Problem 5

> Refs & people discussed with:
>
> https://en.wikipedia.org/wiki/Trinomial_triangle
> b09902100

(1) Asymptotic Notations

(a)

$$\begin{aligned} \ln n! &= \sum_{i=1}^{n} \ln i \leq \sum_{i=1}^{n} \ln n = n \ln n = \ln n^n \\ \Rightarrow \ln n! &= O(\ln n^n) \end{aligned}$$

(b)

$$\begin{aligned} n^{\ln c} &= (e^{\ln n})^{\ln c} = e^{\ln n \cdot \ln c} = c^{\ln n} \\ \Rightarrow n^{\ln c} &= \Theta(c^{\ln n}) \end{aligned}$$

(c)

Assume that $\sqrt{n} = O(n^{\sin n})$ is true, then $\exists\ n_0,\ c > 0$ such that $\forall\ n > n_0,\ \sqrt{n} \leq c \cdot n^{\sin n}$.

Let $n_1 = \lceil \frac{c^2 + n_0}{\pi} \rceil \pi$, we have:

$$\begin{aligned} n_1 &\geq \frac{c^2 + n_0}{\pi}\pi = c^2 + n_0 > n_0 \\ \sqrt{n_1} &\geq \sqrt{\frac{c^2 + n_0}{\pi}\pi} = \sqrt{c^2 + n_0} > c \\ \sin n_1 &= \sin\left(\lceil \frac{c^2 + n_0}{\pi} \rceil \pi\right) = 0 \\ c \cdot {n_1}^{\sin n_1} &= c \cdot {n_1}^0 = c \\ \Rightarrow \sqrt{n_1} &> c = c \cdot {n_1}^{\sin n_1} \end{aligned}$$

This conflicts with the assumption, therefore the assumption is false, $\sqrt{n} \neq O(n^{\sin n})$

(d)

Let $f(x) = x - (\ln x)^3$, we have:

$$
\begin{aligned}
f(x) &= x - (\ln x)^3 \\
f'(x) &= 1 - \frac{3(\ln x)^2}{x} = \frac{x - 3(\ln x)^2}{x} \\
f''(x) &= -\frac{3(2\ln x - (\ln x)^2)}{x^2} = \frac{3\ln x(\ln x - 2)}{x^2}
\end{aligned}
$$

And:

$$
\begin{aligned}
f(e^6) &= e^6 - 6^3 = (e^2)^3 - 6^3 > 0 \\
f'(e^6) &= \frac{e^6 - 3 \cdot 6^2}{e^6} > \frac{e^6 - 6^3}{e^6} > 0 \\
f''(x) &= \frac{3\ln x(\ln x - 2)}{x^2} > 0,\ \forall\, x > e^2 \\
&\Rightarrow \forall\, x > e^6,\ f(x) = x - (\ln x)^3 > 0 \\
&\Rightarrow \forall\, x > e^6,\ (\ln x)^3 < x
\end{aligned}
$$

Choose $n_0 = \lceil e^6 \rceil,\ c = 1$:

$$
\begin{aligned}
&\forall\, n > n_0,\ (\ln n)^3 < c \cdot n \\
&\Rightarrow (\ln n)^3 = o(n)
\end{aligned}
$$

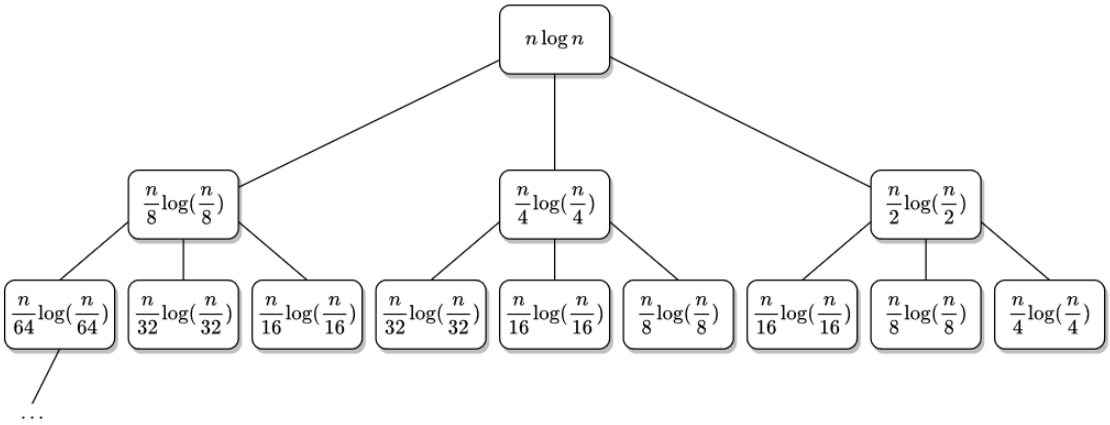
---

(2) Solve Recurrences

(a)

