Novel Sorting Algorithms Using Treshold Decompotision

M. Panvi Tej
Department of Computer Science and Engineering
Mahindra University
Under the supervision of Dr. Garimella Rama Murthy

ABSTRACT

Sorting and selection constitute foundational operations in computer science, underpinning algorithmic design across domains such as data processing, optimization, digital signal analysis, and information retrieval. Conventional approaches to these problems are dominated by comparison based or arithmetic driven paradigms, which, despite their efficiency in general purpose computing environments, present limitations in contexts emphasizing parallelism, hardware efficiency, or logic level simplicity. This study proposes a Boolean matrix based framework for sorting and order statistic computation that reconceptualizes numerical processing through threshold decomposition and Boolean logic.

The central contribution of this work lies in demonstrating that order statistics including the median, p th maximum, p th minimum, and weighted median can be computed exclusively through Boolean operations. A Hybrid Adaptive Distinct Set framework is introduced to optimize threshold evaluation by restricting computation to the distinct values present in the input dataset. By associating interval widths with each threshold, the proposed method reduces computational complexity from linear dependence on the maximum input magnitude to linear dependence on the number of unique values. This refinement enhances scalability and practical feasibility, particularly for datasets with sparse value distributions.

The framework is further extended to weighted data, providing both a mathematical and logical interpretation of the weighted median and establishing its equivalence to the standard median under uniform weighting. Boolean expressions for selection operations are formally derived, reinforcing the suitability of the approach for parallel and hardware oriented implementations. Collectively, this work advances an alternative theoretical and computational foundation for sorting and selection, illustrating that complex numerical operations can be reformulated as logical processes without loss of correctness or interpretability.

I. INTRODUCTION

Sorting and selection problems occupy a central position in theoretical and applied computer science. The ability to order data or extract specific ranked elements underpins a wide range of computational tasks, including database indexing, scheduling, statistical analysis, machine learning preprocessing, and real time signal processing. As a result, extensive research has been devoted to the development, analysis, and optimization of algorithms capable of performing these operations efficiently and reliably.

Classical sorting algorithms such as quicksort, mergesort, and heapsort are predominantly comparison based. Their theoretical properties are well understood, with tight lower bounds established for worst case and average case performance. Alternative approaches such as counting sort and radix sort exploit structural assumptions about the input domain, particularly bounded integer ranges, to achieve linear time complexity under specific constraints. While these methods are highly effective within their intended contexts, they remain fundamentally tied to arithmetic operations and control intensive logic.

In contrast, contemporary computational environments increasingly emphasize parallelism and hardware specialization. Field programmable gate arrays, application specific integrated circuits, and massively parallel processing units prioritize operations that can be expressed in terms of simple logical primitives. Within such architectures, Boolean operations often incur significantly lower cost than arithmetic comparisons or complex branching logic. This observation motivates the exploration of alternative algorithmic frameworks that operate primarily through Boolean logic while preserving correctness and efficiency.

Threshold decomposition offers a conceptual bridge between numerical data and Boolean representation. By expressing each numeric value as a set of Boolean indicators corresponding to threshold comparisons, it becomes possible to preserve ordering information without explicit numerical computation. This representation has been explored in related contexts, particularly in signal processing and median filtering, where threshold logic has been shown to capture essential statistical properties.

The present study extends this conceptual foundation to sorting and selection algorithms. Rather than relying on pairwise comparisons between values, the proposed framework constructs a Boolean matrix whose structure encodes the relative magnitude of each element with respect to a set of thresholds. Order statistics are then derived through row wise aggregation and Boolean decision rules. This shift from value centric to threshold centric computation constitutes a fundamental departure from traditional paradigms.

Two core research questions guide this investigation. First,

can order statistics such as the median and p th extrema be computed accurately using only Boolean operations derived from threshold decomposition. Second, can the computational complexity of such methods be reduced to practical levels by exploiting properties of the input data, specifically the number of distinct values. Addressing these questions requires both theoretical formulation and algorithmic refinement.

The contributions of this work are threefold. It formalizes a Boolean matrix representation for unweighted selection problems, introduces a Hybrid Adaptive Distinct Set optimization that reduces computational overhead, and extends the framework to weighted medians while preserving logical consistency. These contributions collectively challenge the assumption that numerical sorting and selection inherently require arithmetic computation, demonstrating instead that Boolean logic provides a viable and expressive alternative.

