di elemen array index [5] yaitu data 7, dilakukan pengecekan apakah data [5] == 7 ?, karena sama maka pencarian dihentikan dan berhasil menemukan data 7 dalam array.

Gambar 6.2 Ilustrasi Sequential Searching

Berpedoman pada ilustrasi pada Gambar 6.1 maka dapat dituliskan Algoritma untuk pencarian dengan metode Sequential berikut ini:

1. Set-up variable flag untuk menunjukkan "`ditemukan atau tidak ditemukan`"
2. Ambil elemen pertama dalam deret (Array/list)
3. Jika elemen dalam daftar sama dengan elemen yang diinginkan Set flag untuk "`elemen ditemukan`" dan
4. Menampilkan pesan "`elemen yang ditemukan dalam daftar adalah`" Lanjutkan ke langkah 6
5. Jika tidak akhir daftar, mengambil elemen berikutnya (*`next iteration`*). Kembali ke langkah 3
6. Jika flag menandakan elemen yang dicari tidak ditemukan sampai akhir dari deret, maka tampilkan pesan "`elemen tidak ditemukan`"
7. Selesai

Listing program 6.1 memberikan penjelasan mekanisme prosedure sequential searching

```
void sequential_search(int data[], int size, int cari)
{
    int i;
    for (i = 0; i < size; i++)
    {
      if (data[i] == cari)
      {
        printf("Data %d ditemukan di lokasi %d \n", cari, i+1);
        break;
      }
    }
    if (i == size)
        printf(" %d Tidak ada dalam Array\n", cari);
}
```

Listing Program 6.1 Sequential Searching

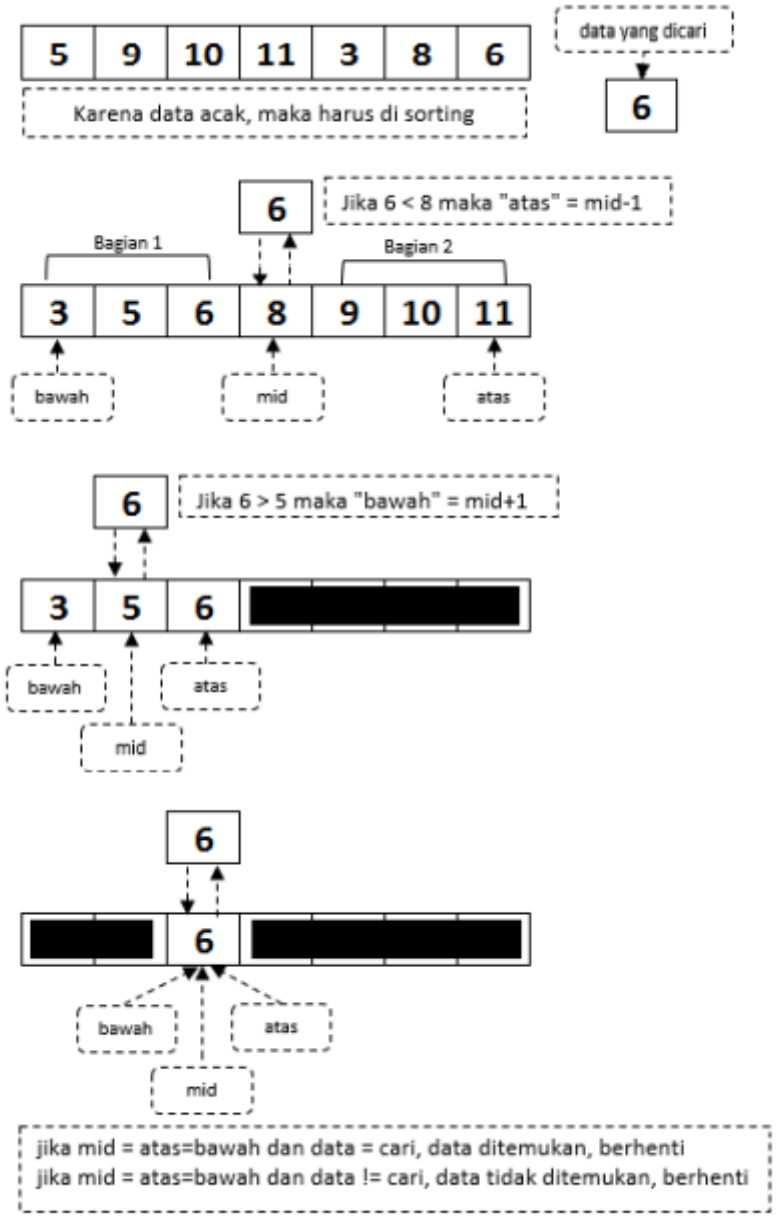
6.3. Binary Searching

Pencarian biner (*binary searching*) didasarkan pada pendekatan *divide - and – conquer* atau membagi-dan-menyelesaikan. Pencarian biner dimulai dengan membagi dua deret yang menjadi ruang permasalahan menjadi bagian satu dan bagian dua. Selanjutnya target (data yang dicari) akan dibandingkan dengan elemen tengahnya, kemudian meninjau berbagai kemungkinan ini:

1. Jika target lebih kecil dari elemen tengah maka program akan mengeliminasi bagian dua dan pencarian dengan cara yang sama dilakukan pada bagian satu sampai target ditemukan.
2. Jika target lebih besar dari elemen tengah maka program akan mengeliminasi bagian satu dan pencarian dengan cara yang sama dilakukan pada bagian dua sampai target ditemukan.
3. Jika target sama dengan elemen tengah maka berhenti, pencarian berhasil.

