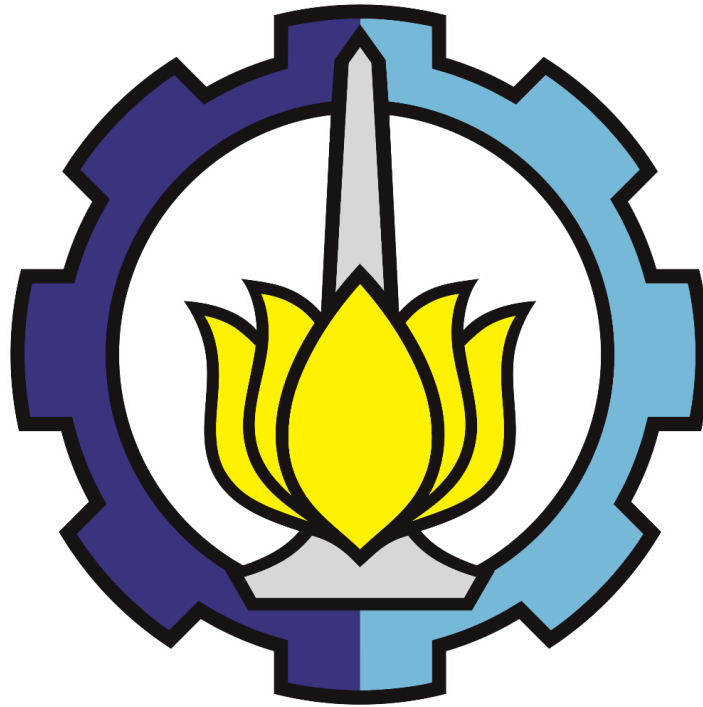


# Quiz 2

## Design and Analysis of Algorithms



1. Ryukazu Andara S – 05111840000129
2. Gema Adi Perwira – 05111840000138
3. Nodas Uziel Separa - 05111840007007

Design and Analysis of Algorithms E

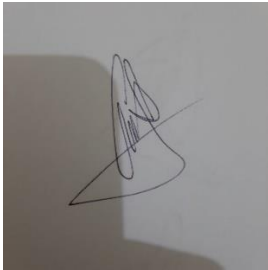
Departement of Informatics Faculty Of Intelligent Electrical And Informatics  
Technology

Sepuluh Nopember Institute of Technology Surabaya

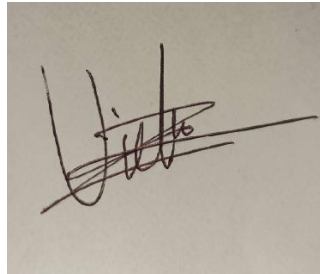
2020

“By the name of Allah (God) Almighty, herewith I pledge and truly declare that I have solved quiz 1 by myself, didn’t do any cheating by any means, didn’t do any plagiarism, and didn’t 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, March 25, 2020

A handwritten signature in black ink on a light-colored background. The signature is stylized, starting with a large 'G' and ending with a long horizontal stroke.

Gema Adi Perwira  
05111840000138

A handwritten signature in black ink on a light-colored background. The signature is stylized, starting with a large 'N' and ending with a long horizontal stroke.

Nodas Uziel Separa  
05111840007007

A handwritten signature in black ink on a light-colored background. The signature is stylized, starting with a large 'R' and ending with a long horizontal stroke.

