

“Maba Problem”

Find the Shortest Path using BFS Algorithm

Members

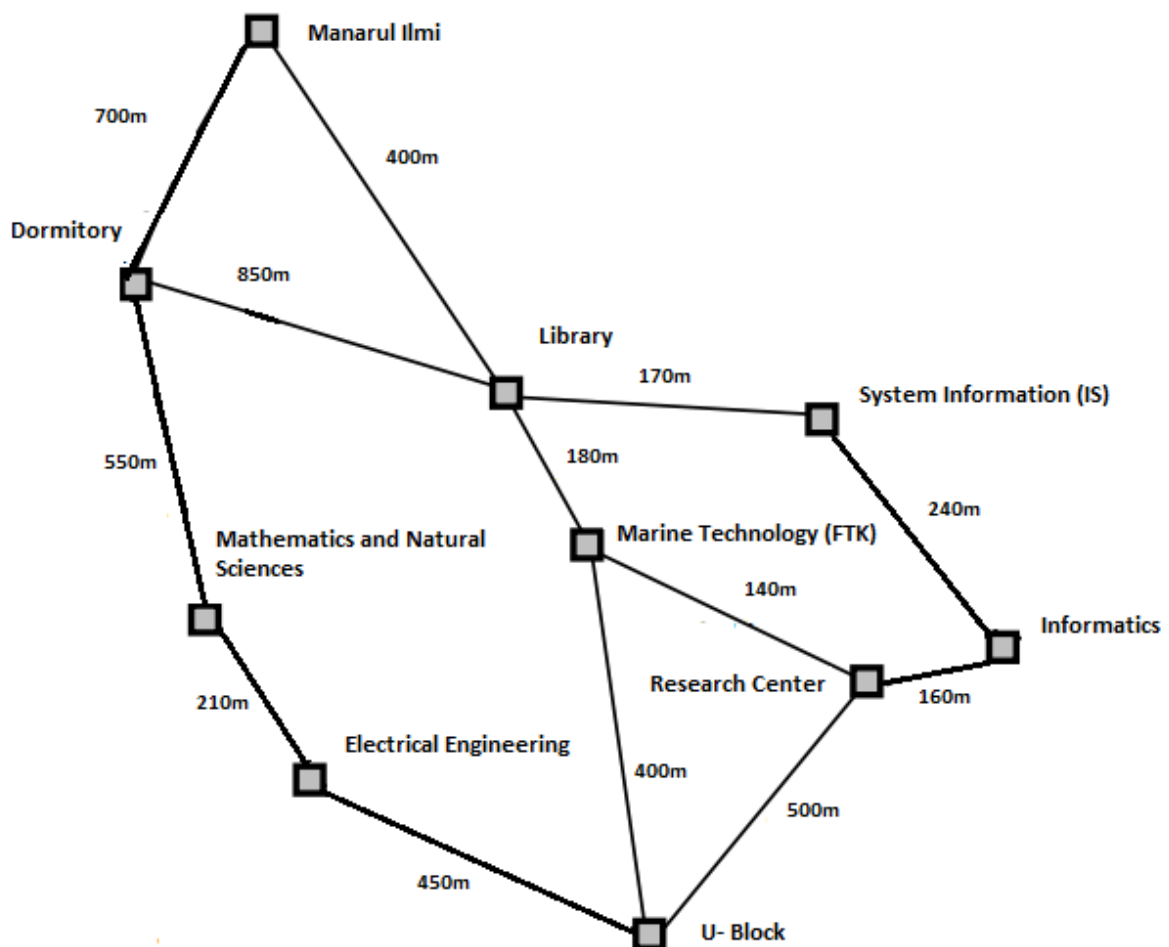
- 05111940000026 - Fais Rafli Akbar Hidya
- 05111940000159 - Salman Damai Alfariq
- 05111940000165 - Rihan Farih Bunyamin

Problem:

Elshe is a new student of its 2020 informatics. because she had never previously studied offline at her campus, she was not familiar with the existing class schedule. One day she woke up late because she slept too late the night before

Therefore, help Elshe to choose the fastest path from where she lives (dormitory) to her class (informatics) from the map provided so that she is not late for her class.

