學號： R08922A20

姓名：洪筱慈

Problem 5

(1)

(a) False.

當 n = $k\pi$, for k = 0, 1, 2….,

$sin\ n = 0$, $n^{sin\ n} = 1$

但是 $\sqrt{n} = \sqrt{k\pi} > 1$, for k = 1, 2,...

所以 g(n)=$n^{sin\ n}$ 不能是 $\sqrt{n}$ 的 upper bound。

(b) True.

1.證明 if $f(n) = O(g(n))$, than $log(f(n)) = O(log(g(n)))$

已知找得到 n_0 ，使得 $0 \le f(n) \le c\,g(n)$, for all $n \ge n_0$ 。

兩邊同取 log: $log(f(n)) \le log(c\,g(n))$

展開: $log(f(n)) \le log(c) + log(g(n))$

欲證 $log(c) + log(g(n)) \le c_1 log(g(n))$, for all $n \ge n_1$ and $n_1 \ge n_0$

把 $log(g(n))$ 項整理在同一邊：

$$log(c) \le (c_1 - 1)log(g(n))$$

$$\frac{log(c)}{log(g(n))} \le (c_1 - 1)$$
$$\frac{log(c)}{log(g(n))} + 1 \le c_1$$

因為 $g(n)$ 為正且遞增，log(g(n)) 增加的趨勢比 log(c) 快，因此必能夠找到一個 c_1 ，對所有 $n \ge n_1$ and $n_1 \ge n_0$ ，使得此式成立。

2. 證明 if $f(n) = \Omega(g(n))$, than $log(f(n)) = \Omega(log(g(n)))$

已知找得到 n_2 ，使得 $0 \le c_2 g(n) \le f(n)$, for all $n \ge n_2$

與 1. 步驟相同，先兩邊同取 log 後再展開，可得欲證可以找到一組 c_3 , n_3 ，使得

$$c_3 log(g(n)) \le log(c_2) + log(g(n)) \quad \text{, for all } n \ge n_3,\ n_3 \ge n_2$$

把 $log(g(n))$ 項整理在同一邊：

$$c_3 \le \frac{log(c_2)}{log(g(n_3))} + 1$$

當 n_3 趨近於無限大，則 $\frac{log(c_2)}{log(g(n_3))}$ 趨近於零，此時是右邊最小的時候，故選取 c_3 =1 ，對所有 $n \ge n_3$ and $n_3 \ge n_2$ ，使得此式成立。

由1. 和 2. 可知， if $f(n) = \Theta(g(n))$, than $log(f(n)) = \Theta(log(g(n)))$ 成立。

Reference:

https://www3.cs.stonybrook.edu/~rob/teaching/cse373-fa15/sol2.pdf

(c) False.

根據 big O 的定義， $f_1(n) \le c_1 g_1(n)$, $f_2(n) \le c_2 g_2(n)$

可推得式一： $f_1(f_2(n)) \le c_1 g_1(c_2 g_2(n))$

若 $f_1 \circ f_2(n) = O(g_1 \circ g_2(n))$ 要成立，代表式二 $f_1 \circ f_2(n) \le c_3 g_1(g_2(n))$ 要成立。

然而式一的 c_2 不能提出來，因此式二不成立。

(d) True.

1. 證明 $(n+a)^b = O(n^b)$

   欲證存在 c, n_0 , $(n+a)^b \le cn^b$, for all $n \ge n_0$

   將 $(n+a)^b$ 用二項式展開：

   $n^b + bn^{b-1}a + \ldots + bna^{b-1} + a^b \le cn^b$

   $bn^{b-1}a + \ldots + bna^{b-1} + a^b \le (c-1)n^b$

   $\frac{bn^{b-1}a + \ldots + bna^{b-1} + a^b}{n^b} \le (c-1)$

   $\frac{bn^{b-1}a + \ldots + bna^{b-1} + a^b}{n^b} + 1 \le c$

   由於分母的次方數比分子最大項還要大，因此分母的增長速度比分子快，隨著 n 越來越大，左邊項只會越來越小。

   選取 $n_0 = 1$ ， c = $ba + \ldots + ba^{b-1} + a^b + 1$ ，則 $(n+a)^b \le cn^b$ holds for all $n \ge n_0$ 。

