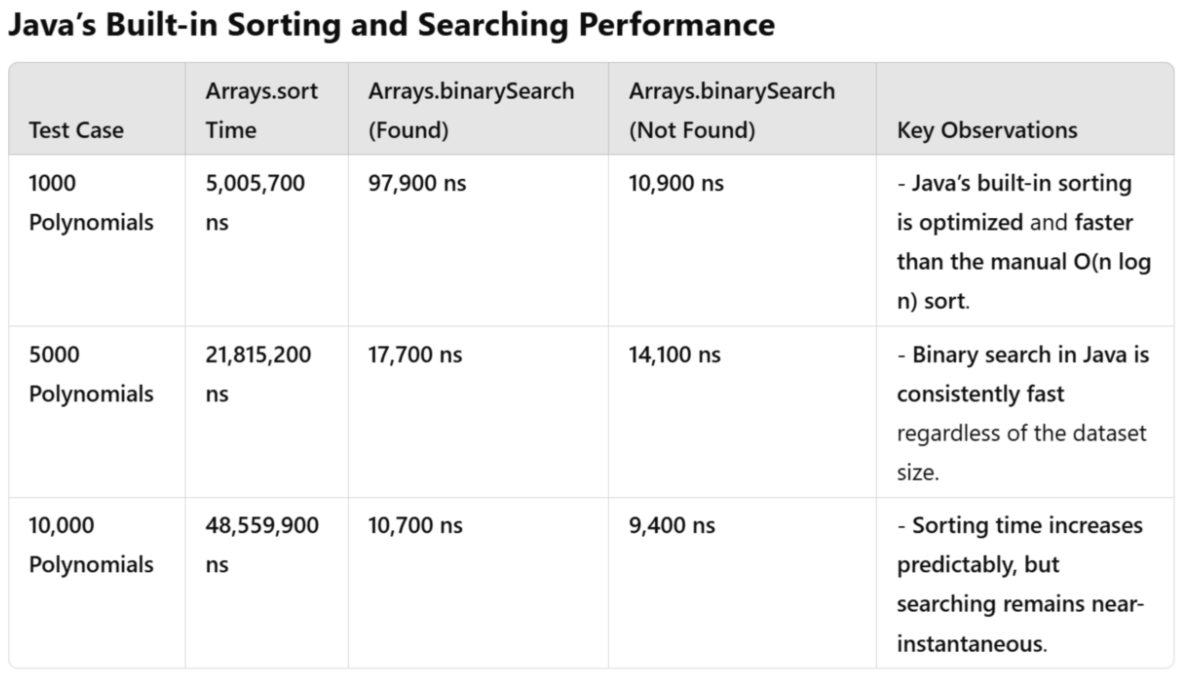
Table 1: Random Polynomial Constructor Output

Polynomial Number	Expression
1	$-1.5x^8 - 3.4x^7 - 1.3x^6 - 1.8x^5 - 4.2x^4 + 3.3x^3 + 2.6x^2 - 3.6x - 4.3$
2	$1.4x^5 - 1.6x^4 - 4.5x^3 + 3.2x^2 - 0.7x + 1.4$
3	$1.5x^3 - x^2 + 0.8x - 2.8$
4	$4.3x^4 + 2.2x^3 - 4.7x^2 + 4.9x + 1.3$
5	$-3.8x^3 + 0.6x^2 - 0.3x + 1.5$

