Praktikum Struktur Data

Modul 8 - Searching

Senin, 30 Mei 2021

Tujuan dari modul ini agar mahasiswa memahami konsep pencarian, serta implementasinya. Kerjakan tugas-tugas yang terdapat dalam modul ini, dengan ketentuan sebagai berikut :

1. Semua jawaban modul dikerjakan dalam format *.ipynb (baik konsep maupun implementasi), jangan lupa diberikan **heading** di setiap cell jawaban, misalkan untuk jawaban konsep Nomor 1, diberikan heading **Konsep_1**. Silahkan dilihat contoh *lecture notes* yang saya telah bagikan melalui website.

2. Penamaan file ipynb adalah : NPM_ModulX_TopikModul.ipynb, misalkan, 200411100077_Modul8_Searching.ipynb

3. Print menjadi file pdf, dokumen ipynb tersebut dengan nama yang sama, hanya saja berekstensi pdf, misalkan, 200411100077_Modul8_Searching.pdf

4. Submit **link collaboratory** yang berisi file ipynb tersebut, dan submit **file pdf**

5. Pilih salah satu nomor, buat juga video pembelajaran dan upload ke channel youtube masing-masing, dengan hashtag yang ditentukan. Format video bebas, wajib diberikan voice over. Akan lebih baik jika mode camera adalah ON.

6. Kejujuran selalu jadi yang utama, kerjakan sendiri, tidak diperkenankan plagiarism

1 Konsep

1.1 searching

Jelaskan perbedaan algoritma sequential search (unordered maupun ordered), binary search, dan hashing. Jika diperlukan berikan contoh masing-masing algoritma

1.2 Hashing

Jelaskan pencarian dengan menggunakan konsep Hashing, antara lain :

- Table Hash, Slot
- Fungsi Hash - Nilai Hash
- Pencarian dengan konsep Hashing
- Collusion

2 Implementasi

2.1 Searching

Buatlah code untuk mengimplementasikan masing-masing algoritma searching (Sequential Search dan Binary Search), dengan ketentuan sebagai berikut :

- data di*generate* secara random sebanyak 500 data (Hint : gunakan modul random yang sudah disediakan python)

- khusus data untuk algoritma binary search, generate data sedemikian hingga, kemungkinan data yang sama adalah kecil

- contoh output dapat dilihat pada Gambar 1

2.2 Hashing

Buatlah code untuk mengimplementasikan algoritma pencarian dengan konsep Hashing, dengan ketentuan sebagai berikut :

- data di*generate* secara random sebanyak 500 data (Hint : gunakan modul random yang sudah disediakan python)

- generate data sedemikian hingga, kemungkinan data yang sama adalah kecil

- [**untuk Kelas E saja**] Gunakan penanganan collusion dengan menggunakan chaining. Contoh output yang dihasilkan dapat dilihat pada Gambar 2

Selamat Mengerjakan, Selalu Latihan, Jujur harus dimulai kapanpun, Bertanya jika kurang mengerti, #StayAtHome, #LearningFromHome