2. 證明 $(n+a)^b = \Omega(n^b)$

   欲證存在 c_2 , n_2 , $c_2 n^b \le (n+a)^b$, for all $n \ge n_2$

   同樣二項式展開

   $cn^b \le n^b + bn^{b-1}a + \ldots + bna^{b-1} + a^b$

   經過類似的整理過程後可得

   $$c_2 \le \frac{bn^{b-1}a + \ldots + bna^{b-1} + a^b}{n^b} + 1$$

   右邊最小的情況是當 n 趨近於無限大，則 $\frac{bn^{b-1}a + \ldots + bna^{b-1} + a^b}{n^b}$ 趨近於零，

   則選 $n_2 = 1$ ， $c_2 = 1$ ，則 $c_2 n^b \le (n+a)^b$, holds for all $n \ge n_2$ 。

由 1. 2. 可知， $(n+a)^b = \Theta(n^b)$ 。

(2)

(a) ANS: $\Theta(\frac{n}{logn})$

1. 證明下界：

考慮 $T_1(n) = T_1(n-1) + \frac{1}{logn}$ 的情況，$T_1(n-1)$ 和 $T_1(n)$ 相差 $\frac{1}{logn}$，$T_1(1)$ 要累加到 $T_1(n)$ 需要 n 次，故 $T_1(n) = O(\frac{n}{logn})$

此為下界：$T(n) = T(n-127) + \frac{127}{logn}$，$T(n) = \Omega(\frac{n}{logn})$。

2. 證明上界：

令 c = 127，並假設 n 可被 127 整除。

$$T(n) = T(n-c) + \frac{c}{logn}$$

$$= \frac{c}{logn} + \frac{c}{log(n-c)} + \ldots + \frac{c}{logc}$$

因為 c 是常數，先令 c=1，簡化問題，則上式會變成

$$\frac{1}{logn} + \frac{1}{log(n-1)} + \ldots + \frac{1}{log2} = \sum_{i=2}^{n} \frac{1}{logi} \leq \int_{2}^{n} \frac{dx}{logx}$$

設 logx = t，$x = e^t$，$\frac{1}{x} = dt$，$dx = xdt$，帶入上式，可得 $\int_{1}^{logn} \frac{e^t dt}{t}$

而 $\int_{1}^{logn} \frac{e^t dt}{t} = O(\frac{n}{logn})$，故上界為 $O(\frac{n}{logn})$。

根據 1. 2.，故 T(n)=$\Theta(\frac{n}{logn})$

Reference:

https://stackoverflow.com/questions/64376765/solving-recursive-time-function-tn-tn-127127-logn

(b) ANS: $\Theta(nlogn)$

猜測 $T(n) = O(nlogn)$，則期望找到 $c,\ n_0$ 使得 $T(n) \le cnlogn$ holds for all $n \ge n_0$

則原式 $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + nlogn$

$$\le c\frac{n}{2}log\frac{n}{2} + c\frac{n}{4}log\frac{n}{4} + c\frac{n}{8}log\frac{n}{8} + nlogn$$
$$\le \frac{cn}{8}(7logn - 11) + nlogn$$
$$= \frac{7c}{8}nlogn - \frac{11cn}{8} + nlogn$$
$$= cnlogn - (\frac{c}{8}nlogn + \frac{11cn}{8} - nlogn)$$

為了使 $T(n) \le cnlogn$ 成立，則需選取 c 使得 $\frac{c}{8}nlogn + \frac{11cn}{8} - nlogn \ge 0$

若 $n \ge 2$，則 $\frac{c}{8}logn + \frac{11c}{8} - logn \ge 0$

$$\frac{c}{8}(logn + 11) \ge logn$$
$$c \ge \frac{8logn}{logn + 11}$$

選取 c = 8，因為右邊分母 logn + 11 成長速度必定比 logn 快，隨著 n 變大，右邊只會越來越小。而最大值為 8。

選取 c = 8, n_0 =2，使得 $T(n) \le cnlogn$ holds for all $n \ge n_0$。

(c) ANS: $\Theta(n^2)$

根據 master theorem, 先看 $n^{log_b{}^a} = n^2$， 成長趨勢比 f(n)= $nlog\,n$ 還快，因此屬於 case 1，則 T(n) = $\Theta(n^{log_b{}^a}) = \Theta(n^2)$ 。

