After graduating from UNM, you start your new job at the exciting investment banking firm Mammon, Inc. The firm is faced with the following problem. They have an array of the predicted prices of a stock over \( n \) days and they want to determine, using this array, exactly one day to buy the stock and one day to sell the stock in order to maximize their profit.

You are given an array of length \( n \). The array contains predicted stock prices (of type int). You want to choose two indices in the array such that if you buy at the low-price, earlier index and sell at the high-price, later index, then you make the most profit.

The array is very large. Your program can’t simply compare every stock price. It has to be smarter than that. The solution is known as dynamic programming and it involves building up the overall solution from simpler sub-problems.

Here is the solution you need to implement:

You will iterate through the array of stock prices and determine the best choice to buy and sell at from only the elements you have seen so far.

For example, when you have only seen the first day, the best you can do is buy and sell on that day and make zero profit.

When you next loop and have seen the first and second day then either you want to keep your original strategy that earns you zero profit or you want to buy on the first day and sell on the second day (but only if the price is higher on the second day).

As you continue to iterate through the list of stock prices, your program will need to keep track of

- the lowest price seen so far (int)
- the index of the lowest price seen so far (int)
- the best profit that can be achieved only considering each stock price seen so far (int)
- the index at which to sell to get the best profit (int)

I will test your code with a large, randomly generated array of ints. For example:
// Initialize and populate stock price array. 
Random generator = new Random(1); // Set the random seed 
int[] stockPrices = new int[100000]; 
for (int i=0; i<stockPrices.length; i++){
    stockPrices[i] = generator.nextInt(9999999); 
} 

I expect output to take the following form exactly (though the values may differ if the stockPrice array is initialized differently):

Max achievable profit is $9999891
from buying on day 60769 at a price of $95
and selling on day 83764 at a price of $9999986

Feel free to use the above code to test your code.