Manual

1.Christofides algorithm

1.1 Setting up running environment.

1.1.1 Python3

I used Python3 programming language in this project. I suppose you must be familiar with Python, but if not. You can quickly get started with Python and know how to print 'Hello World' using python in 10 minutes. Below are some useful links about download and learning Python:

https://realpython.com/installing-python/

https://www.stavros.io/tutorials/python/

1.1.2 Other packages

I didn't use any other third-party packages in this project, you can run my code correctly after installing Python3 successfully.

1.2 Test data

You can change the test data in the bottom of *ca_for_tsp.py* in the **christofides_algorithm** directory by altering the value of variable **data**. The shape of **data** is two-dimension list, the length of data indicates how many cities in this TSP problem, each element in data represents the positions of this city in our coordinate. The first number is the x coordinate value, the second number is the y coordinate value.

1.3 Running code

1.3.1 Running in command line

If you want to run my code in the command line, just enter to the *1. Christofides Algorithm* directory by cd command, then type command *python ca_for_tsp.py.*

1.3.2 Running in PyCharm or other IDLE.

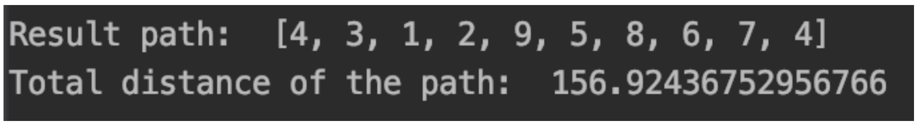
Just run the *ca_for_tsp.py* python file.

1.4 Sample Execution:

Input:

data = [[0, 1], [3, 10], [6, 8], [1, 4], [2, 3], [18, 9], [23, 2], [31, 61], [18, 7], [10, 9]]

Output:



```
Result path:  [4, 3, 1, 2, 9, 5, 8, 6, 7, 4]
Total distance of the path:  156.92436752956766
```

2. Multiplicative Weight Updates

I tried, but I don’t know how to do it.

3. Data Structure

3.1 Fibonacci Heaps

3.1.1 Running Environment:

Just as same as the environment in question 1. Python3 without other third-party packages.

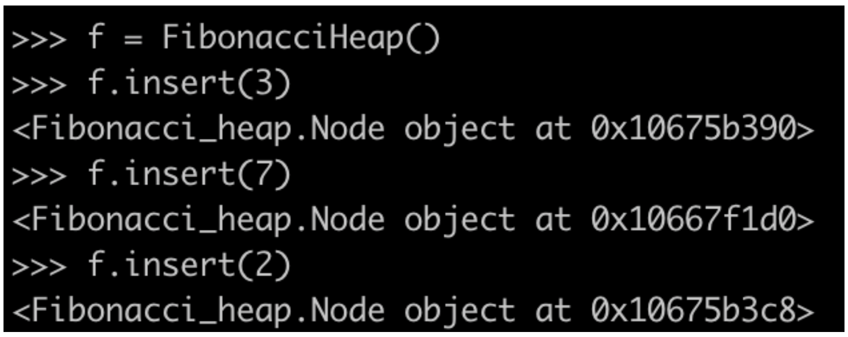
3.1.2 Run the code:

a. If you want to run my test data. Just run the *Fibonacci_heap.py* in ***3.Data Structure*** directory. The two methods to execute python file was shown in the manual of question 1.

b. If you want to test yours data, just follow the sample execution below, initialize a FibonacciHeap object, and using insert method to add values then using other methods to test the function of my Fibonacci Heap structure.

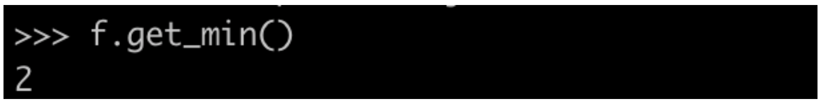
3.1.3 Sample Execution:

Insert method:



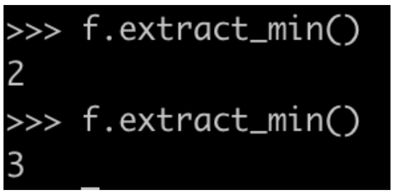
```
>>> f = FibonacciHeap()
>>> f.insert(3)
<Fibonacci_heap.Node object at 0x10675b390>
>>> f.insert(7)
<Fibonacci_heap.Node object at 0x10667f1d0>
>>> f.insert(2)
<Fibonacci_heap.Node object at 0x10675b3c8>
```

Get minimum value in heap:



