No.

Date : :

HW-1 #5

(1) Prove / Disprove Asymptotic Notations

(a) $\ln n! = o(\ln n^n)$

(draft) exists b & $n_0 \geq 0$ s.t. $\ln n! \leq b * \ln n^n$ holds $\forall n \geq n_0$

$\rightarrow \ln n! \leq \ln n^{n \cdot b}$

$\rightarrow n! \leq n^{n \cdot b}$ let $b=1$, $n_0=1$ this inequality holds

By definition

$\ln n! \leq b * \ln n^n$ for all $n \geq n_0$, here $b=1$, $n_0=1$

So $\ln n! = o(\ln n^n)$ proved

(b) $n^{\ln c} = \theta(c^{\ln n})$

(when $n \geq 1$, $c^{\ln n} \leq n^{\ln c} \leq 2 * c^{\ln n}$ when $n \geq 1$

($\xrightarrow{\text{取ln}}$ $\ln n \ln c \leq \ln c \ln n \leq \ln c \cdot (2 \ln n)$))

By definition $b_1 c^{\ln n} \leq n^{\ln c} \leq b_2 c^{\ln n}$ for all $n \geq n_0$, $b_1=1$, $b_2=2$, $n_0=1$

So $n^{\ln c} = \theta(c^{\ln n})$ proved.

△ (c) $\sqrt{n} = O(n^{\sin n})$

Assume for a contradiction, exists b, n_0 s.t.

$\sqrt{n} \leq b * n^{\sin n}$ for all $n \geq n_0$

$\rightarrow n^{\frac{1}{2} - \sin n} \leq b$

$\because -1 \leq \sin n \leq 1 \xrightarrow{} n^{-\frac{1}{2}} \leq n^{\frac{1}{2} - \sin n} \leq b$

$\rightarrow n \leq b^{-2}$ contradict if we choose $n = \max\left(\frac{1}{b^2}, n_0\right) + 1$

(i.e. $n > \frac{1}{b^2}$ && $n \geq n_0$)

$n^{-\frac{1}{2}} \leq n^{\frac{1}{2} - \sin n} \leq 1$

$n \leq \frac{1}{b^2}$ (let $n = \max\left(\frac{1}{b^2}, n_0\right) + 1$

$x^{-\frac{1}{3}} > b^{\frac{2}{3}}$

$x > x^{\frac{2}{3}}$

No.

Date

(d) $(\ln n)^3 = o(n)$

(draft) exists b & $n_0 \geq 0$ s.t. $(\ln n)^3 < b\,n \quad \forall\, b \geq n_0$

$\rightarrow \ln n < b^{\frac{1}{3}} n^{\frac{1}{3}}$

We know that $\ln n = \int_1^n \frac{dx}{x} < \int_1^n \frac{dx}{x^{1-\frac{1}{3}}}$ if $n \geq 1$

$y = x^{\frac{1}{3}}$

$= \int_1^n x^{-\frac{2}{3}}\,dx = \frac{x^{\frac{1}{3}}}{\frac{1}{3}}\Big|_1^n = 3(n^{\frac{1}{3}} - 1)$

$< 3 \cdot n^{\frac{1}{3}}$

so we could have $b^{\frac{1}{3}} = 3 \rightarrow b = 3^3$

By definition

$(\ln n)^3 < b^* n$ for all $n \geq n_0$, with $b = 3^3$, $n_0 = 1$

So $(\ln n)^3 = o(n)$ proved.

No.
Date

$\frac{n}{2}+\frac{n}{4}+\frac{n}{8}=\frac{7n}{8}$ $\quad -\frac{n}{8}-\frac{n}{8}-\frac{3n}{8}$

5(2) Solve Recurrences : give the tight bound of the following recurrence equations

