

数据结构(上)

课程不允许录像, 否则将追究法律责任, 赔偿损失

九章算法强化班 第2章



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

微博: <http://www.weibo.com/ninechapter>

知乎: <http://zhuankan.zhihu.com/jiuzhang>

官网: <http://www.jiuzhang.com>

数据结构: 数据之间的关系, 好的关系可以使得数据处理起来更加高效

1. Union Find 并查集

2. Trie 字典树

Union Find

并查集

一种用来解决集合**查询合并**的数据结构
支持 $O(1)$ find/ $O(1)$ union

1. 查询 Find (递归? 非递归?)

$O(1)$ find

1. 合并 Union

$O(1)$ union

两个操作时间复杂度证明:

[https://en.wikipedia.org/wiki/Proof_of_O\(log*n\)_time_complexity_of_union%E2%80%93find](https://en.wikipedia.org/wiki/Proof_of_O(log*n)_time_complexity_of_union%E2%80%93find)

Log*n 的解释: https://en.wikipedia.org/wiki/Iterated_logarithm

- 模板代码

```
1 int find(int x) {  
2     if (father[x] == x) {  
3         return x;  
4     }  
5     return father[x] = find(father[x]);  
6 }
```

```
1 int find(int x) {  
2     if (father[x] == x) {  
3         return x;  
4     }  
5     return find(father[x]);  
6 }
```

比较有无路径压缩的区别

路径压缩: $O(1)$ 找root的原因

- Key
 - 老大哥之间合并
 - 跟小弟没关系

```
1- public void union(int a, int b) {  
2     int root_a = find(a);  
3     int root_b = find(b);  
4-     if (root_a != root_b) {  
5         father[root_a] = root_b;  
6     }  
7 }
```

BIG BROTHER



**IS WATCHING
YOU**

```
1 public class UnionFind{
2     private int[] father = null;
3     public int find(int x) {
4         if (father[x] == x) {
5             return x;
6         }
7         return find(father[x]);
8     }
9
10    public void union(int a, int b) {
11        int root_a = find(a);
12        int root_b = find(b);
13        if (root_a != root_b)
14            father[root_a] = root_b;
15    }
16 }
```

Connecting Graph

<http://www.lintcode.com/en/problem/connecting-graph/>
<http://www.jiuzhang.com/solutions/connecting-graph/>

589

题意概括：n个节点的图，没有任何边存在。

有两个操作：

1. 在a和b节点之间连上边
2. 询问a和b节点是否连通

590

Connecting Graph II

<http://www.lintcode.com/en/problem/connecting-graph-ii/>

<http://www.jiuzhang.com/solutions/connecting-graph-ii/>

题意概括：n个节点的图，没有任何边存在。

有两个操作：

1. 在a和b节点之间连上边
2. 询问a所在的连通块的节点个数

45分钟

591

Connecting Graph III

<http://www.lintcode.com/en/problem/connecting-graph-iii/>

<http://www.jiuzhang.com/solutions/connecting-graph-iii/>

题意概括：n个节点的图，没有任何边存在。

有两个操作：

1. 在a和b节点之间连上边
2. 询问**当前图**的连通块的个数

Connecting Graph问题的总结

- 并查集原生操作：
 - 查询两个元素是否在同一个集合内
 - 合并两个元素所在的集合
- 并查集的派生操作：
 - 查询某个元素所在集合的元素个数
 - 查询当前集合的个数

并查集实战例题

Google Interview: Number of Islands (九章算法班讲过)

434

www.lintcode.com/zh-cn/problem/number-of-islands

<http://www.jiuzhang.com/solutions/number-of-islands/>

Google Interviewer: Number of Islands II

<http://www.lintcode.com/zh-cn/problem/number-of-islands-ii/>

<http://www.jiuzhang.com/solutions/number-of-islands-ii/>

Facebook Interviewer: Graph Valid Tree

<http://www.lintcode.com/problem/graph-valid-tree>
<http://www.jiuzhang.com/solutions/graph-valid-tree/>

Union Find $O(n)$

Surrounded Regions

<http://www.lintcode.com/en/problem/surrounded-regions/>

<http://www.jiuzhang.com/solutions/surrounded-regions/>



01:12:55

Trie Tree

字典树

Snapshot Interviewer: Implement Trie

<http://www.lintcode.com/en/problem/implement-trie/>

<http://www.jiuzhang.com/solutions/trie/>

Snapshot Interview: Add and Search Word

<http://www.lintcode.com/en/problem/add-and-search-word/>
<http://www.jiuzhang.com/solutions/add-and-search-word/>

Microsoft Interviewer: Word Search II

<http://www.lintcode.com/en/problem/word-search-ii/>

<http://www.jiuzhang.com/solutions/word-search-ii/>

- Hash vs Trie做这道题目的区别
 - 把谁建成Trie树？

- Given a dictionary[aca, acc] and a matrix of upper alphabets, find all words in the dictionary that can be found in the matrix.
 - acaf
 - acad
 - acae
- 解题思路：
 - 把字典建成Trie树。
 - 用dfs的方法遍历矩阵, 同时在Trie上搜索前缀是否存在。
 - 查询所有Trie里面有可能出现的字符。

Word Square

<http://www.jiuzhang.com/solutions/word-squares/>

对于数组["ball","area","lead","lady"]找到
可以组成的所有Word Square

[b a l l]

[a r e a]

[l e a d]

[l a d y]

