ShanghaiTech University

Computer Networks

Spring 2016

# Project 3: Routing Algorithms via Python

*Author:*
Xin Gao

*Advisor:*
Prof. Ziyu Shao

April 19, 2016

## 1 Project Description

**Routing Algorithms via Python.**

## 2 Introduction

The goal of the project is to implement two routing algorithms (link-state routing and distance-vector routing) via Python. The input is the metro map of Shanghai and arbitrary two stations. I use Chinese names of stations. The output is the least delay path or the least switching times with delay as low as possible.

In my project, I use Dijkstra algorithm as the link-state routing algorithm. Link-state routing algorithm is a global routing algorithm. It uses all the information of the network to compute the least cost path between any two points. The information includes the topology of the network and all the link price. The process of Dijkstra algorithm is:

**Algorithm 1** Dijkstra algorithm for source node $u$ (Ref: James F. Kurose, Keith W. Ross, *Computer Network: A Top-Down Approach*)

```
1:  $N = \{u\}$
2:  for all nodes $v$ do
3:      if $v$ is a neighbor of $u$ then
4:          then $D(v) = \infty$
5:      end if
6:  end for
7:  while $N \neq S$ do
8:      find $w$ not in $N$ such that $D(w)$ is a minimum
9:      add $w$ to $N$
10:     update $D(v)$ for each neighbor $v$ of $w$ and not in $N$:
11:     $D(v) = \min(D(v), D(w) + c(w, v)$
12: end while
```

In the project, I use Bellman-Ford algorithm as the distance-vector routing algorithm. Distance-vector routing algorithm is a decentralized routing algorithm. In this algorithm, each nodes calculate path with local information at the beginning. The local information in this project is the set of links connected directly to it. In each iteration, each node will exchange information with it neighborhoods and update the information it has. After some iterations, each node will have all the information of the network and the path it compute is optimal.

**Algorithm 2** Bellman-Ford algorithm for source node $u$

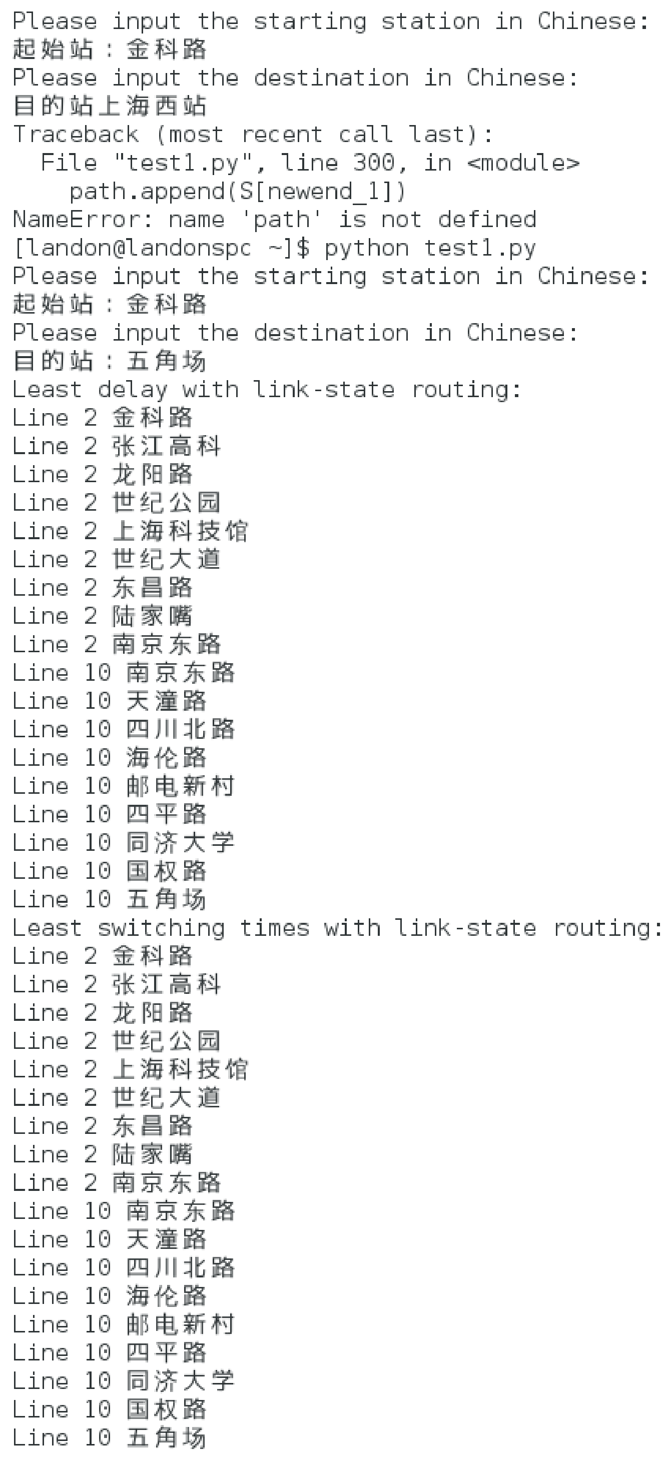
```
1: for all nodes $v$ do
2:     if $v \neq u$ then
3:         $D(v) = \infty$
4:     else
5:         $D(v) = 0$
6:     end if
7: end for
8: $i = 1$
9: for $i \leq |S| - 1$ do
10:     for all edge $w - v$ do
11:         if $D(v) > D(w) + c(w, v)$ then
12:             then $D(v) = D(w) + c(w, v)$
13:         end if
14:     end for
15:     $i++$
16: end for
```

