5_1-SequentialSearch

1 Pencarian/*Searching*

Sequential Searching ***

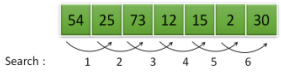
Pencarian data terkadang sangat diperlukan pada proses komputasi, sistem informasi, dan lain-lain. Dengan pencarian data ini maka data yang diperlukan dapat diketahui posisinya terhadap kumpulan data yang lain, sehingga data ini dapat digunakan untuk proses berikutnya. Misalkan pada data mahasiswa, diperlukan data lengkap tentang mahasiswa dengan Nomor induk XXXXXX, maka diperlukan proses pencarian data tersebut dengan *key* berupa nomor induk mahasiswa. Data yang didapat ini dapat digunakan untuk keperluan yang lain, misalkan keperluan administratif.

Pada umumnya, proses pencarian data ini menghasilkan posisi data yang dicari terhadap keseluruhan data, akan tetapi untuk pencarian data yang sederhana, nilai True atau False saja yang dihasilkan, nilai ini menandakan apakah data berada di sekumpulan data ataukah tidak. Proses pencarian sederhana ini sudah disediakan oleh python dengan menggunakan syntax *in*, seperti contoh berikut:

```
In [0]: a=[12,5,9,8,1,10,26]

        11 in a

In [0]: 12 in a

In [0]: 15 in a
```

Algoritma searching yang pertama adalah sequential search.

Sequential Search merupakan teknik pencarian data dengan cara membandingkan data yang dicari dengan seluruh data yang terdapat pada kumpulan data secara satu persatu, mulai dari data pertama sampai dengan data terakhir. Ilustrasi sequential search ini dapat dilihat pada Gambar 1 berikut.



Gambar 1. Sequential Search

Terdapat dua konsep untuk sequential search berdasarkan kumpulan data yang akan dicari, yaitu 1. Section 1.1 2. Section 1.4

1.1 Unordered List Sequential Search

Pada *unordered list sequential search*, data berada pada keadaan acak, tidak terurut, sehingga pencarian harus dilakukan mulai dari indeks awal sampai indeks terakhir dari data, atau pencarian berhenti ketika data sudah ditemukan

1.2 Code

Berikut implementasi algoritma *unordered list sequential search*.

```
In [0]: def seqSearch(listData, data):
            ind = 0
            found = False
            while ind < len(listData) and not found:
                if listData[ind] == data:
                    found = True
                else:
                    ind = ind+1
            return found
```

```
In [0]: a=[12,5,9,8,1,10,26]
```

```
In [0]: seqSearch(a,1)
```

1.3 Latihan - 1:

1. Modifikasi program sequential tersebut agar mengembalikan 'Data tidak ada' jika data tidak ditemukan dan mengembalikan index data jika data ditemukan
2. Modifikasi program sequential tersebut agar tetap meneruskan pencarian sampai data terakhir, dan kembalikan index data semua data yang ditemukan (asumsi, ada data yang sama)

Section 1

1.4 Ordered List Sequential Search

Pada ordered list sequential search, data sudah dalam keadaan terurut, hal ini tentunya dapat mengurangi waktu komputasi pencarian. Ilustrasi algoritma pencarian ini, dapat dilihat pada Gambar 2 berikut:

Gambar 2. Ordered List Sequential Search

Misalkan data yang dicari adalah 10, pada index kedua dari data tersebut adalah 15, oleh karena itu dapat disimpulkan bahwa data 10 tidak terdapat pada list. karena indeks kedua sampai indeks terakhir memiliki nilai lebih besar dari 15. Sehingga data yang sudah dalam keadaan terurut dapat mempercepat waktu komputasi.

Berikut implementasi algoritma sequential search pada ordered list

```
In [0]: def orderedSeqSearch(listData, data):
            ind = 0
            found = False
            stop = False
            position=[]
            while ind < len(listData) and not found and not stop:
                if listData[ind] == data:
                    found = True
```

```
                    position.append(ind)
                else:
                    if listData[ind] > data:
                        stop = True
                    else:
                        ind = ind+1

            if found:
                return ind
            else:
                return ('Data tidak ada')
```

```
In [0]: a=[1,1,5,5,5,8,9,10,12,26]
```

```
In [0]: ind=orderedSeqSearch(a,5)
        print(ind)
```

Section 1