Struktur Data - 2022
Indah Agustien Siradjuddin

```
print('Data = ', data,'\n')
print(seqSearch(data, 100))
print(seqSearch(data, 20))
print(seqSearch(data, 500))
```

```
Data = [51, 8, 18, 25, 20, 7, 57, 22, 20, 52, 26, 99, 29, 27, 1, 87, 78, 59, 29, 8, 71, 59, 11, 84, 58, 27, 52, 94, 47, 19,
65, 97, 13, 80, 3, 8, 75, 74, 86, 19, 3, 10, 21, 4, 4, 34, 28, 19, 53, 77, 88, 75, 53, 7, 67, 39, 39, 19, 94, 70, 47, 22, 4
9, 20, 94, 63, 95, 58, 74, 4, 17, 40, 21, 95, 16, 85, 34, 13, 14, 31, 11, 20, 6, 38, 86, 31, 46, 64, 1, 60, 40, 22, 28, 41,
7, 14, 32, 3, 89, 12, 53, 99, 73, 78, 72, 98, 2, 26, 50, 13, 20, 27, 53, 24, 7, 33, 66, 87, 32, 21, 97, 80, 41, 4, 98, 18, 3
2, 98, 4, 99, 10, 100, 68, 7, 41, 99, 85, 90, 47, 33, 74, 73, 36, 23, 8, 1, 41, 34, 24, 55, 42, 96, 21, 72, 87, 88, 17, 79,
24, 46, 12, 59, 26, 75, 63, 72, 82, 38, 22, 73, 45, 20, 75, 71, 98, 45, 57, 6, 90, 68, 68, 87, 75, 79, 70, 38, 16, 23, 37,
4, 55, 5, 96, 22, 34, 37, 71, 89, 30, 25, 91, 24, 65, 73, 24, 57, 87, 1, 89, 36, 59, 85, 94, 27, 9, 22, 13, 17, 25, 13, 81,
94, 43, 17, 34, 28, 22, 60, 48, 2, 76, 34, 77, 75, 43, 40, 52, 10, 11, 57, 91, 67, 14, 18, 93, 87, 95, 74, 5, 37, 2, 65, 97,
98, 29, 88, 77, 28, 61, 25, 39, 7, 79, 72, 69, 40, 62, 100, 70, 95, 11, 93, 44, 31, 30, 9, 10, 94, 14, 67, 69, 94, 62, 44, 4
7, 48, 96, 8, 76, 94, 1, 25, 67, 100, 70, 51, 81, 78, 41, 89, 72, 72, 3, 15, 30, 82, 37, 56, 91, 85, 93, 54, 95, 46, 37, 19,
5, 96, 47, 90, 30, 85, 62, 58, 62, 17, 35, 99, 71, 9, 86, 95, 29, 48, 32, 63, 55, 15, 97, 44, 39, 52, 11, 13, 90, 86, 55, 3
1, 58, 48, 38, 48, 26, 9, 9, 37, 86, 60, 47, 6, 60, 95, 71, 4, 15, 91, 19, 28, 44, 67, 39, 66, 69, 68, 3, 18, 60, 19, 53, 3
8, 7, 56, 80, 63, 95, 21, 72, 30, 15, 28, 53, 6, 10, 100, 4, 22, 65, 67, 59, 22, 84, 48, 35, 40, 77, 32, 63, 46, 35, 57, 18,
22, 50, 4, 32, 50, 62, 6, 68, 39, 85, 56, 19, 35, 72, 99, 80, 26, 32, 38, 52, 12, 90, 97, 34, 96, 72, 55, 41, 1, 28, 35, 17,
11, 21, 72, 32, 86, 56, 18, 1, 47, 5, 2, 96, 48, 33, 7, 94, 5, 99, 11, 93, 19, 94, 99, 87, 60, 90, 49, 80, 49, 98, 49, 81,
1, 21, 94, 88, 39, 7, 46, 82, 23, 29, 34, 18, 62, 93, 88, 53, 55, 61, 39, 98, 100, 38, 16, 39, 91]

100 is in : [131, 267, 293, 393, 495]
20 is in : [4, 8, 63, 81, 110, 171]
500 is not found
```

