# *Implementasi Depth First Search (DFS) dan Breadth First Search (BFS) pada Python*

M. Fikri Damar Muchtarom 120140077 (*Author*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): mfikri.120140077@student.itera.ac.id

***Abstract*—Python merupakan bahasa programan interpretative multiguna. Python meurpakan sebuah bahasa yang lebih mudah untuk dibaca karena python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sebuah sintaks. Hal ini juga yang membuat python sangat baik untuk dipelajari oleh pemula maupun orang yang sudah menguasai bahasa pemrograman lain. Python juga dapat dijalankan hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah bisa menjalankan Python. Dan kita juga bisa mengimplementasikan strategi algoritma *Depth First Search* (DFS) dan *Breadth First Search* (BFS) dengan menggunakan Python.**

***Keywords—python; dfs; bfs; algoritma; linux***

## I. PENDAHULUAN

Pemrograman merupakan proses menulis, menguji dan memperbaiki (*debug)*, dan memelihara kode yang telah dibangun didalam suatau sistem komputer. Kode ini dapat ditulis dalam berbagai macam bahasa pemrograman atau sering disebut juga dengan bahasa komputer. Pemrograman ditujukan untuk memuat suatu program yang dapat melakukan suatu perhitungan yang sesuai dengan keingingan user. Pemrograman juga merupakan suatu cara dalam membuat satu atau lebih dari satu algoritma dengan menggunakan suatu bahasa pemrograman tertentu sehingga menjadi suatu program komputer. Bahasa pemrograman juga ada banyak dan setiap bahasa memiliki gaya yang berbeda-beda dalam penggunaannya. Gaya dalam pemrograman ini juga dapat disebut dengan paradigma pemrograman.

Algoritma merupakan metode atau langkah yang direncanakan secara tersusun dan saling berurutan yang ditujukan untuk menyelesaikan suatau permasalahan dengan sebuah instruksi atau kegiatan. Algoritma memiliki alur pemikiran untuk memecahkan suatu masalah yang terdiri atas sejumlah langkah matematis.

Strategi algoritma adalah kumpulan metode atau teknik untuk memecahkan suatu masalah untuk mencapai tujuan yang telah ditentukan, dalam hal ini deskripsi metode atau teknik tersebut dinyatakan dalam suatu urutan langkah-langkah penyelesaian.

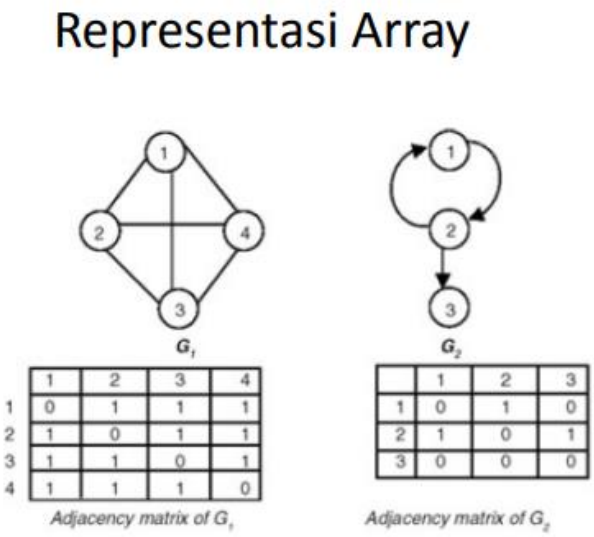
*Depth First Search* (DFS) adalah salah satu algoritma penelusuran dengan struktur graf/pohon berdasarkan kedalaman. Diawali dengan menelusuri root, kemudian ke salah satu folder anaknya, penelusuran dilakukan terus melalui folder anak pertama dari folder sebelumnya hingga mencapai folder yang terdalam.

*Breadth First Search* (BFS) merupakan sebuah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu.

## II. TEORI DASAR

### A. *Depth First Search* (DFS)

Algoritma DFS ini adalah salah satu algoritma yang digunakan untuk pencarian jalur. Algoritma ini mirip dengan algoritma BFS namun bedanya adalah algoritma BFS menggunakanan perhitungan secara terurut dari urutan pertama sampai urutan terakhir, maka algoritma DFS ini melakukan hitungan secara terurut dari urutan terakhir. Setelah menghabiskan semua kemungkinan di titik terakhir, baru algoritma ini akan mundur ke titik-titik sebelumnya sampai kepada titik awal.



