The Algorithm

```
Max-Profit(start_date, end_date, dates)
   if start_date == end_date:
      return (start_date, start_date)
   else:
      middle_date = (start_date + end_date) / 2
      (left_buy_date, left_sell_date) = Max-Profit(0, middle_date)
      (right_buy_date, right_sell_date) = Max-Profit(middle_date + 1, end_date)

      left_lowest = date with lowest price from dates[start_date … middle_date]
      right_highest = date with highest price from dates[middle_date + 1 … middle_date]
      left_half_answer = (left_buy_date, left_sell_date)
      right_half_answer = (right_buy_date, right_sell_date)
      across_answer = (left_lowest, right_highest)
      return the pair of maximum profit between left_half_answer, right_half_answer, and across_answer
```

The gist of it is, getting the best pair of dates from the left half of the array, getting the best pair of dates from the right half, and getting the best pair where the buying date is from the left half and the selling date is from the right half (minimum from left half and maximum from right half), and returning the best pair out of these three. You would make the initial call $Max-Profit(1, n, dates)$.

Proof of Correctness

We will prove by induction that this is correct. A series of dates is composed of a left half and a right half. Define $P(x)$ to mean "the optimal pair of buying and selling dates for the series of dates x."

Base Case

There is only one day. The maximum profit is of course 0 and the buying and selling dates are both that one day.

Inductive Case

Given a series c of dates and their stock prices, define a to be its left half and b to be its right half. Because all the dates are sorted chronologically, the optimal pair of buying and selling dates falls into one of three categories:

1. Both dates are from the left half
2. Both dates are from the right half
3. The buying date is from the left half and the selling date is from the right half

We can assume inductive hypotheses $P(a)$ and $P(b)$. The optimal pair of dates where the buying date is from the left half and the selling date is from the right half would be the date with the lowest price from the left half and the date with the highest price from the right because that creates the biggest profit margin. Choosing the best pair out of those 3 pairs thus gives the optimal solution.

Runtime Complexity Analysis

The number of subproblems at each step doubles because you split the array of dates into halves. The size of each subproblem at each step halves. Finding the maximum and minimum in an array is $O(n)$. $p = q = 2$.

$$T(n) \leq 2T\left(\frac{n}{2}\right) + O(n)$$
$$T(1) \leq O(1)$$

It is altogether $\boldsymbol{O(n \log n)}$.

Space Complexity Analysis

n dates would generate $O(\log n)$ layers in its recurrence tree. Each layer or node in the tree would only need $O(1)$ space because we're always storing a pair of dates. Observe that the algorithm is depth-first. The space is thus $\boldsymbol{O(\log n)}$.