$$
\begin{aligned}
T(n) &= 2T(n-1) + 1 \\
&= 2(2T(n-2) + 1) + 1 = 2^2 T(n-2) + 3 \\
&= 2^2(2T(n-3) + 1) + 3 = 2^3 T(n-3) + 7 \\
&\cdots \\
&= 2^k T(n-k) + (2^k - 1) \\
&= 2^{n-2} T(n - (n-2)) + 2^{n-2} - 1 \\
&= 2^{n-1} - 1 \\
\Rightarrow T(n) &= \Theta(2^n)
\end{aligned}
$$

(b)



Let R_k be the sum of k-th row of the recurrsion tree.

$$
\begin{aligned}
R_1 &= n \log n \\
R_2 &= \frac{n}{2}\log(\frac{n}{2}) + \frac{n}{4}\log(\frac{n}{4}) + \frac{n}{8}\log(\frac{n}{8}) \\
&= \frac{7}{8}n\log n - \frac{(2^2 \cdot 1 + 2 \cdot 2 + 1 \cdot 3)\log 2}{8}n \\
&= \frac{7}{8}n\log n - \frac{11\log 2}{8}n \\
&\leq \frac{7}{8}n\log n
\end{aligned}
$$

$$
\begin{aligned}
R_3 &= \frac{n}{4}\log(\frac{n}{4}) + 2 \cdot \frac{n}{8}\log(\frac{n}{8}) + 3 \cdot \frac{n}{16}\log(\frac{n}{16}) + 2 \cdot \frac{n}{32}\log(\frac{n}{32}) + \frac{n}{64}\log(\frac{n}{64}) \\
&= (\frac{7}{8})^2 n\log n - \frac{154\log 2}{64}n \\
&\leq (\frac{7}{8})^2 n\log n
\end{aligned}
$$

$$
R_k \leq (\frac{7}{8})^{k-1} n\log n
$$

Then:

$$
\begin{aligned}
T(n) &\leq \sum_{k=1}^{\infty} R_k \\
&\leq \sum_{k=1}^{\infty} (\frac{7}{8})^{k-1} n \log n \\
&= \sum_{k=0}^{\infty} (\frac{7}{8})^{k} n \log n \\
&= \frac{1}{1-\frac{7}{8}} n \log n \\
\Rightarrow T(n) &= O(n \log n)
\end{aligned}
$$

And:

$$
\begin{aligned}
T(n) &\geq R_1 \\
&= n \log n \\
\Rightarrow T(n) &= \Omega(n \log n)
\end{aligned}
$$

Therefore, $T(n) = \Theta(n \log n)$.

(c)

Choose $n_1 = e,\ c_1 = 1,\ \epsilon = 0.5$

$$
\begin{aligned}
&\forall\ n > n_1,\ n \log n \geq n \cdot 1 = c_1 \cdot n \\
&\Rightarrow n \log n = \Omega(n^1) = \Omega(n^{(\log_4 2) + \epsilon})
\end{aligned}
$$

Choose $n_2 = 2,\ c_2 = 2$

$$
\forall\ n > n_2,\ 4 \cdot \frac{n}{2} \log \frac{n}{2} = 2n(\log n - \log 2) \leq 2n \log n = c_2 \cdot n \log n
$$

By case 3 of master theorem, $T(n) = \Theta(n \log n)$

(d)

$$
\begin{aligned}
n = 2^m \Rightarrow T(2^m) &= 2^{m/2} T(2^{m/2}) + 2^m \\
F(m) = T(2^m) \Rightarrow F(m) &= 2^{m/2} F(\frac{m}{2}) + 2^m
\end{aligned}
$$

Let $\lg x = \log_2 x$.

