# Penerapan Algoritma *Breadth First Search* pada Penentuan Solusi Jalan Keluar Labirin

Adelline Kania Setiyawan - 13520084
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520084@std.stei.itb.ac.id

***Abstract*—Algoritma *Breadth First Search* merupakan salah satu algoritma pencarian jalur solusi dengan pembetukan dan penelusuran pohon yang sering digunakan. Salah satu implementasi dari algoritma BFS ini adalah penentuan solusi jalan keluar dari suatu labirin, yaitu dengan merepresentasikan jalur labirin tersebut menjadi graf tidak berarah kemudian menentukan jalur solusi berdasarkan graf yang telah terbentuk.**

***Kata Kunci—Breadth First Search; BFS; graf tidak berarah; labirin; jalur; simpul; sisi***

## I. PENDAHULUAN

Algoritma adalah langkah-langkah atau prosedur untuk menyelesaikan suatu permasalahan secara komputasional tertentu yang kemudian dapat diimplementasikan dalam suatu program. Algoritma harus dibuat dan disusun secara logis dan sistematis. Algoritma dapat menerima suatu input tertentu kemudian memberikan output ataupun mengubah perilaku suatu objek. Algoritma yang baik adalah algoritma yang dapat menyelesaikan permasalahan secara keseluruhan untuk setiap data uji yang diberikan dengan waktu komputasi yang efisien dan efektif.

Salah satu algoritma yang secara umum dan sering digunakan adalah algoritma *Breadth First Search* atau BFS yang bertujuan untuk menentukan jalur dari suatu posisi awal ke posisi akhir. Algoritma *Breadth First Search* memanfaatkan konsep pohon yang memiliki simpul-simpul yang menghubungkan dengan simpul lainnya. Algoritma *Breadth First Search* merupakan implementasi dari teknik algoritma traversal graf secara melebar, yaitu menelusuri simpul-simpul suatu pohon secara melebar dari simpul awal untuk menentukan jalur ke posisi akhir yang diinginkan.

Implementasi dari algoritma *Breadth First Search* dalam kehidupan sehari-hari sangat banyak, seperti melakukan *indexing* halaman website untuk pencarian atau *search engines* di internet, sistem navigasi GPS, penentuan nilai dan jalur minimum dari *spanning tree*, *peer to peer network*, hingga penentuan solusi jalan keluar dari suatu labirin. Penentuan solusi jalan keluar dari suatu labirin dengan menggunakan algoritma *Breadth First Search* akan dilakukan dengan merepresentasikan terlebih dahulu jalur-jalur dari labirin menjadi suatu graf tidak berarah kemudian mengimplementasikan algoritma BFS tersebut pada graf sehingga didapatkan jalur solusi dari suatu simpul awal menuju simpul solusi akhir.

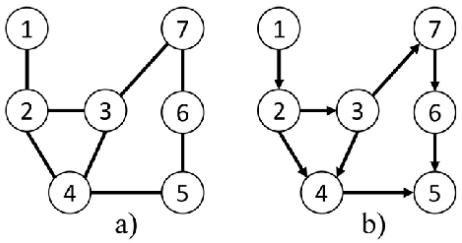
## II. LANDASAN TEORI

### A. *Graf*

Graf adalah suatu struktur yang merepresentasikan hubungan antara simpul-simpul *(vertices)* dan sisi-sisi *(edges)* yang menghubungkan sepasang simpul dari simpul-simpul yang ada. Graf pada umumnya digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut sebagai simpul dan sisi.

Terdapat beberapa jenis graf, yaitu graf sederhana dan graf tidak sederhana. Pada graf sederhana atau *simple graph*, graf tidak memiliki gelang maupun sisi ganda. Sedangkan pada graf tidak sederhana atau *unsimple graph*, graf memiliki gelang atau sisi ganda. Graf juga dapat dibedakan berdasarkan arah orientasinya, yaitu:

1. Graf tidak berarah, yaitu graf yang sisinya tidak memiliki orientasi arah
2. Graf berarah, yaitu graf yang sisinya memiliki orientasi arah terhadap suatu simpul




