210411100085_Modul8_Searching

May 31, 2022

1 Konsep

1.1 Searching

Sequential Search

Algoritma searching sequential search merupakan algoritma yang mencari data dengan cara mengecek setiap elemen yang berada pada list. Proses pengecekan data tersebut akan terus berjalan hingga elemen terakhir pada list. Contoh :

```
"""
[13,32,15,53]
    /\
    ||
data yang di cari adalah 32, dan ditemukan di index 1
maka proses pencarian akan terus berjalan walaupun sudah didapat data yang dicari

"""
```

Sequential Search (Ordered)

Algoritma searching sequential search (ordered) merupakan algoritma yang mencari data dengan cara mengecek setiap elemen yang berada pada list dengan syarat data harus diurutkan terlebih dahulu. Proses pengecekannya sama seperti unordered sequential search tetapi pada ordered sequential search setelah data ditemukan maka proses pencariannya akan berhenti. Contoh :

```
"""
[13,32,15,53]

|Data diurutkan |
|terlebih dahulu|
     \/

[13,15,32,53]

    /\
    ||
data yang di cari adalah 32, dan ditemukan di index 2
maka proses pencarian akan berhenti.
"""
```

Binary Search

Algoritma Binary search merupakan algoritma search yang pencariannya membagi data menjadi 2 dan dimulai dari bagian tengah data dengan syarat data harus terurut.

Hasing

Algoritma hashing merupakan algoritma yang menempatkan data pada tempat yang bernama hash table sehingga data yang akan dicari dapat menggunakan hash table. Pencariannya hanya melakukan 1 kali perbandingan saja apakah data ada di tempat hash table tersebut atau tidak.

1.2 Hashing

Hash Table & Slot

Hash table merupakan tempat yang digunakan untuk menempatkan data yang akan dicari nantinya, sedangkan slot itu merupakan posisi/index dari data tersebut.

Fungsi Hash

Fungsi hash merupakan bagaimana suatu data itu akan ditempatkan pada hash table. Cara menempatkan data tersebut dalam hash tabel dapat menggunakan rumus data%ukuranHashTable. Maka data akan ditempatkan pada posisi slot dari hasil tersebut.

Pencarian dengan konsep Hasing

- Hal pertama yang dilakukan adalah buat hash table yang berukuran lebih dari ukuran list data yang dicari.
- Kemudian tempatkan data tersebut ke dalam hash table dengan menggunakan fungsi hash.
- Setelah selesai menepatkan semua data pada slot di hash table langkah selanjutnya adalah mencari data dengan cara hitung nilai hash dari data yang akan dicari kemudian dari nilai hash tersebut digunakan untuk mendapatkan data dari hash table menggunakan slot dengan nilai hash yang sydah didapat tadi.

Collusion

Collusion pada hashing merupakan kejadian dimana ada data dengan nilai yang sama yang akan ditempatkan pada hash table sehingga terjadi replacing data dengan data baru. Penangannya dapat menggunakan metode chaining yaitu dengan cara menempatkan data dengan nilai hash yang sama pada slot yang sama tetapi pada slot tersebut terdapat ruangan tersendiri untuk setiap data.

2 Implementasi

2.1 Searching

[57]:
```python
import random as rand

def insertionSortAscending(nums) :
    maxIdx = len(nums)-1

    for i in range(maxIdx,0,-1):
        key = nums[i-1]
```

```
        iAfter = i
        while iAfter <= maxIdx and key>=nums[iAfter]  :
            nums[iAfter-1] = nums[iAfter]
            iAfter+=1
        nums[iAfter-1] = key

    return nums


def generateData(totalData,rangeData) :
    return [rand.randint(1,rangeData) for i in range(totalData)]
```

[50]:
```
def seqSearch(data,keyword) :
    dataFound = []
    for i in range(len(data)) :
        if keyword == data[i] :
            dataFound.append(i)

    if len(dataFound) > 0 :
        return "%d is in = %s" % (keyword,dataFound)
    else :
        return "%d is not Found" % (keyword)
```

