Production Systems and Information Engineering
Volume 8 (2019), pp. 39–49
https://doi.org/10.32968/psaie.2019.003

COST ANALYSIS OF THE PREFIX TREE DATA STRUCTURE

Edit Csizmás
Pallasz Athéné University, Kecskemét, Hungary
Department of Informatics
csizmas.edit@gamf.kefo.hu

László Kovács
University of Miskolc, Hungary
Institute of Information Science
kovacs@iit.uni-miskolc.hu

[Accepted July 2017]

Abstract. The data tree is a very widely used data structure in many application areas. The tree structure provides an efficient storage and data manipulation for different data lists or data sets. The prefix tree is a special tree structure to store ordered lists of data elements. In the prefix tree, the lists with common prefix part share the same path. As the prefix tree can be involved in many data manipulation algorithms, the cost estimation of the tree is an important component in the cost function of the whole data manipulation algorithm. This paper provides a cost analysis related to the tree size for a random input set of object lists. The analysis includes both analytical and simulation methods and the main result presented in this paper is an approximation function based on the gamma-distribution.

Keywords: prefix tree, cost function, gamma-distribution

1. Introduction

The ordered list is a good implementation structure for the set representation. The ordering of the elements enables a more efficient implementation of the set operations. For example, the set intersection can be implemented with a sorted merge algorithm. This kind of implementation is used among others in the query engine of the relational databases during the execution of an inner join operation. Thus, we have an object set with a total ordering on it:

$$\langle Oo_i, \leq \rangle$$

The size of O is denoted with M. A set of objects can be represented with an ordered list:

$$l = o_1, ..., o_n \text{ where } o_{i-1} \leq o_i$$

A set of lists, $L = \langle l_j \rangle$ can be represented with a special tree, prefix tree [1], T, where every list l is represented with a path starting from the root element. The prefix tree constructed from the lists on O is denoted by T_O. The paths with similar prefix part share the same segment in the tree. In the tree every node is assigned to an element $o \in O$, except the root, which is empty node. In the tree, every node represents an object sequence related to the nodes along the path from the root. As some lists may be included in other lists as sub-lists, we need a special flag in the nodes to denote the end symbol of the lists. A given list $o_1, ..., o_n$ is contained in T, if there exists a path where the i-th element of the path is assigned to o_i and at the node related to on has end-node flag set to 1. In Fig. 1, a sample prefix tree is shown which contains the following lists: (1,2),(2,3),(3).

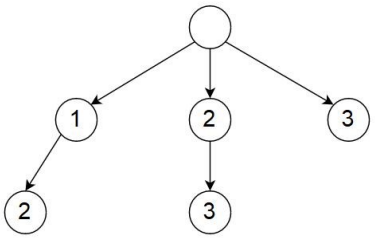

Figure 1. Sample prefix tree

The size of the tree T is equal to the number of nodes in the tree. The prefix tree T_O is a complete tree, if it contains all possible lists on O. The size of the complete prefix tree for the O with M elements is 2^M. This formula can be deducted from the fact that if a new element is added to O, then a node child node assigned to the new element must be appended to every node of the tree. Thus in every step, the size of the tree is doubled (shown in Fig 2.).

In general, the cost of data manipulation operations depend on the size of the data structure. Thus, the size of the prefix tree is an important factor in cost estimation of the data manipulation algorithms. As the size of the data tree depends on the set of input list, the total cost is also a function of the input data. Considering the data set, there are two crucial factors in data generation:

- number of lists

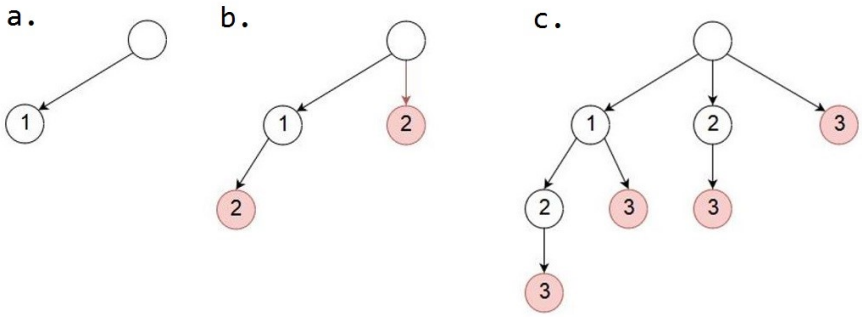

Figure 2. Construction of complete prefix trees

- distribution of the elements in the lists.

