Graph Theory: Homework #4

Lin Hung Cheng B01902059

Problem 1

在第2.5 節Euler 迴路的案例中， 改用連續空間表示圖2.10 的圖、但同時要含SAME 和MARK 兩個欄位。

Solution

說明： First element of DATA is index 1. Elements in DATA: [data, same, mark]
BEG: [1, 5, 11, 15]
DATA: [[4, 18, 0], [2, 8, 0], [2, 9, 0], [2, 10, 0], [4, 17, 0], [3, 13, 0], [3, 14, 0], [1, 2, 0], [1, 3, 0], [1, 4, 0], [4, 15, 0], [4, 16, 0], [2, 6, 0], [2, 7, 0], [3, 11, 0], [3, 12, 0], [2, 5, 0], [1, 1, 0]]

Problem 2

在第2.5 節Euler迴路的案例中，利用連續空間表示圖、據以改寫程式inputgraph。

```
function INPUTGRAPH(V, E)                          ▷ array index start from 1
    DATA ← array(E.length * 2)                     ▷ use DATA to represent [data, same, mark]
    for each element d in DATA do
        d.mark ← 0
        d.same ← 0
    end for
    BEG ← array(E.length * 2)
    for edge in E do
        v[edge.first].push(edge.second)            ▷ first and second are two vertex in the edge
        v[edge.second].push(edge.first)
    end for
    BEG[i] ← 1
    for i ← 2, n do
        BEG[i] ← BEG[i − 1] + v[i − 1].length
    end for
    index ← 1
    for i ← 1, n do
        for each node N in v[i] do
            DATA[index].data ← N
            index ← index + 1
        end for
    end for
    for i ← 1, n do
        for j ← 1, v[i].length do
            if v[i][j].same = 0 then
                sameIndex ← findSameIndex(v[i][j].data, i)
                    ▷ findSameIndex: goto BEG=v[i][j].data and return the nearest index which same = 0
and data = i
                myIndex ← BEG[i] + j − 1
                DATA[sameIndex].same ← myIndex
                DATA[myIndex].same ← sameIndex
            end if
        end for
    end for
    return BEG, DATA
end function
```

Algorithm 1: inputGraph

Problem 3

在第2.5 節Euler 迴路的案例中， 利用連續空間表示圖、據以改寫程式Eulertour。

```
function EULERTOUR(V, E, BEG, DATA)
    tour ← array(E.length)
    tournext ← array(E.length)
    tourprev ← array(E.length)
    new ← 1
    cur ← new
    tour[new] ← 1
    tournext[new] ← 0
    tourprev[new] ← 0
    while cur ≠ 0 do
        i ← tour[cur]
        if BEG[i] ≠ 0 then                                  ▷ BEG has the same usage as ADJ
            new ← new + 1
            tour[new] ← data[BEG[i]]
            tournext[new] ← tournext[cur]
            tourprev[new] ← cur
            cur ← new
            mark[BEG[i]] ← 1
            mark[DATA[BEG[i]].same] ← 1
            j ← DATA[BEG[i]].next
            while j ≠ 0 and mark[j] = 1 do
                j ← DATA[j].next
            end while
        else
            cur ← tourprev[cur]
        end if
    end while
    return tour, tourprev, tournext
end function
```

Algorithm 2: Euler Tour

Problem 4

在聯集尋找問題中， 我們若採用NAME、集合表列與SIZE 的方法， 試證明，若演算法總共做了n-1 次聯集的運算， 那需時會是O(n log n)

Solution

考慮最差情況(使計算量最多)，最後一次運算應為$\frac{n}{2}$ 和$\frac{n}{2}$ 的聯集；而要形成$\frac{n}{2}$，最差的情況是 $\frac{n}{4}$ 和 $\frac{n}{4}$ 的聯集……，以此類推。

直到 1 和 1的聯集，可知$\frac{n}{2}$最多出現1次，$\frac{n}{4}$最多出現2次...，最多共需要$\frac{n}{2}*1+\frac{n}{4}*2+\frac{n}{8}*4+...+1*\frac{n}{2}$次計算，因為此式共有$\log_2 n$項，每項和均為$\frac{n}{2}$，所以其和$\frac{n}{2}\log_2 n \leq n\log_2 n$