Gambar 2.1.1 (a) graf tidak berarah (b) graf berarah
*Sumber: https://www.researchgate.net/figure/a-Undirected-graph-and-b-directed-graph_fig4_337354946*

Terdapat beberapa terminologi graf yang secara umum harus diketahui ketika ingin melakukan pemodelan permasalahan dalam graf, yaitu sebagai berikut:

1. Ketetanggaan atau *adjacent*, yaitu dua buah simpul pada suatu graf dikatan bertetangga apabila terdapat sisi yang secara langsung menghubungan kedua buah simpul tersebut;

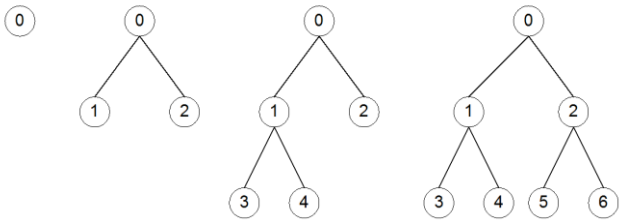
2. Derajat atau *degree*, yaitu jumlah sisi yang berhubungan dengan suatu simpul, biasanya dinotasikan dalam d(v);
3. Lintasan atau *path*, yaitu panjang atau jumlah sisi yang harus dilalui dari suatu simpul awal menuju simpul tujuan
4. Graf kosong atau *null graph*, yaitu graf yang tidak memiliki sisi apapun sehingga tidak ada penghubung antarsimpulnya;

### B. *Algoritma Breadth First Search*

Algoritma *Breadth First Search* atau BFS adalah salah satu algoritma yang umum dan sering digunakan untuk melakukan pencarian jalur. Algoritma *Breadth First Search* merupakan penerapan dari teknik algoritma traversal graf secara melebar. Algoritma *Breadth First Search* akan melakukan pencarian jalur dengan menelusuri simpul-simpul suatu pohon dari simpul utama hingga ditemukan jalur ke simpul yang menjadi tujuan pencarian.

Algoritma *Breadth First Search* akan menggunakan graf yang direpresentasikan sebagai graf statis, yaitu graf yang sudah terbentuk sebelum proses pencarian dilakukan. Graf ini akan direpresentasikan sebagai suatu struktur data. Pada umumnya, struktur data yang digunakan adalah graf yang memiliki simpul-simpul saling bertetangga ataupun matriks *adjacency*, yaitu matriks $n \times n$ yang merepresentasikan hubungan setiap simpul dengan simpul lainnya dalam tipe data *boolean*. Selain itu, struktur data yang dapat digunakan adalah struktur data *map* yang memetakan suatu simpul ke list dari simpul-simpul yang merupakan simpul tetangganya.

Algoritma *Breadth First Search* akan membangun solusi permasalahan dengan melakukan pembangkitan status-status dalam bentuk simpul pohon dengan cara mengaplikasikan operator atau langkah legal kepada status simpul lainnya pada suatu jalur. Jalur dari simpul akar menuju simpul daun atau *goal* akan berisi rangkaian operator yang mengarah pada suatu solusi persoalan. Solusi tersebut dapat bernilai benar atau salah terhadap solusi sebenarnya.




Gambar 2.1.1 Pembangkitan ruang status secara melebar

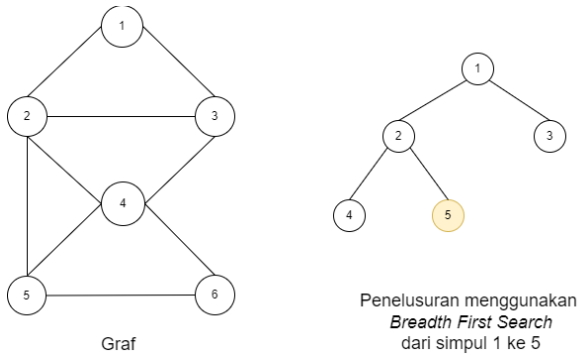
*Sumber:*
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf

Pembangkitan ruang status pada algoritma BFS akan dilakukan secara melebar, yaitu dengan menginisalisasi terlebih dahulu simpul yang merupakan status awal sebagai akar, kemudian menambahkan simpul-simpul anaknya. Semua simpul pada level ke-n akan dibangkitkan terlebih dahulu sebelum pembangkitan simpul-simpul pada level $n + 1$.