Dijkstra algorithm is much faster than Bellman-Ford algorithm. Since Dijkstra algorithm compute the least path using global information from the beginning. In Bellman-Ford algorithm, nodes has to exchange local information with neighbors, it takes long time for convergence. However, Bellman-Ford algorithm is scalable, while Dijkstra algorithm is not.

## 3 Simulation

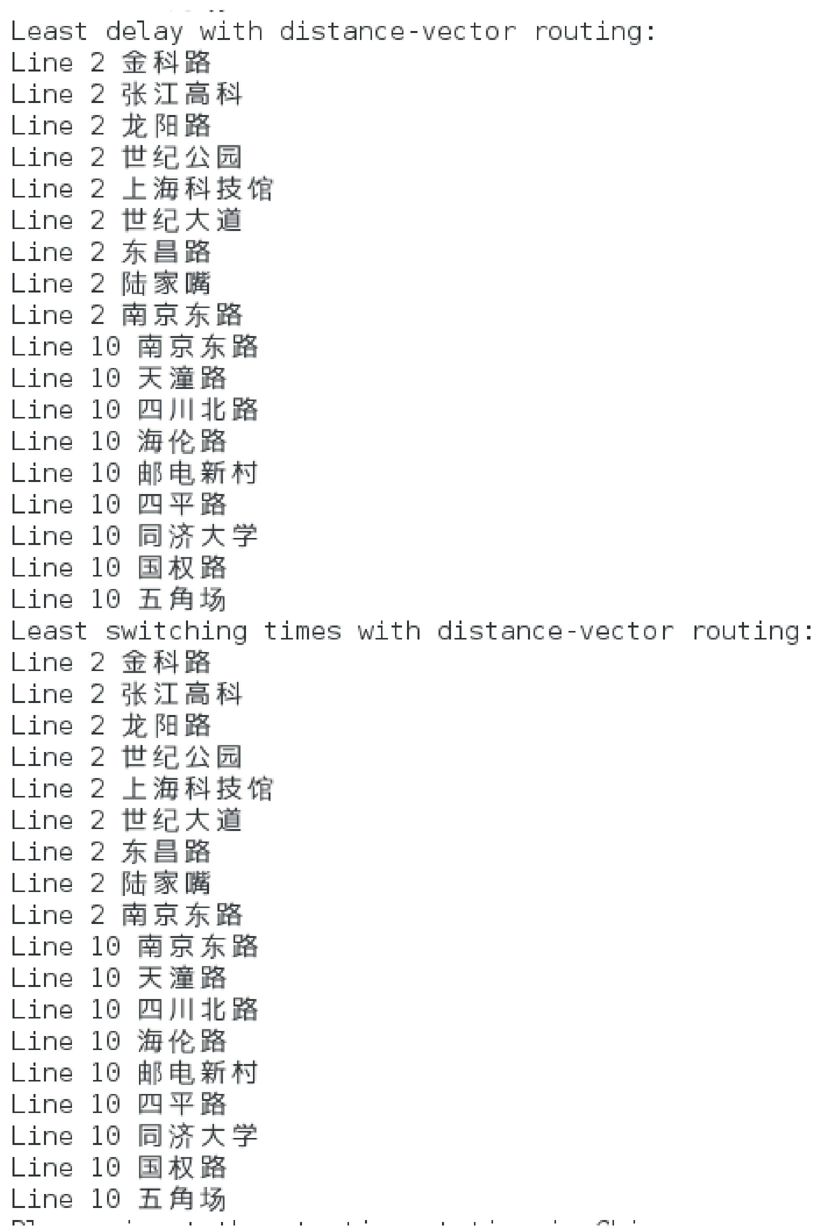
In the simulation, I write the two routing algorithms according to the introduction part of the report. These two algorithms can compute the least delay path directly. However, how to compute the least switch times path can not is a challenge. In my simulation, I refer to the solution of Zhaoxi Wu.