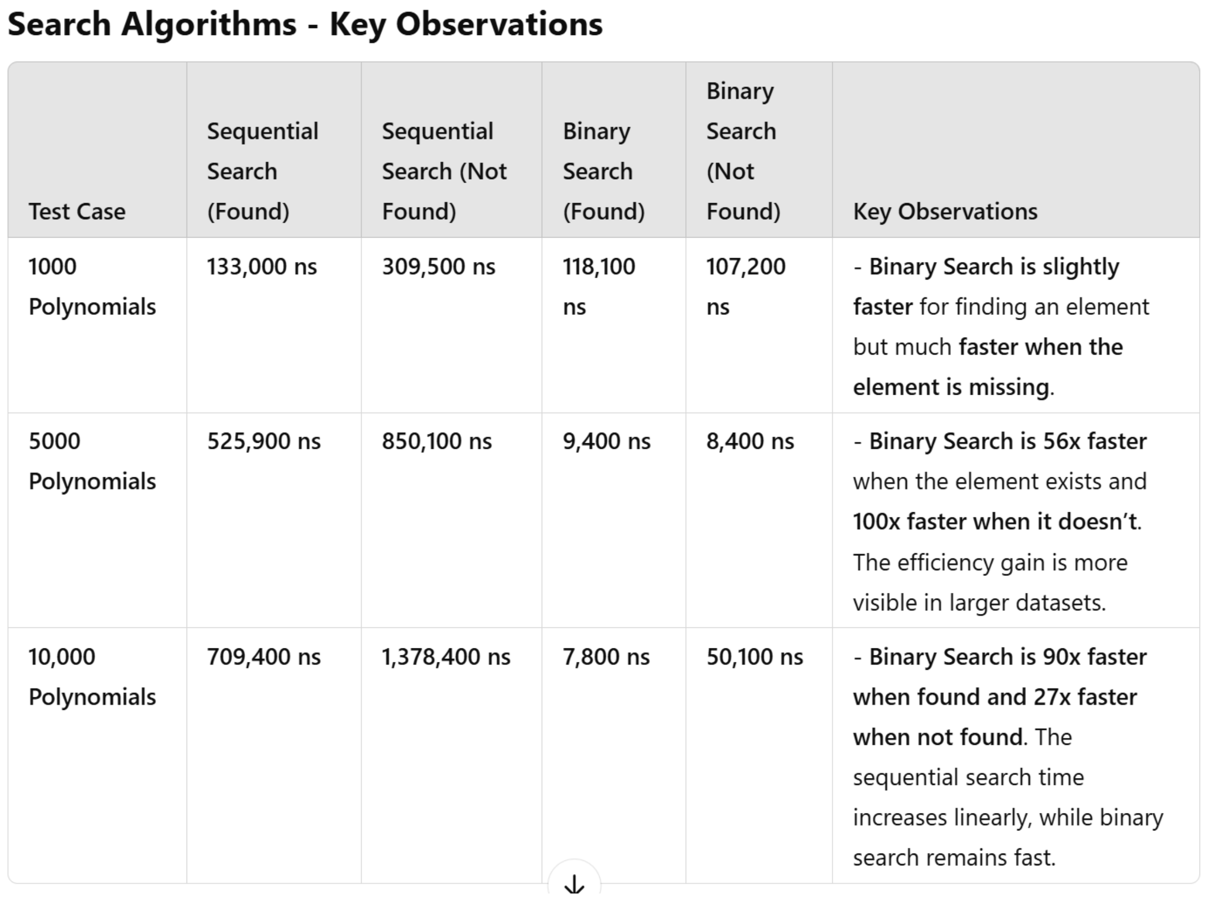
Sorting Algorithms - Key Observations

Test Case	Bubble Sort Time	O(n log n) Sort Time	Key Observations
1000 Polynomials	**96,868,400 ns**	**2,604,300 ns**	- **Bubble Sort is significantly slower** than O(n log n). The performance difference is **about 37x**, confirming that Bubble Sort is impractical for large datasets.
5000 Polynomials	**2,045,426,600 ns**	**14,810,100 ns**	- The performance gap **widens drastically**. Bubble Sort is **138x slower**, making it almost unusable beyond small test cases.
10,000 Polynomials	**9,280,452,200 ns**	**36,083,400 ns**	- The quadratic nature of Bubble Sort is evident: **as input size doubles, its runtime increases exponentially**. In contrast, O(n log n) scaling remains efficient.



Java’s Built-in Sorting and Searching Performance

Test Case	Arrays.sort Time	Arrays.binarySearch (Found)	Arrays.binarySearch (Not Found)	Key Observations
1000 Polynomials	5,005,700 ns	97,900 ns	10,900 ns	- **Java’s built-in sorting is optimized** and **faster than the manual O(n log n) sort**.
5000 Polynomials	21,815,200 ns	17,700 ns	14,100 ns	- **Binary search in Java is consistently fast** regardless of the dataset size.
10,000 Polynomials	48,559,900 ns	10,700 ns	9,400 ns	- **Sorting time increases predictably, but searching remains near-instantaneous**.



Search Algorithms - Key Observations

Test Case	Sequential Search (Found)	Sequential Search (Not Found)	Binary Search (Found)	Binary Search (Not Found)	Key Observations
1000 Polynomials	133,000 ns	309,500 ns	118,100 ns	107,200 ns	- **Binary Search is slightly faster** for finding an element but much **faster when the element is missing**.
5000 Polynomials	525,900 ns	850,100 ns	9,400 ns	8,400 ns	- **Binary Search is 56x faster** when the element exists and **100x faster when it doesn’t**. The efficiency gain is more visible in larger datasets.
10,000 Polynomials	709,400 ns	1,378,400 ns	7,800 ns	50,100 ns	- **Binary Search is 90x faster when found and 27x faster when not found**. The sequential search time increases linearly, while binary search remains fast.