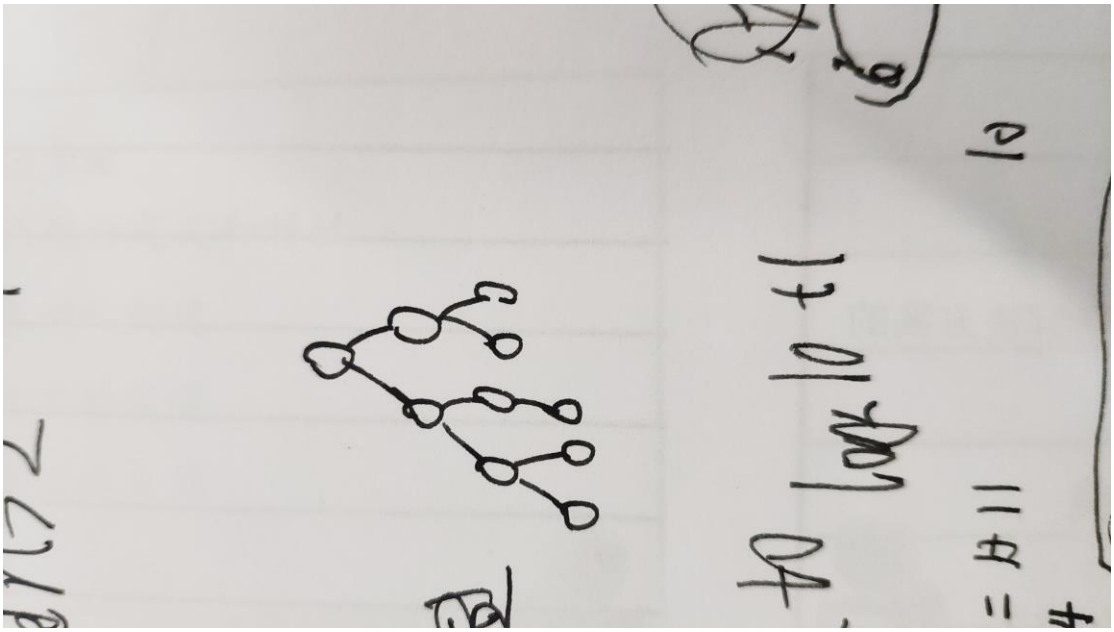
# 一、问题分析

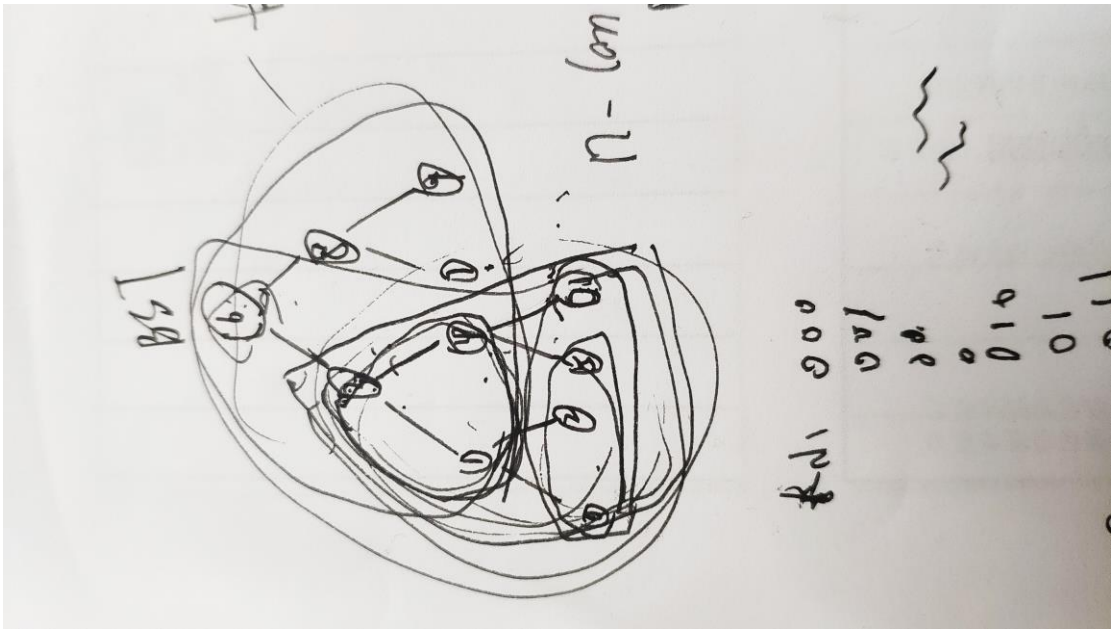
- 分析并确定要处理的对象（数据）是什么
  一组整数
- 分析并确定要实现的功能是什么
  将这组整数储存
  将这组整数插进 BST 中
  对这个 BST 树按层次遍历输出
- 分析并确定处理后的结果如何显示
  按照层次遍历递归输出
- 请用题目中样例，详细给出样例求解过程。
  输入:10