(a) $T(n)=2T(n-1)+1$ use brute force method

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

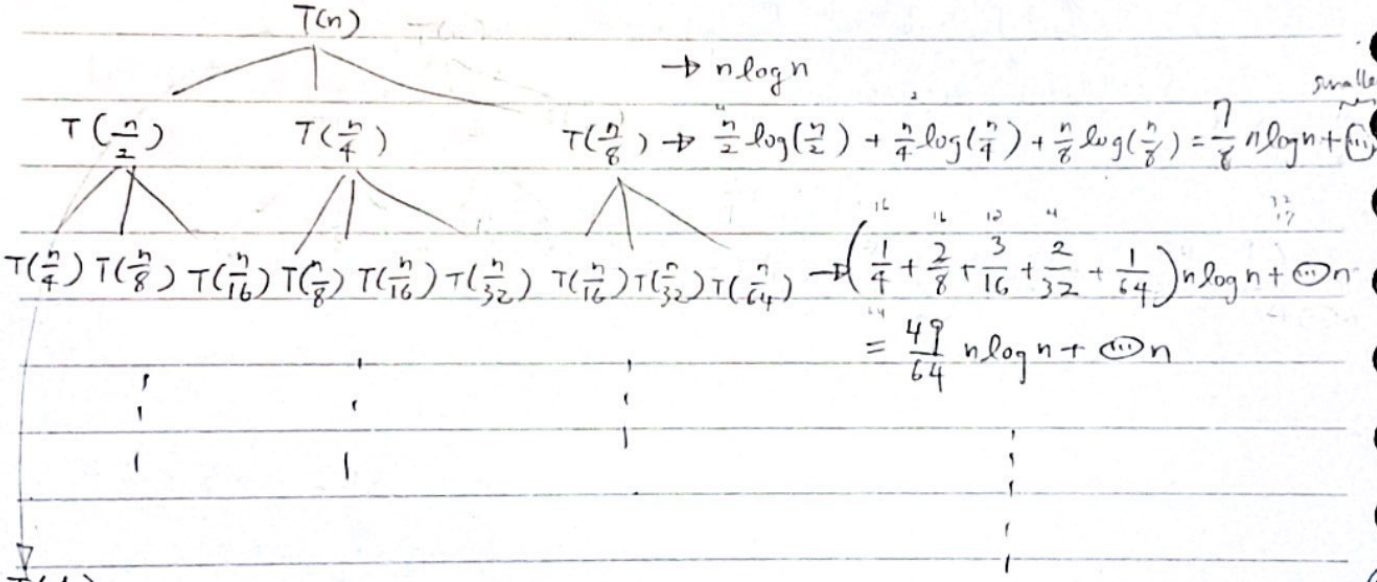
$=2(2T(n-2)+1)+1 = 2^2T(n-2)+1+2$

$=2^2[2T(n-3)+1]+1+2 = 2^3T(n-3)+1+2+2^2$

$= \cdots = 2^{n-1}T(1)+1+2+2^2+\cdots+2^{n-2}$

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

(b) $T(n)=T(\frac{n}{2})+T(\frac{n}{4})+T(\frac{n}{8})+n\log n$, use recursion tree method



$T(n) \le \left[1+\frac{7}{8}+\left(\frac{7}{8}\right)^2+\cdots\right] n\log n + \cdots n = \frac{1}{1-\frac{7}{8}} n\log n + \cdots n = \Theta(n\log n)$

$\therefore\ 8n\log n \le 8n\log n + a\cdot n \le (8+a)n\log n$

No.
Date

(c) $T(n) = 4\,T(\frac{n}{2}) + n\log n$, use master theorem method

(a over 4, b under 2)

$T(n)$ follows the form of recurrence relation: $T(n) = \begin{cases} o(1) & \text{if } n \le 1 \\ a \cdot T(\frac{n}{b}) + f(n) & \text{if } n > 1. \end{cases}$

, and we have $a=4$, $b=2$, $f(n) = n\log n = o(n^{\log_2 4})$ ($n\log n$ 小於 n^2)

(its obvious that $n\log n = o(n^2)$ ∵ $n\log n < c \cdot n^2$ if $n \ge n_0$ if we chose $c=1$, $n=1$)

By Master Theorem, $T(n) = \theta(n^{\log_b a})$

$= \theta(n^2)$

$n = 2^k$

(d) $T(n) = \sqrt{n}\,T(\sqrt{n}) + n$, use variable transformation method.

let $k = \log n \rightarrow T(2^k) = 2^{\frac{k}{2}}\,T(2^{\frac{k}{2}}) + 2^k$

let $S(k) = T(2^k) \rightarrow S(k) = 2^{\frac{k}{2}}\,S(\frac{k}{2}) + 2^k$, By master theorem, we can solve $S(k)$.

w/ $a = 2^{\frac{k}{2}}$, $b = 2$ → $\theta(k^{\frac{k}{2}})$

$\rightarrow k^{\log_b a} = k^{\frac{k}{2}} > 2^k$

$\sqrt{k}^k$

So, $T(n) = \theta(\log n^{\frac{\log n}{2}})$

(1) - text 詳解版本

令 function CountInversionPairs(l, r) 回傳在原 sequence B, index l~r 之間可以找到的 pair 數量，用 divide & conquer 的方式 recursively call 直到終止條件回傳之前，確保 l~r 之間已經由小到大 sort 好。