His solution is: if there is an intersection point of several lines, number this point for different line with different number. If we want the least delay path, set the delays between these different numbered station (in fact they are the same station with different number) equal to 0. If we want the least switch time path, set the delays between these different numbered station equal to infinity. I changed Wu's solution in the following way: when compute the least delay path, it's the same as Wu's solution; however, when compute the least switch times path I set the delays equal to 1000, which is much larger than any delay between two directly connected nodes. The result of the simulation proves that my solution is effective. Some of my simulation result are shown in figures.



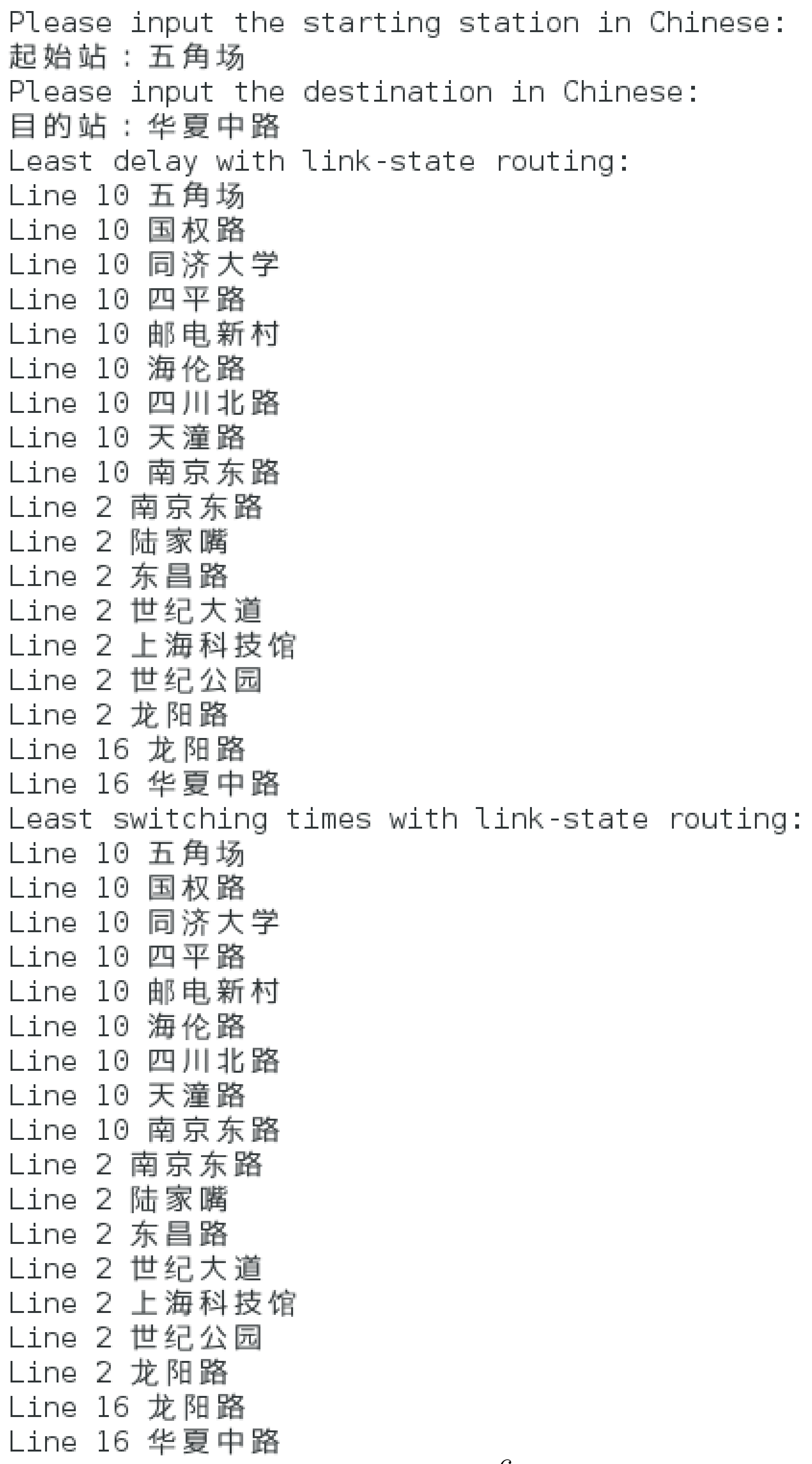
```
Please input the starting station in Chinese:
