Akshay S Bharadw

1BM18CS011

ADS lab - 6 Writeup (B-trees)

```
class node { // class for B-tree node
  int *data; // Array of elements in node
  int m; // Minimum degree / order
  node **child; // child pointer array
  int n; // number of elements in node
  bool leaf; // to check if a node is a leaf

  public :
      node (int m1, bool leaf1) ;
      void insertionnonfull (int item);
      void splitchild (int i, node *y);
      void traverse();
      ~~[illegible]~~

friend class btree; // class for B-tree
}

class btree {

   node *root;
   int m;

   public :
      btree ( int m1 ) {
          root = NULL;
          m= m1;
      }
      void traverse () {
          if (root != NULL)
              root -> traverse();
      }
      void insertion(int item);
};
```

[illegible]

Akshay S Bharadwaj

1BM18CS011

```
node :: node (int m1, bool leaf1) {
    m = m1;
    leaf = leaf1;
    data = new int [2*m - 1];
    child = new node * [2*m];
    n = 0;
}
void btree :: insertion (int item) {
    if (root == NULL) {
        root = new node (m, true);
        root -> data [0] = item;
        root -> n = 1;
    }
    else {
        if (root -> n == 2*m - 1) {
            node *s = new node (m, false);
            s -> child [0] = root;
            s -> splitchild (0, root);
            int i = 0;
            if (s -> data [0] < item)
                i++;
            s -> child [i] -> insertionnonfull (item);
            root = s;
        }
        else
            root -> insertionnonfull (item);
    }
}
void node :: insertionnonfull (int item) {
    int i = n - 1;
    if (leaf == true {
        while (i >= 0 && data[i] > item) {
```

[illegible]

Akshay S Bharadwaj
1BM18CS011

```
                data [i+1] = data[i];
                i--;
            }
            data [i+1] = item;
            n = n+1;
        }
        else {
            while ( i>=0 && data[i] > item )
                i--;
            if (child [i+1] -> n == 2*m-1) {
                splitchild (i+1 ,child [i+1] );
                if (data [i+1] < item)
                    i++;
            }
            child [i+1] -> insertionnonfull (item);
        }
    }
void node :: splitchild ( int i, node *y) {
        node *z = new node (y -> m, y-> leaf);
        z -> n = m-1;
        for (int j = 0; j < m-1 ; j++)
            z-> data [j] = y -> data [j+m];
        if (y-> leaf == false ) {
            for (int j = 0; j < m; j++)
                z -> child [j] = y -> child [j+m];
        }
        y -> n = m-1;
        for (int j = n ; j >= i+1 ; j--)
            child [j+1] = child [j];
        child [i+1] = z ;
        for (int j = n-1 ; j >=i ; j--)
            data [j+1] = data [j];
        data [i] = y -> data [m-1];
        n = n+1 ;
    }
```

AS