II. Background and Related Work

The study of sorting and selection algorithms has produced a substantial body of literature spanning theoretical computer science, algorithm engineering, and applied domains. Early foundational work established comparison based lower bounds and led to the development of optimal general purpose algorithms. Subsequent research explored specialized algorithms for constrained input domains, including integer sorting and selection under bounded ranges.

Order statistics, particularly the median, have received significant attention due to their robustness properties and relevance in statistics and signal processing. Deterministic linear time selection algorithms, such as the median of medians method, demonstrate that selection can be performed efficiently without full sorting. However, these algorithms remain arithmetic and comparison driven, limiting their direct applicability in logic constrained environments.

In parallel, research in signal processing introduced median and weighted median filters as nonlinear alternatives to linear filtering techniques. These filters exhibit desirable properties in noise reduction and edge preservation. Importantly, weighted median filters were shown to admit representations based on threshold logic and Boolean operations. By decomposing input signals into thresholded binary sequences, researchers demonstrated that median filtering could be implemented using logical operators, enabling efficient hardware realization.

This body of work provides critical theoretical grounding for the present study. Threshold decomposition preserves rank order information while abstracting away numeric magnitude. Boolean logic then operates on these abstractions to recover statistical properties. The extension of this approach from filtering to general selection and sorting constitutes a natural yet nontrivial progression.

Boolean matrix representations have also been explored in data mining and matrix factorization, where binary encodings enable efficient pattern extraction and compression. While these applications differ in objective, they reinforce the expressive power of Boolean matrices in capturing structural relationships within data.

Despite these advances, the application of threshold based Boolean representations to general sorting and selection remains relatively unexplored. Existing approaches often rely on exhaustive threshold ranges, resulting in computational complexity proportional to the maximum input value. This limitation has constrained practical adoption and motivates the need for adaptive optimization strategies.

The Hybrid Adaptive Distinct Set concept introduced in this work addresses this gap by recognizing that only distinct input values contribute meaningful threshold transitions. By restricting computation to these values and accounting for interval widths between thresholds, the algorithm retains correctness while significantly reducing computational cost. This insight aligns with broader trends in algorithm design that emphasize data aware optimization.

III. Boolean Matrix Theory and Threshold Decomposition

Threshold decomposition transforms a set of numeric values into a collection of Boolean representations that encode relative ordering information. Consider an input array consisting of non negative integers. For each threshold value within the domain, a Boolean indicator is assigned to each element, denoting whether the element meets or exceeds the threshold. The resulting structure can be interpreted as a Boolean matrix whose rows correspond to thresholds and whose columns correspond to input elements.

Formally, each row of the matrix represents a binary snapshot of the dataset at a given threshold. As the threshold increases, the number of true entries in each row monotonically decreases. This monotonicity property is central to the correctness of subsequent selection operations. The transition points at which Boolean values change from true to false encode the sorted order of the original data.

By aggregating Boolean values across each row, it becomes possible to determine how many elements exceed a given threshold. This information suffices to evaluate whether a threshold lies above or below a particular order statistic. For example, a threshold corresponds to the median if more than half of the elements exceed it while fewer than half fall below it.

The Boolean matrix representation decouples the computation of order statistics from explicit comparisons between elements. Instead, it relies on aggregate properties of thresholded representations. This abstraction not only simplifies logical formulation but also enables parallel evaluation across thresholds or elements.

However, a naive implementation that evaluates all possible thresholds from zero to the maximum input value incurs prohibitive computational cost when the value range is large. Each threshold requires scanning all elements, resulting in complexity proportional to the product of the number of elements and the maximum value. This limitation motivates the refinement introduced in the Hybrid Adaptive Distinct Set approach.

IV. Hybrid Adaptive Distinct Set Boolean Matrix Algorithm

The Hybrid Adaptive Distinct Set algorithm builds upon the Boolean matrix framework by recognizing that meaningful changes in the Boolean representation occur only at thresholds corresponding to distinct input values. Intermediate thresholds between distinct values produce identical Boolean rows and therefore contribute no new information. By collapsing these redundant thresholds into weighted intervals, the algorithm achieves substantial efficiency gains without compromising correctness.