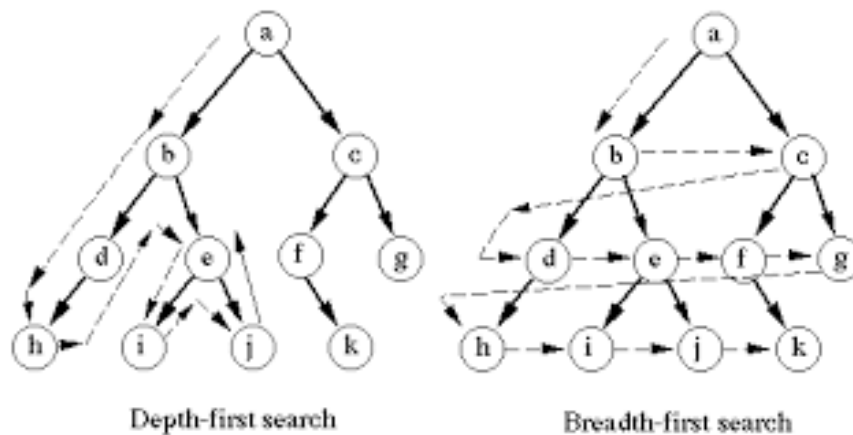
here is a map of Elshe's travel route from the Dormitory to her class(informatics) :



Source Code and Analysis

We are using BFS Algorithm to solve this. Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

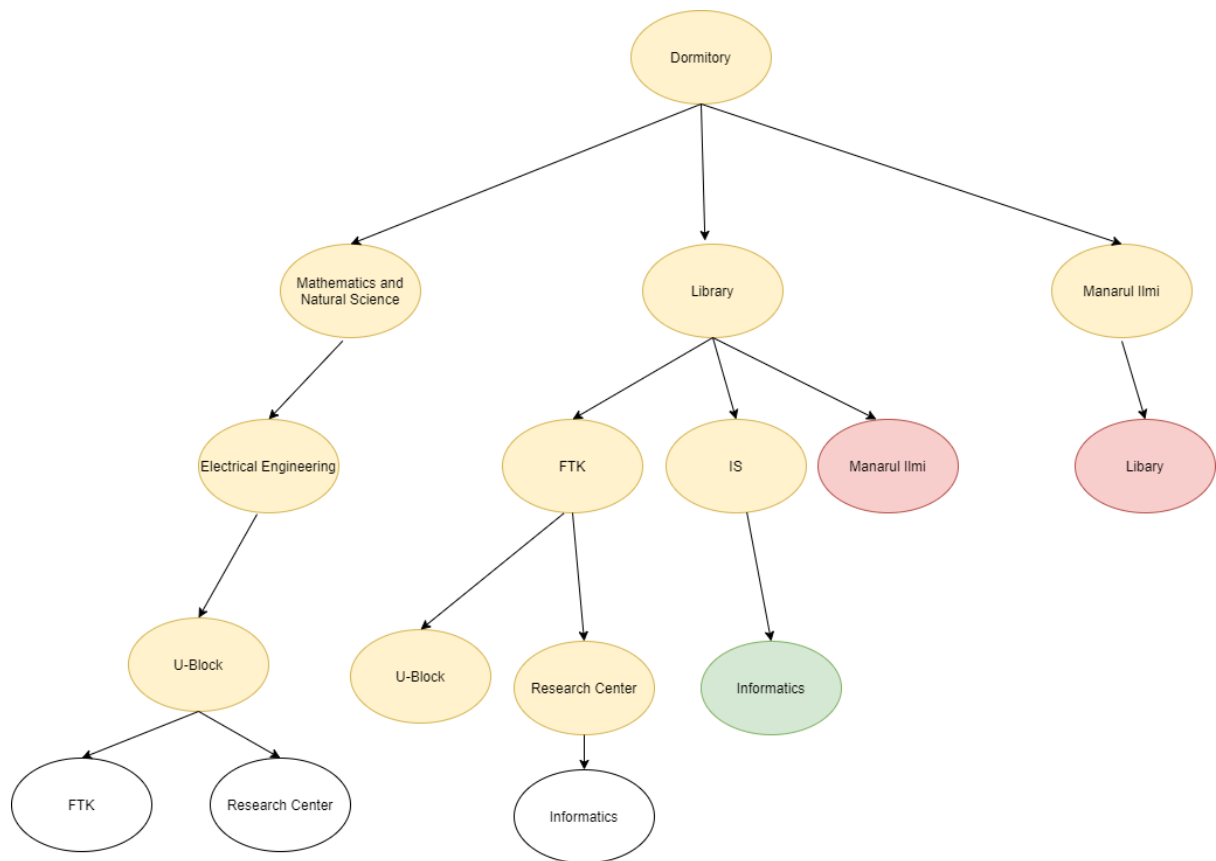
It uses the opposite strategy as depth-first search, which instead explores the node branch as far as possible before being forced to backtrack and expand other nodes. For example, depth n , before visit nodes in depth $n+1$. BFS node-visiting process can be visualized as:



