ADS

Group Activity :-

Y–Trees Data Structure

111903091 Saurabh Deokate
111903132 Yashwant E Ingle
111903033 Ghansham Salunke

- Introduction

About Y – Tress -

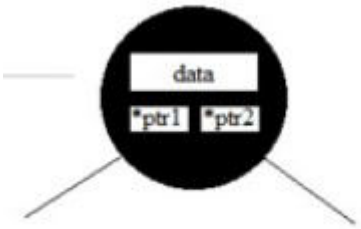
It is basically an acyclic graph where each root node of a level can explore min and max 3 nodes such that:

- It has at least one special node **O** called origin at the zero-level of the tree pointing to the next level nodes.
- The other elements in Y-tree (if any) are sorted in a clockwise sense at each level.

Terminology -

Some Basic Terms:

1) Node : Each Element of a tree is a node where origin is a speacial type of node. A Node contain filelds like- data, Pointer1, Pointer2.



2) Origin : It is a special type of node which consist of of 3 pointers to the other node and no data element is present. It is also called as 0^{th} level node.

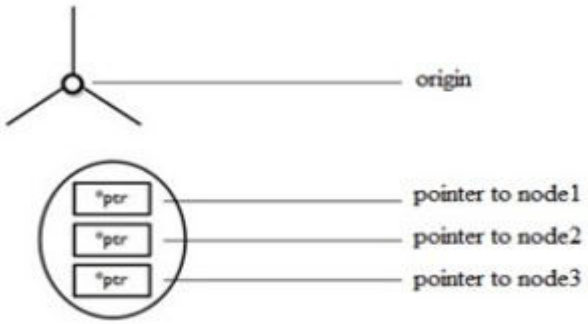
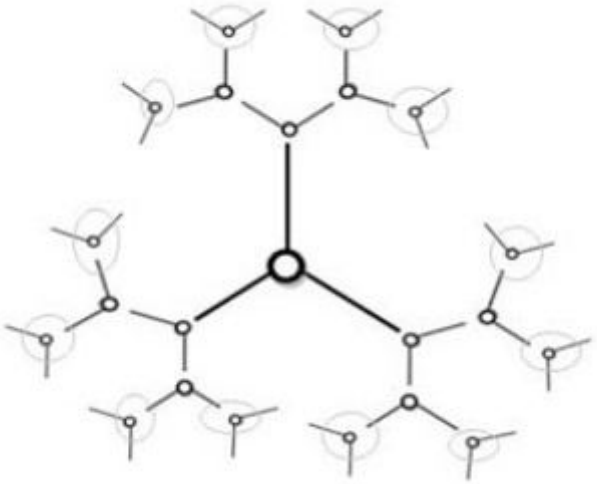


fig 2.2

3) Parent/Child : Immidiate node of preceding level is called parent and immediate node in its succeeding level is called its child.

4) Siblings : If two nodes are at same level then they are called as Siblings.

Some Basic Calculations :



a) Number of nodes in the k th level of a Y-Tree :

Let n k-1 =no. of nodes in k-1 th level.

Then, no. of nodes in k th level = n k-1 * 2

b) Number of distinct paths in the k th level Y-Tree

Let number of distinct paths to reach a kth level node=N

N = 3 C1 x 2 C1 x2 C1 x..... n-1 times

= 3 C1 x (2 C1) n-1

= 3 x 2 n-1

c) Total Number of nodes in the k th level Y-Tree

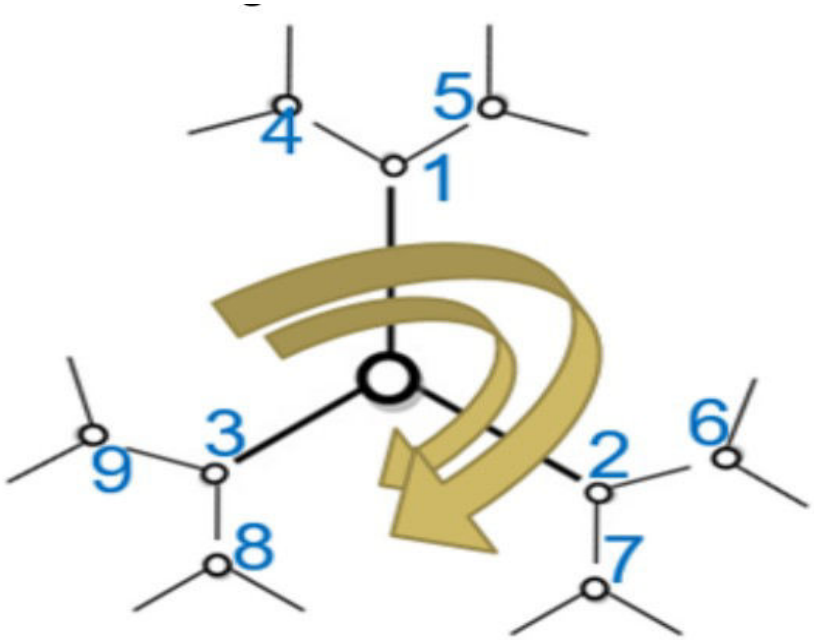
The total number of nodes in a k-level Y-Tree is given as:

T (k) = 3x2 k-1 + 3x2 k-2 +3x2 k-3 + … + 3x2 0

= 3 (2 k-1 + 2 k-2 +2 k-3 + … +1) {by summation of G.P.}

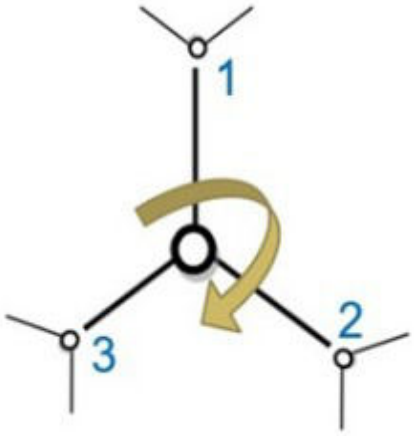
= 3 (2 **k - 1)

Pecular Characteristis -



- The origin node of 1st level contains only pointers to all other nodes.

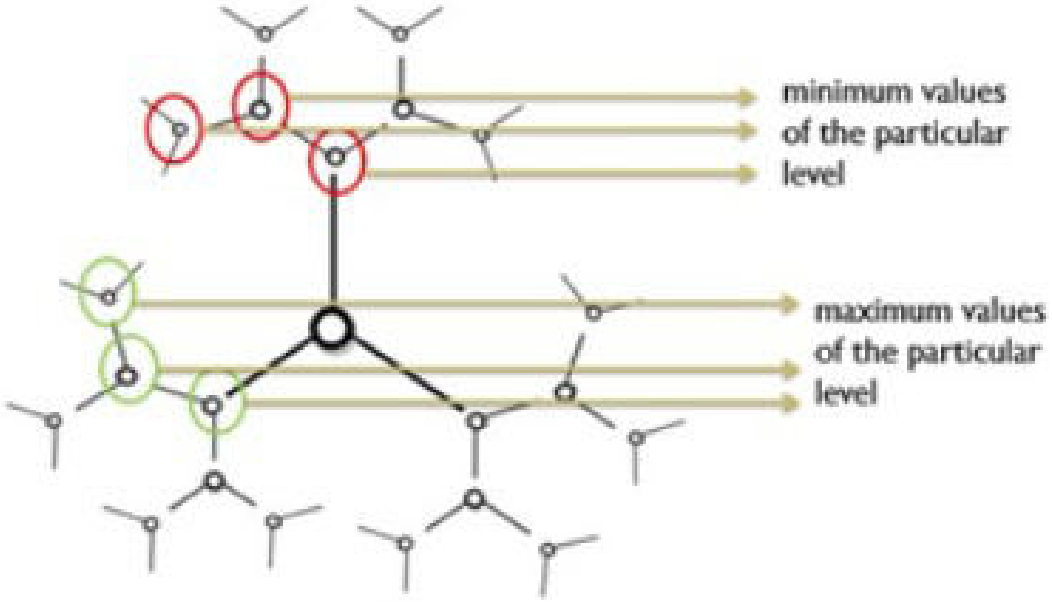
- The child of every origin are arranged in sorted and clockwise direction.



- The Min element of a particular level is present at the Starting position of every level.

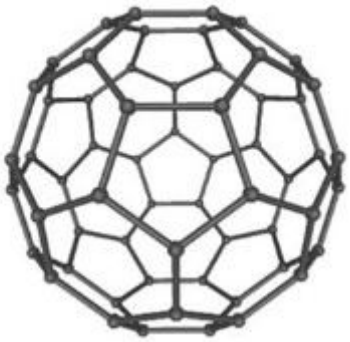
- The Max element of a particular level is present at the Ending position of every level.

- For 1st level, there are three child nodes but for rest of the level there are only two.

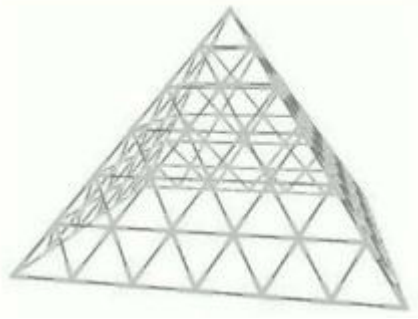


Varients of Y-tress -

1) Football version



2) Pyramid



Operations -

Traverse -

TRAVERSE Let YTREE be a k^{th} level Y-Tree in memory. This algorithm traverses YTREE applying an operation PROCESS to each element in it. START is the pointer to the root of the YTREE and the pointer variable PTR points to the node currently being processed. The variable L is used to run through the levels till level k.

