1 Konsep

1.1 Searching

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

Jawaban

1.1 Searching

1. Sequential Search (Unordered) adalah algoritma pencarian data dengan cara membandingkan data yang di cari dengan seluruh data yang terdapat pada kumpulan data secara satu per satu mulai dari awal hingga akhir dan pencarian akan berhenti jika data di temukan kalau tidak ditemukan pencarian akan berlangsung sampai akhir.
2. Squential Search (Ordered) hampir sama dengan Squential Search (unoerdered) namun data sudah dalam keadaan terurut jadi data akan berhenti jika pencarian melebihi data yang dicari sehingga tidak perlu mencari sampai akhir sehingga mengurangi waktu komputasi pencarian.
3. Binary Search adalah proses pencarian data (dalam keadaan terurut) dengan cara mencari index tengah jika indeks tersebut sama dengan data yang dicari maka pencarian berhenti jika tidak maka akan di cek apakah lebih besar atau lebih kecil jika lebih besar pencarian tidak di mulai dari awal melainkan dari index tengah+1 hingga akhir begitu pula jika lebih kecil pencarian dimulai dari indeks pertama hingga indeks tengah-1. Proses ini terus dilakukan hingga data ditemukan atau tidak ditemukan.

1.2 Hashing

Jelaskan pencarian dengan menggunakan konsep Hashing, antara lain :

1. Table Hash, Slot
2. Fungsi Hash - Nilai Hash
3. Pencarian dengan konsep Hashing
4. Collusion

Jawaban

1.2 Hashing

1.
   - Hash Table adalah sebuah tempat penyimpanan data,untuk memudahka pencarian kita bisa menggunakan list untuk merepresentasikan hash table

- Slot yaitu posisi/indeks yang terdapat pada hash table sebagai tempat penyimpanan setiap data karena slot adalah indeks maka slot berupa integer mulai dari 0 - n.

2. - Fungsi Hash adalah fungsi yang digunakan untuk mengetahui indeks yang sesuai dengan slot dalam table hash.
   - Nilai hash adalah hasil dari proses fungsi hash yang digunakan untuk menentukan slot tempat data akan disimpan dalam tabel.
3. Pencarian dalam konsep hashing dilakukan hanya dengan melakuakan satu perbandingan apakah data yang dicari sama dengan data dengan indeks nilai hash atau tidak hal ini bisa di lakukan karena data sudah di tempatkan pada hash table.
4. Collusion adalah keadaan ketika dua data atau lebih memiliki nilai hash yang sama sehingga menempati indeks yang sama. collusion ini bisa diatasi dengan menggunakan chaining atau linear probing.

2 Implementasi

2.1 Searching

Buatlah code untuk mengimplementasikan masing-masing algoritma searching, dengan ketentuan sebagai berikut :

