# Graph Theory: Homework #3

**Lin Hung Cheng B01902059**

# Problem 1

用gcd(m, n) 表示非負整數m和n的最大公因數。試證：如果m = an+r， 其中a 和r 是整數且0 $\leq r < n$，則gcd(m, n)=gcd(n, r)。並證明：這可以用來構造出一個用O(logm + log n)個除法的演算法

**1.**

令gcd(m, n) = g，則存在x, y 使 m = gx, n = gy, 且x, y互質；此時 r = g(x-ay)。
因為y 和 x-ay互質，所以 $gcd(n, r) = gcd(gy, g(x - ay)) = g$

*Proof.* 設 y 和 x-ay 有 gcd = $h > 1$, 則 $y = hj, x - ay = hk$。 此時 $x = h(aj + k), gcd(x, y) \geq h$, 與x, y互質矛盾

**2.**

$m = an + r$, $r < \frac{1}{2}m$ 當r = 0或n = 1時，可以找到gcd = n 每次執行除法可以使m或n至少減少一半，所以至多需要$\log_2 n + \log_2 m$次除法

*Proof.* 若 $r \leq \frac{1}{2}m$, 則 $a_n < \frac{1}{2}m$, 且 $n < \frac{1}{2}m$, 與 $n > r$ 矛盾，所以每次執行除法可以使m或n至少減少一半

# Problem 2

給定實數列$x_1, x_2, ..., x_n$， 要從中找出一段連續的若干項使其和為最大。試找一個比這個更有效率的演算法。

**Solution**

```
function FINDMAXSEQUENCESUM(array)
    s[0] ← 0
    minsum ← s[0]
    ans ← −∞
    for i = 1, n do
        s[i] ← s[i − 1] + array[i]
        minsum = min(minsum, s[i − 1])
        if s[i] − minsum > ans then
            ans = s[i] − minsum
        end if
    end for
    return ans
end function
```

Algorithm 1: findMaxSequenceSum

可以達到 $O(n)$ 的時間複雜度

## Problem 3

雙單連表列A的資料存在陣列DATA中、而其指標維陣列是NEXT 和PREV。(1)請寫出程式，將新資料x 加入A 中、放在i後面。(2) 請寫出程式，將A 中的一項i 刪除。

**1.**

**while** $A.value \neq i$ **do**
    $A \leftarrow (A \rightarrow next)$
**end while**
$n \leftarrow (A \rightarrow next)$
$(A \rightarrow next) \leftarrow x$
$(n \rightarrow prev) \leftarrow x$
$(x \rightarrow next) \leftarrow n$
$(x \rightarrow prev) \leftarrow A$

Algorithm 2: pushback

**2.**

**while** $A.value \neq i$ **do**
    $A \leftarrow (A \rightarrow next)$
**end while**
$p \leftarrow (A \rightarrow prev)$
$n \leftarrow (A \rightarrow next)$
$(p \rightarrow next) \leftarrow n$
$(n \rightarrow prev) \leftarrow p$
$deleteA$

Algorithm 3: deleteElement

## Problem 4

假設圖G = (V,E) 有n 個頂點v1, v2, . . . , vn 及m 條邊e1, e2, . . . , em， 定義G的相連矩陣（incidence matrix） M = [bij ] 是一個n × m 矩陣， 其中bij = 1, 若vi 和ej 相連；0, 若vi 和ej 不相連。試證： MMT = A + 度數矩陣

### Solution

$MM^T_{ij} = b_{i1}b_{j1} * b_{i2}b_{j2} * b_{i3}b_{j3}... * b_{im}b_{jm}$

若 i ≠ j，其意義為$v_i$和$v_j$是否有在$e_1, e_2, ...e_m$相連，即為$v_i$和$v_j$是否有相連，也就是相鄰矩陣的定義。

若 i = j，則$MM^T_{ij} = MM^T_{ii} = b_{11}b_{11} + b_{22}b_{22} + ...b_{nn}b_{nn} = b_{11} + b_{22} + ...b_{nn} = v_n$的度數。 $MM^T_{ij}$由此可知，等於相鄰矩陣和度數矩陣的和。

## Problem 5

假設圖G = (V,E) 中V = 1, 2, . . . , n 而E 則以m 個有序對表示。我們可以用連續空間（sequential space）來儲存G 中各點的鄰居，也就是在陣列DATA中先存N(1)、然後接續著存N(2)、. . . 依此類推直到放完N(n) 為止； 我們以bi 表示資料N(i) 在DATA 中的起始位置， 於是N(i) 所在的範圍就是從DATA[bi] 到DATA[bi+1 1]這一段（其中定義bn+1 為資料結束後的下一個位置）。bi 的值存放在陣列BEG 內。例如， 設n = 9， 而E = 31, 41, 59, 26, 53, 58, 97, 93, 23, 84，則此時DATA 和BEG的資料分別如下：試寫一

個演算法（或程式），能讓使用者輸入n 和E， 產生出對應的DATA 和BEG。

## Solution

```
function generateDATAandBEG(n, E)
    v ← array(list())
    DATA ← array(E.length * 2)
    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] ← N
            index ← index + 1
        end for
    end for
    return BEG, DATA
end function
```

Algorithm 4: getBEGandDATA