“并查集及其应用”训练题

新考纲分析

三、树与二叉树
(一)树的基本概念
(二)二叉树
1.二叉树的定义及其主要特征
2.二叉树的顺序存储结构和链式存储结构
3.二叉树的遍历
4.线索二叉树的基本概念和构造
(三)树、森林
1.树的存储结构
2.森林与二叉树的转换
3.树和森林的遍历
(四)树与二叉树的应用
1.哈夫曼(Huffman)树和哈夫曼编码
2.并查集及其应用

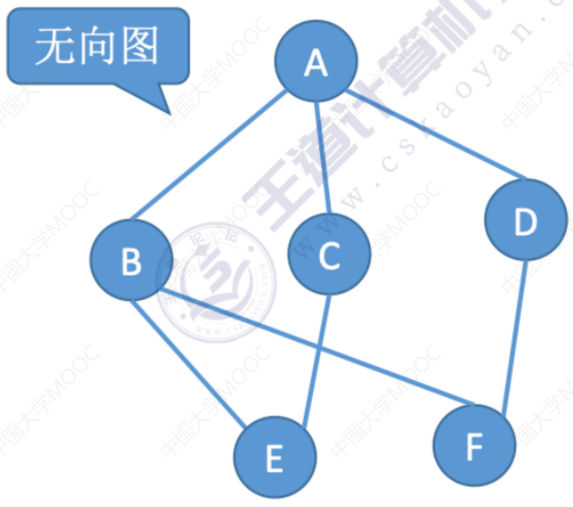
“并查集及其应用” 是今年的408大纲新增考点。在数据结构大纲中，但凡要求为“**XXXX应用**”的考点，都极有可能在大题中考察。并查集的代码实现比较简单，很有可能考算法题。因此，大家必须能够手撕代码。请大家完成下面这些训练。

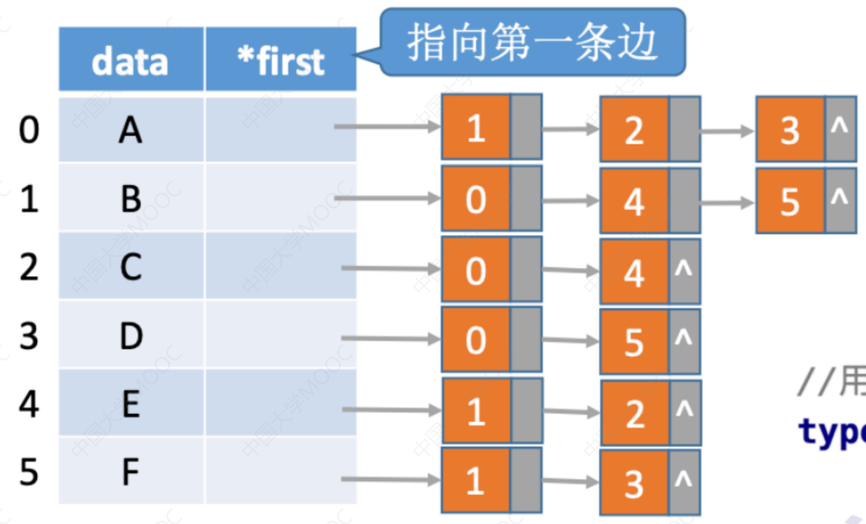
并查集算法题相关训练

用一个长度为n的数组S[n]表示n个元素的并查集，请手撕代码：

1. 实现并查集的初始化
2. 实现 Find (int S[], int x)，找到x所属集合
3. 实现 Union (int S[], int x, int y)，将 x 和 y 所属集合并为一个集合（注：如果 x 和 y 不是根节点也需要合并）
4. 采用”小树合并到大树“的策略优化 Union 操作
5. 采用“压缩路径”的策略优化 Find 操作
6. 应用上面实现的基本操作，解决以下问题：
    - 用二维数组 int Edge[n][n] 表示某无向图的**邻接矩阵**，请实现一个算法，计算无向图中有几个连通分量?
    - 用二维数组 int Edge[n][n] 表示某无向图的**邻接矩阵**，请实现一个算法，判断无向图中是否有环?
7. 请基于下无向图的**邻接表**数据结构定义，再次实现上面两个算法。

