## 程式作業5

## 1. 演算法設計

主要想法源於 dfs 演算法,利用二維 vector 儲存 adj,二維 array 儲存 edge,以及 兩個 stack 儲存 dfs 拜訪序列與最終輸出序列。

利用尤拉迴圈刪除一段,剩下的餘序列仍為尤拉回圈的特性,多設計一個 dfs 拜訪序列主要是避免 dfs 先走入死胡同而提早結束,遇到死路後即進行 pop 直到有新的路可走,直到整個圖形被遍歷,輸出最終輸出序列即可得到答案。

## 2. Pseudo code

```
stack<int> visit;
                    //拜訪序列
stack<int> outputans;
                        //最終輸出序列
int rount[10001][10001]={0}; //edge
vector<vector<int> > listarray(10001);
                                        //adj
main()
   for(int num=1 TO E) //儲存輸入
       cin>>i>>j;
       listarray[i].push_back(j);rount[i][j]=1;
       listarray[j].push_back(i);rount[j][i]=1;
   for(int x=1;x<=V;x++){ //檢查 edge 是否屬於尤拉回圈
       if(listarray[x].size()\%2==1)
           check=1;
       sort(listarray[x].begin(),listarray[x].end()); //排字典序
   visit.push(1);
                    //儲存起點
               //呼叫 dfs 號廻
   Visit(1);
   if(check=0 AND outputans.size()=E+1) //檢查是否符合尤拉回圈
       while(outputans.size()>0) //輸出 outputans stack
           cout < < outputans.top() < < ";
           outputans.pop();
   else
       cout << "not exist";
void Visit(int start)
                    //dfs 號迴
   for(int y=0 TO listarray[start].size()){
       if(rount[start][listarray[start][y]] = 1){
           rount[start][listarray[start][y]]=0;
                                            //刪除走過的路徑
           rount[listarray[start][y]][start]=0;
           visit.push(listarray[start][y]); //存入暫存拜訪序列
```

```
Visit(listarray[start][y]); //遞廻下一點
if(outputans.size()!=E+1){ //無路可走時進行 pop 並存入最終輸出序列
int ans = visit.top();
visit.pop();
outputans.push(ans);
```

3. 時間複雜度分析

主要工作區也就是 Visit 的遞迴部分為 O(E) 排序順便檢查偶邊的時間複雜度為 O(VIgV) 總複雜度 O(VIgV+E)

4. Test outcome

```
■ C\Users\賠償買\Desktop\107403527_賠償夏_程式作業(面) exe

6 6
1 2
2 3
3 1
4 5
5 6
6 4
not exist
Process returned 0 (0x0) execution time : 9.782 s
Press any key to continue.

■
```

