ADA HW 1

Problem 5

(1) Asymptotic Notations

(a)

$n >= n,\ n > n-1,\ n > n-2...$
$n^n\ = n*n*n*n*...*n(共n個)$
$n!\ = n*(n-1)*(n-2)...*1$
$\therefore$ for all x>0, there must be a $b = 1$ that $n^n * b \geq n!$
So $ln\ n! = O(ln\ n^n)$

(b)

$n^{ln\ c} = (e^{lnn})^{ln\ c}\ \ 又(e^{lnn})^{ln\ c} = (e^{lnc})^{ln\ n}$
$\because (e^{lnc})^{ln\ n} = c^{ln\ n}\ \ \therefore n^{ln\ c} = c^{ln\ n}$
因此我們必能找出一個$b = 1$使得
$c^{ln\ n} * b \geq n^{ln\ c} \wedge c^{ln\ n} * b \leq n^{ln\ c}$
所以$n^{ln\ c} = \Theta(c^{ln\ n})$

(c)

ref: 李長諺、石容居

Assume there exist a c that $\sqrt(n) \leq c * n^{sin\ n}$
However,when $n = k\pi\ (k \in N)$, $n^{sin\ n} = 1$, and $\sqrt(k\pi) \geq c$
$\therefore When\ k \geq 1,\ \sqrt{n} > n^{sin\ n}$
So $\sqrt{n} \neq O(n^{sin\ n})$

(d)

ref: 李長諺、石容居

$When\ n = e^k\ (k > 0),\ (ln\ n)^3 = k^3$

When $lim_{n \to \inf} \frac{f(n)}{g(n)} = 0$, it means that when $n \to \inf$, for all $\delta > 0$ making $|\frac{f(n)}{g(n)} - 0| < \delta$

Let $f(n)\ and\ \ g(n) > 0$, it can be transfered to $\frac{f(n)}{g(n)} < \delta$

So $f(n) < \delta * g(n)$, which means that $f(n) = o(g(n))$

And it's obviously to see that $\lim_{n \to \inf} \frac{k^3}{e^k} = 0$ (By L'hospital)

So $k^3 = o(e^k)$, which means that $(ln\ n)^3 = o(n)$

(2) Solve Recurrences

(a)

$T(n) = 2T(n-1) + 1$

$2T(n-1) = 2^2T(n-2) + 2$

.

.

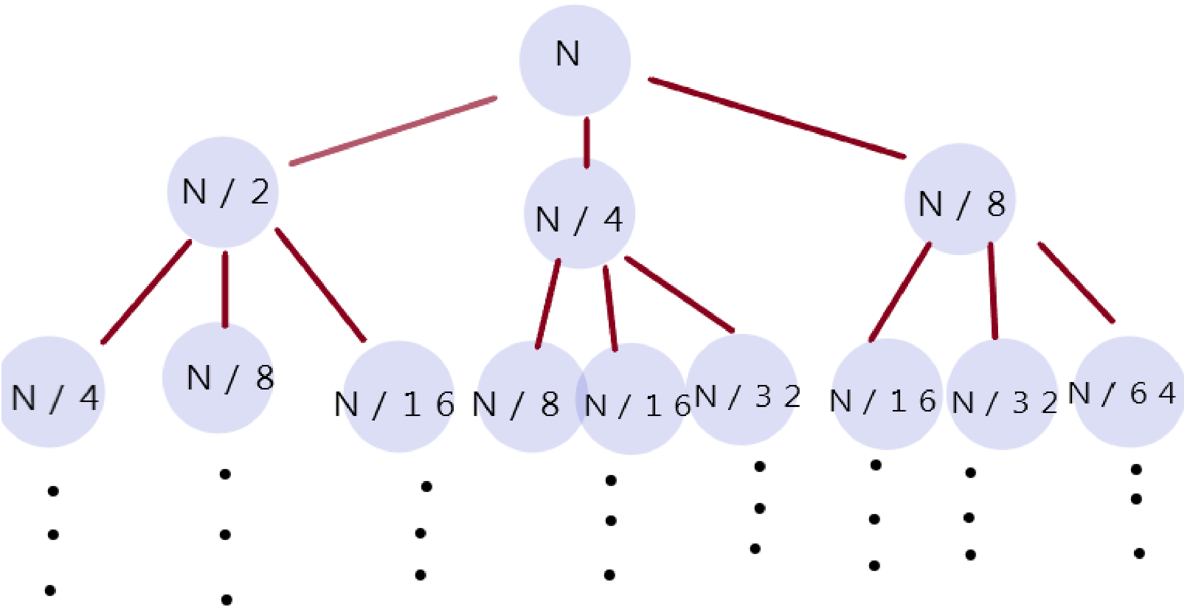
.

