

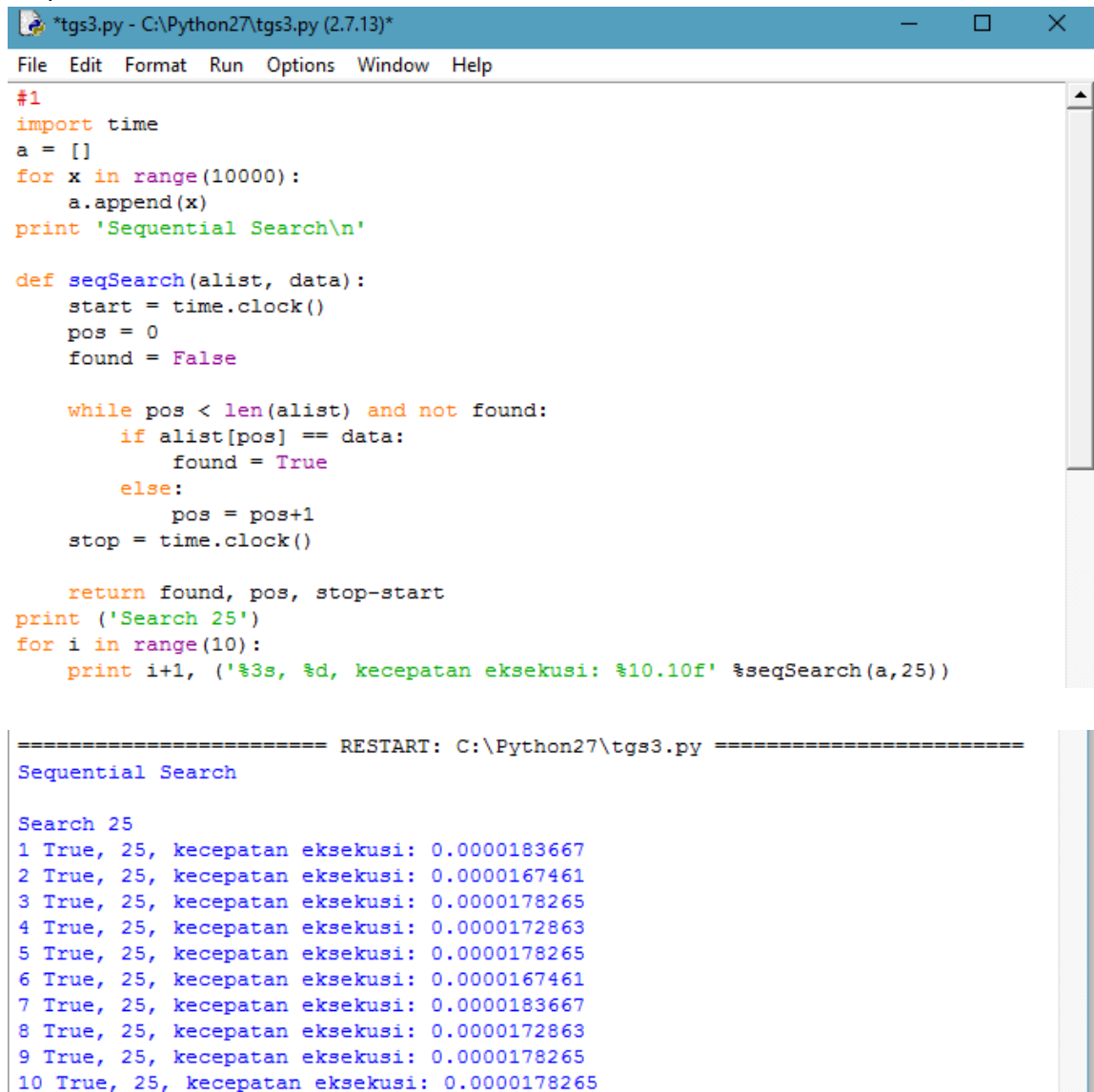
Nama : Anindya Nafsitasari

NIM : L200150081

Kelas : B

TIMER untuk mengukur kecepatan proses eksekusi algoritma:

a. Sequential Search



```
*tgs3.py - C:\Python27\tgs3.py (2.7.13)*
File Edit Format Run Options Window Help

#1
import time
a = []
for x in range(10000):
    a.append(x)
print 'Sequential Search\n'

def seqSearch(alist, data):
    start = time.clock()
    pos = 0
    found = False

    while pos < len(alist) and not found:
        if alist[pos] == data:
            found = True
        else:
            pos = pos+1
    stop = time.clock()

    return found, pos, stop-start
print ('Search 25')
for i in range(10):
    print i+1, ('%3s, %d, kecepatan eksekusi: %10.10f' %seqSearch(a,25))

===== RESTART: C:\Python27\tgs3.py =====
Sequential Search

Search 25
1 True, 25, kecepatan eksekusi: 0.0000183667
2 True, 25, kecepatan eksekusi: 0.0000167461
3 True, 25, kecepatan eksekusi: 0.0000178265
4 True, 25, kecepatan eksekusi: 0.0000172863
5 True, 25, kecepatan eksekusi: 0.0000178265
6 True, 25, kecepatan eksekusi: 0.0000167461
7 True, 25, kecepatan eksekusi: 0.0000183667
8 True, 25, kecepatan eksekusi: 0.0000172863
9 True, 25, kecepatan eksekusi: 0.0000178265
10 True, 25, kecepatan eksekusi: 0.0000178265
```

