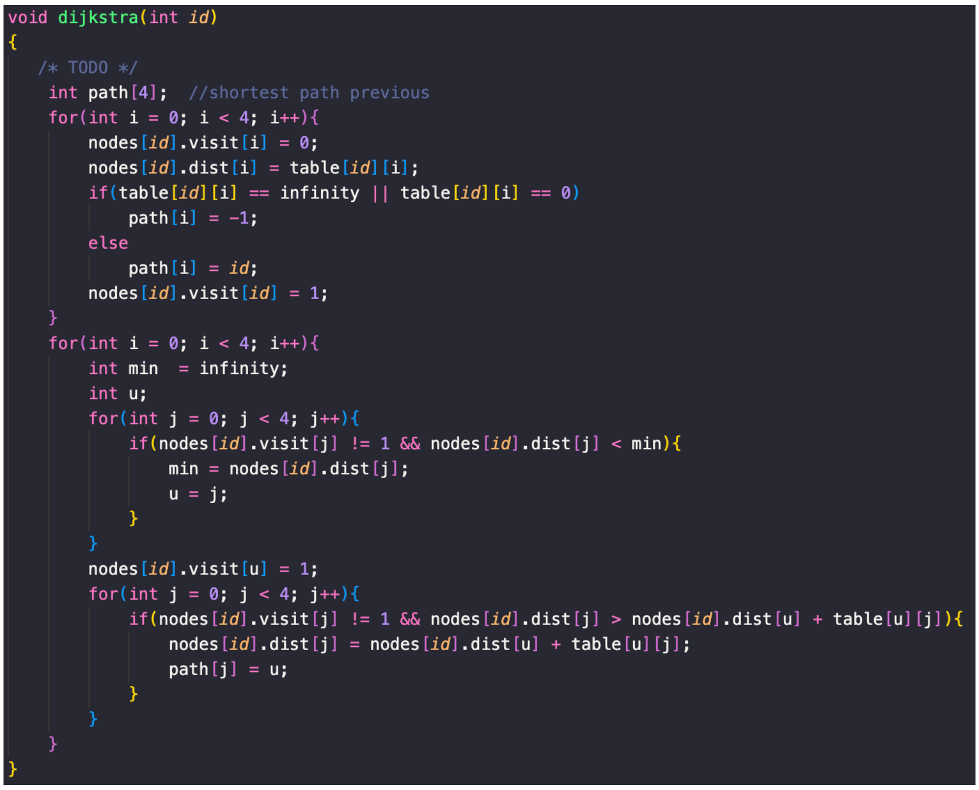
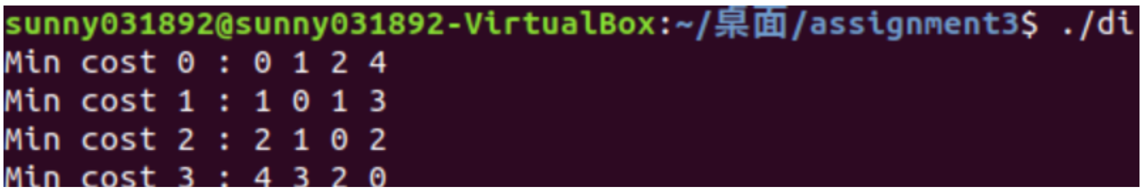
Assignment 3

Part 1 dijkstra



```
void dijkstra(int id)
{
   /* TODO */
    int path[4];  //shortest path previous
    for(int i = 0; i < 4; i++){
        nodes[id].visit[i] = 0;
        nodes[id].dist[i] = table[id][i];
        if(table[id][i] == infinity || table[id][i] == 0)
            path[i] = -1;
        else
            path[i] = id;
        nodes[id].visit[id] = 1;
    }
    for(int i = 0; i < 4; i++){
        int min  = infinity;
        int u;
        for(int j = 0; j < 4; j++){
            if(nodes[id].visit[j] != 1 && nodes[id].dist[j] < min){
                min = nodes[id].dist[j];
                u = j;
            }
        }
        nodes[id].visit[u] = 1;
        for(int j = 0; j < 4; j++){
            if(nodes[id].visit[j] != 1 && nodes[id].dist[j] > nodes[id].dist[u] + table[u][j]){
                nodes[id].dist[j] = nodes[id].dist[u] + table[u][j];
                path[j] = u;
            }
        }
    }
}
```