- Base case (l == r) : return 0 ~O(1)

- Recursive case :

① Divide: 從 middle point m = floor((l+r)/2) 切成左邊和右邊，call them recursively

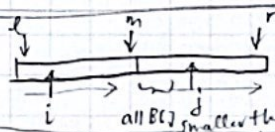
加總 result = CountInversionPairs(l, m) + CountInversionPairs(m+1, r)
左邊 T(⌈n/2⌉) + 右邊 T(⌊n/2⌋)

② Conquer: 計算左半邊每大數，右半邊每小數的 pair 的數量，加總到 result 中。 ~O(n) — (*)

③ Combine: ⓐ 同時將左半邊和右半邊 merge sort 由小到大重回得。

- Return result. (sequence is updated as well) ~O(n) — (*)

(*): Efficient way to conquer:



在 ②, ③ ，可以利用 "two pointers" linearly 完成，因左邊和右邊皆各自由小到大排

- 一個 pointer 在左半邊從 0 → m 代表現在"大數"，另一個 pointer 在右邊從 m+1 → r 代表該位子以左的數都小於目前的大數，加總進 result

- 我們將右邊 pointer ++ 後，本來右邊 pointer 以左仍小於目前的大數，將右邊 pointer 繼續 ++ 找到新的"界線"，以此類推。

在 merge sort 時亦為相同原則，可以在上述 conquer 的過程中另外新增1個 pointer 指向新的排序 away 中現在新增的位子，每次更新任何 pointer 時，將更新之前該值插入排序 away，最後

~O(n)

(e) Explain why $O(N \log N)$

綜合以上, let $T(n)$ = CountInversionPairs(l, r) 的 time complexity, where $n = r - l + 1$.

則我們有:

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + O(n) & \text{if } n \geq 2 \end{cases}$$

proof by induction $T(n) \leq 2b \cdot n \log_2 n + a \cdot n$ when $T(n) = \begin{cases} a & , n=1 \\ T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + b \cdot n \end{cases}$

$n = 1$, trivial

$n > 1$: $T(n) \leq 2 \cdot b \lfloor \frac{n}{2} \rfloor \log_2 \lfloor \frac{n}{2} \rfloor + a \lfloor \frac{n}{2} \rfloor$ (by induction)

$+ 2b \lceil \frac{n}{2} \rceil \log_2 \lceil \frac{n}{2} \rceil + a \lceil \frac{n}{2} \rceil + b \cdot n$

$\lfloor \frac{n}{2} \rfloor < \lceil \frac{n}{2} \rceil < \frac{n}{\sqrt{2}}$

$\leq 2 \cdot b \lfloor \frac{n}{2} \rfloor \log_2 \frac{n}{\sqrt{2}} + a \lfloor \frac{n}{2} \rfloor$

$+ 2b \lceil \frac{n}{2} \rceil \log_2 \frac{n}{\sqrt{2}} + a \lceil \frac{n}{2} \rceil + bn$

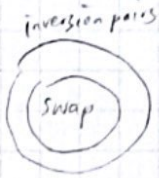
$= 2b \left(\log_2 n - \frac{1}{2} \right) n + a \cdot n + bn$

$= 2b \, n \log_2 n + a \cdot n$ ✱

$\therefore T(n) \sim O(n \log n)$

(3) Prove: num of exchanges (swapping) in bubble sort = num of inversion pairs 得

```
BubbleSort(S)
  for i = S.length-1 downto 0
      for j=0 to i
          if S[j] > S[j+1], swap them
```



① 在BubbleSort中，所有swap的兩數皆inversion pairs，且兩數只能swap一次 (往 相鄰)

<pf> 假設現在某 S[a] 和 S[a+1] swap，表示 S[a] > S[a+1] 得件符合，

則 S[a] 原本的 index 必小於 S[a+1] 原本的 index.

→因為若不是，S[a] 原本在 S[a+1] 的右邊，表示它們曾 swap 過，

表示 S[a+1] > S[a] → 矛盾.

∴ num of inversion pairs ≥ num of swapping

② 在BubbleSort中，所有'inversion pairs 都被swap 過

因為BubbleSort 完之後，原sequence 變成由小到大排列的 sequence，

且每次改變排序只能交換相鄰2數，

所以，對於任何 S[a]，所有在它右邊，且比它小的數 (index > a)，(i.e. 以a為大數的 inversion pairs)

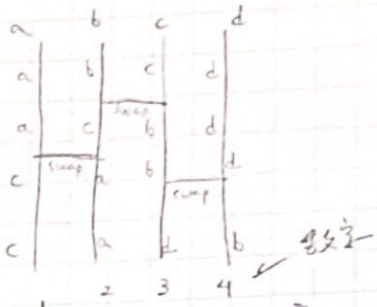
都至少要它交換一次才能排到它左邊，才能符合由小到大的排序.