The algorithm begins by sorting the input array and extracting the set of unique values. These values define the thresholds at which the Boolean representation changes. The difference between consecutive thresholds defines an interval width that captures the number of original thresholds represented by a single Boolean row.

For each distinct threshold, a Boolean row is constructed by evaluating whether each sorted element meets or exceeds the threshold. The number of true and false values is computed, and Boolean decision rules are applied to determine whether the threshold contributes to the desired order statistic. The contribution is then scaled by the interval width, ensuring that the cumulative result matches the value that would have been obtained through exhaustive threshold evaluation.

This adaptive strategy reduces the number of Boolean rows from the maximum input value to the number of distinct values. In datasets where the number of unique values is significantly smaller than the value range, the improvement is substantial. The algorithm thus achieves complexity proportional to the product of the number of elements and the number of distinct values.

The correctness of the approach follows from the observation that order statistics depend only on relative ordering, not on absolute magnitude. By preserving all threshold transition points, the algorithm retains complete ordering information. Interval weighting ensures that omitted redundant thresholds are accounted for in the final accumulation.

An illustrative example demonstrates the algorithm in practice. Given a small array with distinct values, the Boolean rows corresponding to each threshold can be explicitly constructed and evaluated. The accumulated result yields the correct median, maximum, and minimum values, validating the logical formulation.

V. Boolean Expressions for Selection and Mathematical Formulation

A. Problem Definition

Let the input dataset be a finite sequence of non-negative integers

$$a = (a_1, a_2, \ldots, a_N),$$

where $N \in \mathbb{N}$ denotes the number of elements. Define

$$\text{MAX} = \max_{1 \le i \le N} a_i.$$

The objective is to compute order statistics, specifically the median, the p-th maximum, the p-th minimum, and their weighted analogues, using only Boolean operations derived from threshold decomposition.

B. Boolean Threshold Decomposition

For every integer threshold

$$t \in \{1, 2, \ldots, \text{MAX}\},$$

define Boolean variables

$$X_i(t) = \begin{cases} 1, & \text{if } a_i \ge t, \\ 0, & \text{if } a_i < t, \end{cases} \quad i = 1, 2, \ldots, N.$$

The Boolean row vector at threshold t is

$$X(t) = (X_1(t), X_2(t), \ldots, X_N(t)).$$

Stacking these vectors for all thresholds yields a Boolean matrix of dimension

$$(\text{MAX} + 1) \times (N + 1),$$

where rows correspond to thresholds and columns correspond to input elements.

Define the rowwise statistics

$$\text{ones}(t) = \sum_{i=1}^{N} X_i(t), \quad \text{zeros}(t) = N - \text{ones}(t).$$

These quantities capture the number of elements greater than or equal to the threshold and the number strictly below it.

C. Hybrid Adaptive Distinct Set Threshold Reduction

Let the sorted array be

$$a^{\uparrow} = \text{sort}(a),$$

and let the strictly increasing sequence of distinct values be

$$T = \{\tau_1 < \tau_2 < \cdots < \tau_R\},$$

where R is the number of distinct elements.

Instead of iterating over all integer thresholds, computation is restricted to T. Define the width associated with threshold τ_k as

$$w_k = \tau_k - \tau_{k-1}, \quad \tau_0 = 0.$$

This width represents the number of redundant integer thresholds collapsed into a single Boolean evaluation.

For each τ_k, the Boolean row is defined as

$$X_i(\tau_k) = \begin{cases} 1, & \text{if } a_i^{\uparrow} \ge \tau_k, \\ 0, & \text{otherwise.} \end{cases}$$

D. Boolean Decision Variables for Unweighted Selection

Define rowwise Boolean decision variables as follows.