Secara umum, algoritma untuk melakukan pencarian suatu solusi dengan penelusuran ruang status menggunakan algoritma *Breadth First Search* adalah sebagai berikut:

1. Menentukan suatu simpul untuk menjadi akar sebagai titik awal dari pencarian
2. Masukkan simpul tersebut dalam suatu *array* yang menyimpan simpul-simpul yang belum ditelusuri
3. Ambil simpul pertama dari *array* yang menyimpan simpul-simpul yang belum ditelusuri
4. Simpan simpul tersebut ke dalam *array* yang menyimpan simpul-simpul yang sudah ditelusuri
5. Telusuri semua kemungkinan simpul tetangga tersebut berdasarkan graf atau matriks *adjacency* dan pastikan bahwa simpul tetangga tersebut bukan merupakan simpul yang telah ditelusuri atau simpul tetangga tersebut tidak berada dalam *array* yang menyimpan simpul-simpul yang sudah ditelusuri
6. Periksa apakah simpul tersebut merupakan solusi yang ingin dicari
7. Jika simpul merupakan solusi, maka pencarian akan selesai, jika tidak, ulangi langkah nomor 2

Pencarian solusi dengan pembangkitan ruang status pada algoritma BFS ini dapat dilakukan dengan dua cara, yaitu melakukan pembangkitan ruang status hingga hasil solusi pencarian pertama dilakukan atau melakukan pembangkitan ruang status hingga semua kemungkinan solusi telah dilakukan. Apabila melakukan pembangkitan solusi hingga hasil solusi pencarian pertama saja, berarti solusi yang didapatkan dari permasalahan hanyalah satu, sedangkan apabila pembangkitan ruang status terus dilakukan hingga semua status ditelusuri, akan menghasilkan semua alternatif solusi dalam suatu permasalahan. Namun, apabila melakukan penelusuran untuk semua alternatif solusi, waktu yang diperlukan untuk melakukan penelusuran akan cukup lama, terlebih lagi untuk ukuran jumlah simpul dari graf yang cukup besar.




Gambar 2.1.2 Contoh penelusuran ruang status secara melebar menggunakan algoritma Breadth First Search

Algoritma *Breadth First Search* adalah algoritma pencarian yang memanfaatkan traversal graf dengan pembangkitan pohon ruang status secara *blind* atau buta. Disebut demikian karena

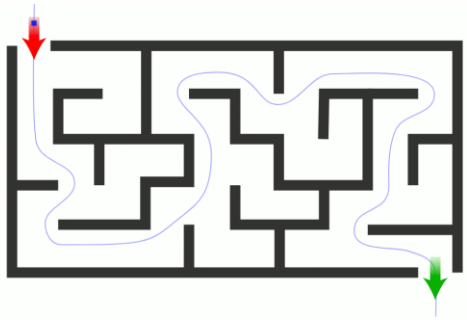
algoritma *Breadth First Search* melakukan pencarian tanpa mempedulikan *cost* atau harga setiap simpul yang ditelusuri. Walaupun *cost* simpul yang ditelusuri tersebut memiliki *cost* tertinggi atau terendah, algoritma *Breadth First Search* akan tetap melakukan penelusuran simpul secara melebar, seperti yang telah dijelaskan sebelumnya.

Walaupun begitu, algoritma *Breadth First Search* adalah algoritma yang *complete* karena algoritma BFS ini akan selalu dapat menghasilkan solusi pencarian jika memang terdapat solusi pencarian yang diinginkan. Solusi pencarian yang akan dikembalikan oleh algoritma *Breadth First Search* adalah solusi pencarian yang paling dekat dengan simpul akarnya. Hal ini disebabkan oleh traversal yang dilakukan dengan pembangkitan simpul secara melebar. Apabila permasalahan yang diselesaikan oleh algoritma BFS ini bukanlah permasalahan dengan simpul yang memiliki *cost* yang berbeda, atau dalam kata lain setiap simpul dibangkitkan memiliki *cost* yang sama, maka algoritma BFS akan mengembalikan solusi yang terbaik.

