Homework1

B08902149 資工三 徐晨祐

January 19, 2022

- Problem 5 : I discussed this problem with 蘇簡宏

1. (a) $\because \quad \ln(n!)$
$= \ln(n) + \ln(n-1) + \ln(n-2) + \cdots + \ln(2) + \ln(1)$
$\leq \ln(n) + \ln(n) + \ln(n) + \cdots + \ln(n) + \ln(n)$
$= \ln(n^n)$
$\therefore \quad \ln(n!) \leq \ln(n^n) \Rightarrow \ln(n!) = O(\ln n^n)$

(b) $\because \quad \ln(n^{\ln c}) = \ln c \cdot \ln n = \ln(c^{\ln n})$
$\therefore \quad n^{\ln c} = c^{\ln n}$
$\because \quad n^{\ln c} \leq c^{\ln n} \Rightarrow n^{\ln c} = O(c^{\ln n})$
$n^{\ln c} \geq c^{\ln n} \Rightarrow n^{\ln c} = \Omega(c^{\ln n})$
$\therefore \quad n^{\ln c} = \Theta(c^{\ln n})$

(c) Assume $\forall c, n_0 \in \mathbb{N}$ such that $\sqrt{n} \leq c \cdot n^{\sin n}$ when $n \geq n_0$.

$$\begin{aligned} &\because \ \sqrt{n} \leq c \cdot n^{\sin n} \\ &\Rightarrow 1 \leq c \cdot n^{\sin n - \frac{1}{2}} \\ &\Rightarrow \frac{1}{c} \leq n^{\sin n - \frac{1}{2}} \end{aligned}$$

There must exist a positive number n_1 such that $n_1 > max(c, n_0)$
and $\sin n_1 \leq -\frac{1}{2}$ such that $\frac{1}{c} > n^{\sin n - \frac{1}{2}}$ when $n = n_1$. So it's a contradiction.
The statement is incorrect.

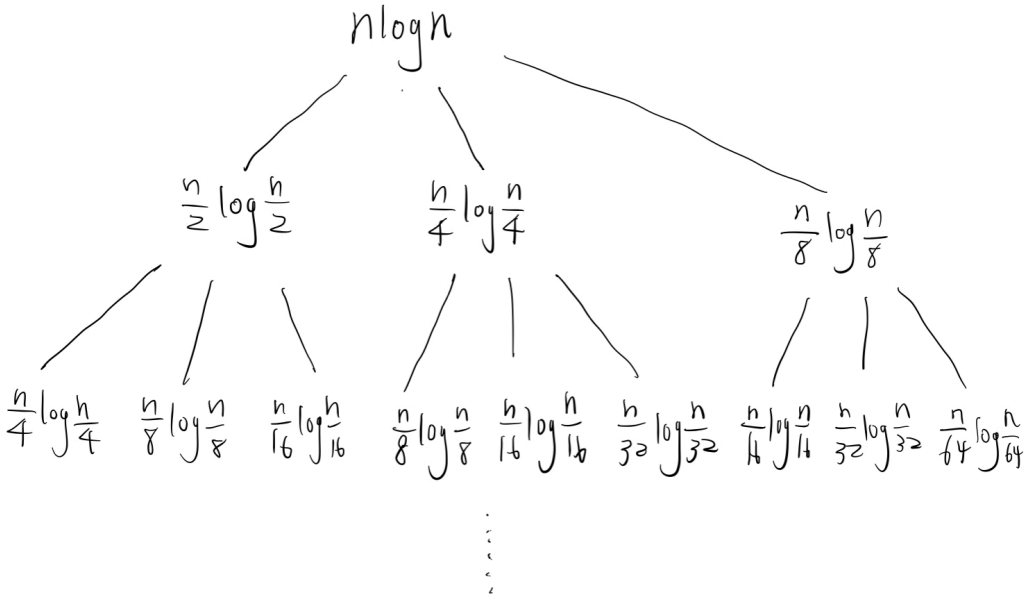
(d)

$$\because \lim_{n\to\infty} \frac{(\ln n)^3}{n} = \lim_{n\to\infty} \frac{6}{n} = 0 \ (By\ L'Hosipital\ rule)$$

$\therefore$ By definition of limit, there exists two positive number ϵ, n_0 such that for all $n \geq n_0$, $\left|\frac{(\ln n)^3}{n}\right| < \epsilon$. $\Rightarrow (\ln n)^3 = o(n)$.
The statement is correct.