1) Median Decision Variable:

$$V_{\text{med}}(\tau_k) = \begin{cases} 1, & \text{if ones}(\tau_k) > \text{zeros}(\tau_k), \\ 0, & \text{otherwise.} \end{cases}$$

2) p-th Maximum Decision Variable:

$$V_{\text{pmax}}(\tau_k) = \begin{cases} 1, & \text{if ones}(\tau_k) \geq P_{\text{max}}, \\ 0, & \text{otherwise.} \end{cases}$$

3) p-th Minimum Decision Variable:

$$V_{\text{pmin}}(\tau_k) = \begin{cases} 1, & \text{if zeros}(\tau_k) < P_{\text{min}}, \\ 0, & \text{otherwise.} \end{cases}$$

E. Accumulation Formulas

The required order statistics are obtained by weighted summation over thresholds.

$$\text{Median} = \sum_{k=1}^{R} w_k \cdot V_{\text{med}}(\tau_k),$$

$$\text{p-th Maximum} = \sum_{k=1}^{R} w_k \cdot V_{\text{pmax}}(\tau_k),$$

$$\text{p-th Minimum} = \sum_{k=1}^{R} w_k \cdot V_{\text{pmin}}(\tau_k).$$

These formulas are algebraically equivalent to summation over all integer thresholds while achieving reduced complexity.

F. Boolean Expression for Median (Explicit Form)

For the special case $N = 5$, the median requires at least three Boolean variables to be equal to one. Let

$$(X_1, X_2, X_3, X_4, X_5) \in \{0, 1\}^5.$$

The Boolean expression is

$$\begin{aligned} Y_{\text{med}} = {} & (X_1X_2X_3) + (X_1X_2X_4) + (X_1X_2X_5) \\ & + (X_1X_3X_4) + (X_1X_3X_5) + (X_1X_4X_5) \\ & + (X_2X_3X_4) + (X_2X_3X_5) + (X_2X_4X_5) \\ & + (X_3X_4X_5). \end{aligned}$$

This expression evaluates to one if and only if at least three inputs are equal to one, matching the median condition exactly.

G. Weighted Median Formulation

Let each element a_i have an associated positive weight $w_i > 0$. Define

$$W_{\text{tot}} = \sum_{i=1}^{N} w_i, \quad T = \frac{W_{\text{tot}}}{2}.$$

After sorting pairs (a_i, w_i) by ascending a_i, define cumulative weights

$$C_k = \sum_{i=1}^{k} w_{(i)}, \quad k = 1, \ldots, N.$$

The weighted median is defined as

$$m = a_{(k)} \quad \text{where} \quad k = \min\{k \mid C_k \geq T\}.$$

H. Boolean Interpretation of Weighted Median

The weighted median can be interpreted as the ordinary median of a multiset where each element a_i is conceptually replicated w_i times. Under this interpretation, Boolean threshold decomposition applies unchanged, and the weighted median coincides with the unweighted Boolean median of the expanded multiset.

I. Equal Weights Consistency Result

If all weights are equal, $w_i = c > 0$, then

$$W_{\text{tot}} = Nc, \quad C_k = kc.$$

The median condition reduces to

$$kc \geq \frac{Nc}{2} \iff k \geq \frac{N}{2}.$$

For odd $N = 2m+1$, the smallest such k is $m+1$, yielding

$$\text{Weighted Median} = \text{Ordinary Median}.$$

This establishes theoretical consistency between weighted and unweighted formulations.

J. Sorting via Boolean Selection

Define the Boolean minimum selection operator as the special case $P_{\text{min}} = 1$. Repeated application of the Boolean minimum or maximum operators yields a complete sorting procedure. Each iteration extracts the next order statistic using the same Boolean threshold formulation, demonstrating that sorting is reducible to Boolean selection.

VI. Boolean Logic Interpretation and Selection Expressions

The Boolean matrix framework described previously provides a structural representation of numerical data through threshold decomposition. While row wise aggregation yields a procedural mechanism for computing order statistics, the same logic can be expressed directly through Boolean expressions. These expressions formalize selection conditions using logical conjunction and disjunction, further reinforcing the idea that sorting and selection can be realized entirely within a Boolean domain.

Consider a Boolean row corresponding to a fixed threshold. Each column variable represents whether a particular element exceeds the threshold. The selection problem then reduces to determining whether a sufficient number of these Boolean variables evaluate to true or false. For an odd number of elements, the median condition requires that strictly more than half of the elements meet or exceed the threshold. This condition can be expressed as a logical formula that evaluates to true if and only if a majority of the Boolean variables are true.

In the specific case of five elements, the median condition requires at least three true values. A canonical Boolean formulation enumerates all combinations of three variables and forms the logical disjunction of their conjunctions. Each conjunction represents a specific triple of elements simultaneously exceeding the threshold, and the disjunction ensures that the expression evaluates to true whenever any such triple exists. This construction guarantees equivalence between the Boolean expression and the numerical median condition.