Suppose the process starts from node a. Then, it will respectively visit b, c, d, e, f, g, h, i, j, and k. BFS can be defined as follows:

```
1  procedure BFS(G, root) is
2    let Q be a queue
3    label root as discovered
4    Q.enqueue(root)
5    while Q is not empty do
6      v := Q.dequeue()
7      if v is the goal then
8        return v
9      for all edges from v to w in G.adjacentEdges(v) do
10       if w is not labeled as discovered then
11         label w as discovered
12         Q.enqueue(w)
```

Node Path



Breadth-First Search algorithm applied on MabaProblem map. Initial node is Dormitory and target is Informatics.

Yellow node means city is visited.

Whites are only explored because the shortest path has been found.

Reds are dead ends because they are already visited before.

Note that even we explore Informatics priorly, we do not set it as goal state and terminate the search. Goal state must be visited.

Clearly we can see that the shortest path to Informatics is through Library - Information System - And Goals Informatics

Source Code & Explanation

First, initiate the graph.

```
6 graph = {
7     'Dormnitory': ['Library', 'Manarul', 'mathematics'],
8     'Library': ['Manarul', 'IS', 'ftk', 'Dormnitory'],
9     'mathematics': ['Dormnitory', 'electrical'],
10    'Manarul': ['Dormnitory', 'Library'],
11    'IS': ['Library', 'Informatics'],
12    'electrical': ['mathematics', 'bloku'],
13    'bloku': ['Rescen', 'ftk', 'electrical'],
14    'ftk': ['bloku', 'Library', 'Rescen'],
15    'Rescen': ['bloku', 'ftk', 'Informatics'],
16    'Informatics': ['IS', 'Rescen'],
17 }
18 # IS = information System , manarul = Manarul Ilmi , electrical = electrical engineering , FTK = Naval Technology
19 # bloku = U-block , rescen = Research Center
20
```

For the BFS algorithm, first we initiate a queue to store the current node we were traversed. While the queue is not empty, we travers all the node untill we reach the goal. There is a list named path to store the current path that we've been traversing, and there is a string variable named node to store the last node we traversed. If node is the same with our goal, then we've found the shortest path then return the path value. If node is not the goal, then we will travers all it's child and then check it again if one of the children is the goal we we want.

```
21 def bfs(graph, start, end):
22     # maintain a queue of paths
23     queue = []
24     # push the first path into the queue
25     queue.append([start])
26     while queue:
27         # get the first path from the queue
28         path = queue.pop(0)
29         # get the last node from the path
30         node = path[-1]
31         # path found
32         if node == end:
33             return path
34         # enumerate all adjacent nodes, construct a new path and push it into the queue
35         for adjacent in graph.get(node, []):
36             new_path = list(path)
37             new_path.append(adjacent)
38             queue.append(new_path)
39
40
41
```

