UNIVERSITAS MIKROSKIL | Fakultas INFORMATIKA

IF2103 – Berpikir Komputasional

Recursion

- Pengenalan Recursion
- Depth First Search (DFS)

Strata - 1

Teknik Informatika

Pokok dan Subpokok Bahasan

Recursion

a. Pengenalan Recursion

b. Depth First Search (DFS)

Pengenalan Recursion

- Rekursi merupakan sebuah metode perulangan yang bersifat **non-iterasi**
- Fungsi rekursif kurang lebih seperti fungsi def pada umumnya, namun karena fungsi ini nantinya **memanggil dirinya sendiri**, maka akan menimbulkan efek perulangan yang akan berhenti ketika mencapai suatu kondisi

Pengenalan Recursion

- Sebagai contoh, disini rekursi akan dimanfaatkan untuk mencetak deret angka mulai dari yang terkecil hingga yang terbesar

```
def rekursif(x):
    if(x > 1):              #kondisi selama nilai x > 1 maka panggil kembali
        rekursif(x-1)       #fungsi rekursif dengan x dikurang 1
    print(x, end= ' ')      #kemudian cetak bilangan tersebut

rekursif(10) #pemanggilan fungsi rekursif dengan nilai x = 10
```


- Hasil Print:

```
1 2 3 4 5 6 7 8 9 10
>>>
```

Pengenalan Recursion

- Pada contoh sebelumnya, pemanggilan fungsi rekursi yang dilakukan adalah seperti berikut. Kompleksitas = **O(n)**

Kondisi	Pemanggilan Rekursif & Perintah Print	Hasil Cetak
10 > 1 = True	rekursif (9) print(10)	10
9 > 1 = True	rekursif (8) print(9)	9 10
8 > 1 = True	rekursif (7) print(8)	8 9 10
7 > 1 = True	rekursif (6) print(7)	7 8 9 10
6 > 1 = True	rekursif (5) print(6)	6 7 8 9 10
5 > 1 = True	rekursif (4) print(5)	5 6 7 8 9 10
4 > 1 = True	rekursif (3) print(4)	4 5 6 7 8 9 10
3 > 1 = True	rekursif (2) print(3)	3 4 5 6 7 8 9 10
2 > 1 = True	rekursif (1) print(2)	2 3 4 5 6 7 8 9 10
1 > 1 = False	print(1)	1 2 3 4 5 6 7 8 9 10

Pengenalan Recursion

- Contoh lain, disini rekursi akan dimanfaatkan untuk menjumlahkan bilangan dari 1 sampai x

```
def sumRekursif(x):
    if(x == 1):                          # Jika nilai x mencapai 1
        return 1                         # Maka nilai yang dikembalikan adalah 1
    else:                                # Jika tidak maka Kembalikan nilai x
        return x + sumRekursif(x-1)  # ditambah fungsi rekursi(x-1) sampai tercapai x=1

print(sumRekursif(10)) #pemanggilan fungsi penjumlahan dengan nilai x = 10
```


- Hasil Print:

```
55
>>>
```

Pengenalan Recursion

- Pada contoh sebelumnya, pemanggilan fungsi rekursi yang dilakukan adalah seperti berikut. Kompleksitas = **O(n)**

Kondisi	Pemanggilan Rekursif	Hasil Jumlah
10 > 1 = True	10 + sumRekursif (9)	10
9 > 1 = True	10 + 9 + sumRekursif (8)	19
8 > 1 = True	10 + 9 + 8 + sumRekursif (7)	27
7 > 1 = True	10 + 9 + 8 + 7 + sumRekursif (6)	34
6 > 1 = True	10 + 9 + 8 + 7 + 6 + sumRekursif (5)	40
5 > 1 = True	10 + 9 + 8 + 7 + 6 + 5 + sumRekursif (4)	45
4 > 1 = True	10 + 9 + 8 + 7 + 6 + 5 + 4 + sumRekursif (3)	49
3 > 1 = True	10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + sumRekursif (3)	52
2 > 1 = True	10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + sumRekursif (1)	54
1 > 1 = False	10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1	55

Pengenalan Recursion

- **Latihan**

  Dengan memanfaatkan rekursi, buatlah algortima untuk menghitung deret bilangan dari yang terbesar hingga yang terkecil (n sampai 1)

Depth First Search (DFS)

- **Depth First Search (DFS)** adalah salah satu metode graph traversal selain BFS yang cukup banyak digunakan
- **Depth First Search** digunakan untuk mencari struktur data tree dengan menelusuri satu cabang sampai menemukan sebuah solusi, jika solusi tidak ditemukan pada suatu cabang, maka perlu dilakukan proses backtracking untuk kembali ke node sebelumnya dan melakukan penelusuran kembali ke cabang lain