Representasi Array

- Salah satu representasi graph adalah dengan matrik $n^2$ (matrik dengan baris dan n kolom, artinya kolom berhubungan ke setiap vertex pada graph).
- Jika ada edge dari $v_i$ ke $v_j$ maka entri dalam matrik dengan index baris sebagai $v_i$ dan index kolom sebagai $v_j$ yang di set ke 1 (adj[$v_i$,$v_j$]=1, jika ($v_i$,$v_j$) adalah suatu edge dari graph G).
- Jika e merupakan total jumlah edge dalam graph, maka ada entri 2e yang di set ke 1, selama G adalah graph yang tidak memiliki arah.
- Jika G adalah graph yang memiliki arah, hanya entri e yang di set ke-1 dalam matrik keterhubungan.

*B. Breadth First Search* (BFS)

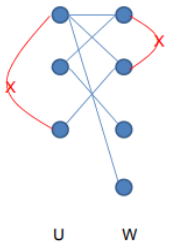
Algoritma BFS merupakan salah satu algoritma yang digunakan untuk mencari sebuah jalur. Algoritma ini adalah satu algoritma pencarian jalur yang sederhana, dimana pencarian dimulai dari titik awal, lalu dilanjutkan ke semua cabang titik tersebut secara berurutan. Jika titik tujuan belum ditemukan, maka perhitungan akan diulang lagi ke masing-masing titik cabang dari masing-masing titik, sampasi titik tujuan itu telah ditemukan.

Properti dan Running Time

- O(V+E)
- G = (V,E), BFS akan mencari seluruh vertek yang bisa di raih dari source s
- Untuk setiap vertek pada level I, path bfs tree antara s dan v mempunyai I edge dan selain path dalam G antara s dan v setidaknya mempunyai i edge
- Jika (u,v) adalah edge maka jumlah level u dan v dibedakan setidaknya satu tingkat
- BFS akan menghitung seluruh jarak terpendek ke seluruh vertek yang bisa diraih olehnya.

Kegunaan BFS

- Memeriksa apakah graf terhubung
- Menghitung spanning forest graph
- Menghitung, tiap vertex dalam graf, jalur dengan jumlah edge yang paling sedikit antara vertex awal dan current vertex atau ketiadaan jalur.
- Menghitung cycle dalam graf atau ketiadaan cycle
- O(V+E).




Bipartite Graph

- G = (V,E) graf yang tidak memiliki arah
- G adalah BG jika ada suatu partisi dari V ke dalam V, W
- Sedemikian rupa sehingga
- Setiap edge memiliki satu end point dalam U dan lainnya dalam W.

*C. Python*

Python merupakan salah satu bahasa pemrograman paling popular di dunia dalam beberapa tahun terakhir. Bahasa pemrograman ini digunakan dalam segala hal mulai dari *machine learning*, membangun situs web, dan pengujian software.

## III. ANALISIS DAN PEMBAHASAN

Pada tugas kali ini saya mengambil topik pengimplementasian *Depth First Search* (DFS) dan *Breadth First Search* (BFS) dengan menggunakan *python*.

Saya akan melakukan beberapa uji coba dengan program kode yang telah saya buat, program ini nantinya akan mencari sesuai dengan DFS dan BFS.

*A. Depth First Search* (DFS)

DFS merupakan salah satu algoritma penelusuran berdasarkan kedalaman. Simpul ditelusuri dari root kemudian ke salah satu simpul anaknya (misalnya prioritas penelusuran berdasarkan anak pertama [simpul sebelah kiri], maka penelusuran akan dilakukan terus melalui simpul anak pertama dari simpul anak pertama level sebelumnya sampai akhirnya mencapai level yang paling dalam. Setelah sampai level paling dalam, penelusuran akan kembali ke 1 level sebelumnya untuk mencari simpul anak kedua dengan pohon biner [simpul sebelah kanan] lalu akan kembali ke langkah sebelumnya dengan menelusuri simpul anak pertama lagi sampai level terdalam.

*B. Breadth First Search* (BFS)

BFS merupakan sebuah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudia

mengunjungi semua simpul yang bertetangga dengan simpul tersebut lebih dahulu. Llau, simpul yang belum dikunjungi dan bertetangga dengan simpul lainnya yang telah dikunjungi, demikian seterusnya. Jika graf berbentuk sebuah pohon berakar, maka semua simpul pada aras d akan dikunjungi lebih dahulu sebelum simpul- simpul pada aras d+1.

