Q1.

You are given recursive CombineSort procedure that sorts input sequence \( A = \{a_1, a_2, \ldots, a_n\} \). Find its recurrence relation \( T(n) \) and use it to calculate worst case complexity of this algorithm. (Note: Combine procedure has \( cn \) worst time complexity).

\[
\text{Start} = 1; \\
\text{Finish} = n; \\
\text{BEGIN} \\
\text{CombineSort}(A, \text{Start}, \text{Finish}) \\
\text{IF (Start < Finish) THEN} \\
\quad \text{Middle} = (\text{Start} + \text{Finish})/2; \\
\quad \text{CombineSort}(A, \text{Start}, \text{Middle}); \\
\quad \text{CombineSort}(A, \text{Middle} + 1, \text{Finish}); \\
\quad \text{Combine}(A, \text{Start}, \text{Middle}, \text{Finish}); \\
\text{END IF} \\
\text{END BEGIN} \\
\text{Answer:} \\
\text{Unless you already got it, it is nothing but MergeSort.} \\
\]

Recurrence relation is
\[
T(n) = 2T(n/2) + cn; \ n > 1 \\
T(n) = a; \ n = 1 \\
\]
This relation can be solved by substitutions
\[
T(n) = 2(2T(n/4)+cn/2)+cn = 4(2T(n/8)+cn/4)+2cn = \ldots 2^k T(1)+kcn = an+cn \log n \\
\]
Hence \( T(n) = O(n \log n) \)

Q2.

Water flows into Hoover dam at a variable rate throughout every month (i.e., 30 days: the tidal cycle) and water is released at a regular interval (once every month). You have the detailed water level data for a period of 30 consecutive days (not necessarily a calendar month, and not necessarily starting or ending with the time of releasing the water).

Give an efficient algorithm to find out exactly when the water was released. (Assume there are no leakage, usage, groundwater absorption or evaporation losses. Also, you may assume that the water is released instantly, and the monthly average of water level does not increase).

Answer:

Since it has been mentioned that the monthly average of water level does not increase, and it applies to any month (30 days), you get the idea that any 30-day period would be repeated.

If we start counting the 30-day water level data (depth vs. time) just after the release and the ending time is just before the release of water, we will get an ascending array \( A \). However, since here we do not know when our 30-day period starts from, we can think
that the ascending array has been “circularly shifted”, and we need to find out for which element of the array it has the highest value.

![Schematic representation of water level](image)

Above is a schematic representation of the water level for any 30 days. The steep fall indicates the release of water. (Note that the rightmost level is just lower than the leftmost level).

1. Start with the full period (30 days) data, i.e., the Array A. Divide the current range \([\text{left}..\text{right}]\) into two (one \([\text{left}..\text{mid}]\) and other \([\text{mid}+1..\text{right}]\).
2. If \(A[\text{left}] \leq A[\text{mid}]\), and also \(A[\text{mid}+1] \leq A[\text{right}]\), return \(\text{mid}\) as the time of release.
3. Else, keep the range for which \(A[\text{left}] > A[\text{right}]\) contains the greatest height, discard the other. Continue in similar fashion.

Complexity: Here it is \(T(N) = T(N/2) + 1\), which gives \(T(N) = O(\log N)\).

Q3.

According to the NASDAQ rules, share prices are announced at the beginning of every minute and the announced price is the value of the share for the next minute. (No intermediate prices are available). Also, in order to bring some sanity after the dotcom meltdown, a rule has been imposed that share prices may rise or fall only by a cent per minute (no fractional change or more than 1 cent change is allowed). However, it may also remain constant for that minute interval.

You have a graph of Priceline.com share value over the day (with minute and cents as \(x\) and \(y\) axis). You want to find out exactly when the price of the share was \(z\) cents (there could be one, multiple or none instances of time when the share was priced a \(z\) cents).

Design an efficient algorithm for finding that and analyze its best & worst case complexity.

Answer:

The graph is essentially a step diagram, and the permissible changes in \(Y\)-axis are \((-1, 0\) or \(+1)\) for a change of 1 in \(x\)-axis.
Suppose the prices are presented by the array $A[i]$, where the price at time $= p$ minute is indicated by $A[p]$.

If we take any time interval $[p,q]$, the maximum the price can vary is $|p-q|$. So, if the following condition satisfies:

$$|z-A[p]| + |z-A[q]| \leq |p-q|$$

then it is possible that there is an occurrence of $z$ in $[p,q]$. Otherwise, the interval cannot have $z$ in it.

We start with $p$ & $q$ as the first and the last point. If it passes the $|z-A[p]| + |z-A[q]| \leq |p-q|$ test, we divide the interval into two and repeat the process for each half (just like the Binary search).

In the worst case (when the price is $z$ throughout the day), the complexity is Theta($N$). However, for all practical purposes, there would be some volume trading, and the price may hit $z$ some fixed number of times. So, the best (or average) case complexity of this algorithm would be Theta($\log N$).