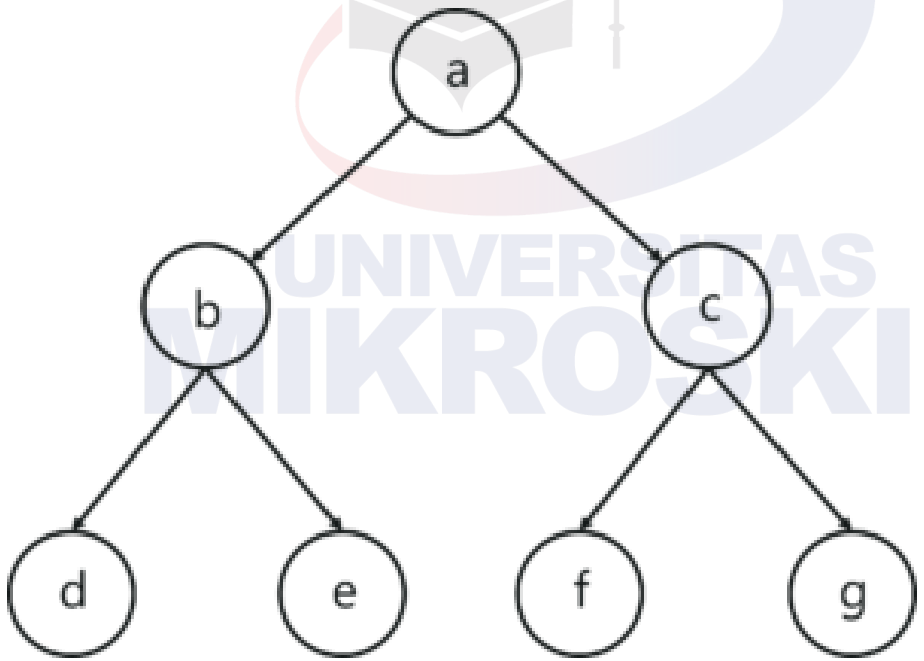
Depth First Search (DFS)

Langkah-langkah yang dilakukan dalam melintasi *graph* dengan menggunakan algoritma DFS :

1. Ambil tumpukan (*stack*) kosong
2. Pilih *node* awal (sebagai *root*) dan masukkan ke dalam tumpukan
3. Selama tumpukan tidak kosong, ambil salah satu *node* dari tumpukan dan masukkan salah satu *node* anaknya ke dalam tumpukan
4. Cetak simpul yang diambil
5. Lakukan backtracking (mengeluarkan isi tumpukan) untuk kembali ke *node* sebelumnya

Depth First Search (DFS)

Perhatikan *graph* di bawah ini, kita akan menggunakan algoritma DFS untuk melintasi *graph*



Depth First Search (DFS)

Dalam kasus ini, kita akan menetapkan simpul ‘a’ sebagai simpul akar dan mulai melintasi ke bawah dan mengikuti langkah-langkah DFS.

1. Simpul ‘a’ akan dianggap sebagai simpul akar dan masukkan ke dalam tumpukan
2. Kemudian ambil salah satu anak dari simpul ‘a’, misalnya ‘b’ lalu masukkan ke tumpulkan
3. Cetak simpul ‘a’
4. Lihat kembali anak dari simpul ‘b’, masukkan salah satunya kedalam tumpukan
5. Ulangi langkah ini sampai simpul tidak memiliki anak lagi, simpul yang sudah dikunjungi bisa dikeluarkan dari tumpukan, dan masukkan kembali simpul yang belum dikunjungi kedalam tumpukan

Depth First Search (DFS)

print a:

Masukkan 'a' kedalam tumpukan, cetak 'a' dan tandai sebagai 'visited'

print b:

Masukkan 'b' kedalam tumpukan, cetak 'b' dan tandai sebagai 'visited'

print d:

Masukkan 'd' kedalam tumpukan, cetak 'd' dan tandai sebagai 'visited'

'd' tidak memiliki anak lagi dan statusnya 'visited', keluarkan dari tumpukan

print e:

Masukkan 'e' kedalam tumpukan, cetak 'e' dan tandai sebagai 'visited'

'e' tidak memiliki anak lagi dan statusnya 'visited', keluarkan dari tumpukan

semua anak dari 'b' statusnya 'visited', keluarkan dari tumpukan

print c:

Masukkan 'c' kedalam tumpukan, cetak 'c' dan tandai sebagai 'visited'

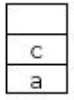
print f:

Masukkan 'f' kedalam tumpukan, cetak 'f' dan tandai sebagai 'visited'

'f' tidak memiliki anak lagi dan statusnya 'visited', keluarkan dari tumpukan

print g:

Masukkan 'g' kedalam tumpukan, cetak 'g' dan tandai sebagai 'visited'



'g' tidak memiliki anak lagi dan statusnya 'visited', keluarkan dari tumpukan

semua anak dari 'c' statusnya 'visited', keluarkan dari tumpukan

semua anak dari 'a' statusnya 'visited', keluarkan dari tumpukan

Hasil Penelusuran:

a b d e c f g

Depth First Search (DFS)

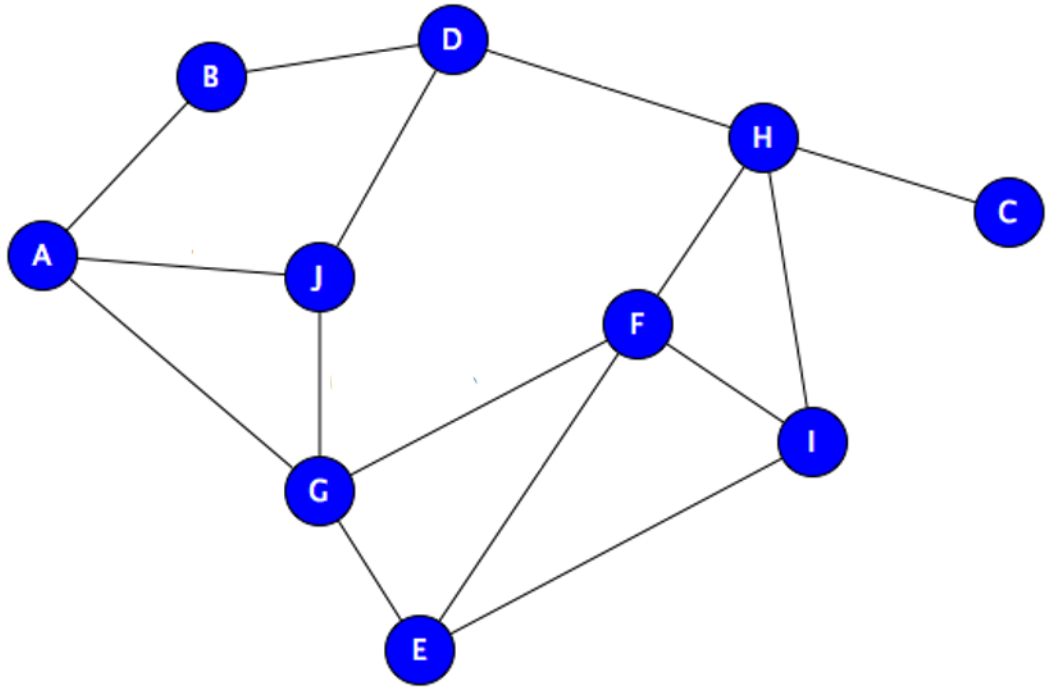
```
def dfs(graph, start, visited=None):            #fungsi rekursi untuk DFS
    if visited is None:                         #jika diawal belum ada node yang dikunjungi
        visited = set()                         #maka buat penampung untuk node yang dikunjungi
    visited.add(start)                          #dan tambahkan node yang pertama kali dikunjungi

    print(start,end="")                         #cetak start untuk menandakan sudah dikunjungi

    for next in set(graph[start]) - visited:    #untuk setiap tetangga dari node yang dikunjungi
        dfs(graph, next, visited)               #panggil fungsi DFS dengan node tetangga yang akan dikunjungi
                                                #beserta penampung node yang sudah dikunjungi

    return visited      #kembalikan node yang sudah dikunjungi setiap fungsi ini dipanggil

graph = {               #mendeklarasikan graph dan node
    'a': ['b', 'c'],    #beserta node tetangganya
    'b': ['d', 'e'],
    'c': ['f', 'g'],
    'd': [],
    'e': [],
    'f': [],
    'g': [],
    }

dfs(graph, 'a')         #memanggil fungsi DFS dengan root 'a' (node yang pertama dikunjungi)
```

Kompleksitas algoritmanya adalah **O(V + E)** dimana V adalah jumlah node dan E adalah jumlah edge (tetangga masing-masing node)

Depth First Search

- **Latihan**

  Dengan memanfaatkan rekursi dan DFS, lakukan pencarian simpul C pada *graph* berikut dan tampilkan rute yang dilalui.



Thanks !

Materi dan diskusi dapat diakses di Microsoft teams