b. Sequential Search (Sorting)

```
#2
import time
a = []
for x in range(10000):
    a.append(x)
print 'Sequential Search (Sorting)\n'

def sortSeqSearch(alist, data):
    start = time.clock()
    pos = 0
    found = False
    stop = False
    while pos < len(alist) and not found and not stop:
        if alist[pos] == data:
            found = True
        else:
            if alist[pos] > data:
                stop = True
            else:
                pos = pos+1
    end = time.clock()

    return found, pos, end-start

for i in range(10):
    print i+1, ('%3s, %d, kecepatan eksekusi = %10.10f' %sortSeqSearch(a,255))
```

```
Sequential Search (Sorting)

1 True, 255, kecepatan eksekusi = 0.0001593577
2 True, 255, kecepatan eksekusi = 0.0001588175
3 True, 255, kecepatan eksekusi = 0.0001577371
4 True, 255, kecepatan eksekusi = 0.0001588175
5 True, 255, kecepatan eksekusi = 0.0001571969
6 True, 255, kecepatan eksekusi = 0.0001571969
7 True, 255, kecepatan eksekusi = 0.0001588175
8 True, 255, kecepatan eksekusi = 0.0001577371
9 True, 255, kecepatan eksekusi = 0.0001582773
10 True, 255, kecepatan eksekusi = 0.0001593577
>>>
```

c. Binary Search

```
#3
import time
a = []
for x in range(10000):
    a.append(x)
print 'Binary Search\n'

def binSearch(alist, data):
    start = time.clock()
    first = 0
    last = len(alist)-1
    found = False
    while first < last and not found:
        midpoint = (first + last)//2
        if alist[midpoint] == data:
            found = True
        else:
            if data < alist[midpoint]:
                last = midpoint-1
            else:
                first = midpoint+1
    end = time.clock()

    return found, end-start

for i in range(10):
    print i+1, ('%2s, kecepatan eksekusi = %10.10f' %binSearch(a,9995))
```

```
...
===== RESTART: C:\Python27\tgs3.py =====
Binary Search

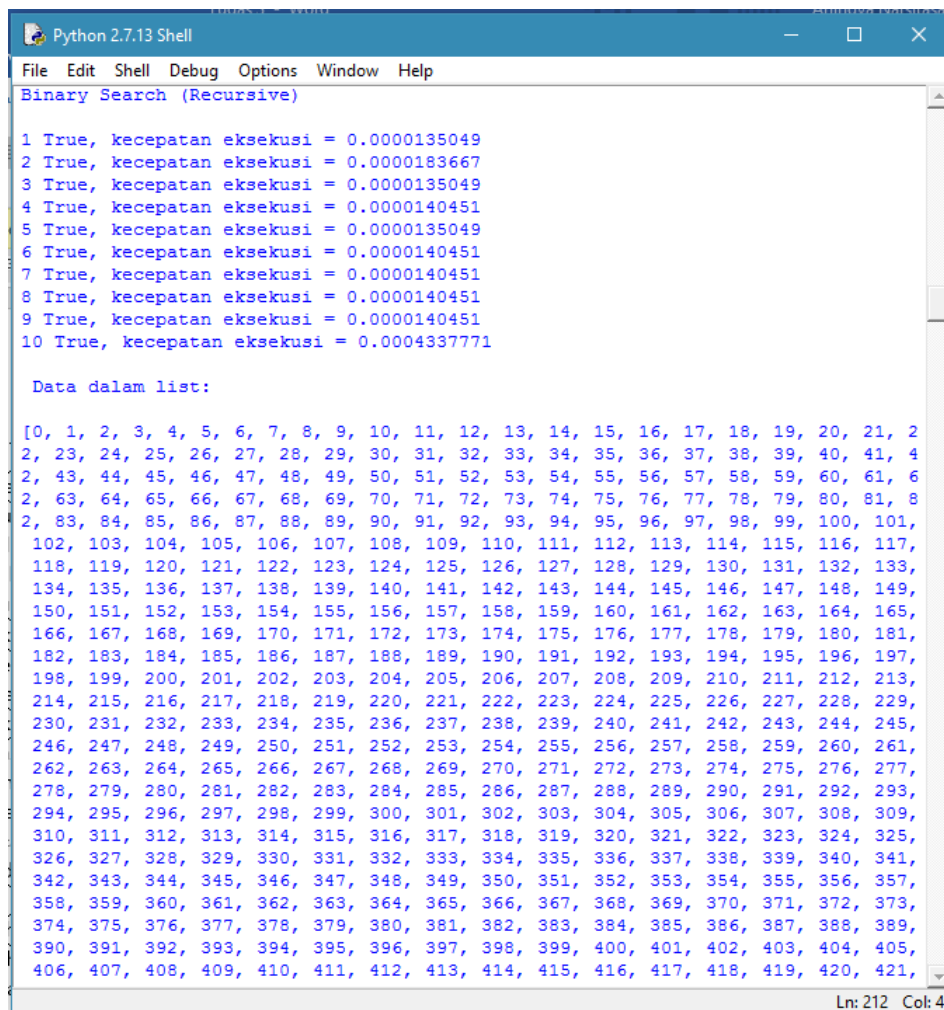
1 True, kecepatan eksekusi = 0.0000729264
2 True, kecepatan eksekusi = 0.0000675245
3 True, kecepatan eksekusi = 0.0000162059
4 True, kecepatan eksekusi = 0.0000172863
5 True, kecepatan eksekusi = 0.0000167461
6 True, kecepatan eksekusi = 0.0000194470
7 True, kecepatan eksekusi = 0.0000162059
8 True, kecepatan eksekusi = 0.0000162059
9 True, kecepatan eksekusi = 0.0000216078
10 True, kecepatan eksekusi = 0.0000167461
```

