# 数据结构转换

## 岗位信息

* 公司：阿里 cbu
* 职级：P5
* 轮次：笔试题

## 题目描述

将形如： [0, "a", 1, "b", 2, "c", 3, "e", 2, "d", 1, "x", 0, "ff"] 的一个数组转化为如下的数据。

{  
 a: {  
 b: {  
 c: {  
 e: null,  
 },  
 d: null,  
 },  
 x: null,  
 },  
 ff: null,  
};

## 前置知识

* 暂无

## 思路

题目描述的不是很清晰。但通过观察发现应该是：

* 每两个一组。这两个中的第一个是深度，第二个是 key。
* 相同的深度不一定父节点相同。也就是说一个节点的父节点并不是深度-1 的节点，因为深度-1 的节点可能有多个。实际上一个节点的父节点应该是左侧离它最近的深度-1 的节点。

根据以上信息，我们可以使用递归来完成。

定义函数 dfs(A, start, d)，其中 A 为题目输入的数组，start 为当前遍历的索引，方便后续退出递归，d 则是一个用于存储深度信息的对象，形如：

{  
 -1: {  
 ...  
 },  
 0: {  
 ...  
 },  
 1: {  
 ...  
 },  
 ...  
}

其中 key 为深度，value 为深度所对应的对象。因此我们只需要返回 d[-1] 即可。

-1 表示 0 层的父节点，你可以将其看成是一个虚拟节点， 其作用仅仅是简化逻辑判断。

有了上述信息，不难写出如下代码：

function dfs(A, start, d) {  
 if (start + 1 >= A.length) return;  
 // do something  
 dfs(A, start + 2, d);  
}  
  
function deserialization(A) {  
 const d = {};  
 dfs(A, 0, d);  
 return d[-1];  
}

接下来，我们只需要完成状态转移就好了。具体来说就是**将当前的 value 挂到父节点，并将当前节点更新到 d 中即可**。

## 关键点

* 理解题意

## 代码

function dfs(A, start, d) {  
 if (start + 1 >= A.length) return;  
 const [depth, v] = [A[start], A[start + 1]];  
 if (d[depth - 1] == void 0) {  
 d[depth - 1] = {};  
 }  
 let next = {};  
 if (  
 start + 2 >= A.length ||  
 (start + 2 < A.length && A[start + 2] < A[start])  
 )  
 next = null;  
 d[depth - 1][v] = next;  
 d[depth] = next;  
 dfs(A, start + 2, d);  
}  
  
function deserialization(A) {  
 const d = {};  
 dfs(A, 0, d);  
 return d[-1];  
}  
  
deserialization([0, "a", 1, "b", 2, "c", 3, "e", 2, "d", 1, "x", 0, "ff"]);

上面代码会输出：

{  
 a: {  
 b: {  
 c: {  
 e: null,  
 },  
 d: null,  
 },  
 x: null,  
 },  
 ff: null,  
};