The importance of this formulation lies not in its combinatorial enumeration, but in its conceptual significance. It demonstrates that selection criteria can be reduced to pure logic without reliance on arithmetic comparison or counting operations. In hardware implementations, such expressions can be mapped directly onto logic gates, enabling highly parallel evaluation.

The same reasoning generalizes to arbitrary numbers of elements and to other order statistics. The p-th maximum condition requires that at least p elements exceed the threshold, while the p-th minimum condition requires that fewer than a specified number of elements fall below it. Each condition admits a corresponding Boolean expression composed of conjunctions of appropriate size and their disjunctions. While the explicit expressions grow combinatorially, their existence establishes theoretical completeness.

These Boolean formulations also clarify the relationship between threshold based aggregation and logical decision making. Rather than counting ones and zeros numerically, the Boolean expressions encode the same decision boundaries implicitly. This equivalence provides flexibility in implementation, allowing designers to choose between numeric aggregation and direct logical evaluation depending on architectural constraints.

VII. Weighted Median Framework

The weighted median extends the notion of central tendency by incorporating relative importance or frequency into the definition of the median. In weighted datasets, each element contributes a specified positive weight, and the median is defined as the smallest value for which the cumulative weight of all elements not exceeding it reaches at least half of the total weight. This concept is widely used in robust statistics, signal processing, and decision making systems.

Within the Boolean matrix framework, the weighted median can be interpreted through two complementary perspectives. The first is the conventional cumulative weight formulation, which operates on sorted value weight pairs and identifies the threshold at which accumulated weight crosses the midpoint. The second is a Boolean interpretation that aligns naturally with threshold decomposition.

The Boolean interpretation conceptualizes each element as being replicated a number of times equal to its weight. Under this expansion, the weighted dataset becomes an unweighted multiset whose median coincides with the weighted median of the original data. This equivalence is well established in statistical theory and provides an intuitive bridge between weighted and unweighted selection.

By adopting this conceptual expansion, the Boolean matrix framework for the unweighted median can be applied directly to the weighted case. Each replicated element contributes identical Boolean columns, and the median condition remains unchanged. Although explicit replication is computationally infeasible for large weights, the conceptual model justifies the use of cumulative weights as a surrogate.

In practical implementation, the algorithm sorts the input values and computes cumulative weights from the smallest value upward. The threshold at which cumulative weight reaches or exceeds half of the total weight is identified as the weighted median. This procedure is equivalent to evaluating a weighted Boolean decision variable that becomes true once sufficient weight lies below or at the threshold.

The equivalence between weighted and unweighted medians under uniform weights provides a consistency check for the framework. When all weights are equal, cumulative weight grows proportionally to the number of elements, and the weighted median reduces exactly to the ordinary median. This result confirms that the weighted extension preserves the logical structure of the unweighted case.

The Boolean matrix perspective thus offers both a conceptual explanation and an implementation pathway for weighted selection. It demonstrates that weighting does not fundamentally alter the nature of the selection problem, but rather modifies the aggregation criterion applied to thresholded representations.

VIII. Sorting Using Boolean Logic

While the primary focus of this work is selection rather than full sorting, the Boolean matrix framework naturally extends to sorting through iterative application of selection primitives. Sorting can be viewed as a sequence of selection operations in

which the smallest or largest remaining element is repeatedly extracted and placed in order.

Within the Boolean framework, the p-th minimum or maximum functions serve as selection primitives. By computing the first minimum, the smallest element of the dataset is identified and removed or marked. The process is then repeated on the remaining elements to identify the second smallest, third smallest, and so on until the dataset is exhausted.

This approach demonstrates that sorting can be decomposed into a sequence of Boolean selection operations without explicit comparisons between elements. Each selection step relies solely on threshold evaluation and Boolean decision rules. While this method may not compete with optimized comparison based algorithms in software environments, it illustrates the expressive power of Boolean logic in representing ordering operations.

The conceptual significance of this result lies in its generality. It shows that sorting, often regarded as inherently arithmetic or comparison driven, can be reformulated as a purely logical process. This reformulation has implications for hardware oriented systems where repeated selection operations can be pipelined or parallelized.

