Optimizing the Search in Hierarchical Database using Quad Tree
Deepak Kumar Sharma, Sonia Vatta
School of Computer Science and Engineering Bahra University Waknaghat Solan, Himachal Pradesh, India

ABSTRACT

This research work is done to optimize the search in hierarchal database by using Quad Tree. Quad Tree is a tree which consists of a node having 4 sub child nodes.[1] Quad Tree is used in many fields such as computer graphics to find out a desired pixel in graphics and collision detection in 3-D gaming. There are various other applications of quad tree in different fields like weather forecasting, geographical survey and study of mutation rate with a change of environment. The main focus of this work was optimizing the search for which the main areas of concentration were various searching techniques using Quad Tree and finding out the best solution. To fulfill the purpose, a new approach is designed which results in better representation of data and enhancing the speed of search. This work is done on the basis of the value of the root node. Further based on this value, the ranges of nodes at each level are calculated. The results are also included with snapshots and comparison is given between earlier and proposed techniques.

Keywords: Quad Tree, Memory Management, Search, Result

I. INTRODUCTION

The quad tree is widely in practice in detecting collision in 2D and 3D games and pixel processing to detect image in different image resolution.[2] Many games require the use of collision detection algorithms to determine when two objects have collided, but these algorithms are often expensive operations and can greatly slow down a game.[3][4][5][6][7] As such the collision detection is very costly process in gaming. So in many games we use Quad Tree to detect collision. The Quad Tree is too used to process the polygon or images in graphics. The techniques used to process the polygons in graphics are point to point processing, edge to edge processing and region by region processing. Commonly used techniques for collision detection is spatial portioning and bit-interleaving [8][9][10][11][12].

In the research paper Quad tree: an approach of traversing and searching published in IJRITCC volume 3 issue 5 [74] the searching time of the information in database is reduce.

II. METHODS AND MATERIAL

A. Searching Algorithm

To search an element in a quad tree, the new design technique is as follows:

Steps for Searching Item

SEARCHING (ROOT)

1. Put ITEM in LOC. Take RANGE = ROOT
2. Calculate range of each sub node of level i
   a. QLN has range 0 to RANGE/2^2*i
   b. BLN has range RANGE/2^2*i+1 to 2*RANGE/2^2*i.
   c. BRN has range 2*RANGE/2^2*i + 1 to 3*RANGE/2^2*i
   d. QRN has all values lies between 3*RANGE/2^2*i+1 to RANGE
3. Selection of initial node to traverse for searching
4. i=1
   If (0<ITEM< RANGE)
{  
    if (ITEM < RANGE/2^2*i)  
        // search QLN  
        {  
            if (LOC == INFO NODE)  
                return to ROOT;  
            i++;  
        }  
    else  
        Traverse QLN SUB NODE;  
    SEARCHING (ROOT);  
}  
else if(ITEM <2*(RANGE/(2^2*i)) to (ITEM>(RANGE/(2^2*i))))  
  {  
    if (LOC == INFO NODE)  
        return to ROOT.  
    i++;  
    else  
        Traverse BLN SUB NODE;  
    SEARCHING (ROOT);  
}  
else if(ITEM <3*(RANGE/(2^2*i)) to (ITEM>2*(RANGE/(2^2*i))))  
  {  
    if (LOC == INFO NODE)  
        return to ROOT.  
    i++;  
    else  
        Traverse BRN SUB NODE Searching (ROOT);  
}  
else if(ITEM <4*(RANGE/(2^2*i)) to (ITEM>3*(RANGE/(2^2*i))))  
  {  
    if (LOC == INFO NODE)  
        return to ROOT.  
    i++;  
    else  
        Traverse QRN SUB NODE Searching (ROOT)  
}  
else  
{  
    ITEM NOT FOUND;  
}

Figure 1: Showing Algorithm Approach Of Searching Node

B. Proposed Algorithm

According to the proposed technique, the nodes are traversed according to the ranges defined. Now when they require node containing the element is searched to find out the new node or for next search pointer backtrack from one previous node to another until root node arrived.

In this technique, the time requires in between one search and next search depends on the number of backtracking done to reach root node.

Suppose 1 millisecond time is requiring for comparison and 1 millisecond time for backtracking from one node to another.

C. Run Time Complexity of Searching Algorithm

The quad tree searching in this algorithm is based on the somewhat divide and conquer approach. In this approach, the nodes are divided into four groups. And the data of specific range is available in specific group. In this tree, the data is already in sorted manner and the ranges will be based on the value of root.

To find the ITEM in the tree, as in the algorithm, the range is checked and then the elements are further searched. In the algorithm, the tree node is divided into quadrants i.e. into four groups of nodes. Thus the recurrence relationship of the algorithm is as

\[ T(n) = T(n/2^2*i) + \{T(n/2^2*i) + c0\} + c1 + c2 + c3 + c4 + c5 + c6 + c7 \]

Where, T(n) is the time complexity and c0, c1, c2, c3, c4, c5, c6, c7 is a cost.
On simplifying the above equation we get
\[ T(n) = T(n/2^2i) + \{T(n/2^2i) + c0\} + C \]
where say, \( C = c1 + c2 + c3 + c4 + c5 + c6 + c7 \)

On further simplifying the equation we have \( T(n) = T(n/2^2i) + S \), where we can say \( S = \{T(n/2^2i) + c0\} + C \)

Now, on simplifying the equation the \( T(n) = T(n/2^2i) + S \) recursively where we assume \( n \)th times of \( i \)th term i.e. \( 2^i \times n = k \) and we assume \( n = 2k \). As we find the complexity in upper bound of time space.

Thus the complexity of the algorithm in run time is \( O(\log_2 n) \)

III. RESULTS AND DISCUSSION

Implementation of Algorithm

a) Example Quad Tree for Implementation

\[ \text{Example Quad Tree} \]

\[ \text{Implemented Code Result} \]

b) Implemented Code Result

```
0—exit, 1—insert, 2—Inorder representation, 3—search item

Enter choice: 1
Enter data to insert: 192

Enter choice: 1
Enter data to insert: 152

Enter choice: 1
Enter data to insert: 56

Enter choice: 1
Enter data to insert: 34

Enter choice: 1
Enter data to insert: 100

Enter choice: 2
34->56->192->100->152->

Enter choice: _
```
IV. REFERENCES


[10] Hanan Samet computer science Department University of Maryland college park, USA


[14] Ivan •Sime•cek, Sparse Matrix Computations using the Quad tree storage format Department of Computer Science and Engineering, Czech Technical University, Prague


[73] Quad Tree: New Approach of Representing and Traversing; International Journal on Recent and Innovation Trends in Computing and Communication; ISSN: 2321-8169; Volume: 3 Issue: 5; ISSN: 2562 – 2565.

[74] Applications of Quad tree: A Review; International journal of engineering research and general science; volume: 3 Issue: 4; ISSN: 2091-2730.