path[4] is to store the previous node of the route. I use a loop to run all the possible node int he graph, then record the visited node. Then I use another loop to calculate the minimum distance between the node by check the node is visit or not and if the distance is smaller then the previous one. In the end, we can get a matrix that record the minimum distance between every node. The screen shot is the result I run in linux.

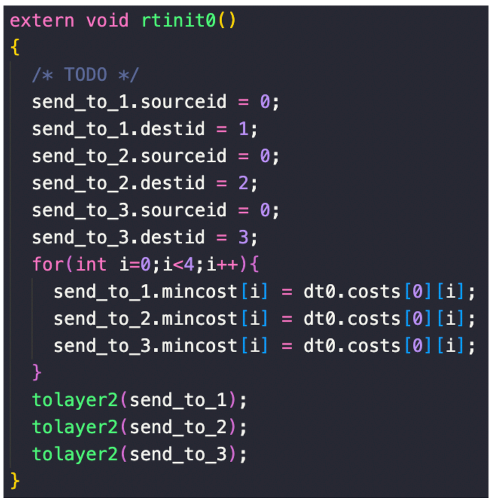


```
sunny031892@sunny031892-VirtualBox:~/桌面/assignment3$ ./di
Min cost 0 : 0 1 2 4
Min cost 1 : 1 0 1 3
Min cost 2 : 2 1 0 2
Min cost 3 : 4 3 2 0
```

Part 2 bellman-ford

In init function, I create 3 rtpkt to record the different situation that node i send to the other 3 nodes. Then record the cost in dt0 into the 3 rtpkt object.

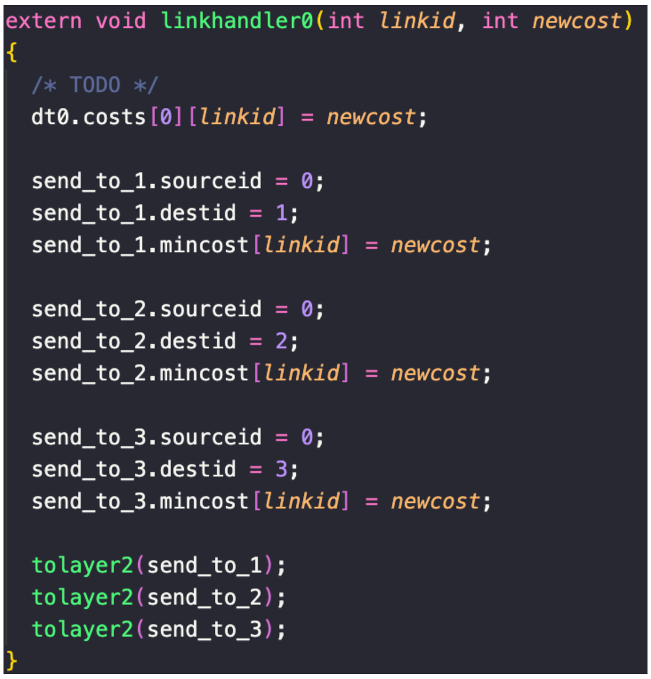
In the update part, first I use a loop to check whether the new route will be better than the previous one, then record the flag, which means the value have been