[53]:
```
data = generateData(500,500)
print("Data = ",data,end="\n\n")
print(seqSearch(data,100))
print(seqSearch(data,20))
print(seqSearch(data,500))
```

```
Data =  [140, 292, 455, 106, 350, 474, 107, 157, 102, 399, 280, 118, 188, 385,
385, 76, 231, 212, 185, 174, 224, 265, 33, 137, 66, 442, 392, 59, 499, 442, 377,
197, 314, 482, 472, 449, 260, 456, 473, 229, 19, 334, 152, 463, 320, 236, 385,
358, 279, 213, 58, 188, 39, 53, 14, 243, 263, 430, 278, 425, 445, 152, 131, 116,
388, 451, 396, 313, 130, 128, 356, 313, 454, 167, 442, 375, 95, 164, 315, 486,
260, 98, 420, 221, 340, 82, 474, 179, 309, 440, 108, 198, 147, 160, 453, 438,
495, 247, 500, 370, 361, 18, 205, 66, 115, 492, 31, 265, 219, 34, 199, 406, 386,
451, 380, 426, 150, 420, 409, 352, 447, 478, 81, 30, 497, 373, 181, 202, 129,
444, 309, 454, 159, 285, 438, 158, 322, 209, 336, 144, 44, 54, 375, 319, 495,
472, 103, 473, 292, 473, 11, 466, 457, 345, 83, 112, 342, 4, 95, 242, 205, 498,
21, 465, 201, 173, 155, 469, 123, 370, 185, 212, 393, 158, 378, 405, 102, 89,
437, 476, 221, 136, 193, 231, 85, 64, 63, 150, 357, 399, 139, 197, 108, 340, 79,
156, 155, 274, 446, 347, 81, 287, 449, 166, 150, 61, 316, 30, 343, 498, 274,
447, 77, 472, 478, 209, 30, 131, 445, 286, 136, 116, 483, 497, 49, 437, 12, 241,
4, 77, 347, 307, 249, 363, 461, 39, 253, 70, 80, 401, 243, 23, 442, 53, 1, 291,
126, 8, 398, 479, 33, 391, 165, 320, 456, 464, 355, 430, 394, 259, 253, 88, 450,
174, 131, 140, 102, 213, 363, 427, 414, 406, 30, 453, 231, 465, 130, 394, 65,
```

```
293, 19, 361, 18, 292, 133, 290, 189, 261, 373, 349, 274, 55, 137, 496, 277, 19,
209, 59, 64, 126, 435, 142, 63, 399, 438, 240, 488, 396, 308, 284, 286, 20, 487,
139, 362, 83, 422, 59, 1, 280, 93, 468, 279, 260, 440, 122, 119, 51, 291, 409,
452, 191, 444, 44, 431, 52, 270, 29, 347, 137, 319, 324, 242, 125, 411, 462,
455, 17, 98, 88, 452, 47, 84, 468, 344, 241, 328, 10, 386, 71, 249, 171, 121,
379, 91, 355, 334, 178, 6, 435, 396, 319, 28, 209, 257, 372, 22, 7, 352, 30,
259, 477, 123, 340, 463, 452, 135, 217, 293, 266, 285, 385, 447, 164, 205, 382,
375, 64, 263, 169, 7, 174, 240, 202, 246, 200, 490, 23, 5, 271, 220, 319, 148,
72, 50, 388, 464, 72, 223, 105, 434, 441, 30, 5, 472, 397, 432, 391, 106, 281,
56, 395, 344, 103, 162, 491, 6, 8, 316, 413, 374, 370, 189, 142, 310, 349, 208,
33, 165, 206, 348, 473, 352, 498, 216, 387, 216, 221, 378, 29, 140, 436, 356,
480, 24, 411, 90, 449, 227, 352, 221, 224, 238, 227, 91, 153, 404, 369, 39, 108,
400, 186, 128, 82, 380, 424, 387, 25, 143, 414, 95, 421, 499, 379, 18, 498, 222,
472, 157, 239]

100 is not Found
20 is in = [311]
500 is in = [98]
```

[62]:
```
def orderedSeqSearch(data,keyword):
    dataFound = False
    idxDataFounded = 0

    for i in range(len(data)) :
        if data[i] == keyword :
            dataFound = True
            idxDataFounded = i
            break

    if dataFound :
        return "%d is in = %d, numberOfIteration = %d" %␣
↪(keyword,idxDataFounded,idxDataFounded+1)
    else:
        return "%d is not found, numberOfIteration = %d" % (keyword,len(data))
```

