# Reviewer 1

Review Report Form

English language and style

( ) English very difficult to understand/incomprehensible  
( ) Extensive editing of English language and style required  
( ) Moderate English changes required  
(x) English language and style are fine/minor spell check required  
( ) I don't feel qualified to judge about the English language and style

|  |
| --- |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Yes | Can be improved | Must be improved | Not applicable |
| Does the introduction provide sufficient background and include all relevant references? | (x) | ( ) | ( ) | ( ) |
| Are all the cited references relevant to the research? | (x) | ( ) | ( ) | ( ) |
| Is the research design appropriate? | (x) | ( ) | ( ) | ( ) |
| Are the methods adequately described? | (x) | ( ) | ( ) | ( ) |
| Are the results clearly presented? | (x) | ( ) | ( ) | ( ) |
| Are the conclusions supported by the results? | (x) | ( ) | ( ) | ( ) |

Comments and Suggestions for Authors

The authors present a new data structure called multiset-trie, which is used to store and process the set of multisets. The manuscript extends the ideas of trie and capture the requirement of the operations that need to be handled in the collection of multisets. The novel part of the manuscript is mathematical modeling, mathematical analysis, and empirical evaluation. The manuscript reads well and is of interest to the algorithms community. I recommend acceptance.

Submission Date

10 December 2022

Date of this review

10 Jan 2023 00:01:43

# Reviewer 2

Review Report Form

English language and style

( ) English very difficult to understand/incomprehensible  
( ) Extensive editing of English language and style required  
( ) Moderate English changes required  
(x) English language and style are fine/minor spell check required  
( ) I don't feel qualified to judge about the English language and style

|  |
| --- |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Yes | Can be improved | Must be improved | Not applicable |
| Does the introduction provide sufficient background and include all relevant references? | ( ) | ( ) | (x) | ( ) |
| Are all the cited references relevant to the research? | (x) | ( ) | ( ) | ( ) |
| Is the research design appropriate? | ( ) | ( ) | (x) | ( ) |
| Are the methods adequately described? | (x) | ( ) | ( ) | ( ) |
| Are the results clearly presented? | (x) | ( ) | ( ) | ( ) |
| Are the conclusions supported by the results? | ( ) | ( ) | (x) | ( ) |

Comments and Suggestions for Authors

The paper proposes a data structure that stores multisets and can answer some natural queries on it.  
Each multiset is represented as a vector of natural numbers, and such vectors are stored in a trie. The queries are answered using naive search on the trie. The model is evaluated in a random scenario, in which each multiset is equally probable.

All in all, I think it would be difficult to improve this paper so that it would deliver results that deserve publishing.

**Reviewer:** I do not see any novelty or importance in those results. The data structure is naive and offers no improvements over a naive approach.

**Response:**

The multiset-trie is different to trie in the following features. 1) Since we are storing multisets the ordering of elements is not important and we use the ordering for the efficient search. The elements of multiset are ordered before they are entered into mset-trie. Experiments show that we achieve the best performance if the elements are ordered by the increasing/decreasing frequency of multiset elements. 2) While the insert, find and delete operations work in the same way as in the ordinary trie, the sub-multiset and super-multiset operations exploit the artificial ordering of multisets for a fast search of multisets (from mset-trie) that are either subsumed or subsume a parameter multiset, respectively.

The artificial ordering of multisets acts as a constraint that narrows down the search space for sub-multiset and super-multiset operations. Consider for instance the following example. Suppose we are searching for all sub-multisets of m={1,3,3,3,5}. In mset-trie we search (using the artificial ordering of elements) the paths from the root to some leaf. In each search step we either take the next element from m, or not! The search space is threfore constrained by the elements of the parameter multiset.

Similarly, the parameter multiset is used as a constraint also in the case of operation super-multiset. In this case, the paths from root to some leaf of mset-trie must include all elements of m={1,4,6}, plus, they can include some additional elements but only when all the preceeding elements match the existing path. For example, after m is matched in prefix {1,4} we can only accept any number of elements 5, but then we have to match an instance of 6, and after we can accept any elements higher than 6 (belonging to some multiset in mset-trie) . Again, the parameter multset m constrains the search space.