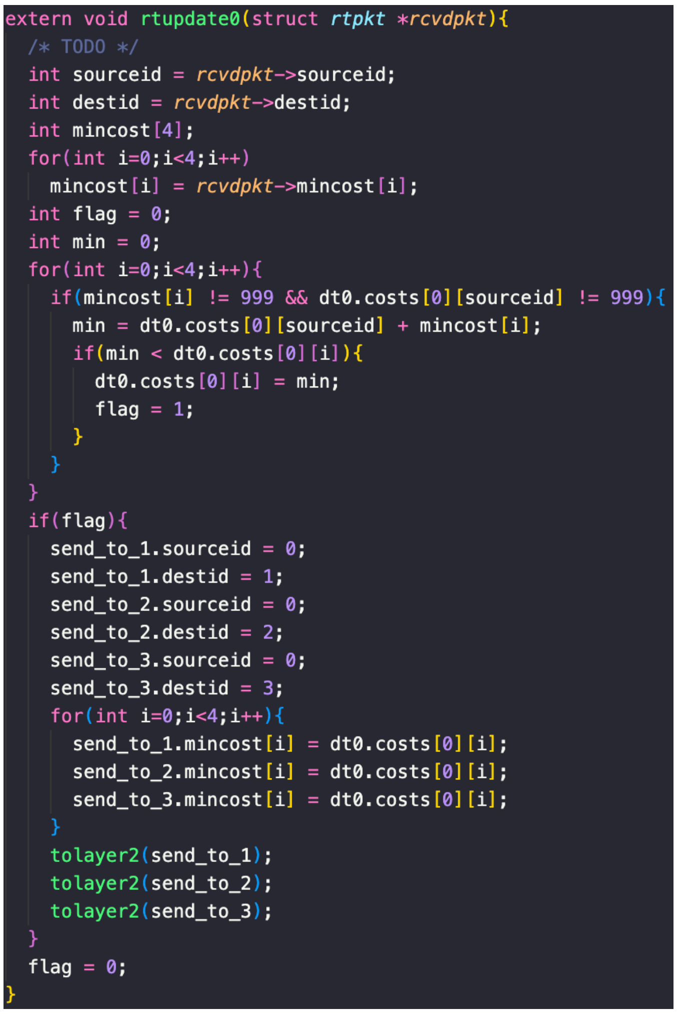
```
extern void rtinit0()
{
  /* TODO */
  send_to_1.sourceid = 0;
  send_to_1.destid = 1;
  send_to_2.sourceid = 0;
  send_to_2.destid = 2;
  send_to_3.sourceid = 0;
  send_to_3.destid = 3;
  for(int i=0;i<4;i++){
    send_to_1.mincost[i] = dt0.costs[0][i];
    send_to_2.mincost[i] = dt0.costs[0][i];
    send_to_3.mincost[i] = dt0.costs[0][i];
  }
  tolayer2(send_to_1);
  tolayer2(send_to_2);
  tolayer2(send_to_3);
}
```

updated. If the value have update, then we should update all of the mincost.