$2^{n-3}T(3) = 2^{n-2}T(2) + 2^{n-3}$

$\therefore\ T(n) = 2^{n-2} + 1 + 2 + \ldots + 2^{n-3} = 2^{n-1} - 1$

Obviously, we can say that $T(n) = \Theta(2^n)$

(b)



the picture denotes the "n" each time T(n) is called

At the first level, it takes $nlog\, n$ time

And the second level , it takes $\frac{7}{8}nlog\, n - k_1\ n$ ($k_1\ \in R$)

And the third level , it takes $\frac{49}{64}nlog\, n - k_2\ n$ ($k_2\ \in R$)

因為n會被$n\ log\ n$吃掉，所以對於每個node的下一層都會是這層複雜度的$\frac{7}{8}$倍，所以複雜度為一公比為$\frac{7}{8}$的數列

因此$T(n) = 8n\ log\ n$

可以得到$T(n) = \Theta(nlog\, n)$

(c)

令a=4, b=2 所以$n^{log_b a}\ = n^2$

而$nlog\, n = O(n^2)$

由master theorem 可以得到$T(n) = \Theta(n^2)$

(d)

ref: 熊育霆

Let $n = 2^k,\ T(2^k) = 2^{\frac{k}{2}} T(2^{\frac{k}{2}}) + 2^k$
Then let $g(k) = \frac{T(2^k)}{2^k}\ \therefore\ g(k) = g(\frac{k}{2}) + 1$
By Master theorum, $g(k) = \Theta(lg\ k)$
And $g(k) = \frac{T(2^k)}{2^k}$, so $T(2^k) = \Theta(2^k\ lg\ k)$
$\because n = 2^k\ \ \therefore T(n) = \Theta(n\ lg(\ lg\ n))$
$\therefore T(n)\ ! = \Theta(n^2)$

Problem 6

(1)
從B的最後面開始讀取，並先建立一個支援插入和查詢都為$log\ n$時間的資料結構
每次都先查詢當前b_i排第幾名，然後加上去後再插入b_i到資料結構裡
資料結構的部分可以用RB tree、BIT或其他平衡樹實作，只要利用旋轉讓他保持高度不超過$nlog\ n$就行

(2)
因為是從最後開始讀取，所以保證每次查詢$i < j$
而插入和查詢共n次、每次$log\ n$所以是$n\ log\ n$

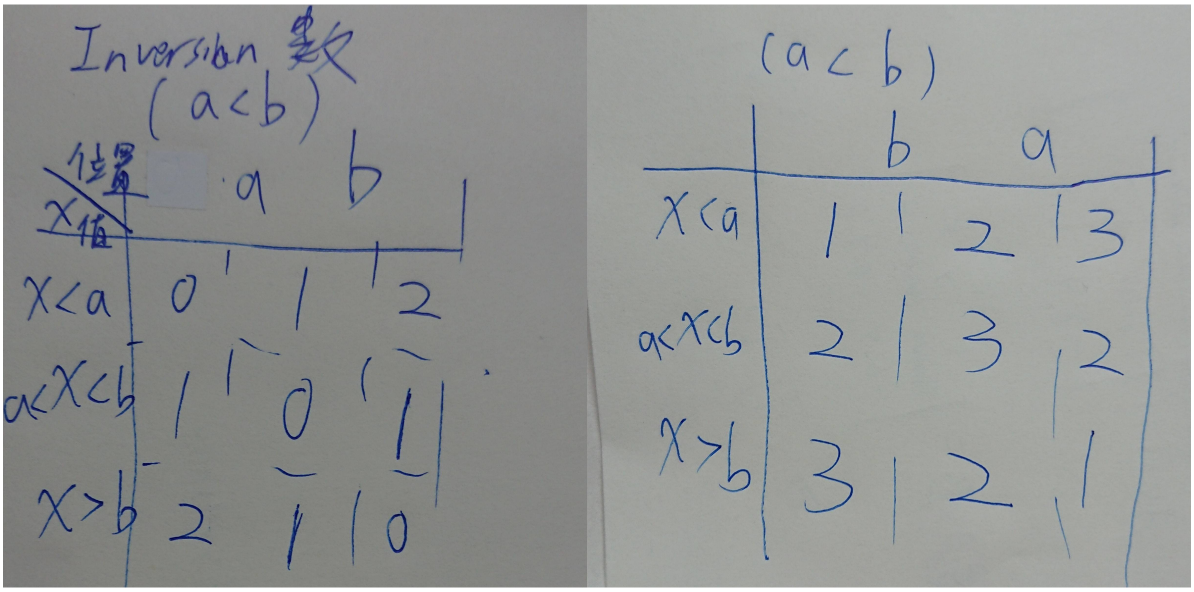
(3)
每個點inversion的數就是他們需要交換多少次才能使得整個數列保持遞增排序。
若有一數列$a_1,\ a_2$可以看出inversion個數就是bubble sort交換次數，接著令數列有n個時假設成立。
當數列有n+1個數時，令新增加的數字是數列的最後一個數。若要使數列達成遞增排序，則他必須和前面所有大於他的人交換位置，也就是bubble sort的交換次數，同時也是inversion數，故假設成立