Also in the case that the above presented extension of trie can be called “naive”, by the best of our knowledge, nobody has used and explored such an extension of trie before. And, in spite of the limited empirical study, we can say that the sub-multiset and super-multiset operations are fast. For example, judged from the empirical analysis, the number of visited nodes (of mset-trie) for the existance queries is low. The same is derived also from the mathematical analysis of the mset-trie.

**Reviewer:** The evaluated model is strange: an a-priori bound on the cardinality is artificial and not justified.

Uniform distribution over all possible multisets is completely unrealistic.

No particular application is given. In a fixed application one could discuss number of attributes, the distribution of values for an item etc.

**Response:**

As with the analysis of any data-structure, it is impossible to predict characteristic properties of data as those differ over various domains. However, in the process of analyzing the data-structure, the usage of uniform distribution seems to be one of standard benchmarks.

We understand that the reviewer asks for a specific application, together with discussion on number of attributes, distribution of input data etc. However, since this is a new data structure which can be used in any domain using multiset containment operations, we felt that providing a discussion (or even fine-tuning the parameters) for any specialized domain would decrease the strength this paper.

**Reviewer:** Important implementation details are not described: for a fixed node the way the children are accessed matters. Using a table makes large overhead. Using a search tree adds time complexity. Using hash tables makes accessing the next node difficult. This is not addressed at all.

**Response:** Implementation of the mset-trie is given in the source code. The conclusion section includes the paragraph stating the inefficient implementation of the links from a node to its children. They are implemented by an array of a fixed size. In the same paragraph we continue “A custom-implemented small and extendable hash table would significantly decrease the amount of space needed to represent a multiset-trie.”

**Reviewer:** The experiments are run on randomly created data with uniform distribution. This gives little insight into real life performance; in fact probably could be calculated.

**Response:** Experiments 1-3 use artificially generated data which *allow us* to investigate the behaviour of mset-trie operations when varying: 1) the size of the alphabet, 2) the maximal degree of a node (maximal multiplicity of elements in multisets), and 3) the density of mset-trie. Further, Experiment 4 uses real-world data (English dictionary). In this experiment we study the influence of the mapping of letters from the alphabet into an interval [0,N] (N is the size of alphabet) to the performance of mset-trie.

We believe that Experiments 1-4 give a general picture of the multiset-trie behavior and performance.

We have added a new Experiment 5 which includes the comparison of the mset-trie with the main-memory implementation of the inverted index.

**Reviewer:** A natural approach to answer your queries is the range orthogonal queries on vectors of natural numbers. This problem in particular has well analyzed and implemented data structures, say k-d trees and others. They literally appear in textbooks. You should at least compare to them.

**Response:** In the case of k-d trees, the values within the query range (or variouos orthognoal ranges) translate to some (usually geometric) meaning. More formally, for range queries to even make sense, a domain of data needs to form a metric space (i.e. must be equipped with a reasonable distance function).

Indeed one can define (edit) distance on the domain of sets, and in our algorithm elements are indeed ordered initially. However this order bears absolutely no meaning and could be selected in arbitrary way with no impact to the resulting output (and maybe with a slight impact on preformance).  
This is however far from the case if one would go and build a k-d tree structure on top of this, as range queries are highly sensitive on the ordering of elements.

Submission Date

10 December 2022

Date of this review

16 Dec 2022 13:10:28

# Reviewer 3

Review Report Form

English language and style

( ) English very difficult to understand/incomprehensible  
(x) Extensive editing of English language and style required  
( ) Moderate English changes required  
( ) English language and style are fine/minor spell check required  
( ) I don't feel qualified to judge about the English language and style

|  |
| --- |
|  |

Comments and Suggestions for Authors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Yes | Can be improved | Must be improved | Not applicable |
| Does the introduction provide sufficient background and include all relevant references? | ( ) | (x) | ( ) | ( ) |
| Are all the cited references relevant to the research? | (x) | ( ) | ( ) | ( ) |
| Is the research design appropriate? | ( ) | (x) | ( ) | ( ) |
| Are the methods adequately described? | (x) | ( ) | ( ) | ( ) |
| Are the results clearly presented? | ( ) | (x) | ( ) | ( ) |
| Are the conclusions supported by the results? | (x) | ( ) | ( ) | ( ) |

**Reviewer:** Some parts of the text are difficult to read. Try to clarify your definitions.

And Appendix of definitions would be very useful.