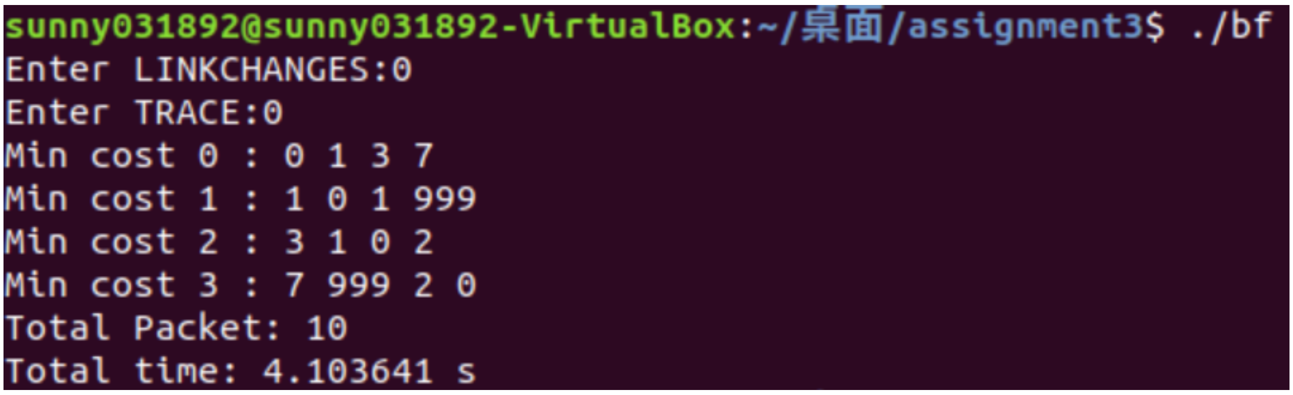
In the linkhandler part, I just assign the new value of each rtpkt of their new minimum cost.



```c
extern void linkhandler0(int linkid, int newcost)
{
  /* TODO */
  dt0.costs[0][linkid] = newcost;

  send_to_1.sourceid = 0;
  send_to_1.destid = 1;
  send_to_1.mincost[linkid] = newcost;

  send_to_2.sourceid = 0;
  send_to_2.destid = 2;
  send_to_2.mincost[linkid] = newcost;

  send_to_3.sourceid = 0;
  send_to_3.destid = 3;
  send_to_3.mincost[linkid] = newcost;

  tolayer2(send_to_1);
  tolayer2(send_to_2);
  tolayer2(send_to_3);
}
```



```c
extern void rtupdate0(struct rtpkt *rcvdpkt){
  /* TODO */
  int sourceid = rcvdpkt->sourceid;
  int destid = rcvdpkt->destid;
  int mincost[4];
  for(int i=0;i<4;i++)
    mincost[i] = rcvdpkt->mincost[i];
  int flag = 0;
  int min = 0;
  for(int i=0;i<4;i++){
    if(mincost[i] != 999 && dt0.costs[0][sourceid] != 999){
      min = dt0.costs[0][sourceid] + mincost[i];
      if(min < dt0.costs[0][i]){
        dt0.costs[0][i] = min;
        flag = 1;
      }
    }
  }
  if(flag){
    send_to_1.sourceid = 0;
    send_to_1.destid = 1;
    send_to_2.sourceid = 0;
    send_to_2.destid = 2;
    send_to_3.sourceid = 0;
    send_to_3.destid = 3;
    for(int i=0;i<4;i++){
      send_to_1.mincost[i] = dt0.costs[0][i];
      send_to_2.mincost[i] = dt0.costs[0][i];
      send_to_3.mincost[i] = dt0.costs[0][i];
    }
    tolayer2(send_to_1);
    tolayer2(send_to_2);
    tolayer2(send_to_3);
  }
  flag = 0;
}
```



```
sunny031892@sunny031892-VirtualBox:~/桌面/assignment3$ ./bf
Enter LINKCHANGES:0
Enter TRACE:0
Min cost 0 : 0 1 3 7
Min cost 1 : 1 0 1 999
Min cost 2 : 3 1 0 2
Min cost 3 : 7 999 2 0
Total Packet: 10
Total time: 4.103641 s
```

Part 3

1. Dijkstra's time complexity is O(V^2), which V is vertices, E is edges. First we should create a nested for loop which cost O(V^2). In the first loop, we should choose a vertex w to be the first node. Then in the second loop we calculate the distance between w and the other node. Therefore, the time complexity is O(V^2).
2. Line 1 of algorithm: The initial for loop runs through each vertex once. Hence this loop runs in O(V) time. Next, consists of 2 loops; one executing V times and the other E times.

So, the time complexity will be O(VE). Last, This loop runs through every edge, therefore running in O(E) time. Thus, the total time complexity is O(V.E).

3. If router A need to send the routing table to router B, we should have A’s routing table & B’s distance table. First, B should search the distance that it go to A. Then search the routing table of A of which router can reach, for example C. Then fill the sum of the two value into A–>C’s blank. Last, find the minimum value of every column to find the minimum distance. So we can get the new routing table.
4. In link state routing, the router will broadcast its packet to all the other router. When a router receive a packet, it would copy it and send to the next router. So we should add the number on the packet so that the router won’t get the same packet twice.
5. If link cost change, the node will recompute the distance. After update its own distance, the node will notify its neighbor that the distance has been change. At the end, all the nodes will have the new value, then keep waiting until any distance have been change.