Algoritma ini membutuhkan sebuah antrian q untuk menyimpan simpul yang telah dikunjungi. Simpul-simpul ini nantinya akan diperlukan sebagai acuan untuk mengunjungi simpul-simpul yang bertetanggan degannya. Tiap simpul yang telah dikunjungi akan masuk ke dalam antrian hanya satu kali. Algoritma ini juga akan membutuhkan table Boolean untuk menyimpan simpul yang telah dikunjungi sehingga tidak akan ada simpul yang dikunjungi lebih dari satu kali.

### *C. Adjacency List*

Adjacency list merepresentasikan sebuah graf sebagai sebuah array dari sebuah linked lists. Index dari array akan merepresentasikan sebuah vertek dan setiap elemen yang ada di dalam linked list merepresentasikan vertek lainnya yang akan membuat sebuah ujung dengan vertek.

Kelebihan dari adjacency list

- Sebuah adjacency list sangat efisien jika digunakan karena kita hanya untuk menyimpan angka untuk setiap ujung. Untuk sebuah graf yang memiliki jutaan vertek and ujung, adjacency list akan sangat membantu untuk masalah penyimpanan.
- Adjacency list juga membantu untuk mencari semua vertek adjacency kepada sebuah vertek dengan mudah.

Kekurangan dari adjacency list

- Mencari sebuah adjacency list tidak secepat mencari adjacency matrix karena semua list yang terhubung ke nodes harus ditelusuri terlebih dahulu untuk menemukannya.

## IV. IMPLEMENTASI

### *A. Depth First Search* (DFS)

Pada DFS ada langkah-langkah dalam pemrosesan kerja algoritma ini sendiri.

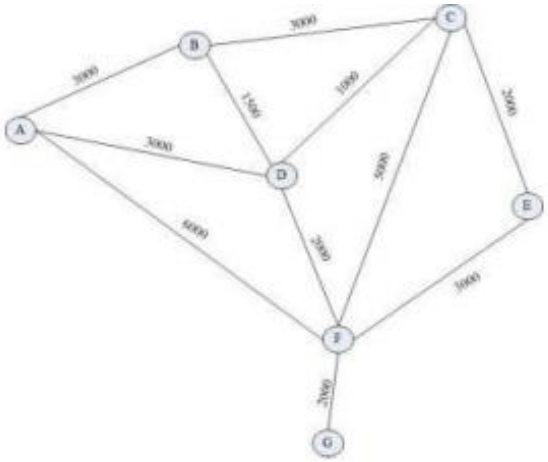
- Proses pencarian pada algoritma DFS akan dilakukan kepada semua anaknya terlebih dahulu, lalu sebelum melakukan pencarian ke node-node yang selevel baru akan dilakukannya pencarian.
- Algoritma DFS akan melakukan suatu perhitungan secara berurut dari urutan yang terakhir. Lalu ketika pengitungan selesai dan mengabiskan semua kemungkinan dari titik yang terakhir, lalu baru akan mundur ke titik-titik yang sebelumnya, dan akhirnya sampai kepada titik awal
- Lalu pencarian tersebut akan dimulai dari node akar menuju level yang lebih tinggi lagi. Proses ini akan diulangi terus menerus hingga sebuah solusi ditemukan.

### *B. Breadth First Search* (BFS)

Dalam algoritma BFS, simpul anak yang telah dikunjungi akan disimpan kedalam antrian. Antrian ini nantinya akan digunakan untuk mengacu simpul-simpul yang bertetanggaan dengan yang akan dikunjungi, lalu sesuai dengan urutan pengantrian untuk memperjelas cara kerja algoritma BFS beserta antrian yang digunakan kita harus menggunakan langkah-langkah sebagai berikut.

- Masukkan simpul ujung (akar) ke dalam antrian.
- Ambil simpul dari awal antrian, lalu cek apakah simpul tersebut merupakan sebuah solusi.
- Jika simpul tersebut merupakan sebuah solusi, maka pencarian akan selesai dan hasil yang ditemukan akan dikembalikan.
- Jika simpul tersebut bukan merupakan sebuah solusi, maka masukkan seluruh simpul yang bertetanggaan dengan simpul tersebut (simpul anak) ke dalam antrian.