1. Set PTR := START, L := 1;
2. Repeat Step 3 while PTR != NULL
3. Repeat for i=0 to $3*2^{L-1}$ on i=i+1
4. Apply PROCESS to PTR->DATA
5. PTR=PTR->SIBLING

[End of Step 3 loop]

6. PTR=PTR->LCHILD
7. L++
8. Go to step 3

[End of Step 2 loop]

9. End

Findmax -

FINDMAX(m) Let YTREE be a k-level Y-Tree in memory. This algorithm sets MAXENTRY as the maximum entry that occurs at level 'm'.

1. Set PTR := START, m := 0
2. Repeat Step 3 for i=1 to m on i=i+1
3. PTR=PTR->RCHILD

[End of Step 2 loop]

4. Set MAXENTRY=PTR->DATA
5. End

Search_by_range -

SEARCHBYRANGE(YTREE, START, ITEM, LNO, NNO)

YTREE is a sorted k-level Y-Tree in memory. This algorithm finds the location of the node where an ITEM of information first appears. Correspondingly it sets LNO to the level number and NNO to the node number of the match or sets them as 0 if no match is found. Variable L is to traverse through the levels.

1. Set PTR=START, LNO=0, NNO=0
2. Repeat Step 3 to 7 for i=1 to k on i=i+1
3. PTR=PTR->RCHILD
4. If PTR->DATA == ITEM
5. Set LNO=i, NNO=$3*2^{i-1}$
6. Else if PTR -> DATA > ITEM
7. Break from loop

[End of If structure]

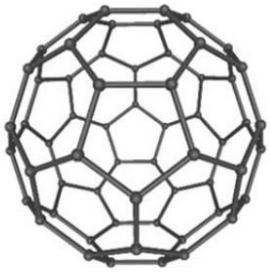
[End of Step 2 loop]

6. Repeat for j=1 to $3*2^{i-1}$ on j=j+1
7. If (PTR->DATA == ITEM)
8. Set LNO=L, NNO=I;
9. Else
10. PTR=PTR->SIBLING

9. End

Applications -

- The presented data model finds numerous applications when it comes to minimizing the search/traverse cycles. Y-trees may be deployed everywhere where there is a data storage system.
- One may apply in concepts in a bank based data storage server or on large scale data storage cells.
- Once a basis is created, one may also link different y-trees together to form geometric structures (fig 8) like a football in which the nodal elements are connected in a football structure or a pyramid in which we maintain the structure of a pyramid.
- This can aid to manage the data more effectively and offer another set of vivid properties.



Complexity analysis -

Insert-

Depends on number of levels in Y-tree

Here number of total nodes in Y-tree is given by,

n = 3*(2**k - 1)

taking log on both sides

we get,

k ~= log(n)

Time complexity - O(log(n))

Traversal -

In traversal we visit each node traverse each node's level and then do it repeatedly for each sibling.Then move to the next level.

We are traversing at each kth level which contain 2/3 nodes.Say total nodes be n so traversing n nodes leads the complexity to be **O(n) where n depends on k(no of levels).**

Search_by_range -

1) Best Case Complexity:

The best case occurs when either the ITEM to be searched is the MAXENTRY/MINENTRY of level k.
As it is apparent from above, we require ‘k’ iterations to reach the MAXENTRY in level ‘k’. Therefore, complexity is of the order O(k).
Total number of nodes (n) = 3(2 k -1)
Hence, complexity in terms of n = O(log(n)/log2) ≈ O(log(n)).

2) Worst Case Complexity

Worst Case occurs when the ITEM of information is present in the 2 nd entry of the k th level in the Y-Tree i.e. just after the MINENTRY.
In this situation, we have to travel to the MAXENTRY first and then traverse through all the SIBLING nodes to reach to the second entry of the k th level.

Time Complexity to reach MAXENTRY = O(log n) {from FINDMAX complexity}
Time Complexity to traverse midway through SIBLING nodes = O(3(2 k-1)) ≈ O(n/2)
Therefore total time complexity = O(log n) + O(n/2) = O(n/2)

3) Average Case Complexity

Average Case occurs when the ITEM of information resides midway between the MAXENTRY and the MINENTRY.
In this case we have to reach the MAXENTRY first and then traverse through the SIBLING pointer links to the ITEM of information.
Time Complexity to reach MAXENTRY = O(log n) {from FINDMAX complexity}
Time Complexity to traverse midway through SIBLING nodes = O(3(2 k-1 /2)) = O(3(2 k-2))

≈ O(n/4)

Therefore total time complexity = O(log n) + O(n/4) = **O(n/4)**

FINDMAX

From algorithm 1 we note that it takes k iterations to find the MAXENTRY in a k-level Y-Tree.
Thus time complexity of the FINDMAX algorithm is O(k).
Now since, total number of nodes in the kth level Y-Tree is 3(2 k -1), the complexity in terms of total number of nodes (n) becomes O(log(n)/log2) ≈ **O(log(n)**