(a) Unordered Sequential Search

```
insertionSort(data)
print('Data = ',data,'\n')
print(orderedSeqSearch(data, 1))
print(orderedSeqSearch(data, 25))
print(orderedSeqSearch(data, 100))
print(orderedSeqSearch(data, 20))
print(orderedSeqSearch(data, 500))
print(orderedSeqSearch(data, 1000))
```

```
Data = [1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7,
7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 13, 13, 13, 1
3, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19, 19, 1
9, 19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 2
4, 24, 24, 24, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 27, 27, 27, 27, 28, 28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 29, 3
0, 30, 30, 30, 30, 31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 32, 33, 33, 33, 34, 34, 34, 34, 34, 34, 34, 35, 35, 35, 3
5, 35, 36, 36, 37, 37, 37, 37, 37, 37, 38, 38, 38, 38, 38, 38, 38, 39, 39, 39, 39, 39, 39, 39, 39, 39, 40, 40, 40, 40, 40, 4
1, 41, 41, 41, 41, 42, 43, 43, 44, 44, 44, 44, 45, 45, 46, 46, 46, 46, 46, 47, 47, 47, 47, 47, 47, 47, 48, 48, 48, 48, 4
8, 48, 48, 49, 49, 49, 49, 50, 50, 50, 51, 51, 52, 52, 52, 52, 52, 53, 53, 53, 53, 53, 53, 54, 55, 55, 55, 55, 55, 55, 5
6, 56, 56, 56, 57, 57, 57, 57, 57, 58, 58, 58, 58, 59, 59, 59, 59, 59, 60, 60, 60, 60, 60, 60, 61, 61, 62, 62, 62, 62, 62, 6
2, 63, 63, 63, 63, 63, 64, 65, 65, 65, 65, 66, 66, 67, 67, 67, 67, 67, 68, 68, 68, 68, 68, 68, 69, 69, 69, 70, 70, 70, 70, 7
1, 71, 71, 71, 71, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 73, 73, 73, 73, 74, 74, 74, 74, 75, 75, 75, 75, 75, 75, 76, 76, 7
7, 77, 77, 77, 78, 78, 78, 79, 79, 79, 80, 80, 80, 80, 80, 81, 81, 81, 82, 82, 82, 84, 84, 85, 85, 85, 85, 85, 85, 86, 86, 8
6, 86, 86, 86, 87, 87, 87, 87, 87, 88, 88, 88, 88, 88, 89, 89, 89, 90, 90, 90, 90, 90, 90, 91, 91, 91, 91, 91, 9
3, 93, 93, 93, 93, 94, 94, 94, 94, 94, 94, 94, 94, 94, 94, 95, 95, 95, 95, 95, 95, 95, 95, 96, 96, 96, 96, 96, 97, 9
7, 97, 97, 97, 98, 98, 98, 98, 98, 98, 98, 99, 99, 99, 99, 99, 99, 99, 99, 100, 100, 100, 100, 100]

1 is in : [0, 1, 2, 3, 4, 5, 6, 7] , numberOfIteration: 9
25 is in : [140, 141, 142, 143, 144] , numberOfIteration: 146
100 is in : [495, 496, 497, 498, 499] , numberOfIteration: 500
20 is in : [109, 110, 111, 112, 113, 114] , numberOfIteration: 116
500 is not found , numberOfIteration: 500
1000 is not found , numberOfIteration: 500
```

