Report

B06504016 林家宏

本次作業我使用 C++

1.

在這個部分，我使用的是著名的 Dijkstra 演算法。以下詳細說明程式內容。

首先讀入輸入的 command

```
string input, first, second;
int position;
int cut = -1;
getline(cin, input);
first = input.substr(0, 2);
second = input.substr(3);
```

如果輸入有依照格式，在輸入的命令是”lf input_file_name.txt”時，這部分程式會打開 input file，接著創建 output file。

```
if(first == "lf"){
    char infilename[second.size() + 1];
    strcpy(infilename, second.c_str());
    fstream fin(infilename);
    fstream fout;
    position = second.find_first_of(".", 0);
    string output;
    output = second.substr(0, position);
    if(cut == -1){
        output = output + "_out1.txt";
    }
    else{
        output = output + "_out2.txt";
    }
    char outfilename[output.size() + 1];
    strcpy(outfilename, output.c_str());
    fout.open(outfilename, ios::out);
```

這部分是讀入資料，並做好初始化。Routing_Table 是動態二維陣列，記錄了

input file 裡面的整張表。Last_router[i][j]是記錄著 Dijkstra 中 i 到 j 最佳路徑中 j 的上一站，shortest_cost[i][j]則記錄著 i 到 j 的最短距離。

```
fin>>n;
int removed[n];
int weight;
int **Routing_Table=NULL;
Routing_Table=new int*[n];
for(int i=0;i<n;i++){
        Routing_Table[i]=new int[n];
        }
for(int i=0;i<n;i++){
    removed[i] = 0;
    for(int j=0;j<n;j++){
        fin>>weight;
        Routing_Table[i][j] = weight;
    }
}
int last_router[n][n];
int shortest_cost[n][n];
for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
            last_router[i][j] = -1;
            shortest_cost[i][j] = Routing_Table[i][j];
            }
        }
```

等待輸入 of

```
while(1){
    getline(cin, input);
    first = input.substr(0, 2);
    if(input.size()> 2 ){
    second = input.substr(3);
    }
```

輸入 of 後，執行 Dijkstra 演算法，每個 router 都當作起點執行一次，總共執行 n 次

```
if(first == "of"){
    for(int i=0;i<n;i++){
        Dijkstra(Routing_Table, last_router[i], shortest_cost[i], i);
    }
```

Dijkstra 演算法內部

用我寫的 FindMinIndex 函數去找到 shortest_cost 中最小的，在 visited 中標示

1，接著將這點所連接到的點做 relax，最後逐步更新 shortest_cost 和 last_router，最後得到最佳解。

```
void Dijkstra(int **R,int *L, int *S, int source){
    bool visited[n] = {0};
    int counting = 0;
    int next;
    L[source] = source;
    for(int i=0;i<n;i++){
        if(R[source][i] > 0){
            L[i] = source;
        }
    }
    visited[source] = 1;
    while(counting != n){
        next = FindMinIndex(S, visited);
        visited[next] = 1;
        if(next == -1){
            break;
        }
        for(int i=0;i<n;i++){
           if(R[next][i] > 0 && visited[i] == 0){
                relax(next, i, R, L, S);
           }
        }
        source = next;
        counting++;
    }
}
```

FindMinIndex 函數

```
int FindMinIndex(int *R, bool *visited){
    int j = 0, k = 999999 ;
    for(int i=0;i<n;i++){
        if(k>R[i] && R[i] > 0 && visited[i] == 0){
            k=R[i];
            j=i;
        }
    }
    if(k == 999999){
        return -1;
    }
    else{
        return j;
    }
}
```

Relax 函數

```
void relax(int node_start, int node_end, int **R, int *L, int *S){
    if(S[node_end] > S[node_start] + R[node_start][node_end] || S[node_end] == -1){
        S[node_end] = S[node_start] + R[node_start][node_end];
        L[node_end] = node_start;
    }
}
```

最後輸出結果

```
  for(int i=0;i<n;i++){
      if(removed[i] == 0){
          fout<<"Routing table of router "<<i+1<<endl;
          for(int j=0;j<n;j++){
              if(removed[j] == 0){
                  fout<<shortest_cost[i][j]<<" "<<find_next_step(last_router[i], i, j)<<endl;
              }
          }
      }
  }
```

Find_next_step 函數，用來找出以某點當作終點時從起點該走的下一步

```
int find_next_step(int *L, int source, int router){
    if(L[router] == -1){
        return -1;
    }
    else{
        while(L[router] != source){
            router = L[router];
        }
        return router + 1;
    }
}
```

釋放記憶體並跳出 while 迴圈

```
            for(int i=0;i<n;i++){
                delete Routing_Table[i];
            }
            delete Routing_Table;
            fin.close();
            fout.close();
            break;
            }
    }
```

2.

其實我兩題是繳交一樣的程式，但是我在這說明移除 router 的部分，其他部分在第一題說明過了。

在第一題的說明中輸入的部分如果輸入的是”rm rn”，n 表示移除的 router 號碼，那麼我就把 Routing_Table 中第 n 行和第 n 列的值全部改成-1，此時我並沒跳出 while 迴圈，因此可以繼續輸入移除 router 的指令。

```
if(first == "rm"){
    string number;
    number = second.substr(1);
    cut = atoi(number.c_str());
    removed[cut - 1] = 1;
    for(int i=0;i<n;i++){
        Routing_Table[i][cut-1] = -1;
        Routing_Table[cut-1][i] = -1;
    }

}
```