[66]:
```
data = insertionSortAscending(generateData(500,800))
print("Data = ",data,end="\n\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, 2, 2, 2, 3, 6, 7, 9, 9, 10, 12, 14, 14, 15, 16, 19, 20, 20, 21,
22, 22, 25, 25, 26, 30, 31, 32, 32, 34, 34, 34, 36, 40, 41, 47, 47, 51, 51, 52,
54, 59, 60, 60, 61, 63, 64, 66, 66, 66, 68, 69, 69, 70, 70, 72, 72, 74, 79, 80,
```

```
81, 82, 83, 84, 86, 86, 89, 90, 92, 92, 92, 92, 97, 97, 99, 100, 101, 101, 101,
104, 106, 107, 108, 108, 109, 109, 111, 111, 112, 115, 115, 115, 116, 117, 117,
119, 120, 121, 124, 125, 127, 128, 129, 131, 135, 135, 136, 137, 139, 140, 140,
141, 141, 144, 145, 148, 149, 152, 153, 156, 156, 157, 157, 160, 162, 168, 171,
171, 173, 173, 174, 177, 177, 177, 177, 179, 180, 181, 183, 183, 186, 187, 188,
192, 194, 194, 195, 198, 200, 204, 205, 206, 207, 207, 208, 209, 210, 211, 211,
212, 212, 212, 213, 213, 213, 214, 215, 223, 224, 224, 225, 225, 226, 230, 235,
242, 247, 252, 254, 257, 257, 258, 259, 262, 264, 265, 265, 269, 269, 270, 271,
274, 277, 278, 278, 279, 282, 282, 283, 287, 289, 289, 291, 291, 292, 293, 296,
299, 303, 304, 305, 308, 314, 315, 318, 318, 320, 325, 330, 331, 334, 335, 338,
339, 342, 342, 345, 346, 348, 349, 350, 351, 352, 352, 352, 355, 357, 359, 360,
362, 365, 366, 366, 367, 369, 369, 372, 374, 380, 382, 382, 383, 386, 388, 388,
389, 394, 397, 398, 399, 402, 403, 404, 404, 407, 407, 410, 411, 415, 417, 417,
420, 420, 424, 424, 427, 428, 429, 434, 435, 435, 435, 437, 437, 445, 445, 446,
448, 449, 450, 450, 452, 454, 454, 455, 459, 461, 463, 464, 466, 466, 469, 470,
471, 472, 473, 474, 475, 477, 478, 482, 488, 489, 491, 493, 494, 496, 497, 498,
499, 500, 501, 502, 502, 503, 505, 507, 510, 510, 511, 512, 512, 512, 520, 521,
524, 524, 528, 529, 529, 533, 533, 533, 535, 536, 539, 539, 540, 542, 542, 543,
543, 544, 545, 551, 552, 552, 552, 552, 553, 554, 554, 555, 558, 559, 559, 569,
569, 571, 571, 571, 575, 576, 577, 577, 578, 581, 582, 583, 586, 586, 587, 588,
588, 591, 592, 592, 592, 594, 595, 596, 598, 600, 606, 607, 607, 608, 610, 611,
612, 614, 614, 614, 620, 628, 629, 630, 630, 631, 633, 635, 638, 643, 647, 649,
652, 652, 653, 658, 658, 661, 661, 669, 669, 670, 670, 674, 674, 676, 684, 691,
696, 696, 696, 700, 700, 701, 704, 706, 707, 709, 710, 711, 711, 714, 715, 716,
719, 720, 724, 725, 727, 728, 729, 731, 739, 741, 741, 741, 742, 745, 745, 746,
746, 748, 748, 749, 750, 750, 751, 755, 757, 758, 759, 759, 760, 761, 764, 765,
771, 771, 771, 774, 774, 779, 781, 784, 785, 788, 789, 791, 791, 792, 793, 794,
795, 795, 796, 799, 799]

1 is in = 0, numberOfIteration = 1
25 is in = 22, numberOfIteration = 23
100 is in = 75, numberOfIteration = 76
20 is in = 17, numberOfIteration = 18
500 is in = 320, numberOfIteration = 321
1000 is not found, numberOfIteration = 500
```

[73]:
```python
def binarySearch(data, keyword):
    first = 0
    last = len(data) - 1
    found = False
    ind = 0
    while first <= last and not found:
        midpoint = (first + last) // 2
        if data[midpoint] == keyword:
            found = True
        elif data[midpoint] > keyword:
            last = midpoint - 1
```

```
        else:
            first = midpoint + 1
        ind += 1

    if found :
        return "%d is in : %d , numberOfItaration : %d" % (keyword,midpoint,ind)
    else :
        return "%d is not found , numberOfItaration : %d" % (keyword,ind)
```

