You are given an array $A$ of length $n$ containing integers (which could be positive or negative). A sub-array $A[i, j]$ of $A$, where $i \leq j$, is defined by the sequence $A[i], A[i + 1], \ldots, A[j]$. Give an $O(n \log n)$ time algorithm to find the sub-array of $A$ with the largest sum. Justify your answer.

**Example:** Suppose $A$ is $\{-8, 10, -6, 17, 2, -1\}$. Assuming that the first element of $A$ is denoted by $A[1]$, the sub-array $A[1, 4]$ has total sum $-8 + 10 - 6 + 17 = 13$, whereas sub-array $A[3, 6]$ has total sum $-6 + 17 + 2 - 1 = 12$.

We use divide and conquer. Divide the array into two sup-arrays, $A_1 = A[1, n/2]$ and $A_2 = A[n/2 + 1, n]$ and solve the two sub-problems recursively. We get two solutions with sums $s_1$ and $s_2$, and let $s$ be the maximum of $s_1$ and $s_2$. We need to see if we can get a better sum by a sub-array which starts before (and including) $n/2$ and ends after $n/2$. This can be done as follows: for each $i \leq n/2$, let $a_i$ be the sum of the elements in the sub-array $A[i, n/2]$ and for each $j > n/2$, let $b_j$ be the sum of the elements in the sub-array $A[n/2 + 1, n]$. Clearly, the best such sub-array will have sum $\max_{i \leq n/2} a_i + \max_{j > n/2} b_j$. We output the maximum of this quantity and $s$. Clearly, the recurrence is $T(n) \leq 2T(n/2) + O(n)$, and so, running time is $O(n \log n)$. 


QUIZ 2 (COL 351)

Name Entry No.

Give precise arguments. Needlessly long explanations will not fetch any marks.

You are given the price of a product for \(n\) consecutive days. Let these be \(P_1, \ldots, P_n\). If you buy it on day \(i\) and sell it subsequently on day \(j\), you make a profit of \(P_j - P_i\). Give an \(O(n \log n)\) time algorithm to figure out when to buy it and when to sell it such that the profit is maximized (you are allowed to buy and sell on the same day). Justify your answer.

**Example:** Suppose the prices for 6 consecutive days are 12,6,8,7,10,7. Then it is best to buy on day 2 and sell on day 5.

We use divide and conquer. Divide the sequence recursively into two sub-sequences: \(P_1, \ldots, P_{n/2}\) and \(P_{n/2+1}, \ldots, P_n\), and solve them recursively. This gives two solutions with profits \(p_L\) and \(p_R\) respectively and let \(p\) be maximum of \(p_L\) and \(p_R\). We now need to see if we can get better profit by buying before (and including) day \(n/2\) and selling after (and including) day \(n/2 + 1\). The best strategy in this case would be buy on the day among 1, \ldots, \(n/2\) where price is minimum and sell among days in \(n/2, \ldots, n\) when the price is maximum, i.e., \(\max_{n/2 < i \leq n} P_i - \min_{1 \leq i \leq n/2} P_i\). We output the maximum of this quantity and \(p\). Clearly, the recurrence is \(T(n) \leq 2T(n/2) + O(n)\), and so, running time is \(O(n \log n)\).