(d) ANS: $\Theta(nloglogn)$

1. 兩邊同除以 n :

$$\frac{T(n)}{n} = \frac{1}{\sqrt{n}}T(\sqrt{n}) + 1$$

令 $n = 2^{2^k}$， $\sqrt{n} = 2^{2^{k-1}}$

代入原式：

$$\frac{T(2^{2^k})}{2^{2^k}} = \frac{T(2^{2^{k-1}})}{2^{2^{k-1}}} + 1$$

再令 $a_k = \frac{T(2^{2^k})}{2^{2^k}}$， 則上式會替換成 $a_k = a_{k-1} + 1$， 則 $a_k = \Theta(k)$

而 $T(2^{2^k}) = a_k \times 2^{2^k}$

故 $T(n) = na_k = \Theta(nk)$

因為 $n = 2^{2^k}$，所以 $k = loglogn$，故 $T(n) = \Theta(nloglogn)$ 。

Reference

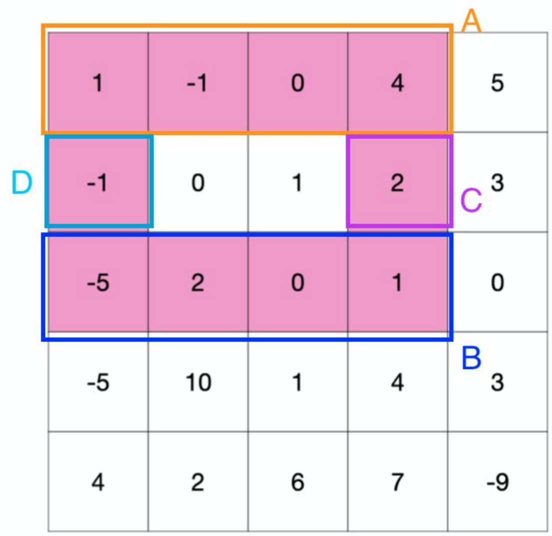
大碩 林立宇 演算法 課本

討論夥伴： 余政倫, 李建德, 洪嘉敏, 王致傑, Ruby Cheng, Tommy Chiang

Problem 6

(1)

a. 先算出每一橫列的 prefix sum P, P[i][j]表示第 i 橫列的 prefix sum，由左往右第 j 個位置的值。再算出每一個 column 的 prefix sum Q, Q[j][i]表示第 j 個 column 的 prefix sum，由上往下第 i 個位置的值。



b. 每個 rectangle 都可以拆成四個小橫列，以例圖來說，可以切成 A, B, C, D 四塊根據題目所給的 a, b，設 a 的座標是 (a_i, a_j) 代表 a 位在第 i 列橫排，第 j 欄；而 b 的座標是 (b_i, b_j)，則：

A 的 weight = P[a_i][b_j] - P[a_i][a_j - 1]
B 的 weight = P[b_i][b_j] - P[b_i][a_j - 1]
C 的 weight = Q[b_j][b_i - 1] - Q[b_j][a_i]
D 的 weight = Q[a_j][b_i - 1] - Q[a_j][a_i]
則這一個以 a, b 構成的 rectangle 的 weight = A + B + C + D.

