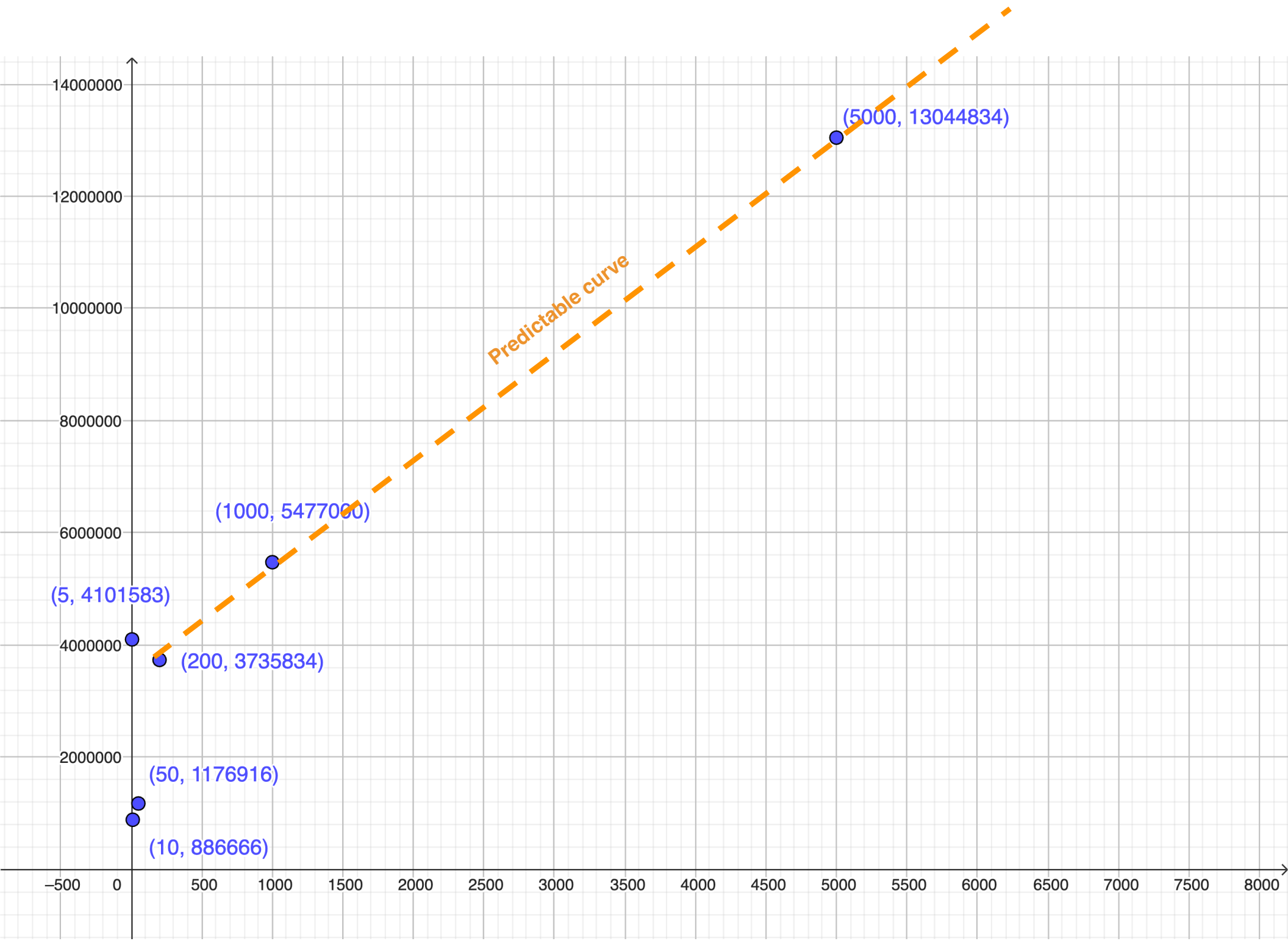
14000000
12000000
10000000
8000000
6000000
4000000
2000000
–500
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
7500
8000
(5000, 13044834)
Predictable curve
(1000, 5477000)
(5, 4101583)
(200, 3735834)
(50, 1176916)
(10, 886666)

Write a small paragraph about what you observed experimentally and what was your analytical/theoretical analysis of time complexity.

I observed that the first test case (array size of 5) always had a significantly higher runtime than other cases with a similar number of elements (array size of 10). This is odd, but may be due to the nature of the input file: many of the trials in this case have B arrays of size 1, which triggers my program to run a different, possibly slower, version of its iteration than other cases would. I also observed that the linear trend upwards became more stable at high array size; that is, there was a more predictable curve (indicated in orange on the plot) when the array size was greater. This is consistent with the fact that since these cases had a greater number of elements, they represents the general trend, rather than a local trend. The time complexity plot from experimental values appears to be somewhat linear, looking more similar to an O(n) time complexity than the O(n^2) time complexity I predicted. This makes sense because my O(n^2) prediction was a worst case runtime analysis, but because of how my method was written the worst case runtime would very rarely occur: there is a nested for-loop triggered by an if-else statement that would always in a worst case scenario, but the if-else is so specific that it usually wouldn't occur. Thus, only one for-loop is looped through, giving an O(n) runtime in most cases, as I observed in the experimental plot.