**NAME: PRANAV CHITALE CLASS: SE COMPS ROLL NO: 14**

**AIM:** To perform creation, insertion and traversal operation on a KD tree.

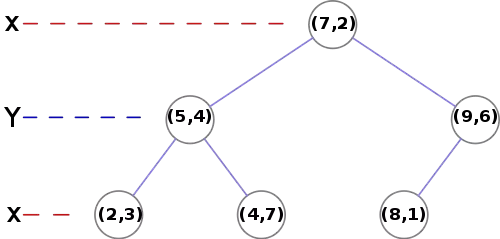
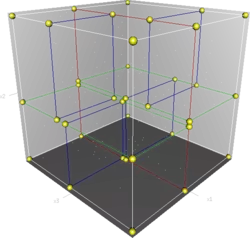
**Theory:**

A K-D Tree(also called as K-Dimensional Tree) is a binary search tree where data in each node is a K-Dimensional point in space. In short, it is a space partitioning data structure for organizing points in a K-Dimensional space.

K-D trees are a useful data structure for several applications, such as searches involving a multidimensional search key

A K-D tree has an attribute called cutting edge which denotes the dimension of comparison for a given level in the tree.

A node is inserted by comparing the dimension of the new node and parent corresponding to the cutting edge.

**Code:**

#include <stdio.h>

#include <stdlib.h>

typedef struct KDTree

{

int dl,dr;

int cuttingEdge;

struct KDTree \*left,\*right,\*parent;

}node;

node\* createNode(int data\_left, int data\_right)

{

node \*newNode = (node\*)malloc(sizeof(node));

newNode->dl = data\_left;

newNode->dr = data\_right;

newNode->cuttingEdge = 0;

newNode->parent = NULL;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

node\* findNode(node \*root, node \*temp)

{

//root = NULL

if(!root){

return temp;

}

if(root->cuttingEdge==0){

if(root->dl > temp->dl){

if(root->left!=NULL){

root->left = findNode(root->left,temp);

}

else{

temp->cuttingEdge = 1;

root->left = temp;

return root;

}

}

else{

if(root->right!=NULL){

root->right = findNode(root->right,temp);

}

else{

temp->cuttingEdge = 1;

root->right = temp;

return root;

}

}

}

else if(root->cuttingEdge==1){

if(root->dr > temp->dr){

if(root->left!=NULL){

root->left = findNode(root->left,temp);

}

else{

root->left = temp;

return root;

}

}

else{

if(root->right!=NULL){

root->right = findNode(root->right,temp);

}

else{

root->right = temp;

return root;

}

}

}

}

node\* insertNode(node \*root, int data\_left, int data\_right)

{

node \*temp = createNode(data\_left,data\_right);

root = findNode(root,temp);

return root;

}

void preOrder(node \*root)

{

if(!root){return;}

printf("(%d,%d) ",root->dl,root->dr);

preOrder(root->left);

preOrder(root->right);

}

void main()

{

node\* root=NULL;

int choice = 1,a,b;

while(choice)

{

printf("Enter Num 1 and Num 2\n");

scanf("%d %d",&a,&b);

root = insertNode(root,a,b);

printf("1: continue 0: end >>> ");

scanf("%d",&choice);

}

printf("Preorder traversal: ");

preOrder(root);

printf("\n");

}

**Input / Output:**

Enter Num 1 and Num 2

3

6

1: continue 0: end >>> 1

Enter Num 1 and Num 2

17

15

1: continue 0: end >>> 1

Enter Num 1 and Num 2

13

15

1: continue 0: end >>> 1

Enter Num 1 and Num 2

6

12

1: continue 0: end >>> 1

Enter Num 1 and Num 2

9

1

1: continue 0: end >>> 1

Enter Num 1 and Num 2

2

7

1: continue 0: end >>> 1

Enter Num 1 and Num 2

10

19

1: continue 0: end >>> 0

Preorder traversal: (3,6) (2,7) (17,15) (6,12) (9,1) (13,15) (10,19)

**Result:**

KD tree was successfully created and traversed in the pre-order manner.

**Conclusion:**

The KD tree is similar to binary search tree with an addition of the cutting edge feature.