An Improved Method for Generating Recursion Trees

Baase and Van Gelder define an algorithm for generating a recursion tree. First let's look at the algorithm they defined. Then we'll repeat our work with Figure 4.5 using the notation Baase and Van Gelder suggest. I believe that this notation makes the manual construction of recursion trees less error-prone for students.

Definition 3.5 (Baase and Van Gelder) Recursion tree rules

1. The *work copy* of the recurrence equation uses a different variable from the original copy; it is called the *auxiliary variable*. Let *k* be the auxiliary variable for the purposes of discussion. The left side of the original copy of the recurrence equation (let's assume it is *T*(*n*)) becomes the size field of the root node for the recursion tree.

2. A node has two fields: size and non-recursive cost. An incomplete node has a value for its size field, but not for its non-recursive cost.

3. The process of determining the non-recursive cost field and the children of an incomplete node is called the *expansion* of that node. We take the size field in the node to be expanded and substitute it for the auxiliary variable *k* in our work copy of the recurrence equation. The resulting terms containing *T* on the right side of that equation become children of the node being expanded; all remaining terms become that node's non-recursive cost

4. If you have generated a recursion tree to its base case, expanding a base case size gives the non-recursive cost field and no children. (usually assume base case cost of one)

In any subtree of the recursion tree, the following equation holds:

Equation 3.7 (Baase and Van Gelder) $size\ field\ of\ root = \sum nonrecursive\ cost\ of\ expanded\ nodes + \sum size\ fields\ of\ incomplete\ nodes$

An example of using the Baase and Van Gelder approach is given in the following two pages.

Source For Improved Recursion Tree

Computer Algorithms: Introduction to Design and Analysis, Third Edition, Sara Baase and Allen Van Gelder, Addison Wesley Longman, 2000.

[no eBook available but the book is in print in the Sheridan Libraries QA76.9.A43 B33 2000 and QA76.6 .B251 1978]

Algorithm is Baase and Van Gelder, Section 3.7 Recursion Trees pages 134-137

Example using Figure 4.5, page 96 Page 1 of 2

Figure 4.5 recurrence $T(n) = 3T(n/4) + cn^2$

Work copy of the recurrence $T(k) = 3T(k/4) + ck^2$ using auxiliary variable *k*

Each node in the improved recurrence tree will be of the form: | *T*(*size*) | *Nonrecurring cost* |

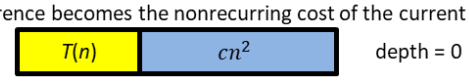
---

(1) Create the root node, depth=0 of the recursion tree. Note that the similarity to Figure 4.5(a). This node is an incomplete node. It is the current node.

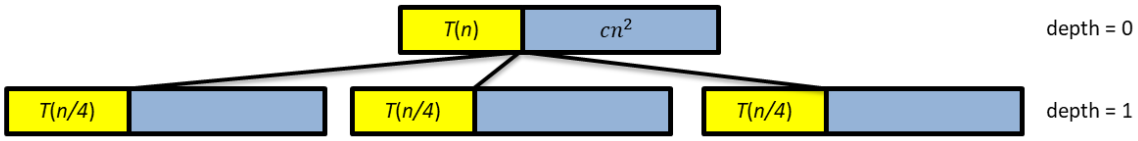
(2) Expand the current node, to form depth=1 nodes of the recursion tree.

(2)(a) Take the size field in the node to be expanded and substitute it for the auxiliary variable *k* in our work copy of the recurrence equation. The size field is *n*. The work copy with *k* = *n* is $T(n) = 3T(n/4) + cn^2$

(2)(b) The nonrecursive cost in the work copy of the recurrence becomes the nonrecurring cost of the current node.



(2)(c) Create children of the current node using the recursive cost term of the work copy of the recurrence. In this case. three child nodes are created each with a size of *n*/4



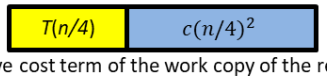
Note the similarity to Figure 4.5(b)

Example using Figure 4.5, page 96 Page 2 of 2

(3) Expand a current node, to form depth=2 nodes of the recursion tree. In this recursion expanding any one of the nodes at depth = 1 produces the same subtree below it for depth = 2. For a correct and complete analysis using a recursion tree all nodes must be expanded so that the work at each level of the tree can be computed. Currently, the sum of nonrecursive costs at depth = 0 is cn^2 and the sum of nonrecursive costs at depth = 1 is unknown. It will be known after expanding each of the nodes at level 1.

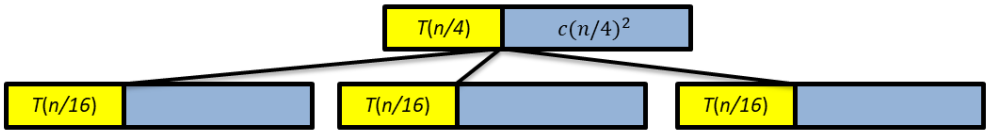
(3)(a) Take the size field in the node to be expanded and substitute it for the auxiliary variable *k* in our work copy of the recurrence equation. The size field is *n/4*. The work copy with *k* = *n/4* is $T(n/4) = 3T(n/16) + c(n/4)^2$

(3)(b) The nonrecursive cost in the work copy of the recurrence becomes the nonrecurring cost of the current node.



depth = 1

(3)(c) Create children of the current node using the recursive cost term of the work copy of the recurrence. In this case. three child nodes are created each with a size of *n*/4



depth = 1

depth = 2

Note the similarity to Figure 4.5(c), remembering that I have illustrated expanding only one of the nodes at depth = 1. The sum of nonrecursive costs at depth = 1 is now known, $3c(n/4)^2$ or $3cn^2/16$ or as in Figure 4.5 (3/16)cn^2.

(4) You would repeat this process of taking the size field in the node to be expanded and substituting it for *k* in the work copy, use the work copy's nonrecursive term to populate the nonrecursive cost part of the current node, and create children of the current node using the recursive cost of the work copy until you reach a base case or can determine the height of the tree and the work at each depth of the tree.

This algorithm can be programmed without having to have a layout utility to make the actual graph. The important thing is the methodical determination of the recursive cost and size of children which often fall to mistakes by students during the process of creating recursion trees.