起始站：金科路
Please input the destination in Chinese:
目的站上海西站
Traceback (most recent call last):
  File "test1.py", line 300, in <module>
    path.append(S[newend_1])
NameError: name 'path' is not defined
[landon@landonspc ~]$ python test1.py
Please input the starting station in Chinese:
起始站：金科路
Please input the destination in Chinese:
目的站：五角场
Least delay with link-state routing:
Line 2 金科路
Line 2 张江高科
Line 2 龙阳路
Line 2 世纪公园
Line 2 上海科技馆
Line 2 世纪大道
Line 2 东昌路
Line 2 陆家嘴
Line 2 南京东路
Line 10 南京东路
Line 10 天潼路
Line 10 四川北路
Line 10 海伦路
Line 10 邮电新村
Line 10 四平路
Line 10 同济大学
Line 10 国权路
Line 10 五角场
Least switching times with link-state routing:
Line 2 金科路
Line 2 张江高科
Line 2 龙阳路
Line 2 世纪公园
Line 2 上海科技馆
Line 2 世纪大道
Line 2 东昌路
Line 2 陆家嘴
Line 2 南京东路
Line 10 南京东路
Line 10 天潼路
Line 10 四川北路
Line 10 海伦路
Line 10 邮电新村
Line 10 四平路
Line 10 同济大学
Line 10 国权路
Line 10 五角场
```

