A FORMULA FOR VERTEX CUTS IN b-TREES

LORENTZ JÄNTSCHI, CARMEN E. STOENOIU, AND SORANA D. BOLBOACĂ

ABSTRACT. The paper communicates a polynomial formula giving the number and size of substructures which result after removing of one vertex from a b-tree. Particular cases of the formula are presented and discussed.

1. Introduction

In computer science, b-trees are tree data structures that are most commonly found in databases and file systems; b-trees keep data sorted and allow amortized logarithmic time insertions and deletions (see [1, 2]). There are at least three domains where the b-trees concepts were use in researches:
Networks: basic operations (Insert, Delete, and Search) algorithms ([3, 4]), dynamic collaboration [5], dynamic information storage [6], dynamic memory management [7, 8], secondary storage data structures [9], mobile databases access [10];
Databases: file organization [11], access and maintain large sets of data [12, 13], searching algorithms [14, 15];
Computational chemistry: topological research [16], and graph theory [17, 18].
It is known that connectivity is one of the basic concepts in graph theory: the minimal number of edges or vertices that disconnect a graph when removed (cuts) [19]. Why the vertex cuts are important? Vertex cuts in a graph can reveal a strong connectivity structure with better properties.

The aim of the research was to found polynomial formula for vertex cuts in b-trees. The applicability on two particular cases of the obtained formula was also assessed.

2. The Problem

A graphical representation of a b-tree is given in figure 1. For $b = 1$ the tree degenerate into a path. For $b = 2$ the tree is the binary tree. The proposed for solving problem is counting of substructures which it results after removing of one vertex from the b-tree. Three remarks can be making: The root vertex has b edges; The leaf vertices have 1 edge; All other vertices have (b +1) edges.

1991 *Mathematics Subject Classification.* 05C05, 05C10, 05C85, 05C90, 11T06.
Key words and phrases. Graph theory, b-tree, Polynomial formula.

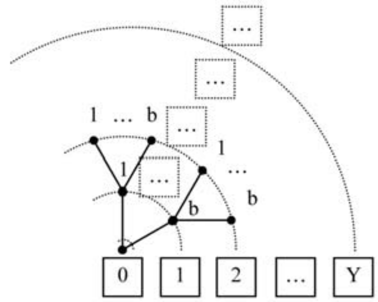

FIGURE 1. $T_{b,Y}$ tree

3. The Solution

The total number of vertices (TNV) in a b-tree with Y levels where counts start from root which has assigned the level 0 (as in figure 1) is given by equation 1. After root removing, it remains b b-trees with $\mid T_{b,Y-1} \mid$ vertices each (equation 2). Number for leafs (one by one) removing is given by equation 3. Number for nodes removing (one by one, from level k, k = $\overline{1, Y-1}$) is given by equation 4. The general formula giving by the all substructures sizes and counts (ASSC) after removing one arbitrary vertex is in equation 5:

$$\mid T_{b,Y} \mid = \frac{b^{Y+1}-1}{b-1} \tag{1}$$

$$\mid T_{b,Y} \mid \backslash Root = bX^{\frac{b^{Y}-1}{b-1}} \tag{2}$$

$$\mid T_{b,Y} \mid \backslash Leaf(s) = b^{Y}X^{b\frac{b^{Y}-1}{b-1}} \tag{3}$$

$$\mid T_{b,Y} \mid \backslash Node_k = b^k(bX^{\frac{b^{Y-k}-1}{b-1}} + X^{\frac{b^{Y+1}-b^{Y+1-k}}{b-1}}) \tag{4}$$

$$ASSC(T_{b,Y}) = bX\frac{b^{Y}-1}{b-1} + b^{Y}X^{b\frac{b^{Y}-1}{b-1}} + \\ + \sum_{k=1}^{Y-1} b^k(bX^{\frac{b^{Y-k}-1}{b-1}} + X^{\frac{b^{Y+1}-b^{Y+1-k}}{b-1}}) \tag{5}$$

where aX^b designate a number of a connected substructures (also trees) with b vertices. Remarks: For Y = 0 only the equation 1 had sense; For Y = 1 the equations 1-3 should be applied; For Y > 1 all equations 1-5 had sense and should be applied.

4. The Polynomial Formula

Assigning the power of 0 at X in formula from equation 1, the polynomial formula giving the number and sizes of substructures (NSS) which it result after removing of one vertex from a b-tree can be written as in equation (6).

Extension of node removing to k = 0 are threated by equation 7, and to k = Y by equation 8. Rewriting of equation 6 by taking into account of equations 7 and 8 gives quation 9. Rearranging of equation 9 leads to 10 (remark: all equations from 6 to 10 assumes that Y > 1):

$$NSS(T_{b,Y}) = \frac{b^{Y+1}-1}{b-1}X^0 + bX^{\frac{b^Y-1}{b-1}} + b^Y X^{b\frac{b^Y-1}{b-1}} + + \sum_{k=1}^{Y-1} b^k (bX^{\frac{b^{Y-k}-1}{b-1}} + X^{\frac{b^{Y+1}-b^{Y+1-k}}{b-1}}) \tag{6}$$

$$| T_{b,Y} \setminus Node_0 | = bX^{\frac{b^Y-1}{b-1}} + X^0 = | T_{b,Y} \setminus Root | - X^0 \tag{7}$$

$$| T_{b,Y} \setminus Node_Y | = b^Y (bX^0 + X^{b\frac{b^Y-1}{b-1}}) = | T_{b,Y} \setminus Leaf(s) | - b^{Y+1}X^0 \tag{8}$$

$$NSS(T_{b,Y}) = \frac{b^{Y+1}-1}{b-1}X^0 - (b^{Y+1}+1)X^0 + + \sum_{k=1}^{Y-1} b^k (bX^{\frac{b^{Y-k}-1}{b-1}} + X^{\frac{b^{Y+1}-b^{Y+1-k}}{b-1}}) \tag{9}$$

$$NSS(T_{b,Y}) = \sum_{k=0}^{Y} b^k (bX^{\frac{b^{Y-k}-1}{b-1}} + X^{\frac{b^{Y+1}-b^{Y+1-k}}{b-1}}) - b\frac{b^{Y+1}-2b^Y+1}{b-1}X^0 \tag{10}$$

