ICN-HW3 Report

> 學號：B06502147 系級：電機三 姓名：林軒毅

1. [Link-State Routing]

第一題要我們實做Link-State Routing Protocol的部份，主要用Dijkstra's algorithm這個Single Source Shortest Path的演算法對每個router算一次來完成。

此次我使用python撰寫，並參考上課投影片以及之前修演算法的pseudo-code，Dijkstra的實現主要寫在class dijkstra的routing，程式碼如下：

```
def routing(self,source):
    self.init(source)
    while len(self.S)!=vertex_num:
        u = self.extract_min()
        self.S.append(u)
        self.check_in_s[u] = True
        for i in range(vertex_num):
            if self.topo_matrix[u][i]!=-1 and i!=u:
                self.relax(u,i)
```

extract_min將不在S中distance最小的index取出，取出後將u加入S中，check_in_s則是讓extract_min用來判斷index是否在S中，之後就是將u的adjacent vertex分別做relax(更新distance值)

在output file時才做routing的動作，load file只有讀檔並將txt檔存進array。

另外就是outputfile時除了輸出最小的weight之外，還要輸出path中source的下一個vertex，這個部份我是從destination的predecessor往回找，直到predecessor=source時，將當前的vertex輸出。

output相關的程式碼如下：

```
elif command[0]=='of':
    D = dijkstra(topo_matrix)
    for i in range(vertex_num):
        D.routing(i)
        D.output_file()
    print('finish process')
    break
```

```
def output_file(self):
    with open(path_prefix+path_name+'_out1.txt', 'a') as f:
        f.write('Routing table of router {}:\n'.format(self.source+1))
        for i in  range(vertex_num):
            if self.distance[i] == 65535 :
                f.write('-1 -1\n')
            elif self.distance[i] == 0:
                f.write('0 {}\n'.format(self.source+1))
            else:
                f.write('{} {}\n'.format(self.distance[i],self.next_router(i)+1))
        f.close()
```

我也有做例外處理，如果command不符就會要求再次輸入，Ctrl+C可以直接結束程式（或是of完）

2.[Router removed]

除了要實現上一題的所有功能，還需移除router及與其他router的path。當command為rf r[router_id]時，我將matrix中的index為router_id-1的col跟row都設成-1，亦即移除，相關程式碼如下：

```
elif command[0]=='rm':
    router_rm = int(command[1].strip('r'))
    rm_set.append(router_rm)
    adj_mat[router_rm-1].fill(-1)
    adj_mat[:][router_rm-1].fill(-1)
    print('remove router{}'.format(router_rm))
```

在output file時，已移除的router就不輸出，程式碼如下（當移除的router在rm_set中就不寫入txt）：

```
def output_filer(self):
    with open(path_prefix+path_name+'_out2.txt', 'a') as f:
        f.write('Routing table of router {}:\n'.format(self.source+1))
        for i in  range(vertex_num):
            if (i+1 in rm_set):
                continue
            if self.distance[i] == 65535 :
                f.write('-1 -1\n')
            elif self.distance[i] == 0:
                f.write('0 {}\n'.format(self.source+1))
            else:
                f.write('{} {}\n'.format(self.distance[i],self.next_router(i)+1))
        f.close()
```

另外有觀察到未必所有會有路徑唯一解（weight會，但路徑可能有很多條），比如case2是環形的，所以像對角的vertex有可能兩個方向都可以走。