[75]:
```
data = insertionSortAscending(generateData(500,800))
print("Data = ",data,end="\n\n")
print(binarySearch(data,1))
print(binarySearch(data,31))
print(binarySearch(data,254))
print(binarySearch(data,1320))
```

```
Data =  [2, 6, 7, 8, 11, 13, 13, 13, 13, 19, 22, 22, 23, 23, 23, 24, 24, 27, 28,
29, 30, 30, 30, 30, 31, 34, 35, 35, 36, 36, 39, 41, 42, 44, 45, 45, 46, 46, 46,
47, 50, 50, 54, 55, 55, 58, 61, 61, 61, 62, 63, 63, 64, 64, 65, 65, 67, 68, 68,
73, 77, 78, 81, 81, 81, 83, 85, 85, 87, 88, 91, 91, 93, 99, 100, 102, 102, 102,
103, 105, 107, 108, 109, 110, 110, 111, 112, 112, 114, 119, 120, 128, 135, 135,
135, 136, 140, 144, 150, 155, 157, 158, 158, 162, 163, 164, 173, 173, 173, 175,
176, 179, 183, 185, 188, 188, 190, 191, 191, 194, 195, 195, 196, 196, 197, 200,
200, 201, 203, 206, 210, 214, 217, 218, 219, 219, 221, 222, 223, 223, 225, 226,
227, 230, 235, 238, 243, 245, 245, 250, 251, 252, 254, 255, 257, 259, 262, 264,
267, 267, 268, 268, 270, 270, 270, 273, 273, 273, 277, 278, 278, 279, 279, 282,
282, 283, 285, 286, 287, 289, 290, 290, 290, 292, 292, 294, 301, 306, 306, 307,
308, 310, 312, 312, 312, 314, 315, 316, 317, 318, 321, 322, 324, 324, 329, 331,
332, 332, 336, 337, 339, 339, 340, 342, 343, 344, 348, 349, 350, 353, 354, 354,
356, 357, 361, 361, 362, 362, 367, 367, 368, 370, 371, 374, 375, 375, 377, 382,
382, 383, 387, 388, 393, 395, 397, 399, 399, 400, 401, 404, 404, 405, 405, 406,
406, 409, 410, 411, 411, 413, 414, 415, 416, 417, 418, 418, 419, 419, 420, 420,
426, 427, 427, 428, 430, 431, 432, 434, 435, 435, 437, 441, 443, 444, 444, 444,
445, 447, 451, 455, 456, 456, 461, 461, 463, 466, 467, 470, 471, 473, 474, 475,
475, 475, 476, 480, 482, 483, 484, 484, 497, 500, 503, 509, 521, 522, 523, 530,
531, 533, 535, 536, 539, 540, 540, 540, 540, 541, 542, 543, 543, 544, 544, 545,
548, 548, 548, 549, 549, 549, 552, 553, 554, 554, 554, 554, 555, 557, 559, 559,
560, 567, 568, 569, 570, 571, 572, 575, 576, 576, 578, 580, 583, 585, 587, 588,
589, 593, 594, 598, 598, 599, 602, 603, 603, 604, 604, 605, 607, 610, 611, 614,
619, 619, 626, 627, 628, 629, 629, 629, 630, 631, 635, 635, 638, 638, 638, 639,
639, 639, 640, 640, 642, 643, 643, 645, 647, 647, 653, 653, 654, 657, 657, 658,
658, 660, 662, 668, 669, 669, 671, 671, 671, 673, 674, 678, 680, 681, 688, 689,
691, 693, 695, 695, 697, 698, 701, 701, 708, 710, 710, 711, 713, 713, 714, 714,
717, 718, 718, 719, 719, 721, 723, 723, 726, 727, 727, 728, 731, 731, 731, 732,
732, 735, 736, 737, 738, 739, 740, 742, 743, 750, 753, 753, 754, 754, 756, 758,
759, 762, 767, 768, 769, 769, 770, 774, 777, 777, 780, 785, 785, 788, 793, 794,
795, 796, 799, 800, 800, 800]
```

```
1 is not found , numberOfItaration : 8
31 is in : 24 , numberOfItaration : 8
254 is in : 152 , numberOfItaration : 9
1320 is not found , numberOfItaration : 9
```

2.2 Hashing

[238]:
```
def createHash(length) :
    return [None for i in range(length)]

def remainderFun(num,lenData) :
    return num%lenData

def hashPut(nums,table) :
    for num in nums :
        idx = remainderFun(num,len(table))
        if table[idx] != None :
            if type(table[idx]) != list :
                table[idx] = [table[idx]]
            table[idx].append(num)

        else :
            table[idx] = num

    return table

def hashSearch(table, keyword) :

    hashVal = remainderFun(keyword,len(table))
    if type(table[hashVal]) == list :
        if keyword in table[hashVal] :
            return [hashVal,table[hashVal].index(keyword)]
        else :
            return False
    else :
        if table[hashVal] == keyword :
            return [hashVal,hashVal]
        else :
            return False
```

[268]:
```
data      = generateData(500,7000)
findData  = 6532
hashTable = hashPut(data,createHash(500))
found     = hashSearch(hashTable,findData)
```

```
if not found :
    print("%d 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:1
Data in slot [32]:[32, 6532]
```