同理，S[a] 左邊比它大的數亦是. — i.e. 以a為小數的 inversion pairs.

∴ num of swapping ≥ num of inversion pairs.

綜合①、② → num of swapping = num of inversion pairs. #

(4) Describe an O(NlogN) algorithm calculates minimum number of ghost leg when $|constraints| \leq N$



- 繪製 horizontal line 的方法：
  - 另一個 array 大小為玩家數量（垂直線數），其 index 由左至右對應到 ghost legs 的垂直線，將有 constraints 的位子對應其 index 填進 array 中，其餘沒 constraints 的位子則由左至右將未 assigned 的目標數字由小到大填完。（目標數字；可用 array<bit> N 看很還沒assign，~O(N)）
  - 對上述 array 作 BubbleSort，每次有"swap"時，就在該兩發生交換的 index 對應到的垂直線之間畫水平線。每次都畫在上一次畫的區域之下。

- 計算繪製了多少 horizontal lines？

由上面的方法，我們得到畫水平線的數量 = 對 array S 做 BubbleSort 時 swap 的數量。由 (3) 小題的推導，該數字就是 inversion pairs 的數量，因此可以將 S 代近 (1) 小題的演算法。

time complexity : T(建立 array S) + O(NlogN)
~O(N)

= O(N) + O(NlogN) = O(NlogN)

(5) Proof the correctness

- 承第四題上半部的說明，因為若我們由每條路徑同時向下走，每次遇到水平線的兩鄰近玩家就會根據 bubblesort 的法則交換位子，因此走到最後，玩家由左自右的排列會符合其被 assign 的數字由小到大。

即修後 vertical line 的編號(也是由小到大 1, 2, 3, ...)

- 另外，將未constraints的位子，由左至右，由小到大 assigned 剩餘未指定的數字，可以保證整些algorithm 拿到最少 horizontal lines 的解。

回答那條就可以使數列中 inversion pairs 最少。

<pf> 若對那些由小排到大，非constraints 指定 assign 的剩餘的數字們任意交換2數 a, b (且 a<b)，則新增一個 inversion pair (a, b)，且原和 b 形成 pair 但和 a 形成 pair 的數，會和 a 形成 pair. (∵ a<b)

原和 a 形成 pair，非 b 的 pair 的數，仍是 a 的 pair (a newindex > a oldindex)

... a ... b

回答把由小到大的 assigned 的數字任意改動成其它形式

只會使 inversion pairs 不減反增，所以原由小到大排列它們是 inversion pairs 最少的方法。

#7. given the friendness f_i for $i=1,2,\dots N$

find continuous part s.t. total friendness is maximized

(1) given $f=[-3, 0, 6, 4, 0, -1, -2, 3, -3, 9]$ (1 2 ... 10) what's the maximum?

用 brute force (或小聰明的方法) 可以找到 maximum 是 i in 2~10，總和 = 16 #

(2) design an O(N) algorithm.

考慮 2 Case：(以下 subarray 的起點 l、終點 r 定義為起點→終點順時針方向)

Case 1: Subarray 在 [1, N] 之間的某個區間

Define A_i = 符合 Case 1，終點在 i 所能找到的最大 subarray，另其對應的起點 $= l_i$

$$\rightarrow A_i = \begin{cases} f_1 & i=1 \quad (l_1=1) \\ \max\{A_{i-1}+f_i,\ f_i\} & \end{cases}$$

if choose this $l_i = l_{i-1}$; if choose this $l_i = i$

Find $k = \underset{i\in[1,N]}{\text{argmax}}\ A_i$，則 Case 1 的 maximum subarray 即從 l_k 到 k，總和 A_k

Case2 Subaway 區間有經過 1 & N

Define $S_i = \sum_{j=1}^{i} f_j = \begin{cases} f_1 & \text{if } i=1 \\ f_i + S_{i-1} & \text{otherwise} \end{cases}$ (1到i的加總)

$R_i = \sum_{j=i}^{N} f_j = S_N - S_{i-1}$ (i到N的加總)