Moreover, partial sorting and rank selection tasks, which are common in practice, can be implemented efficiently using the same primitives. Applications such as top k selection, percentile computation, and order based filtering can benefit directly from the Boolean selection framework without incurring the overhead of full sorting.

IX. Algorithms

A. Boolean Threshold Decompotision Algo + ADS

Algorithm 1: Hybrid ADS Boolean Matrix Selection

Input: Array $A = \{a_1, a_2, \ldots, a_N\}$, parameters P_{max}, P_{min}
Output: Median, p-th Maximum, p-th Minimum

1) Sort A in nondecreasing order
2) Extract distinct values $T = \{\tau_1, \tau_2, \ldots, \tau_R\}$
3) Initialize results: $M \leftarrow 0$, $P^*_{max} \leftarrow 0$, $P^*_{min} \leftarrow 0$
4) Set $\tau_0 \leftarrow 0$
5) **for** each threshold $\tau_k \in T$ **do**
   a) Compute interval width $w_k \leftarrow \tau_k - \tau_{k-1}$
   b) Construct Boolean row $X_i(\tau_k) = 1$ if $a_i \geq \tau_k$, else 0
   c) Compute $ones \leftarrow \sum_i X_i(\tau_k)$
   d) Compute $zeros \leftarrow N - ones$
   e) **Median rule:** if $ones > zeros$, then $M \leftarrow M + w_k$
   f) **p-th max rule:** if $ones \geq P_{max}$, then $P^*_{max} \leftarrow P^*_{max} + w_k$
   g) **p-th min rule:** if $zeros < P_{min}$, then $P^*_{min} \leftarrow P^*_{min} + w_k$
6) **return** M, P^*_{max}, P^*_{min}

Algorithm 2: Weighted Median via Threshold Logic

Input: Values $A = \{a_1, \ldots, a_N\}$, weights $W = \{w_1, \ldots, w_N\}$
Output: Weighted median m

1) Sort pairs (a_i, w_i) by increasing a_i
2) Compute $W_{tot} \leftarrow \sum_{i=1}^{N} w_i$
3) Set threshold $T \leftarrow W_{tot}/2$
4) Initialize cumulative weight $C \leftarrow 0$
5) **for** $i = 1$ to N **do**
   a) $C \leftarrow C + w_i$
   b) **if** $C \geq T$ **then**
      i) $m \leftarrow a_i$
      ii) **break**
6) **return** m

X. Proof of Equivalence Between Weighted and Ordinary Median

The weighted median is a generalization of the ordinary median in which each data element is associated with a positive weight representing its relative importance or frequency. While this formulation is more flexible, it is essential to verify that the weighted median remains consistent with the classical median when all weights are equal. This section formally establishes that result.

A. Problem Setup

Let the input dataset be

$$A = \{a_1, a_2, \ldots, a_N\}$$

sorted in nondecreasing order such that

$$a_1 \leq a_2 \leq \cdots \leq a_N.$$

Assume that each element a_i is assigned the same positive weight

$$w_i = c, \quad c > 0.$$

The total weight of the dataset is therefore

$$W_{\text{tot}} = \sum_{i=1}^{N} w_i = Nc.$$

The weighted median is defined as the smallest value a_k such that the cumulative weight satisfies

$$\sum_{i=1}^{k} w_i \geq \frac{W_{\text{tot}}}{2}.$$

—

B. Cumulative Weight Analysis

Since all weights are equal, the cumulative weight after the k-th element is

$$C_k = \sum_{i=1}^{k} w_i = kc.$$

Substituting into the weighted median condition yields

$$kc \geq \frac{Nc}{2}.$$

Dividing both sides by c gives

$$k \geq \frac{N}{2}.$$

—

C. Median Position for Odd and Even N

Case 1: Odd Number of Elements

Let $N = 2m + 1$. Then

$$\frac{N}{2} = m + \frac{1}{2}.$$

The smallest integer k satisfying the inequality is

$$k = m + 1,$$

which corresponds exactly to the position of the ordinary median.

Case 2: Even Number of Elements

Let $N = 2m$. Then

$$\frac{N}{2} = m.$$

Any value a_k with $k \geq m$ satisfies the condition. This aligns with the standard definition of the median for even-sized datasets, which lies between a_m and a_{m+1}.