Kompleksitas dalam algoritma *Breadth First Search* dalam *big-o notation* adalah $O(b^d)$, dimana $b$ merupakan *branching factor* atau jumlah maksimum percabangan yang mungkin dari suatu simpul dan $d$ merupakan *depth* besar kedalaman dari solusi terbaik atau solusi yang ditemukan pada level terendah. Selain kompleksitas komputasional, algoritma *Breadth First Search* juga memiliki kompleksitas ruang karena memerlukan *array* untuk menampung simpul-simpul yang sudah ditelusuri atau belum. Sehingga kompleksitas ruang algoritma BFS ini akan sama dnegan kompleksitas algoritmanya, yaitu $O(b^d)$. Karena jumlah kompleksitas ruang yang cukup besar, algoritma BFS ini terkadang menjadi kurang baik karena kompleksitas ruang tersebut.

### *C. Labirin*

Labirin atau biasa disebut sebagai *maze*, menurut Kamus Besar Bahasa Indonesia adalah tempat yang penuh dengan jalan dan lorong yang berliku-liku dan simpang siur; sesuatu yang sangat rumit dan berbelit-belit (tentang susunan, aturan, dan sebagainya); sistem rongga atau saluran yang berhubungan.



Gambar 2.2.1 Ilustrasi labirin

*Sumber:*
*https://commons.wikimedia.org/wiki/File:Maze_simple_-_path_animated.gif*

Labirin sering digunakan menjadi sebuah permainan. Permainan tersebut mengharuskan pemainnya menentukan jalan keluar dari suatu titik dimulainya permainan menuju salah satu jalan keluar yang ada. Permainan ini mengharuskan pemain untuk menentukan jalan keluar melalui banyak kemungkinan jalur yang ada. Suatu jalur yang dipilih belum tentu akan mengarahkan pada jalur keluar yang benar. Suatu jalur juga dapat mengarahkan ke jalan buntu. Pada umumnya, apabila sudah dipilih suatu jalur dalam labirin, pemain dapat melakukan jalan mudur atau *backtracking* ke jalur sebelumnya karena jalur tersebut memberikan jalan buntu.

Sejauh ini, belum ditemukan cara yang paling optimal untuk menentukan jalan keluar dari permainan labirin. Sehingga pemain pada umumnya akan melakukan pencarian semua jalur yang dapat mengarahkan ke jalan keluar yang dinginkan. Terdapat teknik yang umum digunakan untuk menyelesaikan permainan labirin, yaitu dengan secara berurut menelusuri jalur yang berada di labirin dengan berpatokan pada salah satu dinding. Sehingga, semua jalur, baik jalur yang mengarah ke arah buntu atau ke arah yang benar, ditelusuri dengan penelusuran tidak lebih dari sekali. Namun, cara ini tentunya akan sangat lama jika semakin terbanyak liku pada labirin. Sehingga diperlukan alternatif cara lain untuk keluar dari labirin, yaitu dengan menggunakan pencarian jalan keluar dengan algoritma *Breadth First Search.*

## III. ANALISIS PENERAPAN ALGORITMA *BREADTH FIRST SEARCH* PADA PENENTUAN SOLUSI JALAN KELUAR LABIRIN

Persoalan penentuan solusi jalan keluar suatu labirin dapat diselesaikan dengan penerapan algoritma *Breadth First Search*. Seperti yang sudah dijelaskan sebelumnya, untuk menentukan solusi jalan keluar suatu labirin dapat dengan secara berurut menelusuri jalur yang berada di labirin dengan berpatokan pada salah satu dinding. Akan tetapi, cara tersebut tergolong penyelesaian masalah secara brute force karena mengharuskan untuk melewati semua jalur di labirin yang ada, baik jalur yang mengarah ke arah buntu maupun ke arah yang benar.