2. (a) $T(n)$
$= 2^1 \cdot T(n-1) + 2^0$
$= 2^2 \cdot T(n-2) + (2^0 + 2^1)$
$= 2^3 \cdot T(n-3) + (2^0 + 2^1 + 2^2)$
$= 2^4 \cdot T(n-4) + (2^0 + 2^1 + 2^2 + 2^3)$
$= \cdots$
$= 2^{n-1} \cdot T(1) + (2^0 + 2^1 + 2^2 + \cdots + 2^{n-2})$
$= 2^n - 1$
$\Rightarrow T(n) = \Theta(2^n)$

(b)



$$T(n) = n\log n + (\frac{n}{2}\log\frac{n}{2} + \frac{n}{4}\log\frac{n}{4} + \frac{n}{8}\log\frac{n}{8})$$
$$+ (\frac{n}{4}\log\frac{n}{4} + \frac{n}{8}\log\frac{n}{8} + \frac{n}{16}\log\frac{n}{16} + \frac{n}{8}\log\frac{n}{8} + \frac{n}{16}\log\frac{n}{16} + \frac{n}{32}\log\frac{n}{32}$$
$$+ \frac{n}{16}\log\frac{n}{16} + \frac{n}{32}\log\frac{n}{32} + \frac{n}{64}\log\frac{n}{64}) + \cdots$$
$$= ((\frac{7}{8})^0 n\log n - C_0 \cdot n) + ((\frac{7}{8})^1 n\log n - C_1 \cdot n) + ((\frac{7}{8})^2 n\log n - C_2 \cdot n) + \cdots$$

$\because \quad T(n) \le 8n\log n - C \cdot n \le 8n\log n$
$T(n) \ge n\log n$
$\therefore \quad T(n) = \Theta(n\log n)$

(c) *Let* $f(n) = n\log(n)$, $a = 4$ *and* $b = 2$
$\Rightarrow T(n) = a \cdot T(\frac{n}{b}) + f(n)$
$\because f(n) = O(n^{\log_2 4 - \epsilon}) = O(n^{2-\epsilon})$ *for some* $\epsilon > 0$
$\therefore T(n) = \Theta(n^2)$

(d) $Let\ n = 2^k,\ T(n) = T(2^k) = 2^{\frac{k}{2}} T(2^{\frac{k}{2}}) + 2^k$

$Let\ S(k) = \dfrac{T(2^k)}{2^k},\ S(k) = S(k/2) + 1$

$By\ Master\ Theorem,\ let \begin{cases} a = 1 \\ b = 2 \\ f(n) = 1 \end{cases}$

$S(k) = \Theta(\log(k)) \Rightarrow T(2^k) = c \cdot 2^k \log(k) \Rightarrow T(n) = \Theta(n(\log(\log(n))))$

- Problem 6 : I discussed this problem with 胡材溢

(1) 我們可以先在函式外新增一個變數叫 **count**，並使用 merge sort 去排序該數列，但在 combine 時只要 $l_i > r_j$ 時就對 **count** 加 $size_l - i$（設左數列的第 i 數爲 l_i，右數列第 j 數爲 r_j，index 從 0 開始，$size_l$ 爲左數列大小），最後 **count** 內所存的值即是原數列的 inversion 數。

(2) Let the total time spent on calculating the number of inversions of an N-sized sequence is $T(N)$. We have:

$$T(N) = \ 2T(N/2) + c \cdot N, \ c > 0$$
$$\Rightarrow By\ Master\ Theorem,\ T(n) = O(N \log N)$$

(3) 由於 bubble sort 的交換規則是兩個相鄰的數比較後，若左數大於右數即交換。故每次交換都是計算到一個 inversion。因爲是透過兩兩交換來把每個數排列到正確的位置，所以每個逆序數對必會交換一次，且兩相鄰的數位置被交換過就不會再被交換了，所以交換的次數即爲 inversion 的數量。

(4) Let C_i be the i^{th} constraint. For each $1 \leq i \leq |constraints|$, there exists $1 \leq n, m \leq N$ such that C_i means "The n^{th} participant must get the m^{th} gift". We use an empty array with size $N + 1$, and let $array[m] = n$ for each $1 \leq i \leq |constraints|$. Next, we insert the indices of unconstrained people increasingly into the blank space of the array from $array[1]$ to $array[N]$. Lastly, we use the algorithm defined in subproblem(1) to get the number of the inversions of the array, and the number we get will be the minimum number of horizontal lines. By the proof in subproblem(2), we know the time complexity is $O(N \log N)$.