**Response:** We assume that this comment concerns the mathematical analysis. The reviewer will find that we added several new definitions and clarified some others.

**Reviewer:** Also, in the experiments is necessary to clarify how is build the input data.

**Response:** The generation of data is described in Section 5, subsection entitled *Data generation* (page 16).

**Reviewer:** And tell us a little bit more about the C++ implementation and how the structure is coded.

**Response:** We include at the beginning of Section 5 two paragraphs describing some aspects of the implementation. We added the paragraph with a description of the implementation of multiset-trie nodes showing that the representation of links between the parent nodes and their children is not space efficient.

**Reviewer:** And finally, regarding the performance, is it possible to compare it with another data structure commonly used?

**Response:** We added a new Experiment 5 that compares mset-trie with the main-memory implementation of the inverted index.

Submission Date

10 December 2022

Date of this review

11 Jan 2023 17:14:35

# Reviewer 4

Review Report Form

English language and style

( ) English very difficult to understand/incomprehensible  
( ) Extensive editing of English language and style required  
( ) Moderate English changes required  
(x) English language and style are fine/minor spell check required  
( ) I don't feel qualified to judge about the English language and style

|  |
| --- |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Yes | Can be improved | Must be improved | Not applicable |
| Does the introduction provide sufficient background and include all relevant references? | (x) | ( ) | ( ) | ( ) |
| Are all the cited references relevant to the research? | (x) | ( ) | ( ) | ( ) |
| Is the research design appropriate? | ( ) | (x) | ( ) | ( ) |
| Are the methods adequately described? | ( ) | ( ) | (x) | ( ) |
| Are the results clearly presented? | ( ) | (x) | ( ) | ( ) |
| Are the conclusions supported by the results? | ( ) | (x) | ( ) | ( ) |

Comments and Suggestions for Authors

The authors present a trie-based data structure for indexing a set of multisets, supporting, among common trie operations, containment queries to report multisets that are supersets or subsets of the queried multiset.

The manuscript is well-structured, and language mistakes are few in numbers.

The topic and the results meet the scope of the journal Algorithms.

The data structure is well-explained, and evaluated under various angles.

**Reviewer:** Nevertheless, I am not satisfied with the mathematical presentation proving the expectancy of various characteristics of the proposed data structure.

**Response:** We hope that the reviewer will find the revised version of the mathematical section more clear. Indeed, we added new definitions, examples, and clarified several parts, as well as captions in our figures.

**Reviewer:** Also, the experiments are lacking any comparison with existing solutions, which are only briefly sketched in one of the last sections of the manuscript.

**Response:** A new experiment that compares mset-trie with the main-memory implementation of the inverted index is added the section Experiment 5.

**Reviewer:** I would think positively about accepting the paper if

(1) the authors can make the proofs in Section 4 better understandable, and

(2) augment the experiments in Section 5 with a thorough comparison with existing data structures.

**Response:** (1) We hope that the reviewer will find the updated version of Section 4 to be more readable, with additional details facilitating understanding.

(2) A new experiment is added comparing mset-trie to inverted index in section *5.5 Experiment 5.* The results of the experiment show that mset-trie outperforms inverted file significantly.

-------------

Detailed comments follow, where numbers at the beginning are the line numbers printed on the right side of the manuscript pages.

**Reviewer:** 4: "such as sub-multiset and super-multiset": these are not operations but objects.

**Response:** Corrected.

**Reviewer:** 8: What precisely are the time and space complexities of your data structure?

**Response:** The time and space complexities of your data structure are presented in the Section 4. While expressions can get a bit technical, the additional space overhead is of order of the input size, i.e. O(|M|). In reality it is mostly ranging between 20-80% of additional nodes in the trie (see Figure 3 for precise plot taking density into account). The time complexity is different for different nodes, but could be (in the worst case) as bad as the size of the whole constructed trie.