Ryukazu Andara S.  
05111840000129

# Tour de East Java using DFS Algorithm

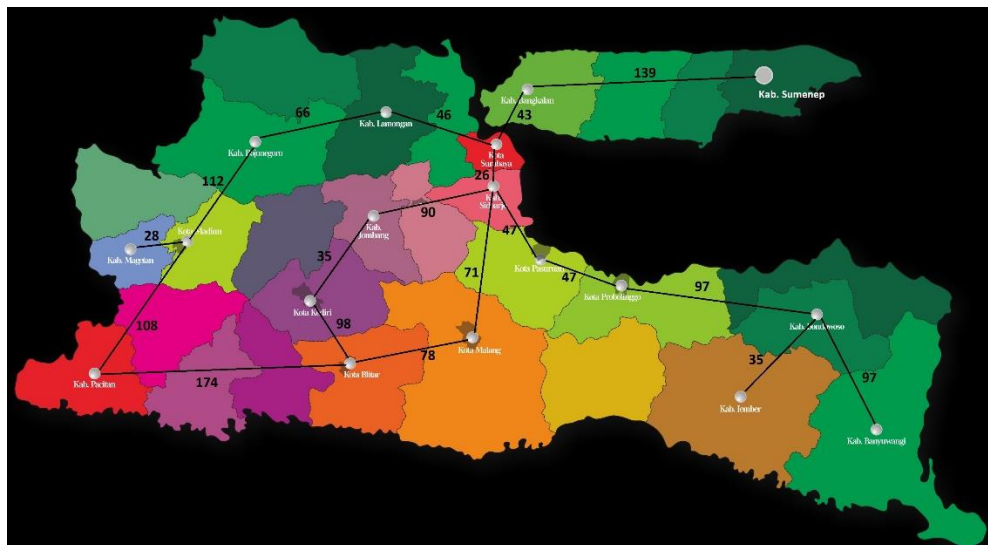
## -Game Definition

In this game, we would like to invite you to travel around East Java. This game contain mini quiz to get you to know how to travel from the city to other city in East Java and also to get you to know the distance that you'll go through and the best ways to go through. If you answer correctly, you will win this game, and if you don't, you will lose and the game will give you the correct answer.

## -Rules

1. First, you must see the East Java map that will given in folder.
2. You must observe that map and remember the city, location of city, and the distance.
3. Then you must close the map and open the game. (Don't cheating)
4. In the game, the question will appear randomly, you just have to input how many ways to get to the city that the question asked, input the possibility ways, and input the distance
5. So simple as that right?

## -Game Explanation



Picture above is the East Java Map that you must see and observe before start the quiz. After that, you open the game and run it. The game will give you randomly question. Example :

```
Ada berapa cara dari Banyuwangi ke Kediri ?  
Jumlah cara =
```

You just have to input the number like the example below.

```
Ada berapa cara dari Banyuwangi ke Kediri ?  
Jumlah cara = 4  
Jawaban Anda salah!  
Jawaban yang benar adalah 3 cara
```

And then, the second question will appear. No matter you answer right or wrong, the second question will appear.

```
Jalur terdekat dari Banyuwangi ke Kediri ?  
Jalur terdekat adalah =
```

You have to input the best ways to get to the destination. Example :

```
Jalur terdekat dari Banyuwangi ke Kediri ?  
Jalur terdekat adalah = Banyuwangi -> Jember -> Malang -> Blitar -> Kediri  
Jawaban Anda salah!  
Jawaban yang benar adalah  
Banyuwangi -> Bondowoso -> Probolinggo -> Pasuruan -> Sidoarjo -> Jombang -> Kediri
```

Then, the final question will appear. The next question is what the distance to get to the destination.

```
Berapa jaraknya ?  
Jarak = 20  
  
Jawaban Anda salah!  
  
Jawaban yang benar adalah 413 km
```

