ALGORITMA DAN STRUKTUR DATA

INF1083

LAPORAN PRAKTIKUM 9 : Searching

Oleh :

Akhmad Qasim

2211102441237

Teknik Informatika
Fakultas Sains & Teknologi
Universitas Muhammadiyah Kalimantan Timur

Samarinda, 2023

Laporan Praktikum 9:
Searching

Pokok Bahasan:

- ❖ Searching
- ❖ Hashing

Tujuan Pembelajaran:

- ✓ Memahami implementasi searching menggunakan Python
- ✓ Memahami implementasi dari metode-metode hashing

Percobaan & Latihan:

1. **Searching**
   a. **Berikan tampilan output dari perintah diatas!**
   Tampilan output:

```
False
True
False
```

   b. **Jelaskan operator in pada perintah diatas!**
   Operator in berfungsi untuk mengecek apakah suatu nilai ada di dalam suatu list atau tidak. Jika ada, maka akan menghasilkan nilai True, jika tidak ada, maka akan menghasilkan nilai False.

2. **Searching**
   a. **Berikan tampilan output dari kedua fungsi tersebut!**
   Output 1:

```
False
True
```

   Output 2:

```
False
True
```

   b. **Analisa perbedaan kedua fungsi tersebut!**
   Analisa 1:

```python
# Fungsi pertama untuk mencari nilai tertentu dalam list
2 usages  akhmadqasim
def sequentialSearch(alist, item):  # alist adalah list yang akan dicari, item adalah nilai yang dicari
    pos = 0  # posisi awal
    found = False  # nilai awal

    while pos < len(alist) and not found:  # selama posisi masih kurang dari panjang list dan nilai belum ditemukan
        if alist[pos] == item:  # jika nilai yang dicari sama dengan nilai yang ada di list
            found = True  # maka nilai ditemukan
        else:  # jika tidak
            pos = pos + 1  # maka posisi akan bertambah 1

    return found  # mengembalikan nilai


testlist = [1, 2, 32, 8, 17, 19, 42, 13, 0]  # list yang akan dicari
print(sequentialSearch(testlist, 3))  # mencari nilai 3
print(sequentialSearch(testlist, 13))  # mencari nilai 13
```

Analisis 2:

```python
# Fungsi kedua untuk mencari nilai tertentu dalam list
2 usages  akhmadqasim +1
def orderedSequentialSearch(alist, item):  # alist adalah list yang akan dicari, item adalah nilai yang dicari
    pos = 0  # posisi awal
    found = False  # nilai awal
    stop = False  # nilai awal

    # selama posisi masih kurang dari panjang list dan nilai belum ditemukan dan nilai belum ditemukan
    while pos < len(alist) and not found and not stop:
        if alist[pos] == item:  # jika nilai yang dicari sama dengan nilai yang ada di list
            found = True  # maka nilai ditemukan
        else:  # jika tidak
            if alist[pos] > item:  # jika nilai yang ada di list lebih besar dari nilai yang dicari
                stop = True  # maka nilai ditemukan
            else:  # jika tidak
                pos = pos + 1  # maka posisi akan bertambah 1

    return found  # mengembalikan nilai


testlist = [0, 1, 2, 8, 13, 17, 19, 32, 42]  # list yang akan dicari
print(orderedSequentialSearch(testlist, 3))  # mencari nilai 3
print(orderedSequentialSearch(testlist, 13))  # mencari nilai 13
```

3. Searching

a. Berikan tampilan output dari kedua fungsi tersebut!

Output 1:

```
False
True
```

Output 2:

```
False
True
```

b. Jelaskan fungsi variabel midpoint dari kedua fungsi diatas!

Midpoint variabel di kedua fungsi binarySearch dan binarySearch2 mewakili indeks elemen tengah dalam daftar alist yang sedang dicari. Ini

dihitung sebagai pembagian bilangan bulat dari jumlah pertama dan terakhir (indeks yang mewakili elemen pertama dan terakhir dari sublist yang sedang dicari) dengan 2. Berfungsi sebagai pivot untuk membagi list menjadi dua bagian dan menentukan setengahnya pencarian berlanjut. Jika item yang dicari sama dengan nilai di alist[midpoint], maka pencarian berhasil dan fungsi mengembalikan True. Jika item tidak ditemukan, pencarian dilanjutkan di bagian bawah atau bagian atas list, tergantung pada apakah item lebih kecil atau lebih besar dari nilai di alist[midpoint].

c. Analisa perbedaan kedua fungsi tersebut!

