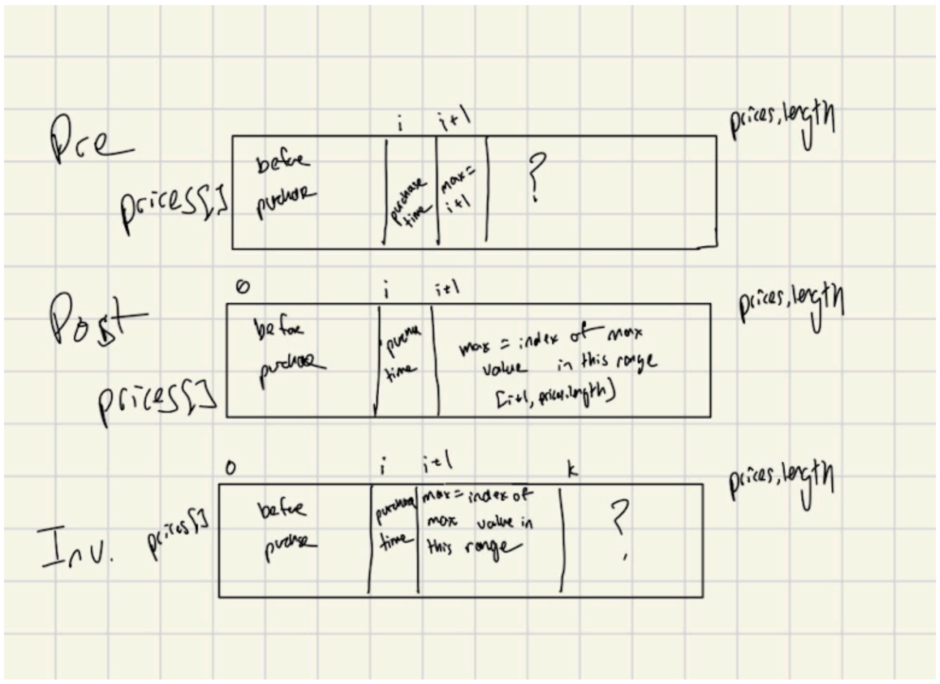
2.1 argmaxTail() Array Diagram



2.4 Runtime analysis of argmaxTail() and optimalTransaction1()

argmaxTail()

Overall worst-case runtime complexity: O(n)

Line Number	Runtime Complexity	Execution Count Complexity
33	O(1)	O(1)
34	O(1)	O(1)
39	O(1)	O(1)
40	O(1)	O(n)
41	O(1)	O(n)
42	O(1)	O(n)
45	O(1)	O(1)

optimalTransaction1()

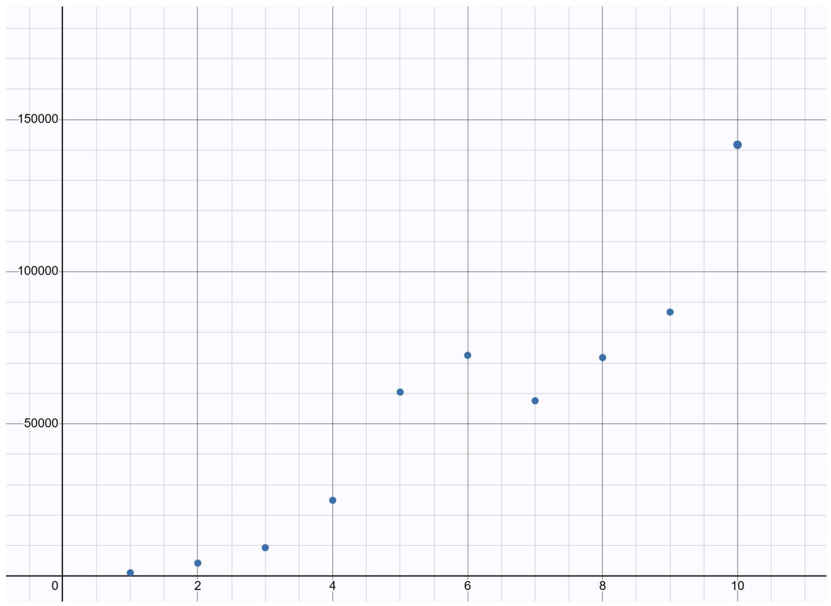
Overall worst-case runtime complexity: O(n^2)