Contoh dari BFS



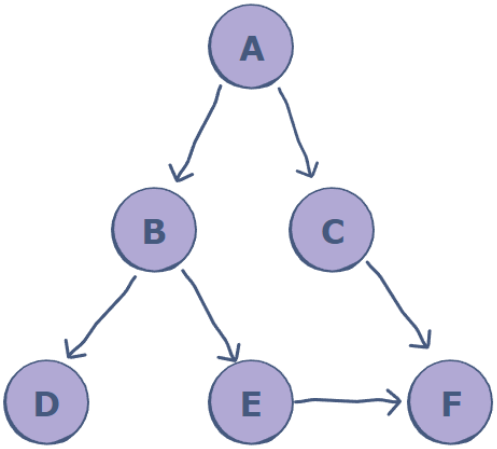

Misalnya kita akan menentukan jalur terpendek dengan BFS dari A menuju E, maka berdasarkan gambar diatas kita dapat mengimplementasikan kedalam BFS untuk memperoleh jalur terpendek dengan langkah-langkah sebagai berikut.

Kita melakukan pencarian jalur terpendek berdasarkan konsep dari BFS, tujuan dari pencarian ini sendiri adalah menemukan titik E. Pencarian akan diawali dengan menelusuri titik pangkal yaitu titik A, karena titik A merupakan titik dengan level tertinggi maka penelusuran akan dilanjutkan dengan menjelajahi titik-titik yang ada pada level di bawahnya yaitu simpul yang bertetangga dengan simpul tersebut (simpul anak). Penelusuran selanjutnya merupakan pada level kedua atau simpul yang bertetangga dengan simpul tersebut yaitu titik B, D dan F, penelusuran selanjutnya adalah bagian untuk memasukkan simpul yang bertetangga dengan B yaitu C dan D, dan dilanjutkan dengan memasyukkan simpul yang bertetangga dengan D yaitu C dan F, tahap selanjutnya akan memasukkan simpul yang bertetangga dengan F yaitu C dan E

dan G. Dalam diagram tersebut titik E merupakan titik tujuan, maka dengan demikian proses pencarian akan berhenti, karena sudah menemukan solusi, seperti yang bisa dilihat dalam gambar.

## V. KODE PROGRAM

### *A. Depth First Search* (DFS)



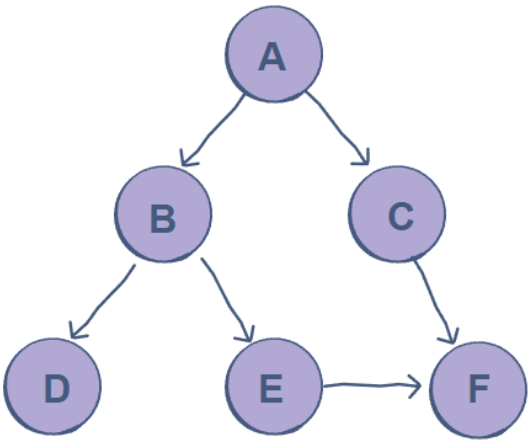

```python
# Using a Python dictionary to act as an adjacency list
graph = {
    'A' : ['B','C'],
    'B' : ['D', 'E'],
    'C' : ['F'],
    'D' : [],
    'E' : ['F'],
    'F' : []
}

visited = set() # Set to keep track of visited nodes.

def dfs(visited, graph, node):
    if node not in visited:
        print (node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

# Driver Code
dfs(visited, graph, 'A')
```

### *B. Breadth First Search* (BFS)




```python
graph = {
  'A' : ['B','C'],
  'B' : ['D', 'E'],
  'C' : ['F'],
  'D' : [],
  'E' : ['F'],
  'F' : []
}

visited = [] # List to keep track of visited nodes.
queue = []     #Initialize a queue

def bfs(visited, graph, node):
  visited.append(node)
  queue.append(node)

  while queue:
    s = queue.pop(0)
    print (s, end = " ")

    for neighbour in graph[s]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
bfs(visited, graph, 'A')
```

## VI. UJI COBA DAN HASIL

### A. *Depth First Search* (DFS)


Penjelasan:

- Pada line 2 - 9 merupakan ilustrasi graf yang direpresentasikan menggunakan adjacency list, untuk memudahkan pencarian DFS menggunakan python, dianjurkan untuk menggunakan sebuah *dictionary* data structrue. Setiap vertek akan memiliki sebuah list adjacency-nya masing-masing pada nodes yang telah disimpan.
- Pada line 11 *visited* adalah sebuah set yang akan digunakan untuk menyimpan semua peristiwa yang telah dikunjungi oleh node.
- Pada line 21 fungsi *dfs* akan dipanggil dan akan meng-oper *visited* set, graf yang telah disediakan oleh sebuah *dictionary dictionary*, dan A, yang merupakan titik mulai node.
- Pada line 13 – 18, *dfs* mengikuti algoritma yang telah di jelaskan diatas.
    1. Pertama ia yang mengecek terlebih dahulu jika node yang sedang digunakan belum dikunjungi, jika iya, maka node akan di append kedalam *visited* set.
    2. Kemudian untuk setiap tetangga dari node tersebut, fungsi *dfs* akan dipanggil lagi.
    3. Node dasar akan dipanggil lagi ketika semua node telah dikunjungi, dan akhirnya fungsi akan *return*.

