A formula for vertex cuts in b-trees

LORENTZ JÄNTSCHI[1], CARMEN ELENA STOENOIU[2], SORANA-DANIELA BOLBOACĂ[3]

[1] Technical University of Cluj-Napoca,
15 Constantin Daicoviciu Street, 400020 Cluj-Napoca, Romania
http://lori.academicdirect.org
lori@j.academicdirect.org

[2] Technical University of Cluj-Napoca,
15 Constantin Daicoviciu Street, 400020 Cluj-Napoca, Romania
http://carmen.academicdirect.ro
carmen@j.academicdirect.ro

[3] "Iuliu Hațieganu" University of Medicine and Pharmacy
13 Emil Isac Street, 400023 Cluj-Napoca, Romania
http://sorana.cademicdirect.ro
sorana@j.academicdirect.ro

Key Words and Phrases: Graph theory, B-tree, Polynomial formula

AMS Subject Classification: 05C05 (Trees), 05C10 (Topological graph theory, embedding), 05C85 (Graph algorithms), 05C90 (Applications), 11T06 (Polynomials)

The paper presents a polynomial formula giving the number and size of substructures that result after removing of one vertex from a b-tree.

The solution proposed for this problem is presented by using of a polynomial formula. Two particular cases are presented.

The obtained polynomial formulas for vertex cuts in b{}-trees can be generalized, allowing calculations of any structures of interest. The obtained formula works also as limit formulas for trivial trees, which are paths.

A formula for vertex cuts in b-trees

Lorentz JÄNTSCHI

Carmen Elena STOENOIU

Sorana-Daniela BOLBOACĂ

Introduction

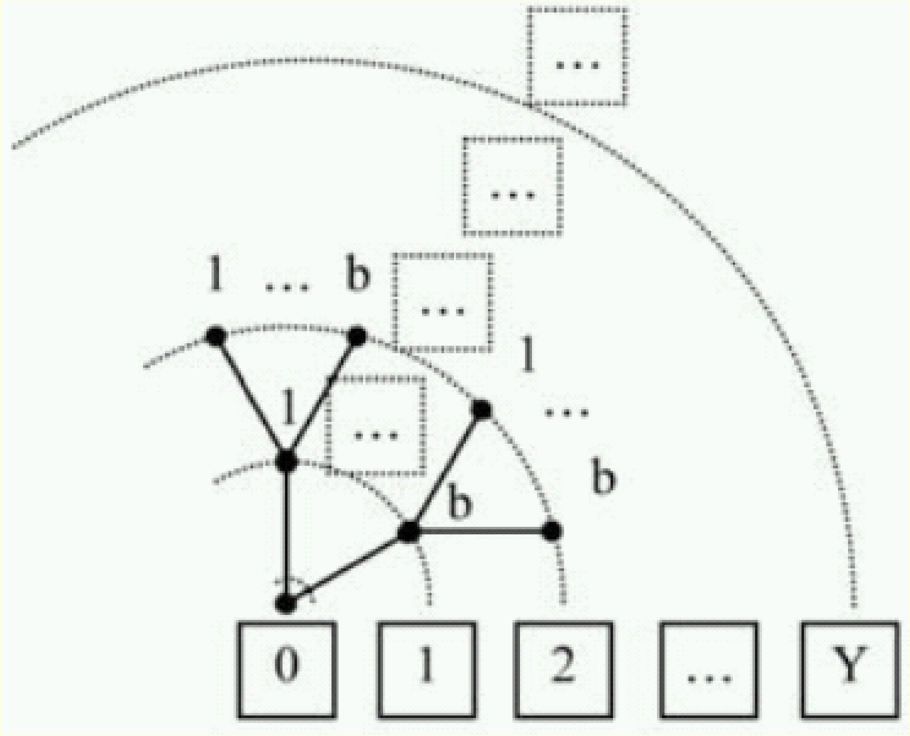
- In computer science, b-trees are tree data structures that are most commonly found in databases and file systems; *b*-trees keep data sorted and allow amortized logarithmic time insertions and deletions (see [1, 2]).
- There are at least three domains where the b-trees concepts were use in researches:
- *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].

Motivation & aim

- 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.

The problem

- A graphical representation of a b-tree is given in figure.
- 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.



The solution

- Three remarks: (1) the root vertex has b edges; (2) the leaf vertices have 1 edge; (3) all other vertices have (b +1) edges.
- The total number of vertices (TNV) in a b-tree with Y levels where counts start from root which has assigned the level 0 is given by 1-st equation.
- After root removing, it remains b b-trees with $|T_{b,Y-1}|$ vertices each (2-nd equation).
- Number for leafs (one by one) removing is given by 3-rd equation.
- Number for nodes removing (one by one, from level k, k = 1..Y-1 is given by 4-th equation.
- The general formula giving by the all substructures sizes and counts after removing one arbitrary vertex is in 5-th equation.