d. Binary Search (Recursive)

```
#4
import time
a = []
for x in range(10000):
    a.append(x)
print 'Binary Search (Recursive)\n'

def binSearchRec(alist, data):
    start = time.clock()
    if len(alist) == 0:
        return False
    else:
        midpoint = len(alist)//2
        if alist[midpoint] == data:
            return True
        else:
            if data < alist[midpoint]:
                return binSearchRec(alist[:midpoint], data)
            else:
                return binSearchRec(alist[midpoint+1:], data)
    stop = time.clock()
    return stop-start

for i in range(10):
    print i+1, ('%2s, kecepatan eksekusi = %10.10f' %binSearchRec(a,55))
print '\n Data dalam list:\n'
print a
```



Python 2.7.13 Shell

File Edit Shell Debug Options Window Help

Binary Search (Recursive)

```
1 True, kecepatan eksekusi = 0.0000135049
2 True, kecepatan eksekusi = 0.0000183667
3 True, kecepatan eksekusi = 0.0000135049
4 True, kecepatan eksekusi = 0.0000140451
5 True, kecepatan eksekusi = 0.0000135049
6 True, kecepatan eksekusi = 0.0000140451
7 True, kecepatan eksekusi = 0.0000140451
8 True, kecepatan eksekusi = 0.0000140451
9 True, kecepatan eksekusi = 0.0000140451
10 True, kecepatan eksekusi = 0.0004337771

Data dalam list:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 4
2, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 6
2, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 8
2, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101,
102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,
134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149,
150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165,
166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197,
198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213,
214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229,
230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245,
246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261,
262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277,
278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293,
294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309,
310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325,
326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341,
342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357,
358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373,
374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389,
390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405,
406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421,
```

Ln: 212 Col: 4

ANALISIS

A. Sequential search

Sequential search adalah cara untuk pencarian data dalam array 1 dimensi. Data yang akan dicari nanti akan ditelusuri dalam semua elemen-elemen array dari awal sampai akhir, dan data yang dicari tersebut tidak perlu diurutkan terlebih dahulu. Terdapat 2 kemungkinan yang akan terjadi dalam waktu pencarian data Sequential Search, diantaranya yaitu :

1. Kemungkinan Terbaik (Best Case)

Best case akan terjadi apabila data yang dicari terletak pada index array yang paling depan, sehingga waktu yang dibutuhkan untuk mencari data sedikit.

2. Kemungkinan Terburuk (Worse Case)

Worse case akan terjadi apabila data yang dicari terletak pada index array yang paling akhir, sehingga waktu yang dibutuhkan untuk mencari data akan sangat lama. Untuk meningkatkan efisiensi pencarian data pada Sequential Search dapat dilakukan dengan cara menghentikan looping dengan menggunakan BREAK apabila data yang dicari sudah ketemu.

B. Binary Search

Binary Search adalah cara untuk pencarian data pada array yang sudah terurut. Karena salah satu syarat dalam binary search adalah data sudah dalam keadaan terurut. Dengan kata lain apabila data belum dalam keadaan terurut, pencarian binary tidak dapat dilakukan. Binary Search ini dilakukan untuk :

1. Memperkecil jumlah operasi perbandingan yang harus dilakukan antara data yang dicari dengan data yang ada di dalam tabel, khususnya untuk jumlah data yang sangat besar ukurannya.
2. Beban komputasi lebih kecil karena pencarian dilakukan dari depan, belakang dan tengah.
3. Prinsip dasarnya adalah melakukan proses pembagian ruang pencarian secara berulang-ulang sampai data ditemukan atau sampai pencarian tidak dapat dibagi lagi (berarti ada kemungkinan data tidak ditemukan).