VE477 Lab5 Report

Student Name: Zhou Haoquan

Student ID: 519370910059

3. Compare the time complexity and running time of `Bellman-Ford` and `Dijkstra`

3.1 Time Complexity

In this lab, we use the `Fibonacci Heap` to implement the `Dijksta` algorithm, which gives a time complexity of

$$\mathcal{O}(|E| + |V|\log|V|)$$

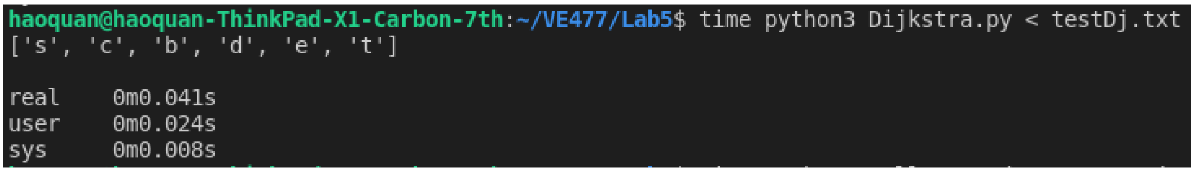
The `Bellman-Ford` will go through `|V|` iterations and each iteration will visit all the `|E|` edges. Then the time complexity is

$$\mathcal{O}(|E||V|)$$

In terms of time complexity, the `Dijkstra` algorithm is better than `Bellman-Ford` algorithm.

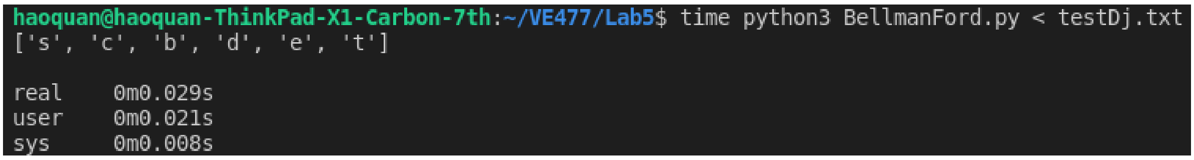
3.2 Real Running time

A small input set is given to both algorithms and the result is shown in the following figurea:



```
haoquan@haoquan-ThinkPad-X1-Carbon-7th:~/VE477/Lab5$ time python3 Dijkstra.py < testDj.txt
['s', 'c', 'b', 'd', 'e', 't']

real    0m0.041s
user    0m0.024s
sys     0m0.008s
```



```
haoquan@haoquan-ThinkPad-X1-Carbon-7th:~/VE477/Lab5$ time python3 BellmanFord.py < testDj.txt
['s', 'c', 'b', 'd', 'e', 't']

real    0m0.029s
user    0m0.021s
sys     0m0.008s
```

We can see that the total running time of `Dijkstra` algorithm is larger than the `Bellman-Ford` algorithm. This is because the `Fibonacci Heap` even performs worse than a `min-heap` in a small scale input. Also, the `Bellman-Ford` algorithm does not require additional data structures other than a `adjacency list`. So this may save it some time as well.

When it comes to larger input, the `Fibonacci Heap` will make `Dijkstra` performs better than `Bellman-Ford`.