(4)
ref: 李長諺、石容居

對每條線標記上他們要走到的終點，沒有指定的則從剩下的終點中從小到大標記。如此花費$O(n)$時間，且可以得到一個數列，接著對這個數列計算inversion個數，時間複雜度為$O(n\ log\ n)$就是所求

(5)
ref: 李長諺、石容居

因為inversion數就是bubble sort需要交換的次數，而每條橫線象徵交換它兩側直線的終點
由此可知inversion數就是橫線數，因此對於其他沒有指定的直線，我們必須找出使得他們inversion數最小的終點排序方式
先以兩未指定位置的數$a<b$，在數列上可以放置指定位置的數x的有a, b右邊、a, b之間還有a, b左邊



由圖中可以看出，不論$x>b$、$x<a$或$b>x>a$，x放在a左邊、a, b之間和b右邊這三個位置時，當$a,\ b$的排列方式為由小到大的時候，inversion數都會較反序少，因此當未指定數列的終點不是遞增排列時，必能找出至少一組點對使得交換他們後能讓inversion數變少。
因此未指定終點者為由小到大排列時inversion數最少

Problem 7

(1)
To maximize friendliness, I pick No.10, 1, 2 ,3, 4
Maximum : 16
(2)
ref: 李長諺、石容居

首先建立兩個大小為n的陣列M, L，分別記錄從第一個到第k個友善度中前綴的最小值，以及從最後一個到第k個友善度中後綴的最大值。至此時間和空間複雜度都是$O(n)$
接著開始遍訪。在遍訪友善度表K的第i個 $(i < n)$時，每次都先計算到當前位置的前綴和再減去M_i作為第一種可能(沒有超出第一個)。
接著再計算當前位置的前綴和及L_{n-i}的和作為第二種可能(超出第一個)，將這兩個數和之前算出的最大值做比較得出當前的最大值。再將全部n個都如此比較過一次就能得到答案，每個計算兩個數為$O(1)$，計算n次所以是$O(n)$
所以時間和空間複雜度都是$O(n)$
(3)
ref: 李長諺、石容居

先建立兩個2*n的dp矩陣，dp1[i][0]紀錄從第0個到第i個間以第i個為最後一個連續數列的最大值，dp1[i][1]紀錄從第0個到第i個間以第i個為最後一個且有1個空格的連續數列的最大值
可以得到轉移方程式為:
dp1[i][0] = max(dp1[i-1][0]+num[i], num[i])
dp1[i][1] = max(dp1[i-1][1]+num[i], dp1[i-1][0])
接著從尾巴一樣建立一個dp二維陣列dp2，紀錄從最後一個到第i個的最大值

首先找出dp1中的最大值M1，作為沒有橫跨第1個時的最大值
接著開始遍訪1~n，每次都讓答案去比較
ans = max(dp1[i][0]+dp2[i+1][0], dp1[i][1]+dp2[i+1][0], dp1[i][0]+dp2[i+1][1], ans, M1)
因為最多可空一個，所以遍訪到i時以i作為分界點查看兩側的最大值

為多少，可以是其中一側空一個或兩側都沒空

如此計算n次，每次需要O(1)所以共O(N)

而計算dp1和dp2也是O(N)

因為建立四個長度為N的陣列所以空間複雜度則是O(N)

因此空間和時間複雜度都為O(N)