Walaupun strategi brute force tersebut cukup baik apabila diimplementasikan, karena dipastikan menemukan solusi dan lingkup pencarian tidak berulang, namun, strategi ini akan menjadi sangat lama apabila terdapat banyak liku atau sisi dari setiap simpul yang ada. Sehingga implementasi algoritma *Breadth First Search* dapat menjadi alternatif dalam menyelesaikan persoalan penentuan solusi jalan keluar dari labirin ini. Hal ini disebabkan oleh kompleksitas algoritma *Breadth First Search* yang lebih kecil daripada strategi brute force dan pada kasus labirin ini, jalan keluar dari suatu labirin dipastikan ada.

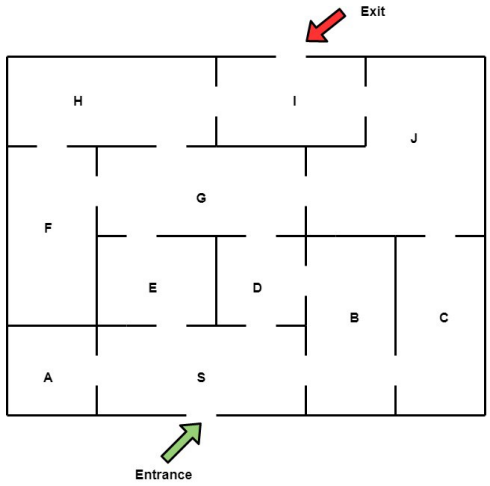
Untuk menentukan solusi jalan keluar dari suatu labirin dengan menggunakan penerapan algoritma *Breadth First Search*, terdapat beberapa langkah-langkah yang perlu dilakukan

### *A. Pemetaan Jalur Labirin Menjadi Graf*

Sebelum dapat melakukan penerapan algoritma *Breadth First Search* untuk menentukan solusi jalan keluar dari labirin, jalur-jalur dari suatu labirin harus dipetakan terlebih dahulu menjadi graf. Hal ini bertujuan untuk mempermudah pembangkitan simpul-simpul dari graf-graf yang saling bertetangga. Pada kasus labirin ini, graf yang digunakan adalah

graf tidak berarah karena pada labirin standar, apabila jalur a dapat dilewati melalui jalur b, maka jalur b dapat dilewati oleh jalur a (hubungan bolak-balik). Berikut ini adalah contoh pemetaan suatu jalur labirin menjadi suatu graf tidak berarah.

Misalkan, terdapat jalur labirin yang akan dipetakan menjadi graf tidak berarah sebagai berikut:

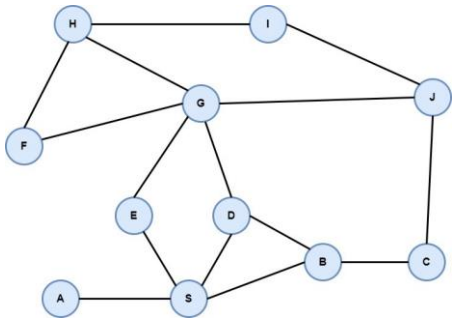



Gambar 3.1.1 Jalur labirin awal
*Sumber: https://python.plainenglish.io/solve-maze-using-breadth-first-search-bfs-algorithm-in-python-7931acbe8a93*

Untuk memetakan jalur labirin tersebut menjadi suatu graf tidak berarah adalah:

1. Memetakan setiap ruangan yang ada, yaitu S, A, B, C, D, E, F, G, H, I, J menjadi suatu simpul tunggal
2. Menghubungkan sisi-sisi setiap simpul yang ada menyesuaikan dengan ruang-ruang pada labirin yang secara langsung bersebelahan dan memiliki penghubung langsung antara kedua ruangan. Misalnya, ruangan S pada labirin bersebelahan langsung atau bertetangga langsung dengan ruang A, D, E, dan B sehingga pada graf terdapat sisi-sisi yang menghubungkan antara simpul A dengan simpul A, D, E, dan B. Atau contoh lainnya adalah ruangan D pada labirin bertetangga langsung dengan ruangan S, B, E, dan G, namun tidak terdapat penghubung langsung antara D dan E, sehingga pada graf, pada simpul D hanya akan terdapat sisi-sisi yang menghubungkan dengan simpul S, B, dan G.