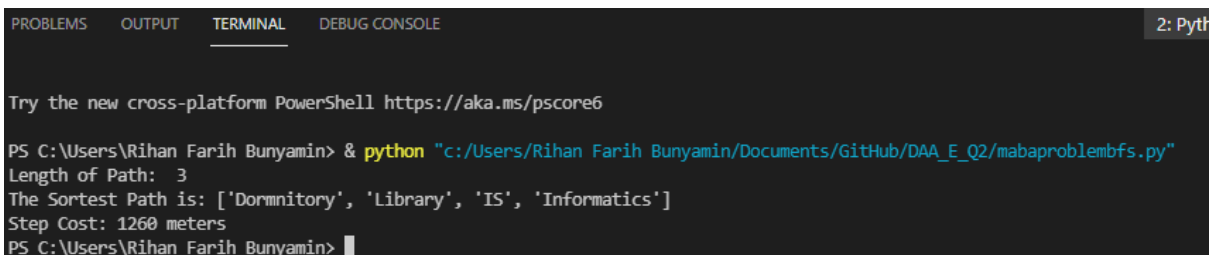
After we got the shortest path we need, we calculated the cost from the starting point to the destination. The distance variable below stored the complete graph also its cost. We provide cost_km variable to store the cost we need. In order to get the cost, we traverse once more the path that we've got earlier while adding the cost from the current node to the next node to cost_km variable. Last, when we finished counting up the cost, return cost_km value to print it.

```

42 def cost(path):
43     distance = {
44         'Dormnitory': ['Library', 'Manarul', 'mathematics', 850, 700, 400],
45         'Library': ['Manarul', 'IS', 'ftk', 'Dormnitory', 400, 170, 180, 850],
46         'mathematics': ['Dormnitory', 'electrical', 550, 210],
47         'Manarul': ['Dormnitory', 'Library', 700, 400],
48         'IS': ['Library', 'Informatics', 170, 240],
49         'electrical': ['mathematics', 'bloku', 210, 450],
50         'bloku': ['Rescen', 'ftk', 'electrical', 500, 400, 450],
51         'ftk': ['bloku', 'Library', 'Rescen', 400, 180, 140],
52         'Rescen': ['bloku', 'ftk', 'Informatics', 500, 140, 160],
53         'Informatics': ['infosys', 'Rescen', 240, 160],
54     }
55
56     cost_km = 0
57
58     for city in range(len(path)-1):
59         next_city_idx = distance[path[city]].index(path[city+1])
60         x = len(distance[path[city]])
61         cost_km += distance[path[city]][int(x/2) + next_city_idx]
62
63     return cost_km
64
65 shortest_path = bfs(graph, 'Dormnitory', 'Informatics')
66 step_cost = cost(shortest_path)
67
68 print ("Length of Path: ", (len(shortest_path)-1))
69 print("The Sortest Path is:",shortest_path)
70 print("Step Cost:" ,step_cost,"meters")
71

```

Output:



```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
2: Pyth

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Rihan Farih Bunyamin> & python "c:/Users/Rihan Farih Bunyamin/Documents/GitHub/DAA_E_Q2/mabaproblembfs.py"
Length of Path:  3
The Sortest Path is: ['Dormnitory', 'Library', 'IS', 'Informatics']
Step Cost: 1260 meters
PS C:\Users\Rihan Farih Bunyamin>

```

note:

although the score is based on contributions on github, in this project we worked on it together and weren't too rigid about commits on github (commits are generally represented by one person). therefore we hope the scores are distributed fairly among the three people. (33.33% each)

References

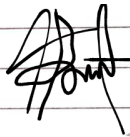
https://en.wikipedia.org/wiki/Breadth-first_search

[Penentuan Rute \(Route/Path Planning\) - Bagian 1: BFS, DFS, UCS, Greedy Best First Search - YouTube](#)[Brushing algorithm](#)

[The shortest path algorithm for breadth-first traversal of BFS - Programmer Sought](#)

By the name of Allah (God) Almighty, here with I pledge and truly declare that I have solved Quiz 2 by myself, did not do any cheating by any means, did not do any plagiarism, and did not accept anybody's help by any means. I am going to accept all of the consequences by any means if it has proven that I have been done any cheating and/or plagiarism.

Surabaya, 10 June 2021



Fais Rafli Akbar Hidayat
05111940000159



Salman Damai Alfariz
05111940000159



Rihan Farih Bonjamin
05111940000165