COL 351 Lecture 3 2023/01/05

Topic: Divide and Conquer:

Local Maximum, Stock Trading

Announcement: Tutorial slot: Thu 18:30 – 19:30

Definition: An element a_i of an array $a_1 \cdots a_n$ is called a <u>local maximum</u> if a_i is $\geq$ its neighbors.

Input: $A = [a_1 \cdots\cdots a_n]$: array of elements from some ordered set

Output: Any local maximum of A.

Obvious alg: (Alg 0): "Brute force"

takes $\Theta(n)$ time in the worst case.

$a_1 \cdots\cdots a_{n/2}$ $\quad$ $a_{n/2+1} \cdots\cdots a_n$

Alg 1:

1. Recursively find a_i, a local max of $a_1 \cdot a_{n/2}$
2. If $i < n/2$ or $(i = n/2$ and $a_{n/2} \geq a_{n/2+1})$

   Return a_i

   Else

   Return a local max of $a_{n/2+1} \cdots a_n$.

$T(n) = 2T(n/2) + c$

$\therefore T(n) = \Theta(n)$

Alg 2:

If $a_{n/2} \leq a_{n/2+1}$

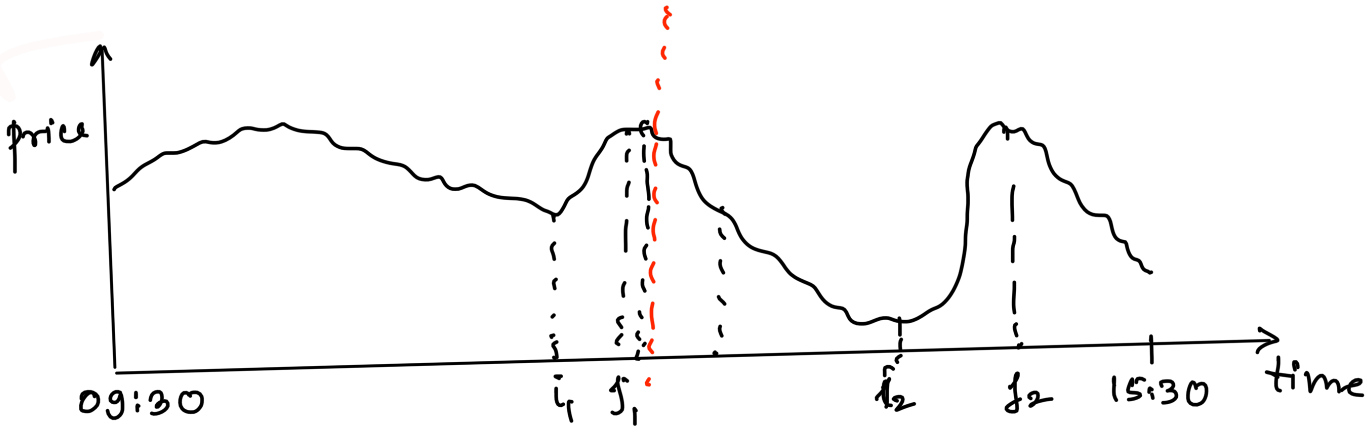
Return a local max of $a_{n/2+1}, \cdots, a_n$

Else

Return a local max of $a_1, \cdots, a_{n/2}$

$T(n) = T(n/2) + c$

$T(n) = \Theta(\log n)$



Input: $a_1, \ldots, a_n$ of numbers;

Output: $\max\limits_{\substack{i,j \in \{1 \ldots n\} \\ i \le j}} a_j - a_i$

Alg 0: Brute-force Running time: $\Theta(n^2)$

Alg 1:

1. $opt_1 \leftarrow \max\limits_{\substack{i,j \text{ in first half} \\ i \le j}} a_j - a_i$ $T(n/2)$

   $opt_2 \leftarrow \max\limits_{\substack{i,j \text{ in second half} \\ i \le j}} a_j - a_i$ $T(n/2)$

2. $opt_3 \leftarrow \max\limits_{j \text{ in second half}} a_j - \min\limits_{i \text{ in first half}} a_i$

3. Return max (opt1, opt2, opt3)

$$T(n) = 2T(n/2) + \underline{cn}$$

$$\therefore T(n) = \Theta(n \log n)$$

Alg 2: Recursive procedure returns the following

1. $\max_{i \le j} a_j - a_i$

2. $\max_{j} a_j$

3. $\min_{i} a_i$

rec stock trading $(a_1, \ldots, a_n)$

$(opt1, max1, min1) \leftarrow$ recstock trading $(a_1, \ldots, a_{n/2})$ — $T(n/2)$

$(opt_2, max_2, min_2) \leftarrow$ recstock trading $(a_{n/2+1}, \ldots, a_n)$ — $T(n/2)$

$$\left.\begin{aligned} opt &= \text{maximum}(opt1, opt2, max_2 - min1) \\ max &= \text{maximum}(max1, max2) \\ min &= \text{minimum}(min1, min2) \end{aligned}\right\} O(1)$$

Return (opt, max, min).

$T(n) = 2T(n/2) + c$

$T(n) = \Theta(n)$

Myth: $O(\ldots)$ is used for worst-case time complexity
$\Omega(\ldots)$ is used for best-case time complexity

Reality: Both $O(\ldots)$ and $\Omega(\ldots)$ can be used in the context of worst-case, best-case, and several other contexts.

Eg: "Merge-sort takes $\Omega(n\log n)$ time in the worst case"

Means: Let $f(n)$: worst case running time of mergesort on an n-element array.

Then $f(n)$ is $\Omega(n\log n)$.