Dalam metode ini setidaknya kita membutuhkan 3 variabel untuk mengidentifikasi elemen pertama, terakhir dan tengah. Kelebihan menggunakan *binary searching* ialah jika pencarian dilakukan pada sejumlah elemen dalam deret, pencarian biner melakukan perbandingan lebih sedikit dibanding sequential. Sedangkan kekurangan dari pencarian ini adalah lebih cepat daripada pencarian linear. Namun, tidak dapat diterapkan pada struktur data di*sorting*, ilustrasi jalannya metode binary searching dapat dilihat pada Gambar 6.2. Diberikan data dalam array 5, 9, 10, 11, 3, 8 dan 6. Kemudian akan dicari data 6, karena data masih dalam kondisi acak, maka harus diurutkan terlebih dahulu, dapat menggunakan salah satu algoritma sorting yang ada dalam Bab 5 buku ini. Selanjutnya dilakukan perbandingan data yang dicari (data 6) dengan data yang berada di tengah array. Apabila data yang dicari lebih kecil dari data tengah (data 8) maka bagian belakang array diabaikan, lakukan langkah tersebut berulang-ulang hingga data ditemukan.

Binary Searching dapat dilakukan dengan dua cara, yang pertama dengan cara rekursif dan kedua dengan iteratif. Kedua cara tersebut sama-sama melakukan perulangan namun untuk rekursif perulangan yang terdapat dalam fungsi dengan intruksi untuk memanggil fungsi itu sendiri. Sedangkan cara iteratif melakukan perulangan terhadap kelompuk instruksi dalam batasan/syarat tertentu, perulangan akan berhenti jika syarat/batasan tidak terpenuhi. Keunggulan perulangan rekursif yaitu mudah untuk melakukan perulangan dalam skala besar, namun tidak bisa menerapkan *nested loop*. Keuntungan perulangan iteratif yaitu mudah dipahami dan mudah dilakukan debugging, dapat menerapkan *nested loop*, namun jika batasannya luas maka pembuatan instruksi program perulangannya termasuk sulit.



Gambar 6.2 Ilustrasi *Binary Searching*

Algoritma *Binary Searching* dengan perulangan rekursif

1. Mulai
2. binary_search (a , kunci , batasatas , batasbawah)
3. Inisialisasi batasbawah = 0; batasatas = n - 1
4. Jika batasbawah > batasatas, return -1
5. Mencari indeks nilai tengah -- mid = (batasatas + batasbawah)/2
6. Perbandingan nilai kunci
   a. jika (kunci < mid) kemudian batasatas = mid - 1 , batasbawah tetap
   b. jika (kunci > mid) kemudian batasbawah = mid + 1 , batasatas tetap
   c. jika (key == mid) Tulis "data ditemukan" nilainya mid;
      panggil kembali fungsi binary_search (a, kunci, batasatas , batasbawah)
7. Jika data tidak ditemukan hingga akhir, tampilkan pesan "`Data tidak ditemukan`"
8. Selesai

Listing Program 6.2 merupakan contoh penerapan *binary search* menggunakan rekursif.

```
/*Fungsi Rekursif Binary Search*/

int binsearch(int a[], int x, int low, int high)
{
   int mid;

   if (low > high)
      return -1;

   mid = (low + high) / 2;

   if (x == a[mid]) {
      return (mid);
   }

   else if (x < a[mid]) {
      binsearch(a, x, low, mid - 1);
   }

   else {
      binsearch(a, x, mid + 1, high);
   }
}
```

Listing Program 6.2 Binary Search dengan Rekursif

Untuk penerapan *binary search* dengan perulangan iteratif disajikan dalam Listing Program 6.3 berikut ini.

```
int binary_search ( int data[10], int keynum, int num)
{
     int low = 1;
     int high = num;
     int mid;
        do
        {
            mid = (low + high) / 2;
            if (keynum < data[mid])
                high = mid - 1;
            else if (keynum > data[mid])
                low = mid + 1;
        } while (keynum != data[mid] && low <= high);

        if (keynum == data[mid])
              printf("Data Ditemukan \n");
        else
              printf("Data Tidak Ditemukan \n");
}
```

Listing Program 6.3 Binary Searching dengan Iteratif

Tugas

Implementasi Algoritma Searching

- Sequential Searching
- Binary Searching (Baik secara rekursif maupun iteratif)
- Cari mana yang paling efektif? Lakukan Analisa.
- Tugas kelompok.
- Analisa laporan dalam bentuk .pdf.