Define B_i = 從任何≥i的位子當起點，加總到N的最大總和，
令其對應的起點為 l_i

則 $B_i = \begin{cases} f_N & \text{if } i=N \\ \max\{B_{i+1}, R_i\} & \end{cases}$

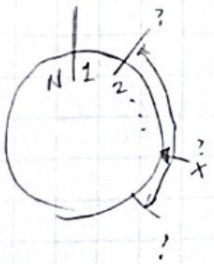
if choose $l_i = l_{i+1}$; if choose $l_i = i$

Find $k = \underset{i\in[1,N-1]}{\text{argmax}}\ S_i + B_{i+1}$ ，則 Case2 的 maximum subaway 即為 l_k 到 N，
總和為 $S_k + B_{k+1}$

比較 Case 1, Case 2，輸出總和比較大的 subaway 為答案。

(3) Can skip at most one in the contiguous part

Case1 Subaway 在 [1, N] 之間的某個區間，且有捨掉一個值



承上題，利用已有的 A_i、l_i

Define D_i = 符合 case 1，終點在 i，且 subaway 中有捨掉一個值，總和最大的 subaway，其對應的起點為 m_i，捨掉的洞為 h_i
$h_i \in [m_i, i]$

則 $D_i = \begin{cases} 0 & \text{if } i=1 \to h_1=1,\ m_1=1 \\ \max\{D_{i-1}+f_i,\ A_{i-1}\} & \end{cases}$

if choose: $h_i = h_{i-1}$, $m_i = m_{i-1}$; if choose $h_i = i$, $m_i = l_{i-1}$

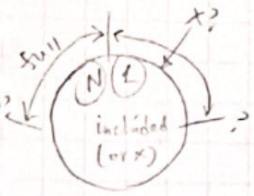
Find $k = \underset{i\in[1,N]}{\text{argmax}}\ D_i$ ，則 Case1 的 maximum subaway 即 m_k 到 k，洞在 h_k，
總和為 D_k

Case 2 Subaway 區間有經過 1 & N，且有找零一個值

Case 2-1 →

Define E_i = 符合 case 2 且終點在 i，且 subaway 中有找一個值在 1~終點之間

令對應的起點 = l_i (exactly the same as (2) case 2)，洞在 h_i



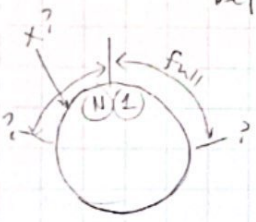
$$E_i = \begin{cases} B_2 & \text{if } i=1,\ h_1 = 1 \\ \max\{ B_{i+1} + E_{i-1} + f_i \ ,\ B_{i+1} + S_{i-1} \} & \end{cases}$$

if choose $h_i = h_{i-1}$; if choose $h_i = i$

Case 2-2 →

Define F_i = 以 1 為起點，加總到任何 ≤ i 的位子為終點的最大結果

令其對應終點 = r_i



$$F_i = \begin{cases} f_1 & \text{if } i=1,\ r_1 = 1 \\ \max\{ F_{i-1},\ S_{i-1} + f_i \} & \end{cases}$$

if choose $r_i = r_{i-1}$; if choose $r_i = i$

Define G_i = 符合 case 2 且起點在 i，subaway 中有找一個值(洞)至起點 i~N 之間

令其對應的終點為 r_i (exactly the same as above)，洞在 h_i

$$G_i = \begin{cases} F_{N-1} & \text{if } i=N,\ h_N = N \\ \max\{ F_{i-1} + G_{i+1} + f_i \ ,\ R_{i+1} + S_{i-1} \} & \end{cases}$$

if choose $h_i = h_{i+1}$; if choose $h_i = i$

Find $k_1 = \underset{i \in [1,N-1]}{\text{argmax}}\ E_i$ 、case 2 且洞在 1~終點之間的 maximum subaway 為 $k_1 \sim r_{k_1}$，且洞在 h_{k_1}

$k_2 = \underset{i \in [1,N-1]}{\text{argmax}}\ G_i$ case 2 且洞在起點~1 之間 … 為 $k_2 \sim r_{k_2}$，且洞在 h_{k_2}

比較 case 2 的 2 個結果、case 1 的答案和 (2) 不是 0 的結果，找出總和最大的答案 — P.S. 題目設定至少 1 個要是，若答案皆為 0，則要 output 第個元最大的答案

以上建立的每個 recurrence relation 都可以用 dynamic programming 來實作，至 bottom-up 的過程每個計算都可以用之前已算好的代入，故 time complexity ~ $O(n)$ 而找最大值也是 $O(n)$ 問題，∴ 全部是 $O(n)$