Line number	Runtime complexity	Execution time complexity
54	O(1)	O(1)
60	O(1)	O(1)
61	O(1)	O(n)
62	O(n)	O(n)
63	O(1)	O(n)
64	O(1)	O(n)
66	O(1)	O(n)
69	O(1)	O(1)

2.7 Simulation Analysis

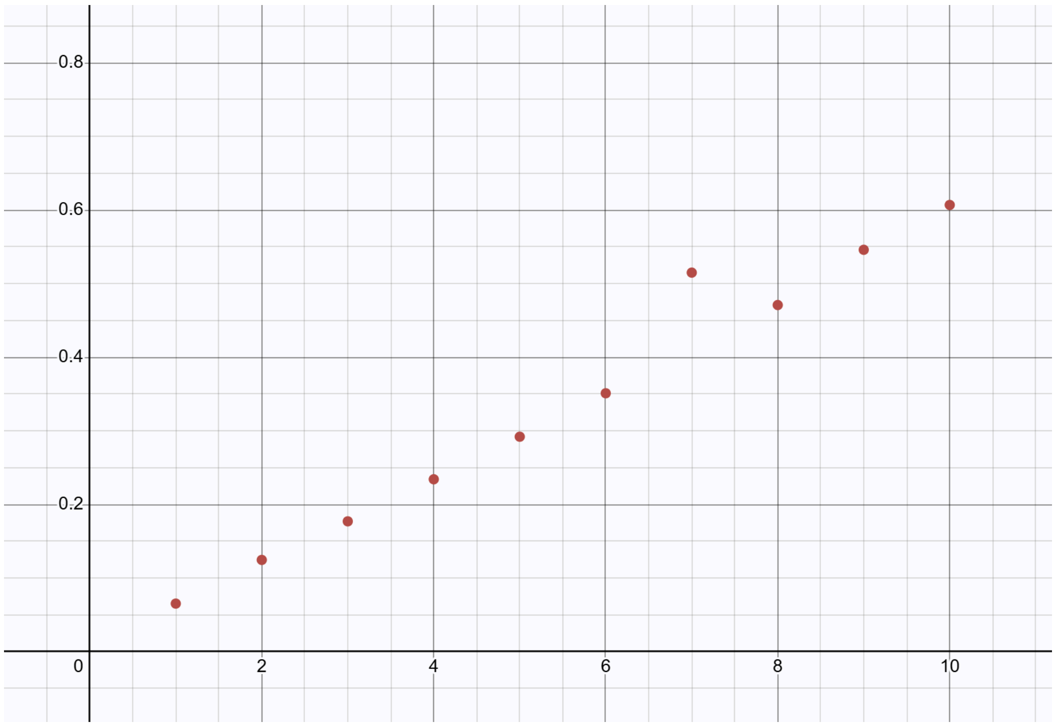
We generated our random price data by using a geometric brownian motion, which follows the formula $dS_t = \mu S_t\, dt + \sigma S_t\, dW_t$. We assumed a growth rate of 5% for μ and assumed σ, which is volatility, to be 20%. We also used the fact there were 252 trading days in a year to create a time step size of 1/252-this gave us the necessary data to use this equation to generate random stock prices.

Plotting the runtime of optimalTransaction1(), we get:



The X-axis is input size in 100000s, while the y-axis is the time elapsed in milliseconds. This graph aligns with the worst-case runtime complexity of this method because it shows a quadratic growth in runtime as the input size increases, which corresponds to the O(n^2) complexity found previously.

Plotting the runtime of optimalTransaction2(), we get:



The X-axis is input size in 100000s, while the y-axis is the time elapsed in milliseconds. This graph aligns with the worst-case runtime complexity of this method because it shows a linear growth in runtime as the input size increases, which corresponds to the O(n) complexity found previously.

2.8 Timing

Henry Ji (hj465) and Jonathan Lin (jl4522)
We spent a collective 10 hours working on this assignment. We met because we were friends from a previous class and we decided to continue our partnership after A2. To split the work, Jonathan took the lead implementing the methods through writing the code, while Henry took the lead on documenting time complexity. However, to ensure good pair programming practice, we always worked together and shared ideas to ensure that we did not split up the work and engaged with the entirety of the assignment.