Fungsi 1:

```
2 usages  akhmadqasim +1
def binarySearch(alist, item):  # alist adalah list yang akan dicari, item adalah nilai yang dicari
    first = 0  # posisi awal
    last = len(alist) - 1  # posisi akhir
    found = False  # nilai awal

    # selama posisi awal lebih kecil dari sama dengan posisi akhir dan nilai belum ditemukan
    while first <= last and not found:
        midpoint = (first + last) // 2  # nilai tengah
        if alist[midpoint] == item:  # jika nilai yang dicari sama dengan nilai yang ada di list
            found = True  # maka nilai ditemukan
        else:  # jika tidak
            if item < alist[midpoint]:  # jika nilai yang dicari lebih kecil dari nilai yang ada di list
                last = midpoint - 1  # maka posisi akhir akan berkurang 1
            else:  # jika tidak
                first = midpoint + 1  # maka posisi awal akan bertambah 1

    return found  # mengembalikan nilai
```

Fungsi 2:

```
4 usages  akhmadqasim
def binarySearch2(alist, item):  # alist adalah list yang akan dicari, item adalah nilai yang dicari
    if len(alist) == 0:  # jika panjang list adalah 0
        return False  # maka nilai tidak ditemukan
    else:  # jika tidak
        midpoint = len(alist) // 2  # nilai tengah
        if alist[midpoint] == item:  # jika nilai yang dicari sama dengan nilai yang ada di list
            return True  # maka nilai ditemukan
        else:  # jika tidak
            if item < alist[midpoint]:  # jika nilai yang dicari lebih kecil dari nilai yang ada di list
                return binarySearch2(alist[:midpoint], item)  # maka akan dicari lagi di list sebelum nilai tengah
            else:  # jika tidak
                return binarySearch2(alist[midpoint + 1:], item)  # maka akan dicari lagi di list setelah nilai tengah


testlist = [0, 1, 2, 8, 13, 17, 19, 32, 42]  # list yang akan dicari
print(binarySearch2(testlist, 3))  # mencari nilai 3
print(binarySearch2(testlist, 13))  # mencari nilai 13
```

4. Hashing

a. Berikan tampilan output dari perintah tersebut!

Tampilan output:

```
Nilai ordinal dari 'c' adalah 99
Nilai ordinal dari 'a' adalah 97
Nilai ordinal dari 'r' adalah 114
Nilai ordinal dari 'i' adalah 105
```

b. Jelaskan fungsi operator *ord()* diatas!

Fungsi ord() untuk mengubah karakter menjadi nilai ordinal. Ordinal adalah nilai yang merepresentasikan posisi karakter dalam tabel ASCII

c. Uji coba dengan menggunakan tanda baca, apakah tanda baca memiliki nilai ordinal? Berikan tampilan output!

Output:

```
Nilai ordinal dari '?' adalah 63
```

Tanda baca juga memiliki nilai ordinal, salah satu contohnya adalah tanda tanya '?' yang bernilai 63.

5. Hashing

a. Berikan tampilan output dari perintah tersebut!

Output:

```
8
```

b. Jelaskan fungsi hash() diatas!

Fungsi hash(astring, tablesize) adalah sebuah implementasi sederhana dari fungsi hash yang mengubah sebuah string (astring) menjadi sebuah nilai integer. Fungsi ini menggunakan pendekatan penjumlahan nilai ordinal (ASCII) dari setiap karakter dalam string, kemudian melakukan operasi modulo (%) dengan tablesize untuk memastikan nilai yang dihasilkan berada dalam rentang indeks yang valid di dalam tabel hash.

c. Uji coba dengan mengubah nilai variable MyString dengan tanda baca, apakah tanda baca memiliki slot ditabel hash? Berikan tampilan output!

Dalam implementasi fungsi hash yang diberikan, tanda baca (seperti "?", ".", ",", dll.) juga akan dianggap sebagai karakter dan akan dihitung dalam perhitungan nilai hash. Oleh karena itu, tanda baca juga akan memiliki slot dalam tabel hash dan akan dihitung dalam perhitungan indeks hasil hash.

Tampilan output:

```
8
```

6. Hashing

a. Berikan tampilan output dari perintah tersebut!

Tampilan output:

```
['bird', 'goat', 'pig', 'chicken', 'dog', 'lion', 'tiger', None, None, 'cow', 'cat']
chicken
tiger
duck
None
```

b. Jelaskan fungsi kelas HashTable diatas!

Tabel hash adalah struktur data yang menggunakan fungsi hash untuk mengonversi kunci (key) menjadi indeks dalam array atau daftar (slots) untuk menyimpan nilai (data). Fungsi-fungsi utama dalam kelas HashTable adalah sebagai berikut:

1. __init__(self): Metode inisialisasi (constructor) yang digunakan untuk membuat tabel hash dengan mengatur ukuran (size) slots, dan menginisialisasi slots dan data dengan nilai None.

2. put(self, key, data): Metode untuk menambahkan pasangan kunci-nilai (key-value pair) ke dalam tabel hash. Metode ini menggunakan fungsi hash (hashfunction) untuk mengubah kunci menjadi indeks dalam slots. Jika slots pada indeks tersebut kosong (None), maka kunci dan nilai akan dimasukkan ke dalam slots dan data pada indeks tersebut. Jika slots pada indeks tersebut sudah terisi, maka metode akan mencari slot kosong berikutnya dengan menggunakan fungsi rehash (rehash) hingga menemukan slot kosong untuk menempatkan kunci dan nilai.