Figure 1: Metro Topology



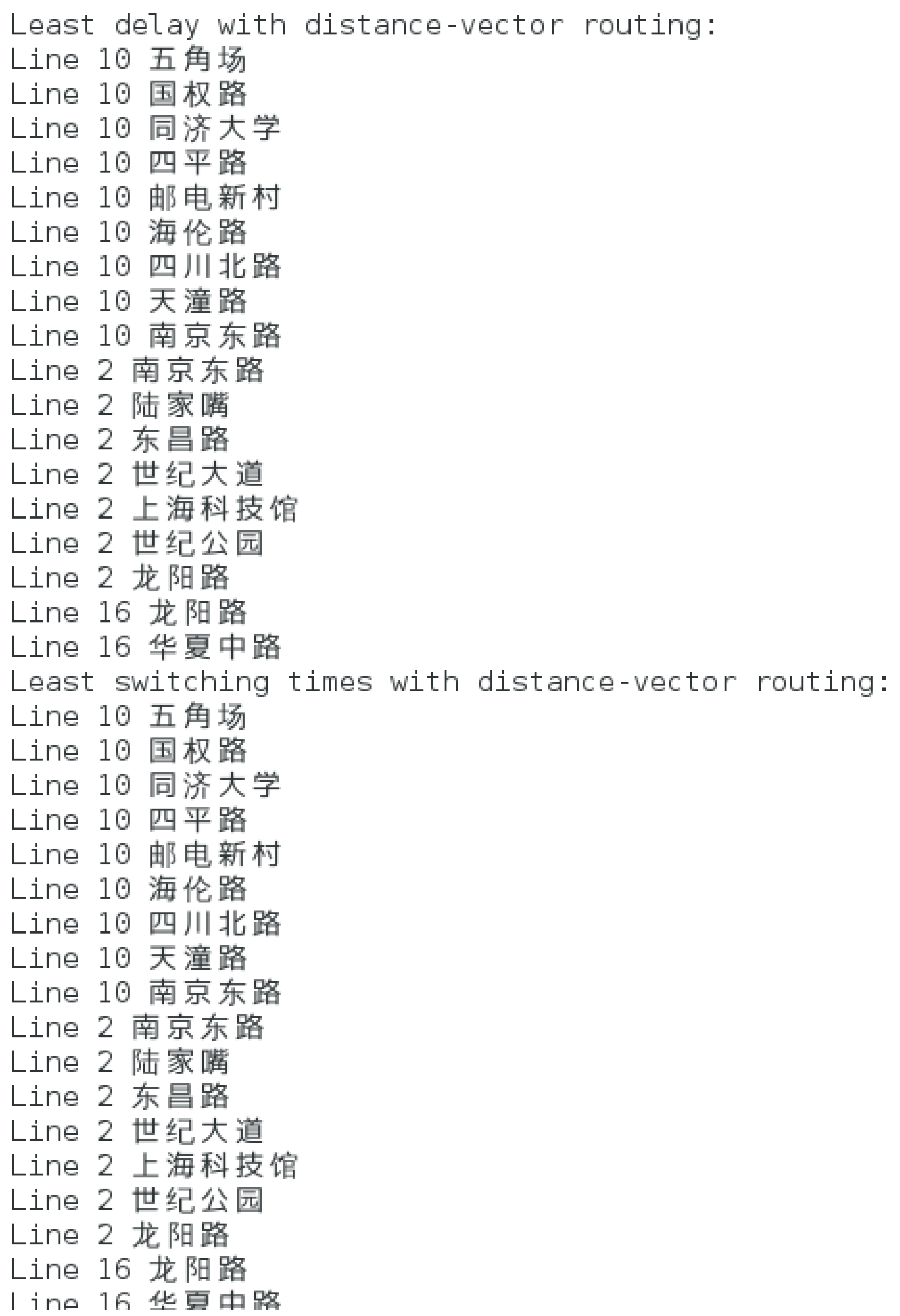
```
Least delay with distance-vector routing:
Line 2 金科路
Line 2 张江高科
Line 2 龙阳路
Line 2 世纪公园
Line 2 上海科技馆
Line 2 世纪大道
Line 2 东昌路
Line 2 陆家嘴
Line 2 南京东路
Line 10 南京东路
Line 10 天潼路
Line 10 四川北路
Line 10 海伦路
Line 10 邮电新村
Line 10 四平路
Line 10 同济大学
Line 10 国权路
Line 10 五角场
Least switching times with distance-vector routing:
Line 2 金科路
Line 2 张江高科
Line 2 龙阳路
Line 2 世纪公园
Line 2 上海科技馆
Line 2 世纪大道
Line 2 东昌路
Line 2 陆家嘴
Line 2 南京东路
Line 10 南京东路
Line 10 天潼路
Line 10 四川北路
Line 10 海伦路
Line 10 邮电新村
Line 10 四平路
Line 10 同济大学
Line 10 国权路
Line 10 五角场
```