Hasil pemetaan yang didapatkan dari labirin tersebut menjadi suatu graf tidak berarah adalah sebagai berikut:




Gambar 3.1.1 Graf tidak berarah yang didapat
*Sumber: https://python.plainenglish.io/solve-maze-using-breadth-first-search-bfs-algorithm-in-python-7931acbe8a93*

Setelah memetakan jalur labirin menjadi suatu graf tidak berarah, dalam program, graf tersebut dapat direpresentasikan menjadi suatu kelas Graf yang memiliki atribut berupa *dictionary* yang memiliki *keyword* berupa identitas suatu simpul yang menghasilkan suatu list yang merupakan tetangga-tetangga dari simpul tersebut. *Dictionary* digunakan untuk memudahkan dan mempercepat akses terhadap informasi ketetanggaan suatu simpul. Dari graf tersebut akan didapatkan *dictionary* list sebagai berikut:

```
[
   A: [S],
   B: [C, D, S],
   C: [B, J],
   D: [B, G, S],
   E: [G, S],
   F: [G, H],
   G: [D, E, F, H, J],
   H: [F, G, I],
   I: [H, J],
   J: [C, G, I],
   S: [A, B, D, E]
]
```

Kelas graf pada program dalam bahasa Python adalah sebagai berikut:

```
from collections import defaultdict
class Graph:
     # Konstruktor
     def __init__(self):
          # Dictionary yang digunakan untuk
          # menyimpan ketetangaan suatu simpul
          self.graph = defaultdict(list)
     # Fungsi untuk menambahkan sisi suatu
     # simpul
```

```
    def addEdge(self,u,v):
        self.graph[u].append(v)
```

Menambahkan sisi-sisi simpul pada program utama dapat dilakukan dengan:

```
g = Graph()
g.addEdge("A", "S")
g.addEdge("S", "A")
g.addEdge("B", "C")
g.addEdge("C", "B")
# dan seterusnya…
```

*B. Implementasi Algoritma Breadth First Search pada Graf*

Setelah melakukan pemetaan dari jalur labirin menjadi suatu graf, hal yang dapat dilakukan selanjutnya adalah implementasi algoritma *Breadth First Search* ini pada graf tersebut. Untuk mengimplementasikannya, akan dibuat suatu prosedur di dalam kelas Graf yang dikhususkan untuk melakukan pencarian secara *Breadth First Search* dengan pembangkitan simpul secara melebar dari atribut yang dimiliki kelas Graf tersebut. Prosedur ini akan memberikan output berupa jalur labirin yang ditemukan berdasarkan input titik awal simpul atau akar dan input jalan keluar atau solusi.

Secara umum, untuk menentukan jalur solusi dari labirin yang telah dimodelkan sebagai graf dengan menggunakan algoritma BFS ini adalah sebagai berikut:

1. Meminta input berupa titik awal simpul atau akar dan simpul jalan keluar atau solusi dari graf
2. Menandakan semua simpul yang terdapat pada graf sebagai simpul yang belum dikunjungi kemudian membuat sebuah *array* yang menyimpan nilai-nilai tersebut
3. Membuat sebuah antrian simpul-simpul yang telah diekspansi namun belum dilakukan penelusuran terhadap simpul-simpul tersebut
4. Menambahkan simpul akar ke dalam antrian dan menandai simpul akar sebagai simpul yang telah ditelusuri
5. Mengambil simpul pertama dari antrian yang menyimpan simpul-simpul yang belum ditelusuri untuk dilakukan penelusuran
6. Telusuri semua kemungkinan simpul tetangga tersebut berdasarkan kelas graf yang telah dibuat dan pastikan bahwa simpul tetangga yang akan ditelusuri tersebut bukan merupakan simpul yang telah ditelusuri atau simpul tetangga tersebut tidak berada dalam antrian yang menyimpan simpul-simpul yang sudah ditelusuri
7. Masukkan semua simpul tetangga yang valid ke dalam antrian
8. Periksa apakah simpul tetangga yang sekarang diperiksa merupakan solusi yang ingin dicari
9. Jika simpul merupakan solusi, maka pencarian akan selesai, jika tidak, ulangi langkah nomor 4