  1 2 3 4 5 6 7 8 9 0

  首先构建一棵含 10 个元素的二叉空树



  然后层次遍历该树，并按从小到大的顺序插入这组数



## 二、数据结构和算法设计

### 抽象数据类型设计：二叉树

### 物理数据对象设计

物理数据对象：基于链表实验的二叉树

```
template<typename T>
class linkTree :public BinTree<T>
{
private:
	linkNode<T>* Root;
	int size;
public:
	linkTree();
	~linkTree();
	void setRoot(linkNode<T>* rt) //设置根节点
	bool BitreeEmpty(BinNode<T>* rt)；//判断二叉树是否为空
	void clear();//清除所有元素
	void Preorder(BinNode<T>* rt, queue<T>& a);//前序输出
	void Inorder(BinNode<T>* rt, queue<T>& a);//中序输出
	void Postorder(BinNode<T>* rt, queue<T>& a);//后序输出
	void LevelOrderTraverse(queue<T>& a);//层次遍历
	linkNode<T>* get_Root() ；//返回根节点
};
```

### 算法思想的设计

输入模块

1、将输入的数储存在数组中

2、排序

构建模块

1、获取根节点位置

2、将根节点左边的数存入根节点左子树

3、将根节点右边的数存入根节点右子树

遍历模块

1、按序遍历树，将遍历元素存入 queue 中

2、将队列中的元素输出

## 关键功能的算法步骤（不能用源码）

输入模块

基本操作：cin

```
Void input（）
{
    int n;
    cin >> n;
    int a[n];
    for (int i = 0; i < n; i++)
        cin >> a[i];
    sort(a, a + n);
}
```

构建模块

基本操作：

SetLeft（），setelement（），setright（）

```
linkNode<int>* build(int* a, int pos)
{
    if (pos == 0) return NULL;
    int deep = 1;//求根节点位置
    while (pow(2, deep) - 1 < n) deep++;
    if (deep == 1) pos= 0;
    else pos= min(pow(2, deep - 1) - 1, n - pow(2, deep - 2));
    int left = position(pos);
    int right = pos - left - 1;
    linkNode<int>* node = new linkNode<int>;
    node->setElement(a[left]);//给根节点赋值
    node->setLeft(build(&a[0], left));//根节点右子树
    node->setRight(build(&a[left + 1], right));//连接左子树
    return node;
}
```

输出模块

基本操作：empty（），get_val(),right(),left()

```
void linkTree<T>::LevelOrderTraverse(queue<T>& a)
{
	queue<BinNode<T>*> q;
	if (Root == 0)
		return;
	q.push(Root);
	while (!q.empty())
	{
		//出队保存队头并访问
		BinNode<T>* front = q.front();
		a.push(front->get_val());
		q.pop();
		//将出队结点的左子树根入队
		if (front->left())
			q.push(front->left());
		//将出队结点的右子树根入队
		if (front->right())
			q.push( front->right());
	}
}
```

# 三、算法性能分析

输入模块：O（nlogn）
构建模块：O（n）
输出模块：O（n）