Iteration 1

Problem Identification

The availability of some care services are not always accessible by people around the Philippines in terms of distance. Because of this the survival rate of some patient decreases due to the distance required to travel to a further care center.

Decomposition

- The geographical location of an individual.
- The type of illness that a person may have
- The medium of transportation used to travel to their destination.
- A persons background, whether a civilian or an ambulance driver

Pattern Recognition

The distance required poses a threat when it comes to the survival of an patient.

Abstraction

Relevant : Illness, Distance, Transportation
Irrelevant :

- Situation of the road, whether traffic or not.
- ---

Graphic Organizer

Iteration 2

Problem Identification

How will I be able to get the least distance required in order to get to a care center?

Decomposition

- How to utilize the total distance to figure out the survival.

Pattern Recognition

To get the least distance required, simply use an algorithm that can disseminate a given list in order to figure out the list distance among them. But what possible algorithm can be used?

Abstraction

Graphic Organizer

Iteration 3

Problem Identification

The dijkstra algorithm can be used in combination to using nodes to simulate the roads to obtain the least distance but how can the distance be used to figure out the survival of a patient?

Decomposition

How will you be able to calculate the comparison between the distance and the time.

Pattern Recognition

Utilizing both a given golden hour of an illness can be used to compare the given output of the least distance required.

Abstraction

Graphic Organizer

Code Breakdown

```python
class GoldenTime():
    def __init__(self): # A pre-existing available time is already here
        self.illness = {
            'heart attack' : 3600,
            'stroke' : 21600,
            'trauma' : 3600,
            'asthma attack' : 3600,
            'severe dehydration' : 7200,
            'septic shock' : 3600,
            'dengue hemorrhagic fever' : 86400,
            'severe allergic reaction' : 3600,
            'third-degree burns' : 3600,
            'tetanus' : 7200,
            'severe pneumonia' : 3600,
            'severe malaria' : 43200
            }

    # Simply shows the current data inside the dictionary.
    def show_golden_time(self):
        print(f'\ncurrently stored golden times (name : seconds) : \n{self.illness}')

    # This can be used to insert a new Illness with its corresponding time on
    # the dictionary, don't mind this if you are not the user since you will
    # actually need to edit the dictionary directly if you want to save it for
    # later use...
    def add_golden_time(self):
        while True:
            try:
                ill  = str(input('\nPlease input the name of the illness : '))
                time = int(input('Please input the Golden Time in SECONDS of the illness : '))
                break
            except ValueError:
                print('PLease insert the time in whole integers!')
        ill = ill.lower()
        self.illness[ill] = time
        print(f'updated [{ill} : {time}] to the dictionary.')
        self.show_golden_time()
```

```python
    # simply calculates whether some common possible illnesses
    # survivable with the given distance both in the minimum an
    # compared to its golden hour expressed in seconds.
    def golden_survival(self, min_distance):
        min_speed    = round((min_distance / 16.67), 4) # m/s
        max_speed    = round((min_distance / 27.78), 4) # m/s
        min_survival = True
        max_survival = True
        print(f'\n{min_speed} is the amount of time it will tak
        print(f'{max_speed} is the amount of time it will take
        for i in self.illness:
            if self.illness[i] < min_speed:
                print(f'{i} failed the minimum speed')
                min_survival = False
            if self.illness[i] < max_speed:
                print(f'{i} failed the max_speed')
                max_survival = False
            else: print(f'{i} passed both test!')
        if min_survival is True and max_survival is True:
            print('\nall common illnesses are survivable!')
        else: print('\nthere were some illnesses that failed either or both minimum and maximum s
        return min_survival, max_survival
```

Code Breakdown

```python
SimulatedRoads = { # Distances of each node from their adjacent nodes.
    'A' : [('B', 100), ('C', 150), ('D', 800)],
    'B' : [('A', 100), ('C', 250), ('E', 350)],
    'C' : [('A', 150), ('B', 250), ('HOSPITAL', 300)],
    'D' : [('A', 800), ('HOSPITAL', 1000)],
    'E' : [('B', 350), ('HOSPITAL', 30)],
    'HOSPITAL' : [('E', 30), ('C', 300), ('D', 1000)]
}


# dijkstra algorithm will be used here, for now the program will manually input
# the minimum distance...
def GoldenSort(roads, start, end, golden_time):
    pass

if __name__ == '__main__':
    program = GoldenTime()
    while True:
        x = input('\n1 = add illness\n2 = show current illnesses\n3 = simulate golden time\n0 = e
        if x == '1':
            program.add_golden_time()
        elif x == '2':
            program.show_golden_time()
        elif x == '3':
            temp = 450 # this is a sample minimum di
            print(f'shortest distance : {temp}')
            program.golden_survival(temp)
        elif x == '0':
            break
        else: print('invadid input!')
    print('Program Terminated, goodbye!')
```

```
1 = add illness
2 = show current illnesses
3 = simulate golden time
0 = end program
X = 3
shortest distance : 450

26.9946 is the amount of time it will take going on
16.1987 is the amount of time it will take going on

heart attack passed both test!
stroke passed both test!
trauma passed both test!
asthma attack passed both test!
severe dehydration passed both test!
septic shock passed both test!
dengue hemorrhagic fever passed both test!
severe allergic reaction passed both test!
third-degree burns passed both test!
tetanus passed both test!
severe pneumonia passed both test!
severe malaria passed both test!

all common illnesses are survivable!
```