The enumeration of tree structures is an intensively investigated topic in the literature. Regarding the general complexity analysis of tree, we can find many contributions [4], [5], [6]. On the other hand, there are no enumeration studies on the prefix tree structure as a specific tree in the literature. The goal of our contribution is to provide an initial analysis on the size complexity of the prefix tree structures.

In our investigation, as an initial step, an independency assumption is applied, that means that the probability of the existence of element o_i in the list, is independent from the probability of any other element o_j. The main parameters of the input data generation are the followings:

- M: the number of elements in O,
- K: the number of lists in the input data set,
- p_i: the probability of the element o_i in the input lists,
- $p_O = (p_1, ..., p_M)$: the probability vector for O.

2. Cost Analysis of the prefix tree size

Two basic approaches are analyzed in our work. The first is based on application of exact probabilistic formulas to determine the size of the generated prefix trees. This method provides precise values but the formulas become too complex when $M > 2$. The second method uses simulation to generate random lists to build up prefix trees. The size of the generated prefix trees are investigated using statistical methods. Based on the experiences, this approach is more suitable to approximate the size of the tree for larger data sets too.

2.1. Enumeration formula

For the case, $M = 2$, the analytical, exact formulas can be constructed on a straightforward way. The formula for the tree size is built up from the following components.

- $P_0 = (1 - p_a)(1 - p_b)$, probability that none of the two elements occurs in the list,
- $P_a = p_a(1 - p_b)$, probability that only element a occurs in the list,
- $P_b = (1 - p_a)p_b$, probability that only element b occurs in the list,
- $P_{ab} = p_a \cdot p_b$, probability that both elements a and b occur in the list.

It can be verified that

$$P_0 + P_a + P_b + P_{ab} = (1 - p_a)(1 - p_b) + p_a(1 - p_b) + (1 - p_a)p_b + p_a p_b = 1$$

Considering the size of the resulted prefix tree, the probabilities of the different size values can be calculated. The probability of the size i is denoted by P_i and the value can be calculated on the following way:

$$P_1 = P_0^K$$

$$P_2 = \sum_{i=1}^{K} \binom{K}{i} P_a^i P_0^{K-i} + \sum_{i=1}^{K} \binom{K}{i} P_b^i P_0^{K-i}$$

$$P_3 = \sum_{i=1,j=1}^{K} \binom{K}{i}\binom{K-i}{j} P_a^i P_b^j P_0^{K-i-j} + \sum_{i=1,j=0}^{K} \binom{K}{i}\binom{K-i}{j} P_a^j P_{ab}^i P_0^{K-i-j}$$

$$P_4 = \sum_{i=1,j=1,l=0}^{K} \binom{K}{i}\binom{K-i}{j}\binom{K-i-j}{l} P_a^l P_{ab}^i P_b^j P_0^{K-i-j-l}$$

The average value of the tree size is calculated with:

$$E[Ntree] = P_1 + 2 \cdot P_2 + 3 \cdot P_3 + 4 \cdot P_4$$

The exact formulas were calculated in a Matlab application for different element probability values. For the sake of simplicity, both elements have the same probability: $p_a = p_b$. The resulted size-function is shown in Figure 3.

As for an arbitrary M, the number of tags in the formula increases to 2^M, it is not possible to use this approach for analysis of larger problems. Thus the application of the exact enumeration formulas is very limited.