5. Discussion of Two Particular Cases

The binary tree ($b = 2$) formula is obtained easily from equation 6 replacing b with 2:

$$NSS(T_{2,Y}) = (2^{Y+1}-1)X^0 + 2X^{2^Y-1} + 2^Y X^{2^{Y+1}-2} + + \sum_{k=1}^{Y-1} 2^k (2X^{2^{Y-k}-1} + X^{2^{Y+1}-2^{Y+1-k}}) \tag{11}$$

For Y = 0 (only the root is present): $NSS(T_{2,0}) = X^0$, meaning that no vertex cuts are available; our tree has just one vertex. For Y = 1 (1 root, 2 leafs): $NSS(T_{2,1}) = 3X^0 + 2X + 2X^2$. For Y = 2 (1 root, 2 nodes, 4 leafs): $NSS(T_{2,2}) = 7X^0 + 2X^3 + 4X^6 + 2(2X + X^4)$. The unary tree (path) formula 12 is obtained as limit formula ($b \longrightarrow 1$) of equation 10 (remark: formula 12 is according with the expected result; rearranging of 12 leads to 13):

$$NSS(T_{1,Y}) = \sum_{k=0}^{Y} (X^{Y-k} + X^k) - (1-Y)X^0 \tag{12}$$

$$NSS(T_{1,Y}) = 2\sum_{k=0}^{Y}(X^k) + (1-Y)X^0 = 2\sum_{k=1}^{Y}(X^k) + (Y+1)X^0 \tag{13}$$

In fact, there are (Y + 1) vertices, and cutting by each vertex leads to 13.

6. Concluding Remarks

The obtained polynomial formulas for vertex cuts in b-trees can be generalized, as present work do, allowing calculations of structures for any b and any Y, formula working also as limit formulas for trivial trees, the paths ($b = 1$).

References

[1] Wikipedia, B-tree definition, http://en.wikipedia.org/wiki/B-tree, (2006).
[2] Bayer R: Binary b-Trees for Virtual Memory, ACM-SIGFIDET, 5B (1971) 219-235.
[3] Shasha D, Goodman N: Concurrent Search Structure Algorithms, ACM T Database Syst, 13 (1988) 53-90.
[4] Lu H, Sahni S: A B-tree dynamic router-table design, IEEE Trans Comput, 54 (2005) 813-824.
[5] Awerbuch B, Scheideler C: The Hyperring: A Low-Congestion Deterministic Data Structure for Distributed Environments. Proc Ann ACM-SIAM Symp Discr Algorit, 15 (2004) 311-320.
[6] Edemenang EJA, Garba EJD: Dynamic information storage algorithms. Advanc Model Anal A, 19 (1994) 17-64.
[7] Laszloffy A, Long J, Patra AK: Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive hp finite element simulations, Parallel Comput, 26 (2000) 1765-1788.
[8] Vitter JS: External memory algorithms and data structures: deaimg with massive data, ACM Comput Surv, 33 (2001) 209-271.
[9] Ko P, Aluru S: Obtaining provably good performance from suffix trees in secondary storage, Lect Not Comp Sci, 4009 (2006) 72-83.
[10] Yang X, Bouguettaya A, Medjahed B, Long H, He W: Organizing and Accessing Web Services on Air, IEEE Trans Syst, Man, Cybern Part A: Syst Human, 33 (2003) 742-757.
[11] Comer D: Ubiquitous b-tree, ACM Comput Surv, 11 (1979) 121-137.
[12] Schrapp M: 1-Pass Top-Down Update Schemes for Search Trees. Design, Analysis and Application, Forts-Berich VDI-Zeitsch, R 10: Angew Inform, 38 (1984) 106p.
[13] Lehman PL, Yao SB: Efficient Locking for Concurrent Operations on b-Trees, ACM T Datab Syst, 6 (1981) 650-570.
[14] Skopal T, Krátký M, Pokorný J, Snášel V: A new range query algorithm for Universal B-trees, Inform Syst, 31 (2006) 489-511.
[15] Kim S-W: On batch-constructing B+-trees: Algorithm and its performance evaluation, Inform Science, 144 (2002) 151-167.
[16] Wang L-S, Yuan S-G, Ouyang Z, Zheng C-Z: Important algorithms used in the target parsing system, J Chin Chem Soc, 59 (2001) 241-246.
[17] Sorensen MM: b-tree facets for the simple graph partitioning polytope, J Comb Optim, 8 (2004) 151-170.
[18] Diudea MV, Gutman I, Jäntschi L: Molecular Topology, Nova Science, Huntington, New York, (2002).
[19] Diestel R: Graph Theory, Springer-Verlag, New York, (2000).

Technical University of Cluj-Napoca, 400641 Cluj, Romania
E-mail address: lori@academicdirect.org

Technical University of Cluj-Napoca, 400641 Cluj, Romania
E-mail address: carmen@j.academicdirect.ro

"Iuliu Hatieganu" University of Medicine and Pharmacy, 400349 Cluj, Romania
E-mail address: sorana@j.academicdirect.ro