**Reviewer:** 13: "a particular" : I think any element is allowed to have duplicates in a multiset, not just a particular element.

**Response:** Corrected.

**Reviewer:** 62/63: What are the merits for being height-balanced for this use case? Any trie that is "full" forms a complete n-ary tree.

**Response:** The primary reason for choosing the heigh-ballanced tree is in the simple design that allows storing and searching multisets effectively.

**Reviewer:** 73: What is the "time complexity space"?

**Response:** The statement was updated: “We carefully designed the experiments to unravel the main features of the search space. We observe how the number of visited nodes depends on various parameters, such as the density of the tree, and the ordering of multiset elements.”

**Reviewer:** 74: empirical analyses, ->  empirical, analyses

**Response:** Updated.

**Reviewer:** 100: Multiset -> A multiset

**Response:** Corrected.

**Reviewer:** 104: What is $n$? Is $n$ a user-defined constant?

**Response:** $n$ is described in line 105.

**Reviewer:** Figure 1: I would write the stored sets as leaf-labels. Further, I see no advantage for writing $c\_j$ instead of just $j$.

**Response:** Please note that the levels of multiset-trie represent the symbols from \Sigma, the numbers represent the indexes of multiset elements, and $c\_i$-s denote the multiplicities of elements.

**Reviewer:** 156: For what is the variable $dev$ used? Can you give an intuition behind it?

**Response:** We have added a paragraph in the introduction of Section 3 after listing the operations of the multiset-trie. The paragraph presents the use of the parameter $dev$.

**Reviewer:** You described up to Section 3.3 standard trie operations, which you could put into the appendix since there is nothing new to see for those readers familiar with trie data structures.

**Response:** The multiset-trie is defined as a tree where all the leaves have equal depth. The definition is different from the standard definition of the trie since the multiset-trie was intended to store unordered multisets and not sequences. In addition, the multiset containment operations can be implemented elegantly.

First, we consider that the reader might want to see the implementation of trie operation in the context of multiset-trie. Secondly, a reader might benefit from reading the implementation of trie operations as an introduction to the implementation of multiset containment operations.

**Reviewer:** 190: tire -> trie

**Response:** Corrected.

**Reviewer:** I think Section 3.5 is identical to Section 3.4 due to the symmetry of the problem.

**Response:** We have joined the two sections in one. The first function submsetExistence is presented in more detail while we describe only the selected aspects of the function supermsetExistence linking the description to the former function submsetExistence.

**Reviewer:** 254: What do you mean with a "quite" precise upper bound?

**Response:** Indeed, the sentence was misleading, so we rephrased it.

**Reviewer:** 255: "appears" : Is this not clear by symmetry?

**Response:** The subset and superset queries are fundamentally different by character. Rather then symmmetric, they are dual to each other. In plain words, if certain subset query requires a lot of time, it will usually be the case that the same query for superset will be computed very fast. Although the difficult situations differ greatly for those two operations, it indeed appears that the final worst-time complexity coincide.

**Reviewer:** 269: Since you need $\Omega(n\sigma)$ time to transform a given input multiset into your type Multiset representation, I am not convinced with your O(1) time.

**Response:**

It is important to distinguish the asymptotic time needed for various operations. In particular, the critical operations like existSubset, or existSuperset should be measured separately from addition of elements, or importing dataset.

As for other data-structures (e.g. priority heap, stack, hash tables), one may not need/want to transform the whole dataset at the same time; one may rather consider starting with an empty collection of sets which becomes populated as the user adds multisets to it.

As a sidenote, it is not true that we need $\Omega(n\sigma)$ to transform a given input multiset into our type Multiset representation.

**Reviewer:** 279: The selection of the same letter $M$ and $\mathcal{M}$ might be confusing. Maybe you can select a different letter for one of the two entities?

**Response:** The choice of notation was carefully selected to be similar-yet-different. Indeed, the set of all leaf-nodes is a (major) part of the whole trie. For this reason, we think the choice is reasonable.

**Reviewer:** Lemma 1 : I think you should find Lemma 1 in literature for the standard trie data structure, which you can cite here.