Polynomials calculations

$$(1) \qquad | T_{b,Y} | = \frac{b^{Y+1} - 1}{b - 1}$$

$$(2) \qquad | T_{b,Y} | \backslash Root = bX^{\frac{b^Y - 1}{b-1}}$$

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

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

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

$$+ \sum_{k=1}^{Y-1} b^k (bX^{\frac{b^{Y-k} - 1}{b-1}} + X^{\frac{b^{Y+1} - b^{Y+1-k}}{b-1}})$$

Polynomials remarks

- aX^b designate a number of a connected substructures (also trees) with b vertices.
- 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.
- Assigning the power of 0 at X in formula from eq.1, the polynomial formula giving the number and sizes of substructures which it result after removing of one vertex from a b-tree is as in eq.6.
- Node removing to k = 0 are in eq.7.
- Node removing to k = Y are in eq.8.
- Eq.6 + eq.7 + eq.8 produces eq.9. Rearranging of eq.9 leads to eq.10
- All eq.6-eq.10 assumes Y>1)

Rearranging polynomial formula

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

$$+ \sum_{k=1}^{Y-1} b^k (bX^{\frac{b^{Y-k}-1}{b-1}} + X^{\frac{b^{Y+1}-b^{Y+1-k}}{b-1}})$$

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

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

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

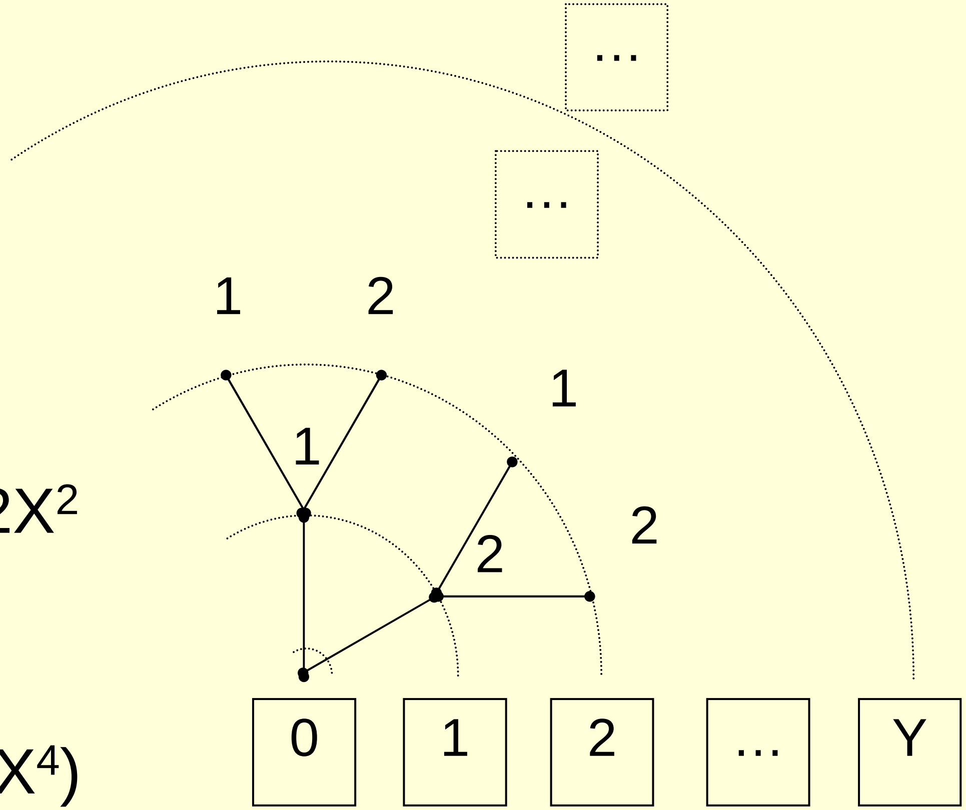
$$+ \sum_{k=1}^{Y-1} b^k (bX^{\frac{b^{Y-k}-1}{b-1}} + X^{\frac{b^{Y+1}-b^{Y+1-k}}{b-1}})$$

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

Binary trees

- Y=0 (just root): $NSS(T_{2,0})=X^0$
- Y=1 (1 root, 2 leafs): $NSS(T_{2,1}) = 3X^0 + 2X^1 + 2X^2$
- Y=2 (1 root, 2 nodes, 4 leafs): $NSS(T_{2,2}) =$

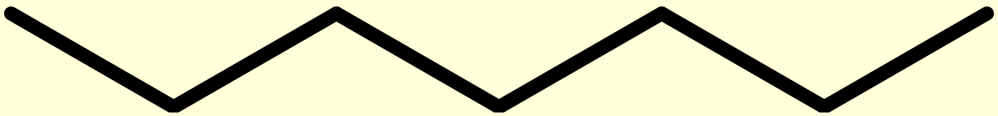
$7X^0 + 2X^3 + 4X^6 + 2(2X + X^4)$



$$NSS(T_{2,Y}) = (2^{Y+1}-1)X^0 + 2X^{2^Y-1} + 2^Y X^{2^{Y+1}-2} +$$

$$+\sum_{k=1}^{Y-1} 2^k (2X^{2^{Y-k}-1} + X^{2^{Y+1}-2^{Y+1-k}})$$

Paths



- The unary tree (path) formula is obtained as limit formula (b → 1) of general formula (10)
- Rearranging of the formula shows that In fact, there are (Y + 1) vertices, and the cutting by each vertex leads to:

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

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

Conclusions

- The obtained polynomial formulas for vertex cuts in b-trees are generalized, allowing calculations of structures for any b and any Y.
- The obtained formula works also as limit formulas for trivial trees, which are paths.
- The b-trees, also called dendrimers, having important fundamental applications, were systematically cut and resulted number of substructures and sizes were expressed as polynomials formulas, which allow particularization for any specific structure, shorting thus the calculation time of these.

Acknowledgments

- This research was partly funded by UEFISCSU Romania.
- First author thank to Prof. Mircea V. DIUDEA from “Babeş-Bolyai” University of Cluj-Napoca, Romania, for the helpful discussions on topics related to the subject, counting polynomials on square matrices.

- Thank you for your attention!

References (1/4)

- [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.

References (2/4)

[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.

References (3/4)

[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, Kratky M, Pokorny J, Snasel 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.

References (4/4)

[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, Jantschi L: Molecular Topology, Nova Science, Huntington, New York, (2002).

[19] Diestel R: Graph Theory, Springer-Verlag, New York, (2000).