Figure 2: Metro Topology



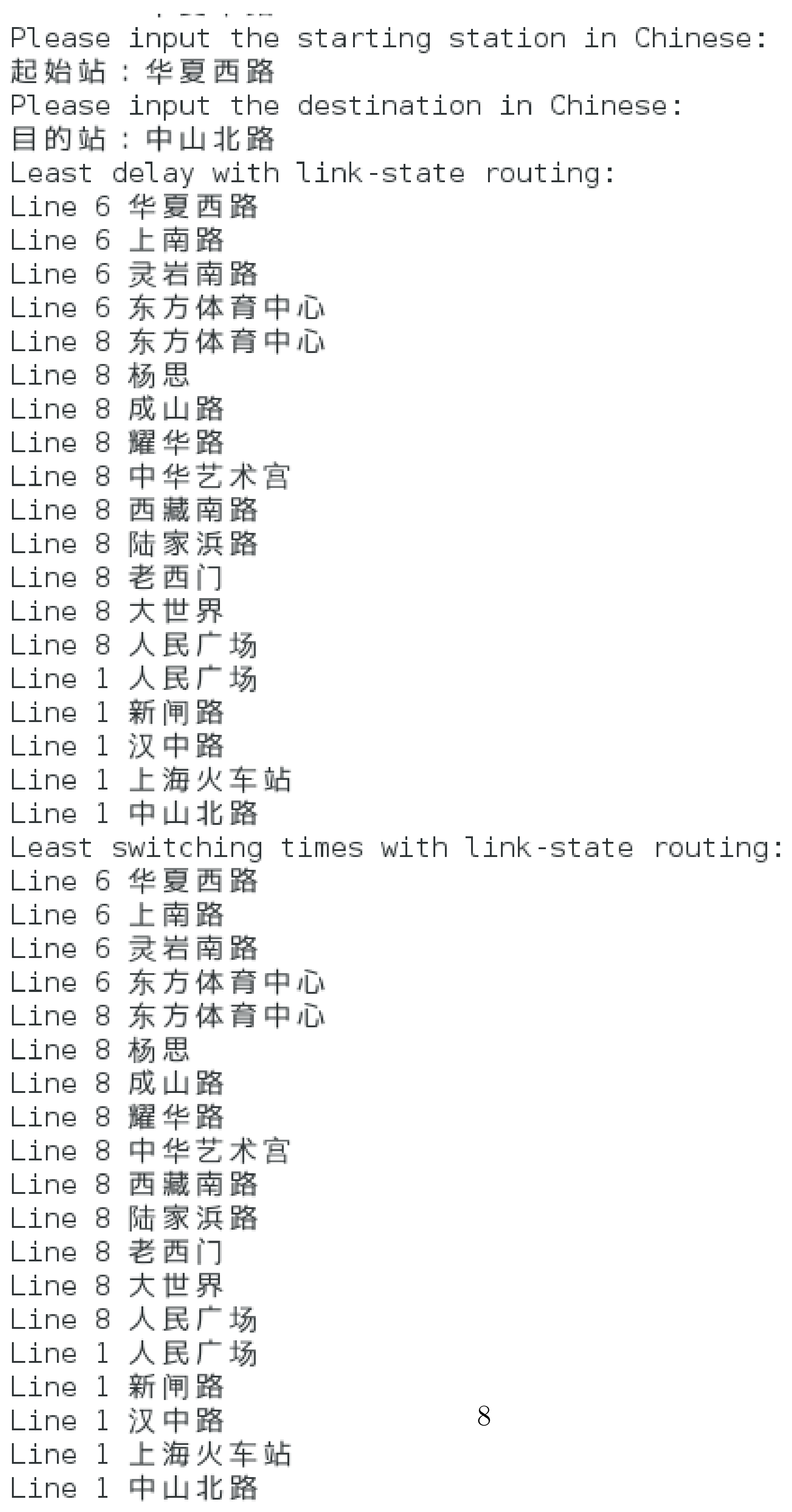
```
Please input the starting station in Chinese:
起始站：五角场
Please input the destination in Chinese:
目的站：华夏中路
Least delay with link-state routing:
Line 10 五角场
Line 10 国权路
Line 10 同济大学
Line 10 四平路
Line 10 邮电新村
Line 10 海伦路
Line 10 四川北路
Line 10 天潼路
Line 10 南京东路
Line 2 南京东路
Line 2 陆家嘴
Line 2 东昌路
Line 2 世纪大道
Line 2 上海科技馆
Line 2 世纪公园
Line 2 龙阳路
Line 16 龙阳路
Line 16 华夏中路
Least switching times with link-state routing:
Line 10 五角场
Line 10 国权路
Line 10 同济大学
Line 10 四平路
Line 10 邮电新村
Line 10 海伦路
Line 10 四川北路
Line 10 天潼路
Line 10 南京东路
Line 2 南京东路
Line 2 陆家嘴
Line 2 东昌路
Line 2 世纪大道
Line 2 上海科技馆
Line 2 世纪公园
Line 2 龙阳路
Line 16 龙阳路
Line 16 华夏中路
```