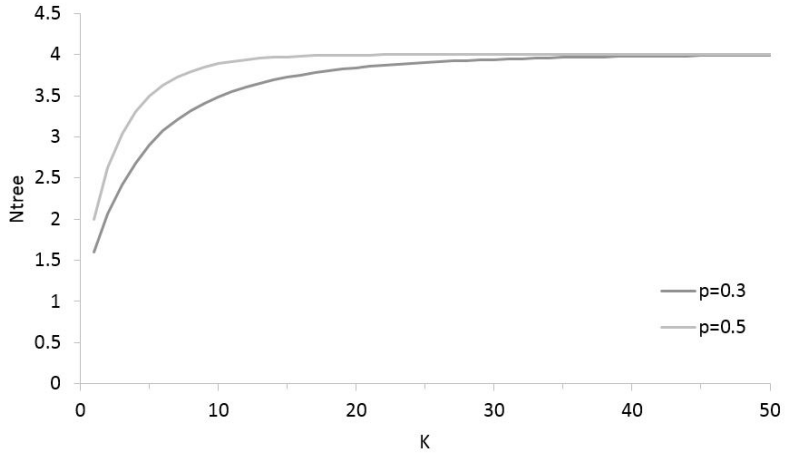

Figure 3. The average size for different element probabilities (M=2)

2.2. Cost Analysis with Simulation

In the simulation, prefix trees are built up from random lists. The generation function to construct the random lists is based on the Monte Carlo method and it depends on three parameters:

- M: the number of elements in O,
- K: the number of lists in the input data set,
- p: the common probability of the elements to appear in the list.

The goal of the simulation is to determine the dependency between the tree size and the input parameters. In our implementation, the lists are stored in a binary matrix form, where the columns correspond to the elements and the rows denote the generated lists. The matrix element $m(i, j)$ is equal to 1 if the i-th list contains the j-th element, otherwise the value is equal to 0.

The size of the trees are calculated with the following algorithms:

- ordering of the input lists by the length of the list,
- eliminating the lists contained in other lists,
- tree construction from the reduced set of lists.

In the simulation, N denotes the number of performed tests. The average of the test results is used to construct the size function. The empirical cost function yielded by the tests for $M = 2$, is shown in Fig. 4.

As it is expected, the simulated cost function for larger N values is more smoother than the functions for small N values. In the investigated parameter

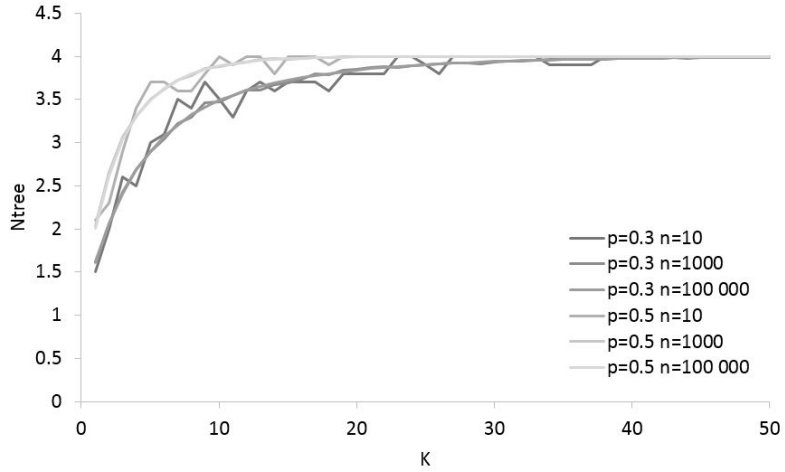

Figure 4. The experimental cost function in dependency from K. M = 2; p = 0,3 and 0,5; N = 10; 1 000; 100 000

ranges, the simulated cost functions are very similar to the exact calculated cost functions. The corresponding error values, calculated as the difference between the simulated and calculated cost values, are given in Table 1.

Table 1. The error of the simulated cost function

	N = 10	N = 1000	N = 100000
p = 0.3	0.6981146	0.098185	0.010077
p = 0.5	0.549177	0.064796	0.007265

For larger M and K values, as the run time of the simulation became significantly high, only lower number of tests were executed. The parameter range investigated in the test series are given in Table 2.