邻接表法（顺序+链式存储）






```
//用邻接表存储的图
typedef struct{
    AdjList vertices;
    int vexnum,arcnum;
} ALGraph;
```

```
//"边/弧"
typedef struct ArcNode{
    int adjvex;              //边/弧指向哪个结点
    struct ArcNode *next;    //指向下一条弧的指针
    //InfoType info;         //边权值
}ArcNode;
```

```
//"顶点"
typedef struct VNode{
    VertexType data;     //顶点信息
    ArcNode *first;      //第一条边/弧
}VNode,AdjList[MaxVertexNum];
```


> 思考：对于**邻接矩阵**表示的无向图，如果想要让每条边只被处理一次，我们可以只遍历邻接矩阵上三角部分。而对于**邻接表**表示的无向图，如果想要让每条边只被处理一次，可以怎么做?

其他训练

抽空玩一玩408快乐站，体会并查集的 Find、Union操作如何实现。点击链接打开快乐：https://www.cs.usfca.edu/~galles/visualization/DisjointSets.html

球球大家考前别再刷抖音了，有那时间还不如进408快乐站复习各种数据结构，408所有算法的可视化模拟，在这个快乐小网站里几乎都能找到：

https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

cs.usfca.edu

David Galles
Computer Science
University of San Francisco

- ~~Reversing a String~~
- N-Queens Problem

- Indexing
  - Binary and Linear Search (of sorted list) → 二分查找 和 顺序查找
  - Binary Search Trees → 二叉查找树BST
  - AVL Trees (Balanced binary search trees) → 平衡二叉树AVL
  - Red-Black Trees → 红黑树
  - Splay Trees
  - Open Hash Tables (Closed Addressing) → 散列表（拉链法解决冲突）
  - Closed Hash Tables (Open Addressing) → 散列表（开放定址法解决冲突）
  - Closed Hash Tables, using buckets
  - Trie (Prefix Tree, 26-ary Tree)
  - Radix Tree (Compact Trie)
  - Ternary Search Tree (Trie with BST of children)
  - B Trees
  - B+ Trees → B树、B+树 （注：心中有B树就行，408不要求掌握B+树的插入删除）
- Sorting
  - Comparison Sorting
    - Bubble Sort → 冒泡排序
    - Selection Sort → 选择排序
    - Insertion Sort → 插入排序
    - Shell Sort → 希尔排序
    - Merge Sort → 归并排序
    - Quck Sort → 快速排序
  - Bucket Sort
  - Counting Sort
  - Radix Sort → 基数排序（该网站实现方式和教材中讲授方式不同）
  - Heap Sort → 堆排序
- Heap-like Data Structures
  - Heaps
  - Binomial Queues
  - Fibonacci Heaps
  - Leftist Heaps
  - Skew Heaps
- Graph Algorithms
  - Breadth-First Search → 图的广度优先遍历
  - Depth-First Search → 图的深度优先遍历
  - Connected Components
  - Dijkstra's Shortest Path → 迪杰斯特拉算法
  - Prim's Minimum Cost Spanning Tree → Prim算法求MST
  - Topological Sort (Using Indegree array) → 拓扑排序算法
  - Topological Sort (Using DFS)
  - Floyd-Warshall (all pairs shortest paths) → Floyd算法
  - Kruskal Minimum Cost Spanning Tree Algorithm → Kruskal算法
- Dynamic Programming
  - Calculating nth Fibonacci number
  - Making Change
  - Longest Common Subsequence
- Geometric Algorithms
  - 2D Rotation and Scale Matrices
  - 2D Rotation and Translation Matrices
  - 2D Changing Coordinate Systems
  - 3D Rotation and Scale Matrices
  - 3D Changing Coordinate Systems
- Others ...
  - Disjoint Sets → 并查集
  - Huffman Coding (available in java version)