**Response:** We looked at literature and could not find such standard trie data structure, as mentioned by reviewer. While we never claim the lemma to be anything deep (after all, it is just a lemma), our trie-setting with probability space assumed to be behaving uniformly at random seems a bit too specific to be found in some standard textbook about trie data structure.

**Reviewer:** Lemma 2: What is a "generating function"?

**Response:** Just before first usage we added a paragraph which provides definition of the probability generating functions.

Text, letter

Description automatically generated

**Reviewer:** Page 10:

*I could not follow the proofs since it seems that some notations have not been introduced in a sufficiently detailed way:*

*- What does $\tilde$ mean?*

**Response:**

It represents the so-called »equality in distribution«.   
While tilde symbol can be found use in this context, there is another, more explicit operator to indicate the equality in distribution, so we replaced the symbol in question with .

We also explain this within the text. The replacement was done consistently.

***Reviewer:*** *Page 10:*

*- What is $\mathcal{B}\_0$?*

**Response:** It is a zero-truncated binomially distributed random variable on parameters $n$ and $p\_{i+1}$. In revised version we include this description just before stating Lemma 2. Also see more detailed explanation just below.

***Reviewer:*** *Page 10:*

*- What is $Bernoulli()$?*

**Response:** $Bernoulli()$ stands for the Bernoulli distributed random variable with the corresponding parameter.

Understanding this analysis requires moderate understanding of probability theory. We argue that if the reader is not familiar with such a notion, it might not benefit much from reading the rest of the proof, even if it would be written with much more clarity and precision.

As per reviewer's commend, we added the definition of Bernoulli to the section.

Text

Description automatically generated

***Reviewer:*** *- Referring to the book of Gardiner [27] to possibly omit crucial definitions and keep the proof shortly is in stark contrast to the elaborative description of standard trie operations in the previous section. A more detailed description in how the proof works is more than welcome.*

**Response:** We equipped the mathematical section with additional examples, explanations, definitions, so that it is now accessible to wider audience.

***Reviewer:*** *- What is a probability generating function?*

**Response:** As stated in the comment above, current version now precisely give definition of a probability generating function. Furthermore we give an elaborate example which illustrates the generating functions in general.

Text, letter

Description automatically generated

***Reviewer:***  *What is $1^-$?*

**Response:** Following reviewer's suggestion, we explain this in detail, in this new version.

Text, letter

Description automatically generated

- What is $G'\_X$?

**Response:** The above is the consequence of:

* For function F(z), we denote by F'(z) its derivative.
* For random variable X, we denote its PGF by G\_X

We believe that in the revised version this was made clear enough.

**Reviewer:** Definition 1: What are the intuitions behind $\alpha$ and $\beta$?

**Response:** In this part of the paper we estimate the sizes of number of visited nodes for the subset or superset queries. Alpha corresponds to the former ones, while beta to the latter ones. Both of them relate to maximal number of nodes on the corresponding depth of our tree (parametrized by multiplicities, as well as maximal multiplicity).

Mostly, they aid us in making equation in Corollary 1 better readable.

**Reviewer:** 321: previous Definition 1 -> Definition 1

**Response:** fixed

**Reviewer:** 336: Here, the time bound of $O(|M|)$ is fine with me since this is also the number of elements we report, so we are output-sensitive for this query.  
However, if we have a query like ${1^n, 2^n, ..., \sigma^0}$, then the time stays the same with fewer up to none elements to report, which is not optimal.  
Can you give some informal conclusion on what the cumbersomely long equations for the time complexities mean?  
It would be also good to make a distinction between expected and worst case scenarios. Currently, the analysis seems to mix both worlds.

**Response:** It is frustrating to us that, despite amazing benchark results on several experiments, this structure seems to be very difficult to analyze. The particular approach that we used (and to which we applied some bounds afterwards) unfortunately did not give as good results as we hoped for.

Indeed, as you report, our results do to output optimal values in certain extremal regimes. We are keen on finding an approach which would shed more insight into the mathematical analysis of our data-structure.

With this in mind, we wrote an informal conclusion where we communicated the above.