Time Complexity Analysis:
(1) 建 P 和 Q ： $O(n^2 + n^2)$
(2) 計算每一個 k 的 weight: O(1)
(3) 共有 k 個 rectangle 要計算： $O(k)$
故整體時間複雜度 = $O(n^2 + k)$。

(2)

一個由 a, b 構成的 rectangle， 它的 perimeter:

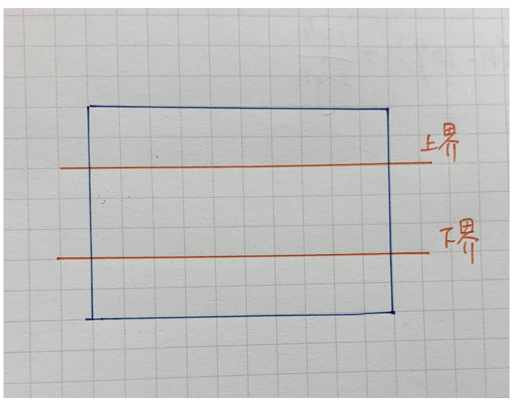
L = (b_i - a_i + 1) x 2 + (b_j - a_j - 1) x 2

根據這個目標，窮舉所有 a 和 b 的可能配對，並算出這些可能配對的 weight， 找出最大值。

Time Complexity Analysis:
任選出 a : n^2， 任選出 b: n^2， 則窮舉出所有可能 a 和 b 的可能配對的時間複雜度= O($n^2 \times n^2$) = O(n^4)
算出所有的 weight: O($n^2 + k$)
故整體時間複雜度 = O($n^4 + n^2 + k$) = O(n^4)。

(3)

1. 窮舉選取上界與下界，如下圖所示：

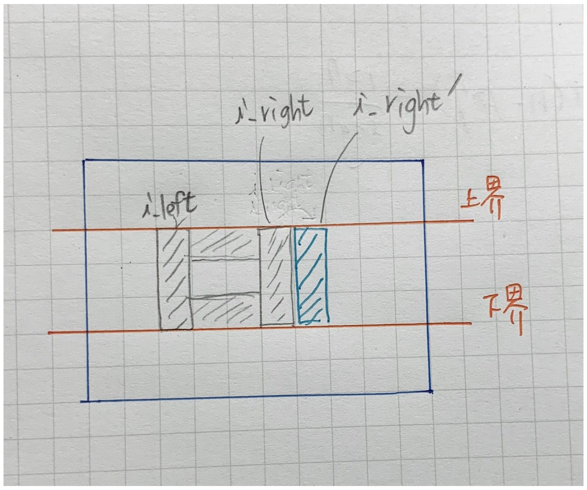


在上下界所夾的區域中，從最左邊開始，尋找最佳的左界與右界，使得形成的 rectangle 有最大的 weight。

尋找最佳左界與右界的方法：
1. 用兩個 index i_right, i_left， 分別代表右界跟左界。
2. 只要有右界跟左界，就可以用 (1) 中的方法得到此個 rectangle 的 weight 總和。
3. 另一個參數： max 紀錄目前能夠得到的最大 weight，以及該 rectangle 的 i_right, i_left 。

Reference:
討論夥伴： 余政倫, 李建德, 洪嘉敏, 王致傑, Ruby Cheng, Tommy Chiang

4. 一開始 i_right, i_left 都在最左邊，當 i_right 往右移動一格到 i_right'，確認此時形成的 rectangle weight 是否大於 max weight，如果有，更新 max weight。



5. 確認 i_left：目前的 i_left 會是當前的最佳左界，而當 i_right 往右一格到 i_right' 的時候，原本 i_right 的位置就成了 i_left 的新選擇。因此，比較 i_left 跳到 i_right 會不會比目前的 weight 大，如果有，就跳，並且更新 max weight。
6. 直到 i_right 走到最右邊到底結束。選出所有上下界中得到的最大 max weight，即為所求。

Time Complexity Analysis:
窮舉上下界： $O(n^2)$
每一個上下界組合都需要掃過一次： $O(n)$
故整體複雜度：$O(n^3)$。

(4)

延續 (3) 的作法，但同時紀錄目前 rectangle 的 perimeter，在每次 i_right 往右一格，並且形成的 rectangle weight 會更大時，先確認會不會超過 L。若沒超過，一切照舊。如果有超過，就比較移動左界使 L 縮小（跳到原本的 i_right）的情況是否有更大的 weight。

Time Complexity Analysis:
窮舉上下界： $O(n^2)$
每一個上下界組合都需要掃過一次： $O(n)$
故整體複雜度：$O(n^3)$。