Table 2. The investigated parameter ranges in the simulations

	min	max	step
M	5	50	5
K	200	12000	200
p	0.3	0.5	0.2

In Fig. 5 and Fig. 6, the tree size is shown in dependency of K for different M values. It can be seen that for a higher K value, the prefix tree will be complete and the cost remain constant. The shape of the cost function in Fig

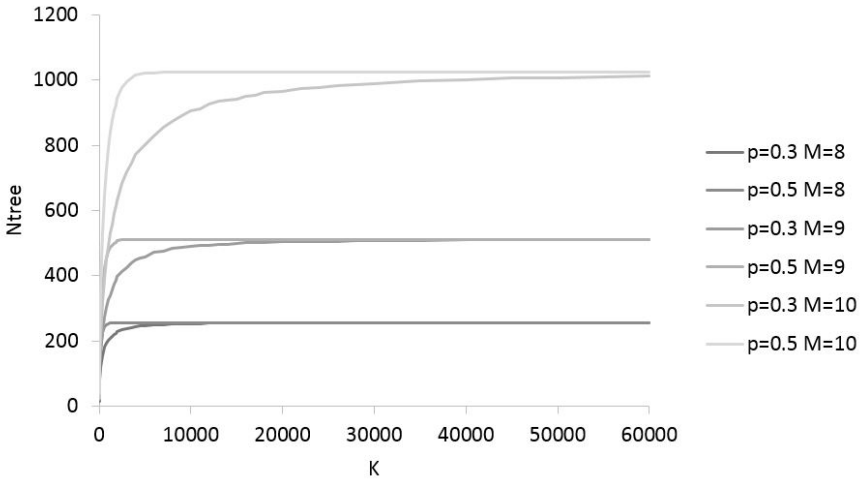

Figure 5. Simulated cost function for large K values

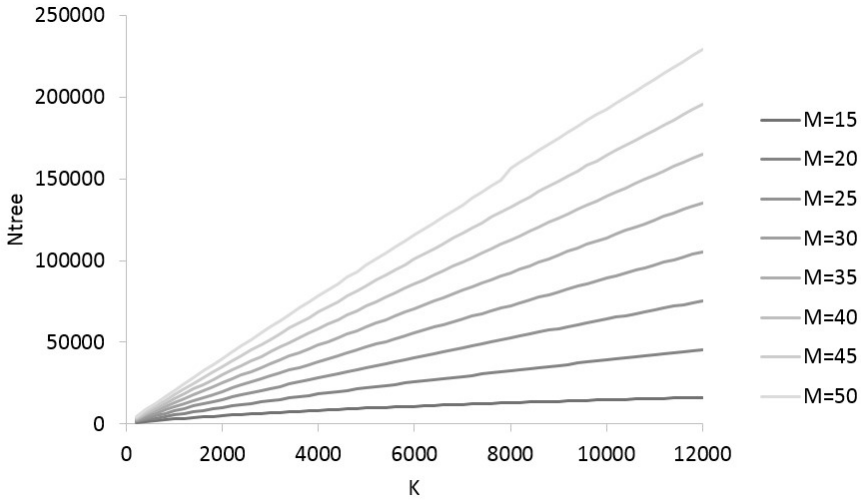

Figure 6. Simulated cost function for small K values

6 shows that the investigated parameter range is still far from the saturation threshold.

We have tested the value of the saturation threshold in the dependency of the M value. The result is shown in Fig. 7. Our experience is that the K

saturation value is an exponential function of M and the increase is larger for $p = 0.3$ than for $p = 0.5$.