After answer all the question, the game will show the conclusion of all question asked.

```
Jadi, dari quiz game di atas dapat disimpulkan bahwa kemungkinan jalur yang dapat dilalui adalah :

1 .
Banyuwangi -> Bondowoso -> Probolinggo -> Pasuruan -> Sidoarjo -> Surabaya -> Lamongan -> Bojonegoro -> Madiun -> Pacitan -> Blitar -> Kediri

2 .
Banyuwangi -> Bondowoso -> Probolinggo -> Pasuruan -> Sidoarjo -> Jombang -> Kediri

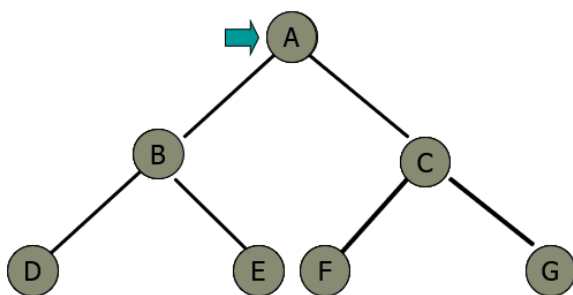
3 .
Banyuwangi -> Bondowoso -> Probolinggo -> Pasuruan -> Sidoarjo -> Malang -> Blitar -> Kediri

Jumlah jalur yang bisa ditempuh adalah: 3 jalur
Jalur yang terdekat adalah: Banyuwangi -> Bondowoso -> Probolinggo -> Pasuruan -> Sidoarjo -> Jombang -> Kediri
Dengan jarak 413 km
```

## -Explanation

Basically, this game can be solved using a Deep First Search (DFS) algorithm. This algorithm is used to find the possibility ways to destination city.

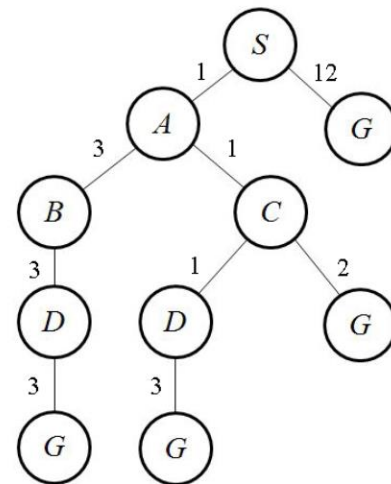
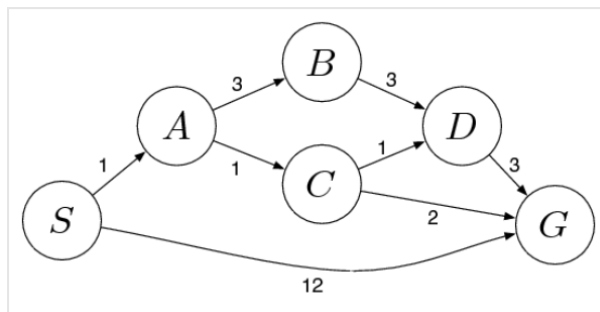
```
def dfs_paths(source, destination, path=None):
    if path is None:
        path = [source]
    if source == destination:
        yield path
    for next_node in set(GRAPH[source].keys()) - set(path):
        yield from dfs_paths(next_node, destination, path + [next_node])
```



If A = Pacitan, B = Madiun(108), C = Blitar(174), D = Magetan(28), E = Bojonegoro(112), F = Malang(78), G = Kediri(98). So if we run this DFS, the search result will be  $A > B > D > E > C > F > G$ .

And then, we find the closest distance using the traverse function that named def terdekat(). This is uniform cost search algorithm. Uniform cost search is Strategy: Expand the lowest cost node. Implementation: the fringe is a priority queue: lowest cost node has the highest priority. In order to be optimal, must test at expansion, not generation, time.

```
def terdekat(source, destination):
    from queue import PriorityQueue
    priority_queue, visited = PriorityQueue(), {}
    priority_queue.put((0, source, [source]))
    visited[source] = 0
    while not priority_queue.empty():
        (cost, vertex, path) = priority_queue.get()
        if vertex == destination:
            return cost, path
        for next_node in GRAPH[vertex].keys():
            current_cost = cost + GRAPH[vertex][next_node]
            if not next_node in visited or visited[next_node] >= current_cost:
                visited[next_node] = current_cost
                priority_queue.put((current_cost, next_node, path + [next_node]))
```



Initialization: { [ S , 0 ] }

Iteration1: { [ S->A , 1 ] , [ S->G , 12 ] }

Iteration2: { [ S->A->C , 2 ] , [ S->A->B , 4 ] , [ S->G , 12 ] }

Iteration3: { [ S->A->C->D , 3 ] , [ S->A->B , 4 ] , [ S->A->C->G , 4 ] , [ S->G , 12 ] }

Iteration4: { [ S->A->B , 4 ] , [ S->A->C->G , 4 ] , [ S->A->C->D->G , 6 ] , [ S->G , 12 ] }

Iteration5: { [ S->A->C->G , 4 ] , [ S->A->C->D->G , 6 ] , [ S->A->B->D , 7 ] , [ S->G , 12 ] }

Iteration6 gives the final output as S->A->C->G (shortest cost).

