# k-array Predictive Search: A Distribution-Aware Search Algorithm

Author: Rohan Gupta

Data: August 18, 2025

Status: Preprint – Seeking Feedback

## Abstract

I introduce k-array Predictive Search, a novel search algorithm designed to locate a target value within sorted arrays by leveraging prior knowledge or assumptions about the underlying data distribution. Unlike traditional binary search, which implicitly assumes uniform distribution and divides arrays evenly by index, my algorithm divides arrays into $k$ subarrays and predicts the likely subarray containing the target value. This enables superior performance across arbitrary distributions, achieving logarithmic complexity $O(\log_k n)$ under accurate predictions.

## 1. Introduction

Efficient search algorithms are fundamental to computer science, underpinning applications ranging from databases to real-time analytics. Binary search, optimal under uniform data distribution assumptions, consistently achieves $O(\log_2 n)$ complexity. However, real-world datasets often follow diverse, non-uniform distributions. To address this, I present k-array Predictive Search, an adaptive algorithm specifically designed to exploit known or estimated distribution, thus enhancing search efficiency beyond conventional methods.

# 2. Problem Statement

Given:

- Sorted array $A$ of length $n$.
- Target value $T$.
- User-specified integer $k \geq 2$.
- Distribution model or heuristic $D$.

Find the index of $T$ efficiently using predictive subdivision and recursive refinement.

# 3. Algorithm Description

## Input Parameters

- $A$: Sorted Array
- $T$: Target Value
- $k$: Number of Partitions
- $D$: Prediction heuristic based on data distribution

## Algorithm Steps

1. Divide the array $A$ into $k$ approximately equal-sized subarrays.
2. Use the provided distribution $D$, along with the first and last element of array $A$, to predict the subarray most likely containing $T$.
3. Evaluate boundaries of the predicted subarray:
    - If $T$ lies within these boundaries, recursively search within it.
    - If $T$ is smaller than the smallest element, recursively search the subarrays to the left.
    - If $T$ is larger than the largest element, recursively search the subarrays to the right.
4. Continue recursively until $T$ is found or determined absent.

### Complexity

- Best Case: $O(\log_k n)$, given accurate prediction.
- Worst Case: $O(\log_2 n)$, if predictions consistently fail.

## 4. Advantages Over Traditional Methods

- **Distribution-Agnostic:** Unlike binary search, my algorithm accommodates arbitrary distributions such as skewed, exponential, clustered, multimodal, and more.
- **User-Defined k:** Flexibility to optimize performance dynamically based on data characteristics.
- **Predictive Efficiency:** Leverages data-driven insights or domain-specific heuristics to accelerate search.

## 5. Comparative Analysis

| Algorithm | Complexity | Distribution Assumption |
|---|---|---|
| Binary Search | $O(\log_2 n)$ | Uniform |
| Interpolation Search | $O(\log\log n)(average)$, $O(n)(worst)$ | Linear/Uniform |
| k-array Predictive Search | $O(\log_k n)(best)$, $O(\log_2 n)$ $(worst)$ | Arbitrary/predictive |

## 6. Example

Consider an array of 1000 sorted values following a skewed distribution. Using $k = 10$ and predictive model reflecting this skewness, the target value of 850 is immediately localized to a subarray at the higher end, significantly reducing the search iterations compared to binary search.

## 7. Applications

- Database indexing and search
- Log analysis with known access patterns
- Real-time sensor data processing
- Predictive autocompletion

## 8. Limitations

- Performance highly dependent on predictive accuracy.
- Optimal $k$ selection can require experimentation or domain knowledge.
- Overhead in distribution modeling may affect small datasets negatively.

## 9. Future Work

- Automating $k$ selection using machine learning.
- Real-time adaptive distribution learning.
- Extension to multi-dimensional data structures.

## 10. Conclusion

k-array Predictive Search presents a robust alternative to binary search by capitalizing on known or learned data distributions. Its inherent flexibility and distribution awareness position it effectively across diverse real-world datasets, paving the way for enhanced algorithm efficiency in practical scenarios.