- data digenerate 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

```
1 !git clone https://github.com/ali-vian/Data-Structure.git
```

```
1 import sys
2 sys.path.insert(0,'/content/Data-Structure')
```

```
1 # 2.1 Searching
2
3 import random as rd
4 data = [rd.randint(1,100) for i in range(500)]
5
6 def seqSearch(dataList,data):
7     position = []
8     for i in range(len(dataList)):
9         if dataList[i] == data :
10            position.append(i)
11    if position != []:
12        return f"{data} is in : {position}"
13    else:
14        return f"{data} not found"
15
16
```

```
print('Data =',data,'\n')
print(seqSearch(data,100))
print(seqSearch(data,50))
print(seqSearch(data,500))
```

```
Data = [85, 37, 12, 39, 7, 55, 56, 9, 57, 32, 48, 22, 24, 29, 37, 49, 70, 37, 50, 77,

100 is in : [40, 107, 138, 151, 152, 160, 232, 280, 287, 380]
50 is in : [18, 27, 136, 244, 346]
500 not found
```

```
def orderedSeqSearch(data,num):
    i = 0
    position=[]
    while i < len(data) and data[i] <= num:
        if data[i] == num :
            position.append(i)
        i+=1
    if position != []:
        return f"{num} is in : {position}, numberOfIteration: {i}"
    return f"{num} is not found, numberOfIteration: {i}"

import sorting as srt
srt.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 = [2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 7,

1 is not found, numberOfIteration: 0
25 is in : [111, 112, 113, 114], numberOfIteration: 115
100 is in : [490, 491, 492, 493, 494, 495, 496, 497, 498, 499], numberOfIteration: 50
20 is in : [85, 86, 87, 88], numberOfIteration: 89
500 is not found, numberOfIteration: 500
1000 is not found, numberOfIteration: 500
```

```
def binarySearch(data,find):
    i = 0
    first=0
    last=len(data)-1
    while first <= last :
        mid = (first+last)//2
        i+=1
        if data[mid] == find:
            return f"{find} is in : {mid}, numberOfiteration: {i}"
        elif find < data[mid]:
            last = mid-1
```

```
        else:
            first = mid+1
    return f"{find} is not found, numberOfIteration: {i}"

def createDataUnique(size):
    res = []
    i=0
    while i <= size:
        temp = rd.randint(1,999)
        if temp not in res :
            res.append(temp)
            i+=1
    return res

data1 = createDataUnique(500)
srt.insertionSort(data1)
print(data1)
```

```
[4, 5, 6, 8, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 28, 29, 30, 31, 36, 38, 41, 44,
```

```
print(binarySearch(data1,1))
print(binarySearch(data1,31))
print(binarySearch(data1,254))
print(binarySearch(data1,1320))
```

```
1 is not found, numberOfIteration: 8
31 is in : 17, numberOfiteration: 9
254 is not found, numberOfIteration: 9
1320 is not found, numberOfIteration: 9
```

2.2 Hashing

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

- data digenerate secara random sebanyak 500 data (Hint : gunakan modul random yang sudah disediakan python)
- generate data sedemikian hingga, kemungkinan data yang sama adalah kecil
- Gunakan penanganan collusion dengan menggunakan chaining. Contoh output yang dihasilkan dapat dilihat pada Gambar 2

```
# 2.2 Hashing

def remainderFunction(num,data):
    return num%data

def createHashTable(size):
    return [ [None] for i in range(size)]

def putData(data,table):
```

```
    for i in range(len(data)):
        idx = remainderFunction(data[i],len(table))
        if table[idx][0] == None :
            table[idx][0] = data[i]
        else:
            table[idx].append(data[i])
    return table

def searchHash(data,table):
    hasVal = remainderFunction(data,len(table))
    for i in table[hasVal]:
        if data ==  i :
            return [hasVal,table[hasVal].index(i)]
    return False
```

```
data2 = [rd.randint(0,9999) for i in range(500)]
numOfSlot =100
hashTable = createHashTable(numOfSlot)
hashTable = putData(data2,hashTable)
```

```
findData = 6532
found = searchHash(findData,hashTable)
if not found:
    print('%d is not in the hashTable'%(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: 3
Data in slot[32] : [2832, 8732, 4732, 6532]
```

```
numOfSlot1 = 300
hashTable1 = createHashTable(numOfSlot1)
hashTable1 = putData(data2,hashTable1)
```

```
findData1 = 6532
found1 = searchHash(findData1,hashTable1)
if not found1:
    print('%d is not in the hashTable'%(findData1))
else:
    print('%d is in slot %d indeks: %d'%(findData1,found1[0],found1[1]))
    print('Data in slot[%d] : %s'%(found1[0],hashTable1[found1[0]]))
```

```
6532 is in slot 232 indeks: 1
Data in slot[232] : [4732, 6532]
```

```
numOfSlot2 = 500
hashTable2 = createHashTable(numOfSlot2)
hashTable2 = putData(data2,hashTable2)
```

```
findData2 = 6532
found2 = searchHash(findData2,hashTable2)
if not found2:
    print('%d is not in the hashTable'%(findData2))
else:
    print('%d is in slot %d indeks: %d'%(findData2,found2[0],found2[1]))
    print('Data in slot[%d] : %s'%(found2[0],hashTable2[found2[0]]))
```

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