Figure 3: Metro Topology



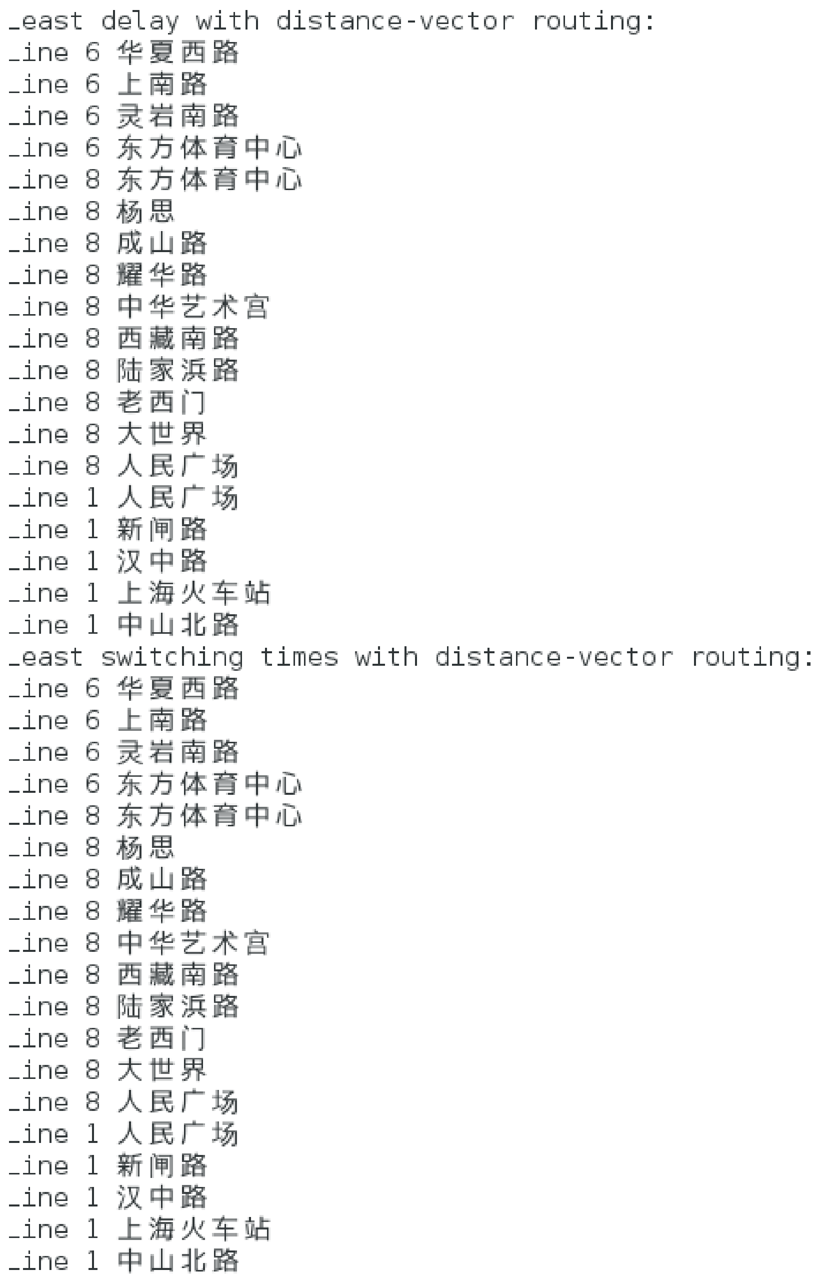
```
Least delay with distance-vector routing:
Line 10 五角场
Line 10 国权路
Line 10 同济大学
Line 10 四平路
Line 10 邮电新村
Line 10 海伦路
Line 10 四川北路
Line 10 天潼路
Line 10 南京东路
Line 2 南京东路
Line 2 陆家嘴
Line 2 东昌路
Line 2 世纪大道
Line 2 上海科技馆
Line 2 世纪公园
Line 2 龙阳路
Line 16 龙阳路
Line 16 华夏中路
Least switching times with distance-vector routing:
Line 10 五角场
Line 10 国权路
Line 10 同济大学
Line 10 四平路
Line 10 邮电新村
Line 10 海伦路
Line 10 四川北路
Line 10 天潼路
Line 10 南京东路
Line 2 南京东路
Line 2 陆家嘴
Line 2 东昌路
Line 2 世纪大道
Line 2 上海科技馆
Line 2 世纪公园
Line 2 龙阳路
Line 16 龙阳路
Line 16 华夏中路
```