(5) Because adding a new horizontal line will only make the two adjacent numbers switch their positions. Let array $arr = \{1, 2, \cdots, N\}$. We can keep switching the positions of two adjacent numbers so that $arr[m] = n$ for each constraint that requires the n^{th} participant to get the m^{th} gift. The number of exchanging two adjacent numbers to get the status of arr equals to the number of exchanging in order to bubble-sort arr increasingly, and the number of exchanging equals to the number of the horizontal lines we need. The purpose of inserting the indices of unconstrained people increasingly is minimizing the number of inversions, that is to minimize the number of the horizontal lines. So the number of inversions we get from arr is exactly the minimum number of horizontal lines. The algorithm is correct.

- Problem7 : I discussed this problem with 胡材溢、蘇簡宏、洪郁凱
(7-2, 7-3 所有用到 dp 的部分，都會以一個 size 爲 N 的 array 去紀錄)

(1) The answer is 16.

(2) 我們可以用先對 f 陣列從 $i = 1$ 到 $i = N$ 做 dp 找出帶有最大 friendliness 總和的 subarray，Transition Equation 如下，其中 $F1(i)$ 是以 $f[i]$ 爲結尾之 subarray 的 friendliness 最大值。

$$F1(i) = \begin{cases} max(f[i],\ F1(i-1) + f[i]),\ 2 \le i \le N \\ f[1],\ i = 1 \end{cases}$$

接著再對 f 陣列從 $i = 1$ 到 $i = N$ 做 dp 找出帶有最小 friendliness 總和的 subarray，Transition Equation 如下，其中 $F2(i)$ 是以 $f[i]$ 爲結尾之 subarray 的 friendliness 最小值。

$$F2(i) = \begin{cases} min(f[i],\ F2(i-1) + f[i]),\ 2 \le i \le N \\ f[1],\ i = 1 \end{cases}$$

因爲每次算每個 $F1(i)$, $F2(i)$，都只需要考慮 $i-1$ 時所得到的值，所以以上的 dp 都只需要花 $O(N)$ 的時間複雜度即可，也只需要花 $O(N)$ 大小的 array 來記錄每個 $F1(i)$, $F2(i)$。接下來我們只需要花 $O(N)$ 的時間個別找到 $F1(i)$ 的最大值和 $F2(i)$ 的最小值，我們令 a, b 爲:

$$a = max(F1(1), F1(2), \cdots, F1(N))$$
$$b = \begin{cases} \infty,\ if\ min(F2(1), F2(2), \cdots, F2(N)) = \sum_{i=1}^{N} f[i] \\ min(F2(1), F2(2), \cdots, F2(N)),\ otherwise \end{cases}$$

其中 a 指的就是由 $f[1]$ 到 $f[N]$ 所有 subarray 的區間總和最大值，b 指的就是由 $f[1]$ 到 $f[N]$ 所有 subarray 的區間總和的最小值，但如果該連續區間是整條 f array 的話，b 就是 ∞。
最後，我們比較 a 和 $\sum_{i=1}^{N} f[i] - b$ 哪個大，哪個就是該 circular array 的區間總和極大值。如果我們得到的答案是 a，那就在剛才用來算 $F1(i)$ 的 dp array 中先找到 a 發生的地方，並持續往前 go through，每次都對 a 扣掉 $f[i]$ 直到 a 變爲 0，則所經過的所有 index 就是讓區間總和最大的 subarray。而如果答案是 $\sum_{i=1}^{N} f[i] - b$(此時 b 一定等於 $min(F2(1), F2(2), \cdots, F2(N))$)，那就在剛剛計算 $F2(i)$ 的 dp array 中先找到 b 發生的地方，並持續往前 go through，每次都對 b 減掉 $f[i]$ 直到 b 變成 0，而 $\{1, 2, \cdots, N\}$ 扣掉剛剛經過的 index，就會是讓區間總和最大的 subarray。以上計算 index 的方法也只需要從頭到尾掃過 dp array 一次即可，時間複雜度爲 $O(N)$。