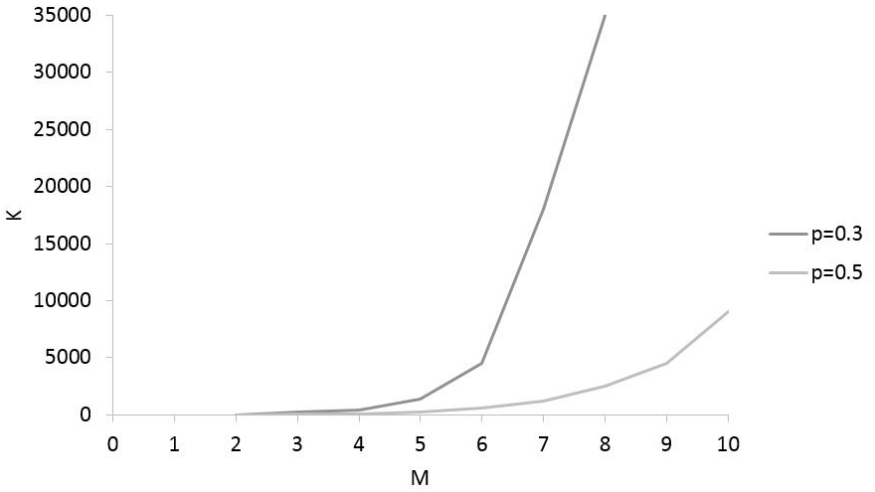

Figure 7. The saturation value (K) in dependency of M, p=0,3; p= 0,5

3. Approximation of the cost function

In order to find an appropriate analytical approximation function, we have first normalized the experimental functions. The normalization on the axis y means that the cost value is transformed into the [0,1] interval. The same normalization can be performed along the axis x too. With the application of the normalization, we can transform the shape of the cost function into a form independent from the current value of M. Thus the resulted graphs can be used to describe the cost function for every M value. Fig. 8 and Fig. 9 show the normalized form of the cost function given with a red color.

The analysis of the normalized cost functions shows that the have a common shape which is very similar to the gamma distribution. The gamma distribution is a continuous probability distribution with two parameters [2]. The two parameters are the shape parameter (k) and the scale parameter (θ). The probability density function of the gamma distribution is given with the following formula:

$$f(x, k, \theta) = \frac{x^{k-1} e^{-\frac{x}{\theta}}}{\theta^k \Gamma(k)}$$

where

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$$

The corresponding distribution function is given with

$$P(k, \theta x) = \frac{1}{\Gamma(a)} \int_0^{\theta x} z^{k-1} e^{-z} dz$$

The approximation algorithm was implemented in Matlab using the system function gamcdf [3].

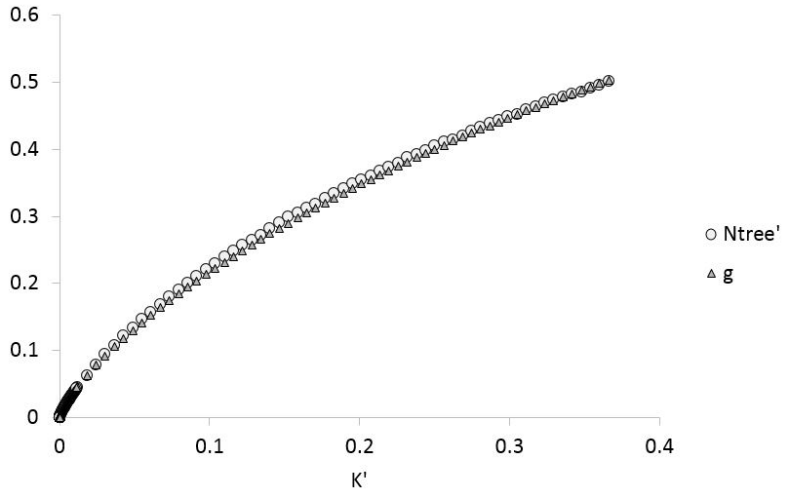

Figure 8. The normalized cost function in dependency of normalized K values, M = 15-50 p = 0.5, k = 0.75; theta = 0.8

Using the regression method for (k, θ) to determine the best matching $f(x, k, \theta)$ gamma distribution, we could achieve a very good approximation of the experimental cost functions. In Fig 8. and Fig 9, the approximation gamma distribution functions are given in color blue.

