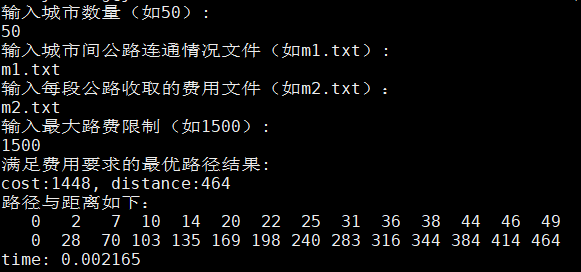
程序运行结果：（./route.bin）



源节点为0，目的节点为49

最优路径为：0 -> 2 -> 7-> 10-> 14-> 20->22-> 25-> 31-> 36-> 38-> 44-> 46-> 49

路费：1448

路径长度：464

运行时间：0.002165s

算法伪代码：

1. //map\_dist  各城市间的距离矩阵
2. //map\_cost 各城市间的费用矩阵
3. //num      城市数量
4. //maxcost   最大费用限制
5. //target     目标城市
6. //min\_dis    各城市间的最短路径
7. //min\_cost   各城市间的最小费用
8. nearestpath(map\_dis, map\_cost, num, maxcost, target, min\_dis, min\_cost)
9. {
10. distance[num]   //记录从源节点到各节点的距离
11. cost[num]       //记录从源节点到各节点的费用
12. visited[num]     //记录节点是否被访问
13. **for** i <- 1 to num **do**
14. distance[i] <- INF
15. cost[i] <- maxcost
16. visited[i] <- **false**
18. vector cities
19. cities.push\_back(0)
21. visited[0] <- **true**
22. pre <- 0
23. res <- INF
25. **while** cities.size() != 0 **do**
26. city <- cities.top()
27. **if** city = target **do**
28. pre <- city
29. **if** distance[city] < res **do**
30. res <- distance[city]
31. cities.pop\_back()
32. visited[city] <- **false**

35. next <- pre + 1
36. **while** next && (visited[next] || map\_dist[city][next] = INF) **do**
37. next <- next+1
39. **if** next =num **do**
40. pre <- city
41. cities.pop\_back()
42. visited[pre] <- **false**
43. **else** **if** cost[city] + map\_cost[city][next] + min\_cost[next][target]
44. <= maxcost
45. && distance[city] + map\_dis[city][next] + min\_dis[next][target]
46. <= res **do**
47. cities.push\_back(next)
48. cost[next] <- cost[city] + map\_cost[city][next]
49. distance[next] <- distance[city] + map\_dis[city][next]
50. visited[next] <- **true**
51. pre <- -1
52. **else** **do** pre <- next
54. **return** res
55. }

算法流程图：