3. hashfunction(self, key, size): Metode untuk menghitung nilai hash dari kunci. Metode ini menggunakan operasi modulo (%) untuk menghitung sisa bagi antara kunci dan ukuran slots.

4. rehash(self, oldhash, size): Metode untuk menghitung ulang nilai hash jika slot yang diinginkan sudah terisi. Metode ini menggunakan operasi modulo (%) untuk menghitung sisa bagi antara (oldhash + 1) dan ukuran slots.

5. get(self, key): Metode untuk mengambil nilai (data) dari tabel hash berdasarkan kunci (key) yang diberikan. Metode ini menggunakan fungsi hash (hashfunction) untuk menghitung indeks dalam slots, dan kemudian mencari nilai yang sesuai dengan kunci pada indeks tersebut. Jika tidak ditemukan, metode

akan mencari ke slot berikutnya dengan menggunakan fungsi rehash (rehash) hingga menemukan nilai yang sesuai atau menemui slot kosong.

6. __getitem__(self, key): Metode untuk mengambil nilai (data) dari tabel hash menggunakan notasi indeks []. Metode ini hanya memanggil metode get(self, key).

7. __setitem__(self, key, data): Metode untuk menambahkan pasangan kunci-nilai (key-value pair) ke dalam tabel hash menggunakan notasi indeks []. Metode ini hanya memanggil metode put(self, key, data).

Dalam contoh kode di akhir, objek H dari kelas HashTable dibuat dan beberapa pasangan kunci-nilai ditambahkan ke dalam tabel hash. Kemudian, nilai dari slots dan data dalam tabel hash ditampilkan menggunakan atribut slots dan data. Selanjutnya, nilai dari tabel hash dapat diambil menggunakan notasi indeks [] atau metode get(key). Jika nilai dengan kunci yang sama sudah ada dalam tabel hash, metode put(key, data) akan menggantikan nilai tersebut dengan nilai baru.

c. Berikah penjelasan pada baris 67, 68 dan 69 pada program diatas!

Baris 67:
H[20] = "chicken": Fungsi ini menambahkan nilai "chicken" ke dalam tabel hash H dengan kunci (key) 20. Nilai "chicken" akan di-hash (diubah menjadi bilangan hash) menggunakan fungsi hash yang telah didefinisikan sebelumnya, dan hasil hash tersebut akan digunakan sebagai indeks untuk menyimpan nilai "chicken" dalam tabel hash.

Baris 68:
print(H.slots): Fungsi ini akan menampilkan slot-slot (indeks-indeks) yang ada dalam tabel hash H. Slot-slot ini adalah tempat di dalam tabel hash di mana nilai-nilai akan disimpan berdasarkan hasil hash yang dihasilkan.

Baris 69:
print(H.data): Fungsi ini akan menampilkan data yang ada dalam tabel hash H. Data ini adalah nilai-nilai yang telah dimasukkan ke dalam tabel hash menggunakan kunci (key) dan nilai (value) yang telah ditentukan sebelumnya.

Kesimpulan:

Searching (pencarian) adalah proses untuk mencari keberadaan suatu data atau informasi dalam suatu struktur data. Beberapa metode pencarian yang umum digunakan antara lain linear search, binary search, dan hash-based search.

Hashing adalah teknik yang digunakan untuk mengkonversi data menjadi bentuk hash atau bilangan acak. Hash digunakan sebagai indeks atau alamat untuk menyimpan dan mencari data dalam struktur data hash table (tabel hash). Hashing sangat efisien dalam pencarian data, karena memungkinkan pencarian dalam waktu konstan (O(1)) pada kasus terbaik.

Implementasi searching menggunakan Python dapat dilakukan dengan menggunakan metode pencarian seperti linear search, binary search, atau hash-based search. Linear search adalah metode pencarian sederhana yang melibatkan pencarian secara berurutan dari awal hingga akhir data. Binary search adalah metode pencarian yang hanya dapat digunakan pada data yang sudah diurutkan, dan mengurangi jumlah langkah pencarian secara signifikan. Hash-based search menggunakan struktur data hash table (tabel hash) untuk menyimpan dan mencari data menggunakan indeks atau alamat yang dihasilkan dari fungsi hash.

Implementasi metode-metode hashing dapat dilakukan dalam Python dengan membuat fungsi hash yang menerima input data dan menghasilkan nilai hash. Nilai hash tersebut kemudian dapat digunakan sebagai indeks atau alamat untuk menyimpan dan mencari data dalam tabel hash. Penting untuk memilih fungsi hash yang baik agar hasil hash merata dan menghindari kolisi (collision) yang dapat mempengaruhi kinerja pencarian dalam struktur data hash table.