### B. *Breadth First Search* (BFS)


Penjelasan:

- Pada line 3 – 10, merupakan ilustrasi graf yang direpresentasikan menggunakan adjacency list, untuk memudahkan pencarian DFS menggunakan python, dianjurkan untuk menggunakan sebuah *dictionary* data structrue. Setiap vertek akan memiliki sebuah list adjacency-nya masing-masing pada nodes yang telah disimpan.
- Pada line 12 *visited* merupakan sebuah list yang digunakan untuk mencatatat semua riwayat dari nodes yang telah dikunjungi.
- Pada line 13, *queue* merupakan sebuah list yang digunakan untuk mencatat semua riwayat dari node yang sedang berada di *queue.*
- Pada line 29, argumen dari fungsi *dfs* adalah *visited* list, graf yang telah disediakan oleh sebuah *dictionary dictionary*, dan A, yang merupakan titik mulai node.
- Pada line 15 – 26, *dfs* mengikuti algoritma yang telah dijelaskan diatas:
    1. Pertama ia akan memeriksa dan memanggil node awal menuju ke *visited* list dan *queue.*
    2. Llau, ketika semua *queue* mengandung elemen-elemen, ia akan terus mengambil node dari *queue* yang ada. Memanggil tetangga node untuk menuju *queue* jika mereka belum terpanggil, dan menandai jika mereka telah dikunjungi.
    3. Proses ini akan berulang terus menerus sampai *queue* tidak ada isinya lagi.

## KESIMPULAN

Dengan menggunakan algoritma *Depth First Search* (DFS) dan *Breadth First Search* (BFS) kita dapat menyelesaikan persoalan-persoalan yang ada, algoritma DFS menyelesaikan sebuah program dengan menyeluruh sesuai dengan urutan dari akhir ke awal dan algoritma BFS menyelesaikan sebuah program dengan cara menyebar.

Namun tiap persoalan juga memiliki penyelesaian optimalnya masing-masing, maka dapat dikatakan bahwa algoritma DFS dan BFS sama pentingnya dalam menyelesaikan sebuah persoalan.

Saya berharap dapat lebih mengembangkan ilmu saya lagi dan bisa lebih teliti lagi untuk tidak menggunakan persoalan yang simple tetapi menggunakan sebuah persoalan lebih rumit lagi yang dapat berguna unrtuk membantu permasalahan ini.

## UCAPAN TERIMA KASIH

Sekian implementasi sederhana tentang *Depth First Search* (DFS) dan *Breadth First Search* (BFS) pada python untuk tugas kali ini. Semoga dapat memberi penjelasan yang jelas seperti apa yang saya tujukan dan dapat digunakan untuk selanjutnya ataupun dapat dikembangkan lebih baik lagi kedepannya, bila ada kesalah kata saya mohon maaf. Sekian Terimakasih.

## REFERENCES

[1] Adjacency list (With code in C, C++, Java and Python). (n.d.). Programiz: Learn to Code for Free. https://www.programiz.com/dsa/graph-adjacency-list

[2] Edpresso Team. (2019, June 24). How to implement depth-first search in Python. Educative: Interactive Courses for Software Developers. https://www.educative.io/edpresso/how-to-implement-depth-first-search-in-python.

[3] Edpresso Team. (2019, November 8). How to implement a breadth-first search in Python. Educative: Interactive Courses for Software Developers. https://www.educative.io/edpresso/how-to-implement-a-breadth-first-search-in-python.

[4] Gustiana, R. (2020, November 2). Implementasi depth first search (DFS) Dan breadth first search (BFS) pada Python. Medium. https://rinagustiana.medium.com/implementasi-depth-first-search-dfs-dan-breadth-first-search-bfs-pada-python-62afe680925d.

[5] (n.d.). Portal E-Jurnal Universitas Kristen Immanuel Yogyakarta. https://www.e-jurnal.ukrimuniversity.ac.id/file/5Jurnal-DINO-InFact-V1N2NOV2016.pdf.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandar Lampung, 28 Maret 2021

Nama dan NIM