4. Conclusion

In the performed analysis, we have tested the size function of the prefix tree in dependency from the size and value distribution of the input data on an analytical and experimental way. Test results show that simulation-based experiments provide a good approximation of the analytical cost functions. After normalization of the experimental cost functions, we have found that

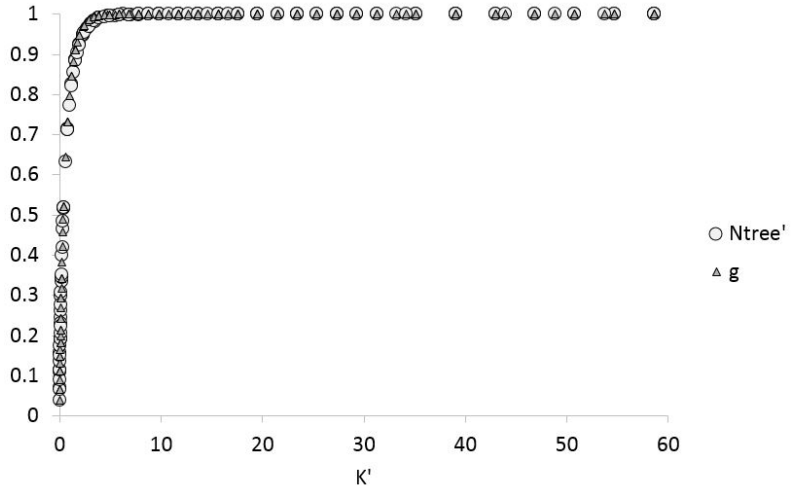

Figure 9. The normalized cost function and the gamma approximation in dependency of normalized K values, M =8; 9; 10; p = 0.5; k = 0.75; theta = 0.8;

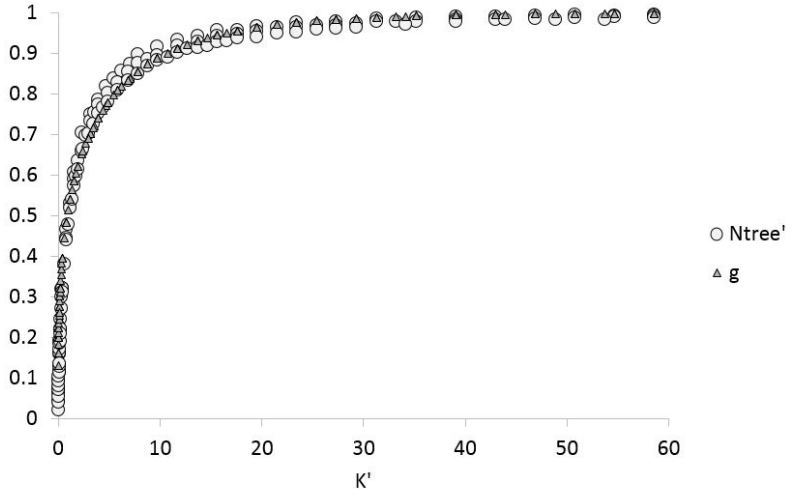

Figure 10. The normalized cost function and the gamma approximation in dependency of normalized K values, M = 8; 9; 10; p = 0.3; k = 0.3; theta = 12

the normalized cost functions can be efficiently approximated with a gamma distribution. In the next phase of the research project, the goal of the analysis

will be the development of an efficient method for calculation of the scale and shape parameters of the best matching gamma distribution.

REFERENCES

[1] M. Hamedanian, M. Nadimi and M. Naderi: *An Efficient Prefix Tree for Incremental Frequent Pattern Mining*, International Journal of Information and Communication Technology Research, Vol 3 (No 2), 2013, pp. 49-56

[2] L. Kenneth: *Numerical analysis for statisticians*, Springer Science and Business Media, 2010.

[3] MATLAB Documentation. [Online]. Available: https://www.mathworks.com/help/stats/gamcdf.html. [date: 19-12-2016].

[4] A. Cayley : *A theorem on trees*, Quart. J. Maths. Vol. 23 (1889), pp. 376-378

[5] C. Chauve, S. Dulucq and O. Guibert: *Enumeration of some labelled trees*, Private Communication

[6] T.C. Cheng: *On computing distinguishing numbers of trees and forests*, The Electronic Journal of Combinatorics Vol. 13 (2006) *https://doi.org/10.37236/1037*