Figure 4: Metro Topology



```
Please input the starting station in Chinese:
起始站：华夏西路
Please input the destination in Chinese:
目的站：中山北路
Least delay with link-state routing:
Line 6 华夏西路
Line 6 上南路
Line 6 灵岩南路
Line 6 东方体育中心
Line 8 东方体育中心
Line 8 杨思
Line 8 成山路
Line 8 耀华路
Line 8 中华艺术宫
Line 8 西藏南路
Line 8 陆家浜路
Line 8 老西门
Line 8 大世界
Line 8 人民广场
Line 1 人民广场
Line 1 新闸路
Line 1 汉中路
Line 1 上海火车站
Line 1 中山北路
Least switching times with link-state routing:
Line 6 华夏西路
Line 6 上南路
Line 6 灵岩南路
Line 6 东方体育中心
Line 8 东方体育中心
Line 8 杨思
Line 8 成山路
Line 8 耀华路
Line 8 中华艺术宫
Line 8 西藏南路
Line 8 陆家浜路
Line 8 老西门
Line 8 大世界
Line 8 人民广场
Line 1 人民广场
Line 1 新闸路
Line 1 汉中路
Line 1 上海火车站
Line 1 中山北路
```

Figure 5: Metro Topology



```
_east delay with distance-vector routing:
_ine 6 华夏西路
_ine 6 上南路
_ine 6 灵岩南路
_ine 6 东方体育中心
_ine 8 东方体育中心
_ine 8 杨思
_ine 8 成山路
_ine 8 耀华路
_ine 8 中华艺术宫
_ine 8 西藏南路
_ine 8 陆家浜路
_ine 8 老西门
_ine 8 大世界
_ine 8 人民广场
_ine 1 人民广场
_ine 1 新闸路
_ine 1 汉中路
_ine 1 上海火车站
_ine 1 中山北路
_east switching times with distance-vector routing:
_ine 6 华夏西路
_ine 6 上南路
_ine 6 灵岩南路
_ine 6 东方体育中心
_ine 8 东方体育中心
_ine 8 杨思
_ine 8 成山路
_ine 8 耀华路
_ine 8 中华艺术宫
_ine 8 西藏南路
_ine 8 陆家浜路
_ine 8 老西门
_ine 8 大世界
_ine 8 人民广场
_ine 1 人民广场
_ine 1 新闸路
_ine 1 汉中路
_ine 1 上海火车站
_ine 1 中山北路
```

Figure 6: Metro Topology

Comparing the simulation result with path result of Gaode Map app, the two result are exactly the same. This proves that my algorithms are correct.

## 4 Conclusion

From the project I understand further about the link-state algorithm and distance-vector algorithm. I realised the these two algorithms using python. To compute the least switch times path, I use a solution referred to my classmate Zhaoxi Wu. I did some change about the solution and the new solution is available. From the simulation I conclude that link-state algorithm is much faster than distance-vector algorithm since the previous one is a global algorithm while the latter one is a distributed algorithm. However, distributed algorithm is scalable, while the global algorithm is not. But in this project, the scalability of link-state algorithm is not reflected.