## Main Function :

```
def main():
    print("Daftar Kota di Jawa Timur : ")
    p = open("Listtown.txt", "r")
    print(p.read())
    print("\n")
    cities = open('horizoncity.txt').readlines()
    city = cities[0]
    words = city.split()
    source = random.choice(words)
    goal = random.choice(words)
    if source == goal:
        print("Generate random mengambil asal kota dan tujuan kota yang sama!")
        sys.exit(1)
    count = 0
    paths = dfs_paths(source, goal)
    cost, jalur_terdekat = terdekat(source, goal)
    for path in paths:
        #a = (' -> '.join(city for city in path))
        count+=1
    print("Ada berapa cara dari " + source + " ke " + goal + " ?", end=' ')
    ans1 = int(input("\nJumlah cara = "))
    benar = 0
    if ans1 == count:
        print ('Selamat jawaban Anda benar!', end='\n')
        benar += 1
    else:
        print('Jawaban Anda salah!', end='\n')
        print("Jawaban yang benar adalah", count, "cara")
    print("\nJalur terdekat dari " + source + " ke " + goal + " ?", end='\n')
    d = " -> ".join(jalur_terdekat)
    # print(d)
    ans2 = input('Jalur terdekat adalah = ')
    if ans2 == d:
        print ('Selamat jawaban Anda benar!', end='\n')
        benar += 1
    else:
        print('Jawaban Anda salah!', end='\n')
        print('Jawaban yang benar adalah ')
        print(' -> '.join(city for city in jalur_terdekat))
    print('\nBerapa jaraknya ?')
    ans3 = int(input("Jarak = "))
    if ans3 == cost:
        print ('\nSelamat Jawaban Anda benar!', end='\n')
        benar += 1
    else:
        print('\nJawaban Anda salah!', end='\n')
        print("\nJawaban yang benar adalah", cost, "km\n")
    if benar == 1:
        print("\nNilai anda adalah 50 ")
    elif benar == 2:
        print("\nNilai anda adalah 75 ")
    elif benar == 3:
        print("\nNilai anda adalah 100, Selamat anda benar semua!")
```

```

elif benar == 0:
    print("\nDibaca mapnya lagi yaaa")
    print("\nJadi, dari quiz game di atas dapat disimpulkan bahwa kemungkinan jalur
yang dapat dilalui adalah :")
    paths = dfs_paths(source, goal)
    hitung = 0
    for path in paths:
        print("\n")
        hitung+=1
        print(hitung, ".")
        print(' -> '.join(city for city in path))
        #print ("\nJumlah kota yang dilewati adalah sebanyak", len(path) , "kota") #
        masih salah disini, harusnya ngeprint semua jalur
    print("\n")
    print("Jumlah jalur yang bisa ditempuh adalah:", count, "jalur")
    print("Jalur yang terdekat adalah:",( ' -> '.join(city for city in
jalur_terdekat)))
    print("Dengan jarak",cost, "km\n")

if __name__ == '__main__':
    main()

```

In main function, first we open list of the city and then we generate random the city from file randcity.txt. This random city used to get the value of variables source and goal. Next, if the city from source and goal is equal, so we need to run the program again. And then if not, we continue to make a question for the quiz.

Link Github :

[https://github.com/rykz-s/DAAE\\_Q2\\_Group8-2019-2020-](https://github.com/rykz-s/DAAE_Q2_Group8-2019-2020-)