(3) 我們首先計算不越過 $f[1]$ 和 $f[N]$ 分界線的所有最多只空一格的連續區間之最大總和值：
先對 $f[1]$ 到 $f[N]$ 做 dp，令 $F1(i)$ 爲以 $f[j]$ 爲起頭 $(j \le i)$ 且以 $f[i]$ 結尾的所有連續區間中，總和值最大者。其 Transition Equation 爲：

$$F1(i) = \begin{cases} max(F1(i-1) + f[i],\ f[i]),\ 2 \le i \le N \\ f[1],\ i = 1 \end{cases}$$

接著再對 $f[N]$ 到 $f[1]$ 反向做 dp，令 $F2(i)$ 爲以 $f[j]$ 爲起頭 $(j \ge i)$ 且以 $f[i]$ 結尾的所有連續區間中，總和值最大者。其 Transition Equation 爲：

$$F2(i) = \begin{cases} max(F2(i+1) + f[i],\ f[i]),\ 1 \le i \le N-1 \\ f[N],\ i = N \end{cases}$$

最後令 $F3(i)$ 爲跳過 $f[i]$ 的連續區間中總和值最大者：

$$F3(i) = \begin{cases} F2(2), & i = 1 \\ F1(i-1) + F2(i+1), & 2 \le i \le N-1 \\ F1(N-1), & i = N \end{cases}$$

$F1,\ F2$ 的計算每次都是 $O(1)$ 時間做狀態轉移，所以皆是 $O(N)$ 時間 dp 完成，$F3$ 每個值也只需要 $O(1)$ 時間計算，因爲 $F1,\ F2$ 都有用陣列紀錄，故到目前爲止只花 $O(N)$ 時間計算。
接下來我們去計算 $F1(1), F1(2), \cdots, F1(N), F3(1), F3(2), \cdots, F3(N)$ 中最大的值，可以得到不越過 $f[1]$ 和 $f[N]$ 分界線的所有最多只空一格的連續區間之最大總和值，令其爲 a。
接著，我們要計算有越過 $f[1]$ 和 $f[N]$ 分界線的所有最多只空一格的連續區間之最大總和值：
我們令 $F4(i)$ 爲以 $f[1]$ 開頭且以 $f[j](j \le i)$ 爲結尾的所有恰空一格之連續區間的最大總和值，計算方法如下：

```
sum = 0;
temp_max = -inf;
toDelete = inf;
for i = 1 to N :
    sum += f[i];
    toDelete = min(f[i], toDelete);
    F4[i] = max(temp_max, sum-toDelete);
```

再令 $F5(i)$ 爲以 $f[N]$ 爲開頭且以 $f[j](j \ge i)$ 爲結尾的所有沒空一格之連續區間的最大總和值，計算方法如下：

```
sum = 0;
temp_max = -inf;
for i = N to 1 :
    sum += f[i];
    F5[i] = max(temp_max, sum);
```

計算 $F4(1)+F5(2),\ F4(2)+F5(3), \cdots,\ F4(N-1)+F5(N)$ 中的最大值，即是恰在 f array 中的 prefix 空一格且越過分界線的所有連續區間中的最大總和值，令其爲 b。
我們可以用同樣的方法，以計算 $F4$ 的概念去計算所有恰空一格的 suffix 中的最大總和值，並用計算 $F5$ 的概念去計算所有沒有空一格的 prefix 中總和值最大者，再用計算 b 的方法得到恰空一格於 suffix 時，所有穿過分界線的連續區間之總和最大值，令其爲 c。用計算 $F5$ 的同樣概念，來計算所有沒空一格的 prefix 的最大總和值，以及所有沒空一格的 suffix 的最大總和值，再用計算 b 的方法可以得到所有沒空一格且穿過分界線的連續區間中最大的總和值，令其爲 d。
由上面的 pseudo code 可知，每次的計算都只花 N 次 iterations，可知加總只花 $O(N)$ 時間。
我們可以比較 $a,\ b,\ c,\ d$ 中的最大者，就能找到至多空一格的連續區間中的最大總和值。並且用 subproblem2 中計算 index 的方法即能得到組合出該區間的所有 index(因爲都是相同概念的 dp)，這個找 index 的過程也只需 $O(N)$ 時間即可。所以總共的時間複雜度爲 $O(N)$，同時我們每次計算也都只花大小爲 N 的 array 紀錄，所以總共的空間複雜度爲 $O(N)$。