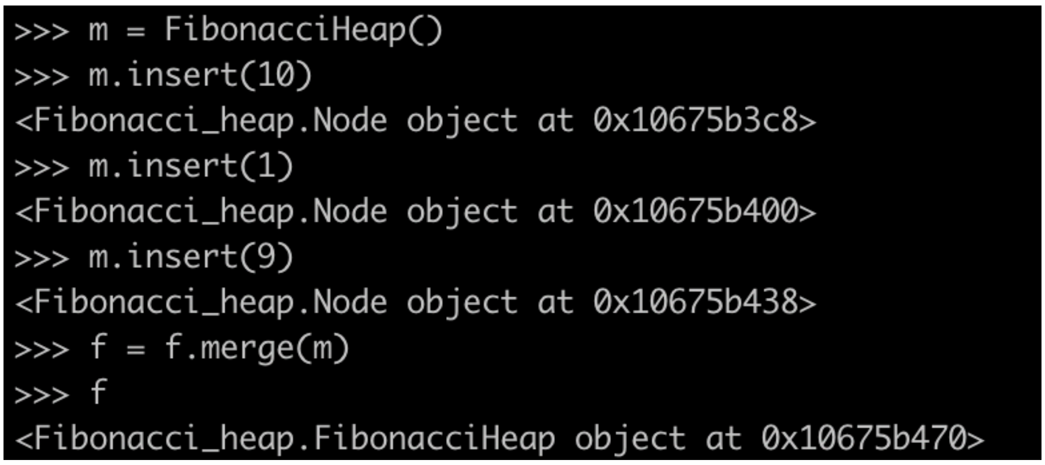
```
>>> f.get_min()
2
```

Extract minimum value in heap:



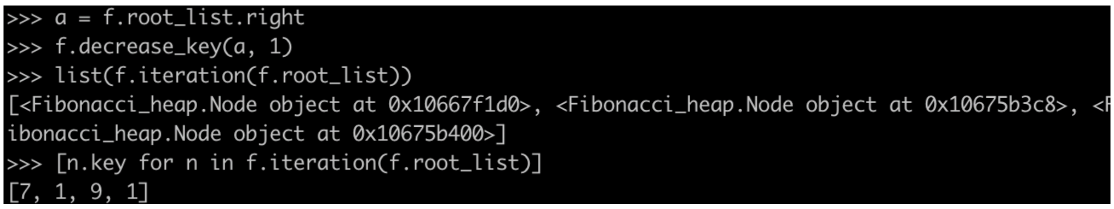
```
>>> f.extract_min()
2
>>> f.extract_min()
3
```

Merge two heaps:



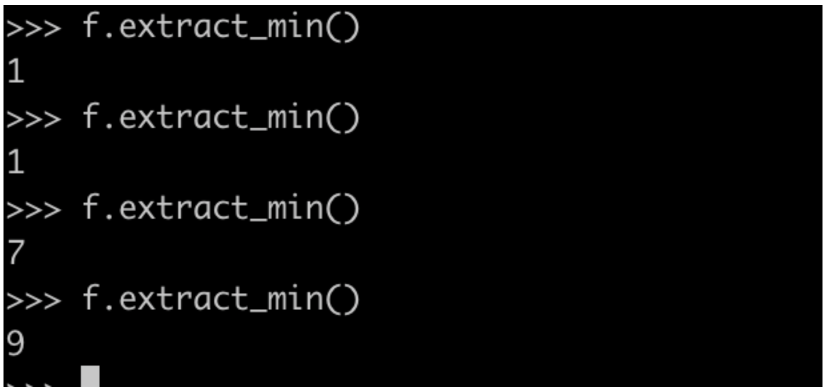
```
>>> m = FibonacciHeap()
>>> m.insert(10)
<Fibonacci_heap.Node object at 0x10675b3c8>
>>> m.insert(1)
<Fibonacci_heap.Node object at 0x10675b400>
>>> m.insert(9)
<Fibonacci_heap.Node object at 0x10675b438>
>>> f = f.merge(m)
>>> f
<Fibonacci_heap.FibonacciHeap object at 0x10675b470>
```

Print the values in the new heap:



```
>>> a = f.root_list.right
>>> f.decrease_key(a, 1)
>>> list(f.iteration(f.root_list))
[<Fibonacci_heap.Node object at 0x10667f1d0>, <Fibonacci_heap.Node object at 0x10675b3c8>, <F
ibonacci_heap.Node object at 0x10675b400>]
>>> [n.key for n in f.iteration(f.root_list)]
[7, 1, 9, 1]
```

Extract minimum value in new heap:



```
>>> f.extract_min()
1
>>> f.extract_min()
1
>>> f.extract_min()
7
>>> f.extract_min()
9
```

3.2 Hollow Heaps

I tried, but I don't know how to do it.