Text

Description automatically generated

**Reviewer:** 345: on the -> the

**Response:** Fixed.

**Reviewer:** 399: the relation : with respect to what?

**Response:** We rephrased the sentence.

**Reviewer:** Figure 2: I have not understood what probability is addressed in this figure.

**Response:** We expanded the caption making the figure in question much more accessible.

Chart, line chart

Description automatically generated

**Reviewer:** Figure 3: I would rename the y-axis to the ratio you compute.

**Response:** We expanded the caption making the figure in question much more accessible.

Graphical user interface, diagram

Description automatically generated**Reviewer:** 398: Note that analyzing -> Analyzing

**Response:** fixed

**Reviewer:** (15): What is $p$ in this equation? It seems to be only used in the limit?

**Response:** Note that \xi\_i can only be defined after setting basic parameters such as n, \sigma, and p. The first two are already set to 10, and 26, respectively.

**Reviewer:** Figure 8 caption: Exsitence -> Existence

**Response:** Corrected.

**Reviewer:** 638: Why is your time complexity constant?

**Response:** This was an unfortunate choice of words, in the discussion of our multiset data-structure compared with ordinary set-tries. We removed the problematic part.

**Reviewer:** 677-679: Can you explain in detail where the problem was and how it got solved?

**Response:** The significant causes that differentiate the performance of the IR and RDBMS implementations are the join algorithms employed and the hardware cache utilization. The differences in join algorithms and the cache access method that makes IR queries faster were identified precisely. First, the multi-predicate merge join of the IR engine is different from the standard merge join, and the index nested-loop join algorithms. Second, the study of the utilization of cache in the multi-predicate merge join, standard merge join, and index-nested loop joins has been done, identifying more precisely the differences in the algorithms.

Text was corrected.

**Reviewer:** 680-691: Are you aware of the Leapfrog Triejoin algorithm, which seems to be closely related to the problem addressed here.

Todd L. Veldhuizen: Incremental Maintenance for Leapfrog Triejoin. CoRR abs/1303.5313 (2013)

**Response:** Unfortunately, the time constraints for the preparation of the manuscript do not allow us to go into further details. The paper seems to be interesting, and we will consider it in the future.

**Reviewer:** 695: The performance also depends on what type of queries you want to optimize.

**Response:** We have optimized in particular the multiset containment queries.

**Reviewer:** 707: How can the space complexity of your data structure be O(|M|) when storing a constant number of elements (|M| = O(1)) already costs you $\Omega(\sigma)$ nodes in the trie?

**Response:** We kindly ask the reviewer to keep in mind that the cardinality of the input collection of sets M is not of magnitude O(1).

**Reviewer:** 715-719: Specifying the implementation of your data structure as you did here would make sense to me if you would have practically evaluated the time/space of your data structure. It seems that such an experiment is missing, in particular with comparison to other approaches.

**Response:** We have added a new section Experiment 5 where we compare mset-trie to the inverted file.

**Reviewer:** Finally, I could not compile the code (https://github.com/nick-ak96/mstrie) since it seems that some header files are not included such as <memory> needed for std::shared\_ptr<>.

Maybe you can check whether you can make your code more portable?

g++ -DHAVE\_CONFIG\_H -I. -I..    -std=c++14 -Ilib -g -O2 -MT core/mstrie.o -MD -MP -MF $depbase.Tpo -c -o core/mstrie.o core/mstrie.cpp &&\

mv -f $depbase.Tpo $depbase.Po

In file included from core/mstrie.cpp:17:

core/mstrie.hpp:68:7: error: ‘shared\_ptr’ in namespace ‘std’ does not name a template type

   68 |  std::shared\_ptr<std::vector<std::shared\_ptr<MstrieNode>>> mult\_switch;

      |       ^~~~~~~~~~

core/mstrie.hpp:13:1: note: ‘std::shared\_ptr’ is defined in header ‘<memory>’; did you forget to ‘#include <memory>’?

12 | #include <queue>

  +++ |+#include <memory>

**Response:** Another co-author verified that the code compiles without errors that you mention. He is using the following system configuration



with only the standard (integrated) development environment, Xcode. Let us remind the reviewer of the installation instructions, which are provided within the mentioned repository (https://github.com/nick-ak96/mstrie).