Claim: $F(m) \leq (2 \lg m) 2^m\ \forall\ m \geq 2$

For $m = 2, F(2) = T(4) = 2 \cdot T(2) + 4 = 6 \leq 2 \cdot 2^2 = 8$
If it's true for $m = k/2$:

$$\begin{aligned} F(k) &= 2^{k/2} F(\frac{k}{2}) + 2^k \\ &\leq 2^{k/2}(2(\lg k - \lg 2)2^{k/2}) + 2^k \\ &= (2 \lg k)2^k - 2^k \\ &< (2 \lg k)2^k \end{aligned}$$

By induction, $F(m) \leq (2 \lg m)2^m \; \forall \; m \geq 2 \Rightarrow F(m) = O((\log m)2^m)$.

Claim: $F(m) \geq (\lg m)2^m \; \forall \; m \geq 2$

For $m = 2, F(2) = T(4) = 2 \cdot T(2) + 4 = 6 \geq 2^2 = 4$
If it's true for $m = k/2$:

$$\begin{aligned} F(k) &= 2^{k/2} F(\frac{k}{2}) + 2^k \\ &\geq 2^{k/2}((\lg k - \lg 2)2^{k/2}) + 2^k \\ &= (\lg k)2^k \end{aligned}$$

By induction, $F(m) \geq (\lg m)2^m \; \forall \; m \geq 2 \Rightarrow F(m) = \Omega((\log m)2^m)$.

$\Rightarrow F(m) = \Theta((\log m)2^m) \Rightarrow T(2^m) = \Theta((\log m)2^m) \Rightarrow T(n) = \Theta(n \log \log n)$

Problem 6

> Refs & people discussed with:
>
> b09902100

(1)

Use merge sort to sort the sequence B and start with $|I(B)| = 0$.
During the merging of two subarrays A_1, A_2, assuming A_1 is the one at front, each time when an element from A_2 is being merged, increase $|I(B)|$ by "$\#$ of elements in A_1 not merged yet".
After merge sort is done, $|I(B)|$ is also calculated.

(2)

The time complexity of merge sort $T(N)$ has this recurrence relation:
$T(N) = 2T(\frac{N}{2}) + O(N)\ \forall\ N \geq 2$.
The extra calculation done for each merge is $O(N)$, because "$\#$ of elements in A_1 not merged yet" can be acquired by using a variable to store length of A_1 and substract by 1 when an element from A_1 is merged.

Adding this cost to the original recurrence relation yields my algorithm's recurrence relation:
$T(N) = 2T(\frac{n}{2}) + 2 \cdot O(N) = 2T(\frac{n}{2}) + O(N)$.

Because the recurrence relation is the same as merge sort, my algorithm also has the same time complexity $O(N \log N)$.

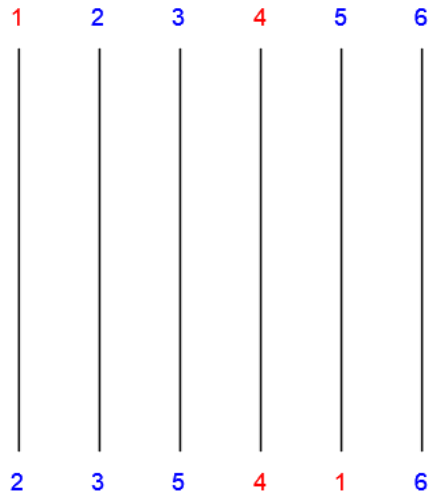
(3)

When an exchange happens during bubble sort, because it reverse a neighboring inversion, number of inversions decreases by 1. And since the number of inversions in a sorted array is 0, the total number of exchanges during bubble sort must be $|I(S)|$.

(4)

1. Organize constraints as tuples. (i, j) means when i should go to j.
2. Loop through all starting points. If point i isn't constrained, constrain it to the first finish point that hasn't been designated.

For example, in this picture the constraints are $(1, 5),\ (4, 4)$. The rest of starting points will be assigned a finish point in order, that is, adding these constraints: $(2, 1),\ (3, 2),\ (5, 3),\ (6, 6)$.



3. Sort these tuples by the first element (starting point) with quick sort.
4. Apply the algorithm described in (1) to calculate the number of inversions.
5. The minimum number of horizontal lines needed is the number of inversions.

(5)

When a horizontal line is added, the corresponding pair of lines exchange their finish point. This is similar to an exchange during bubble sort. We can think of a "constraint" as the initial position of an element in an unsorted array.

To use the minimal number of horizontal lines is to use the minimal number of exchanges, therefore other unconstrained starting points should have their finish points in the same relative order. And to calculate number of exchanges, we can apply the algorithm used to calculate the number of inversion, solving this problem in $O(N \log N)$ time.