(b) Ordered Sequential Search

```
insertionSort(data)
print('Data = ',data,'\n')
print(binarySearch(data, 1))
print(binarySearch(data, 31))
print(binarySearch(data, 254))
print(binarySearch(data, 1320))
```

```
Data = [4, 5, 16, 35, 50, 53, 65, 67, 75, 87, 91, 103, 111, 118, 137, 144, 151, 182, 198, 200, 201, 203, 204, 210, 214, 22
4, 227, 233, 234, 236, 237, 255, 258, 268, 270, 272, 297, 306, 307, 309, 310, 326, 334, 335, 337, 337, 345, 351, 353, 354, 3
56, 357, 361, 370, 376, 379, 386, 390, 396, 398, 399, 408, 414, 425, 425, 428, 454, 457, 458, 459, 468, 476, 477, 484,
495, 503, 519, 520, 521, 531, 539, 551, 553, 569, 569, 573, 577, 589, 590, 611, 633, 637, 638, 640, 644, 653, 658, 661, 661,
667, 678, 680, 685, 690, 696, 698, 700, 703, 704, 725, 730, 740, 747, 752, 752, 761, 784, 790, 795, 803, 808, 811, 815, 817,
824, 824, 857, 862, 874, 874, 878, 880, 889, 895, 900, 905, 911, 911, 932, 968, 971, 973, 978, 980, 997, 1006, 1007, 1012, 1
027, 1047, 1050, 1054, 1055, 1055, 1058, 1070, 1087, 1092, 1096, 1097, 1105, 1106, 1107, 1110, 1111, 1112, 1116, 1125, 1133,
1137, 1140, 1159, 1160, 1174, 1184, 1187, 1188, 1192, 1194, 1203, 1213, 1214, 1219, 1224, 1237, 1249, 1252, 1252, 125
8, 1258, 1258, 1278, 1288, 1298, 1310, 1314, 1319, 1320, 1327, 1328, 1335, 1336, 1337, 1349, 1351, 1354, 1361, 1365, 1368, 1
371, 1372, 1373, 1375, 1378, 1380, 1380, 1387, 1398, 1405, 1405, 1410, 1418, 1420, 1424, 1425, 1429, 1435, 1437, 1453, 1457,
1459, 1465, 1472, 1474, 1476, 1480, 1491, 1494, 1516, 1516, 1521, 1526, 1536, 1536, 1537, 1537, 1548, 1553, 1568, 1580, 158
6, 1587, 1589, 1600, 1601, 1605, 1622, 1626, 1627, 1629, 1640, 1641, 1643, 1646, 1661, 1668, 1670, 1678, 1681, 1683, 1691, 1
709, 1711, 1715, 1724, 1724, 1732, 1754, 1754, 1764, 1765, 1782, 1783, 1787, 1790, 1801, 1823, 1830, 1834, 1835, 1846, 1865,
1867, 1874, 1876, 1876, 1879, 1888, 1891, 1906, 1913, 1916, 1919, 1925, 1926, 1932, 1932, 1941, 1965, 1969, 1976, 1989, 199
4, 1996, 1998, 1999, 2011, 2011, 2012, 2017, 2019, 2024, 2026, 2026, 2043, 2047, 2056, 2066, 2075, 2085, 2093, 2106, 2112, 2
115, 2119, 2122, 2124, 2126, 2132, 2137, 2140, 2144, 2144, 2148, 2150, 2154, 2162, 2163, 2166, 2173, 2177, 2192, 2193, 2211,
2214, 2220, 2237, 2239, 2240, 2243, 2245, 2271, 2288, 2289, 2291, 2299, 2307, 2307, 2317, 2323, 2325, 2330, 2351, 2355, 236
1, 2367, 2373, 2374, 2378, 2380, 2388, 2393, 2396, 2401, 2417, 2420, 2428, 2437, 2440, 2447, 2448, 2456, 2464, 2464, 2469, 2
472, 2482, 2484, 2489, 2490, 2509, 2518, 2518, 2525, 2527, 2535, 2536, 2546, 2548, 2556, 2558, 2561, 2568, 2576, 2582, 2596,
2602, 2611, 2612, 2620, 2621, 2630, 2638, 2639, 2641, 2645, 2646, 2652, 2655, 2658, 2661, 2667, 2674, 2688, 2690, 2691, 269
5, 2699, 2701, 2714, 2718, 2724, 2729, 2731, 2734, 2739, 2746, 2754, 2763, 2765, 2767, 2771, 2777, 2784, 2796, 2796, 2
808, 2809, 2827, 2829, 2838, 2842, 2854, 2854, 2855, 2864, 2865, 2869, 2872, 2874, 2879, 2880, 2886, 2893, 2894, 2895, 2900,
2903, 2904, 2908, 2911, 2917, 2930, 2936, 2946, 2949, 2954, 2956, 2964, 2966, 2969, 2972, 2978, 2988, 2990, 2995, 2997]

1 is not found , numberOfIteration: 8
31 is not found , numberOfIteration: 9
254 is not found , numberOfIteration: 8
1320 is in : 199 , numberOfIteration: 8
```

(c) Binary Search

3

Gambar 1: Searching

```
numOfSlot=100
hashTable=createHashTable(numOfSlot)
hashTable=putData(data,hashTable)
```

```
findData=6532
found=searchHash(findData,hashTable)
if not found:
    print('%d is not in the hash table'%(findData))
else:
    print("%d is in slot %d indeks:%d"%(findData,found[0],found[1]))
    print("Data in slot[%d]:%s"%(found[0],hashTable[found[0]]))
```

```
6532 is in slot 32 indeks:2
Data in slot[32]:[9732, 26732, 6532, 40932, 5032, 34532]
```

(a)

```
numOfSlot=300
hashTable=createHashTable(numOfSlot)
hashTable=putData(data,hashTable)
```

```
findData=6532
found=searchHash(findData,hashTable)
if not found:
    print('%d is not in the hash table'%(findData))
else:
    print("%d is in slot %d indeks:%d"%(findData,found[0],found[1]))
    print("Data in slot[%d]:%s"%(found[0],hashTable[found[0]]))
```

```
6532 is in slot 232 indeks:0
Data in slot[232]:[6532, 5032]
```

(b)

```
numOfSlot=500
hashTable=createHashTable(numOfSlot)
hashTable=putData(data,hashTable)
```

```
findData=6532
found=searchHash(findData,hashTable)
if not found:
    print('%d is not in the hash table'%(findData))
else:
    print("%d is in slot %d indeks:%d"%(findData,found[0],found[1]))
    print("Data in slot[%d]:%s"%(found[0],hashTable[found[0]]))
```

```
6532 is in slot 32 indeks:0
Data in slot[32]:[6532, 5032, 34532]
```

(c)

Gambar 2: Searching - Hashing