腾讯云笔试题

题一 数组转换树

```
function buildLocationTree(locationList) {
  // 1. 创建一个Map，方便通过id快速查找节点
  const locationMap = new Map();
  locationList.forEach(location => {
    locationMap.set(location.id, { ...location, subLocations: [] }); // 拷贝一份，
并初始化subLocations数组
  });

  // 2. 存储所有根节点
  const rootNodes = [];

  // 3. 遍历所有节点，构建层级关系
  locationList.forEach(location => {
    const currentNode = locationMap.get(location.id);

    if (currentNode.pid === undefined) { // 如果没有pid，那就是根节点
      rootNodes.push(currentNode);
    } else { // 否则，找到它的父节点，并把它添加到父节点的subLocations中
      const parentNode = locationMap.get(currentNode.pid);
      if (parentNode) { // 确保父节点存在
        parentNode.subLocations.push(currentNode);
      }
    }
  });

  // 4. 返回根节点集合，符合 LocationTree 接口的 root 属性
  // 确保只有一个顶级根节点，如果存在多个，这里会只取第一个
  if (rootNodes.length === 1) {
    return { root: rootNodes[0] };
  } else if (rootNodes.length > 1) {
    console.warn("发现多个顶级根节点，但LocationTree接口只支持一个root。将使用第一个根节点
作为主根。");
    return { root: rootNodes[0] };
  } else {
    // 列表中没有根节点的情况
    return { root: null }; // 或者抛出错误
  }
}

// 用法:
var locationList = [
  { id: 0, name: "中国" },
  { id: 1, pid: 0, name: "广东省" },
  { id: 2, pid: 1, name: "深圳市" },
  { id: 3, pid: 1, name: "广州市" },
];

var locationTree = buildLocationTree(locationList);
console.log(JSON.stringify(locationTree, null, 2));
```

题二 对大量数据进行排序

```
/**
 * 将两个已排序的子数组合并成一个有序数组的辅助数组
 * @param {Array<Object>} left - 左侧已排序的子数组。
 * @param {Array<Object>} right - 右侧已排序的子数组。
 * @returns {Array<Object>} 合并后的有序数组。
 */
function merge(left, right) {
    const result = [];
    let i = 0; // 左侧数组的指针
    let j = 0; // 右侧数组的指针

    // 比较左右两个数组的元素，将较小的放入结果数组
    while (i < left.length && j < right.length) {
        if (left[i].timestamp <= right[j].timestamp) {
            result.push(left[i]);
            i++;
        } else {
            result.push(right[j]);
            j++;
        }
    }

    // 将左侧数组中剩余的元素添加到结果数组
    while (i < left.length) {
        result.push(left[i]);
        i++;
    }

    // 将右侧数组中剩余的元素添加到结果数组
    while (j < right.length) {
        result.push(right[j]);
        j++;
    }

    return result;
}

/**
 * 实现归并排序，对消息列表按 timestamp 字段进行排序。
 * @param {Array<Object>} messages - 待排序的消息数组。
 * @returns {Array<Object>} 排序后的消息数组。
 */
function mergeSort(messages) {
    // 递归终止条件：如果数组为空或只有一个元素，则已排序
    if (messages.length <= 1) {
        return messages;
    }

    // 分解：将数组分成两半
    const mid = Math.floor(messages.length / 2);
    const left = messages.slice(0, mid);
```

```
    const right = messages.slice(mid);

    // 解决：递归地对左右两半进行排序
    const sortedLeft = mergeSort(left);
    const sortedRight = mergeSort(right);

    // 合并：将已排序的左右两半合并
    return merge(sortedLeft, sortedRight);
}

// 用法

// 模拟 10000 条消息数据
const messages = [];
for (let i = 0; i < 10000; i++) {
    messages.push({
        eventId: i,
        timestamp: Math.floor(Math.random() * 10000000000) + 14000000000, // 模拟一个范围内的timestamp
        user: "auto"
    });
}

// 打乱顺序，确保不是有序的
for (let i = messages.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [messages[i], messages[j]] = [messages[j], messages[i]];
}

console.log("原始消息（部分）：", messages.slice(0, 5)); // 打印前5条原始消息

const sortedMessages = mergeSort(messages);

console.log("排序后的消息（部分）：", sortedMessages.slice(0, 5)); // 打印前5条排序后的消息

// 验证是否真的排序了
let isSorted = true;
for (let i = 0; i < sortedMessages.length - 1; i++) {
    if (sortedMessages[i].timestamp > sortedMessages[i + 1].timestamp) {
        isSorted = false;
        break;
    }
}
console.log("数据是否已排序：", isSorted);
```

题三 正则表达式匹配字符串

```
var cookie = 'PHPSESSID=0kvneb776baqhf58o2tm2c94e5; pgv_pvi=9051466752;
pgv_si=s7186281472;qcloud_visitId=a4855986717854a66aa83f24d3354709;
qcloud_uid=6d95ce375053a1b31d00fce7af9ba02c;pv_uid=257127f177879f702fe6ef94b9d76c
b2;pv_land=http%3A%2F%2Fpassport.qcloud.com%2Findex.php;
_ga=GA1.2.516642758.1463555939;monitorouin=1e19%2BwNnG39Skt0uEiG%2FuMZCqIknLcLH6q
t0aj0xngVjhx7jWWVKewYVkBnbC0PmF4I;monitorappid=8af3cU8qz4kqPoYzLCGRXhLcHpW2w%2BAC
TZuwjaw%2Fh2PMUBz0yIgdTMG5S5guq7qSygo;ptui_loginuin=3052202501%20;pt2gguin=o3052
02501;uin=o 3052202501
;ptisp=ctc;ptcz=9aa1cf3992c8ed2b581f04d51333f0f0b79adfc74e494daa5ad73703d16f93ff;
userinfo=reg%3D1%26un%3Dpolo;nodesess=s%3AO2gAWpD7S4wN9gAu2rYM2Xvf.wq26bg5tscM%2F
3X9FEpx86K0rzF5w%2BbDRetkKugR19I0; nick=polo; appid=1251000011;
moduleId=1251000011;ownerUin=2407912486; regionId=1';

// 正则表达式来精确匹配 'uin=' 后面跟着的数字
const uinRegex = /(?:^|;)\s*uin\s*=\s*(?:o\s*)?(\d+)(?=\s*;|$)/;

const match = cookie.match(uinRegex);

if (match && match[1]) {
  const uin = match[1];
  console.log("精确匹配到的uin是：", uin);
} else {
  console.log("未找到精确匹配的uin。");
}
```