—

D. Illustrative Table

Table I summarizes the cumulative weight behavior for equally weighted elements and highlights the equivalence with the ordinary median.

TABLE I
EQUIVALENCE OF WEIGHTED MEDIAN AND ORDINARY MEDIAN UNDER EQUAL WEIGHTS

Index i	Value a_i	Weight w_i	Cumulative Weight C_i
1	a_1	c	c
2	a_2	c	$2c$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
m	a_m	c	mc
$m+1$	a_{m+1}	c	$(m+1)c$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	a_N	c	Nc

—

E. Conclusion of the Proof

The analysis demonstrates that when all weights are equal, the condition defining the weighted median reduces exactly to the condition defining the ordinary median. Therefore, the weighted median is a strict generalization of the classical median and preserves consistency under uniform weighting.

XI. Optimization and Hardware Considerations

The Boolean matrix framework is particularly well suited to optimization strategies that exploit parallelism and low level hardware capabilities. Several techniques can enhance practical performance without altering the underlying logic.

The use of unique thresholds, as introduced in the Hybrid Adaptive Distinct Set algorithm, is the most significant optimization. By eliminating redundant thresholds, the algorithm reduces computational load while preserving correctness. This optimization aligns with data aware algorithm design principles that tailor computation to input characteristics.

Bit vector representations provide another powerful optimization. Boolean rows can be stored as compact bit arrays, enabling logical operations to be executed using bitwise instructions. Modern processors and hardware accelerators support wide bit operations that process multiple Boolean values simultaneously, significantly accelerating evaluation.

Parallelization further enhances performance. Threshold evaluations are independent of one another, allowing row wise computations to be distributed across processing units. Similarly, Boolean operations within a row can be parallelized across columns. This inherent parallelism makes the framework attractive for implementation on graphics processing units or reconfigurable logic devices.

From a hardware design perspective, the reliance on simple logical gates simplifies circuit complexity. Boolean expressions for selection can be mapped directly onto combinational logic, reducing latency and power consumption. The absence of complex arithmetic units further contributes to efficiency.

XII. Conclusion and Future Directions

This work has presented a comprehensive Boolean logic based framework for sorting and selection, grounded in threshold decomposition and Boolean matrix representation. By transforming numerical data into logical structures, the framework enables computation of order statistics using only Boolean operations, challenging conventional assumptions about the necessity of arithmetic computation in these tasks.

The introduction of the Hybrid Adaptive Distinct Set algorithm addresses a key limitation of naive threshold based approaches by reducing computational complexity through data aware optimization. The extension to weighted medians further demonstrates the flexibility and conceptual coherence of the framework, while Boolean expressions for selection establish a formal logical foundation.

Beyond its immediate contributions, this work opens several avenues for future research. Formal complexity analysis under different data distributions could further characterize performance bounds. Hardware implementations and empirical evaluation on parallel architectures would provide practical validation. Extensions to multidimensional data and other statistical measures may broaden applicability.

Ultimately, the results presented here illustrate that logical abstraction offers a powerful lens through which classical algorithmic problems can be reexamined. By reframing sorting and selection as Boolean processes, this work contributes to a deeper understanding of computation itself and highlights new possibilities for efficient algorithm and system design.

REFERENCES

[1] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1998.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[3] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," *Journal of Computer and System Sciences*, vol. 7, no. 4, pp. 448–461, 1973.

[4] T. S. Huang, G. J. Yang, and G. Y. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 1, pp. 13–18, Feb. 1979.

[5] S. J. Ko and Y. H. Lee, "Center weighted median filters and their applications to image enhancement," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 9, pp. 984–993, Sep. 1991.

[6] D. R. Brownrigg, "The weighted median filter," *Communications of the ACM*, vol. 27, no. 8, pp. 807–818, Aug. 1984.

[7] J. Serra, *Image Analysis and Mathematical Morphology*. London, U.K.: Academic Press, 1982.

[8] D. Bitton and D. J. DeWitt, "Duplicate record elimination in large data files," *ACM Transactions on Database Systems*, vol. 9, no. 2, pp. 255–265, Jun. 1984.

[9] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. New York, NY, USA: Oxford University Press, 2010.

[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed. San Francisco, CA, USA: Morgan Kaufmann, 2017.