Penerapan algoritma *Breadth First Search* pada kelas Graf yang telah dibuat secara umum adalah sebagai berikut:

```
def BFS(self, s):
  # Menandakan semua simpul yang ada sebagai
  # simpul yang belum ditelusuri
  visited = [False] * (max(self.graph) + 1)

  # Membuat antrian simpul-simpul yang belum
  # ditelusuri tetapi sudah pernah dikunjungi
   queue = []

  # Menandakan simpul akar sudah dikunjungi
  queue.append(s)
  visited[s] = True

  while queue:
  # Mengambil simpul pertama dari antrian
    s = queue.pop(0)

    # Menelusuri tetangga dari simpul pertama
    # antrian
    for i in self.graph[s]:
      if visited[i] == False:
        queue.append(i)
        visited[i] = True
```

Implementasi algoritma *Breadth First Search* pada persoalan labirin ini dipastikan dapat memberikan solusi jalur dari suatu titik awal menuju titik akhir apabila memang solusi tersebut terdapat pada graf. Seperti yang sudah dijelaskan sebelumnya, algoritma *Breadth First Search* ini akan memiliki kompleksitas $O(b^d)$, dimana $b$ merupakan *branching factor* atau jumlah maksimum percabangan yang mungkin dari suatu simpul dan $d$ merupakan *depth* besar kedalaman dari solusi terbaik atau solusi yang ditemukan pada level terendah.

Pada potongan kode di atas, terlihat terdapat struktur data tambahan yang digunakan untuk membantu pencarian solusi menggunakan algoritma *Breadth First Search* ini, yaitu struktur data *queue* yang menyimpan antrian simpul yang belum

ditelusuri tetapi sudah pernah dikunjungi dan struktur data list *visited* yang menyimpan nilai simpul apakah sudah ditelusuri atau belum. Panjang list *visited* adalah sesuai dengan jumlah simpul yang ada. Sedangkan, panjang antrian terpanjang adalah jumlah kemungkinan simpul tetangga dari setiap simpul yang tersedia. Atau, jika dimodelkan dalam *big-o notation*, panjang antriannya adalah sama dengan kompleksitas algoritmanya, yaitu $O(b^d)$

## IV. KESIMPULAN

Penentuan solusi jalur keluar labirin dapat diselesaikan dengan menggunakan algoritma *Breadth First Search*. Jalur-jalur labirin akan dimodelkan terlebih dahulu dalam suatu graf tidak berarah, kemudian akan dilakukan pencarian solusi dengan melakukan penelusuran simpul-simpul secara melebar dari graf yang telah dibuat hingga ditemukan solusi yang dicari atau semua kemungkinan telah ditelusuri. Penggunaan algoritma *Breadth First Search* akan menghasilkan solusi yang paling dekat dengan simpul awal karena *cost* penelusuran setiap simpul pada labirin ini memiliki *cost* yang sama.

## V. UCAPAN TERIMA KASIH

Terima kasih kepada Allah SWT, atas rahmat dan hidayatnya, saya bisa menyelesaikan penulisan makalah ini. Terima kasih juga kepada Pak Rinaldi yang telah membimbing, mengajar, dan membantu saya dalam perkuliahan Strategi Algoritma ini. Yang terakhir, pastinya terima kasih kepada keluarga dan teman-teman yang selalu mendukung saya selama ini.

## REFERENSI

[1] Munir, Rinaldi. 2021. *Graf (Bag.1)*. [*Online*] Tersedia dalam https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf. Diakses pada tanggal 16 Mei 2022.

[2] Munir, Rinaldi. 2021. *Breadt/Depth First Search (BFS/DFS) Ba*. [*Online*] Tersedia dalam https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf. Diakses pada 14 Mei 2022.

[3] Soularidis, Andreas. *Solve Maze Using Breadth-First Search (BFS) Algorithm in* Python. https://python.plainenglish.io/solve-maze-using-breadth-first-search-bfs-algorithm-in-python-7931acbe8a93. Diakses pada 16 Mei 2022.

[4] GeeksForGeeks. *Breadth First Search or BFS for a Graph*. https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/. Diakses pada 16 Mei 2022.

## LINK VIDEO YOUTUBE

https://youtu.be/fiXIxpoQTSA

## 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.

Bandung, 22 Mei 2022

Adelline Kania Setiyawan
13520084