```
// 数据结构如下
export const data: IMenu[] = [
  {
    name: "1",
    children: [
      { name: "4" },
      { name: "5", children: [{ name: "5-1", children: [{ name:
"5-1-1", children: [{ name: "5--1-1-1" }] }] }] },
    ],
  },
  { name: "2" },
  { name: "3", children: [{ name: "6" }, { name: "7" }, { name:
"8" }] },
]

type IMenu = { name: string; children?: IMenu[] }
// 要求遍历出结果如下:
// [root, 1, 2, 3, 4, 5, 6, 7, 8 ]
// 打印结果如下:
// [root]
// [1, 2, 3]
// [4, 5, 6, 7, 8]

const bst = (root: IMenu[]) => {
  const quene: IMenu[] = [] // 一个队列, 用于遍历, 所有遍历的节点在这里排
队
  for (let k = 0; k < root.length; k++) {
    quene.push(root[k])
  }

  let deep = 0 // ! 记录层数
  while (quene.length > 0) {
    // 对当前队列内容进行遍历
    const size = quene.length

    const temp: string[] = [] // ! 将每层出列的节点暂时放在这个数组

    for (let i = 0; i < size; i++) {
      // 当前节点出列
      const node = quene.shift()
      // 看看当前节点有没有孩子节点
```

```
      let child: IMenu[] = []
      if (node) {
        if (node.children) child = node.children
        // 把每层出列的节点装起来
        temp.push(node.name)
      }

      if (child) {
        for (let i = 0; i < child.length; i++) {
          quene.push(child[i])
        }
      }
      // 在每层开始时 加一层
      if (i === 0) deep = deep + 1
    }
    // for 外层循环结束, 可以打印下这个 temp
    console.log("第" + deep + "层 ---> ", temp)
  }
}

bst(data)
```