Dynamic connectivity

\*quick find ~O(n)

\*quick union ~O(n)

\*weigthted quick uinion O (logn)

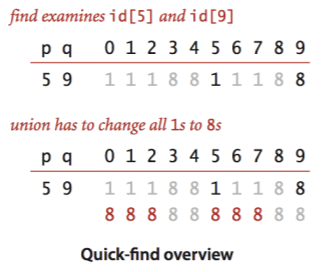
1. 第一種方法 Quick find: 利用陣列的id值去判斷 object 有無 connect

步驟一: 新增一個陣列

步驟二: 新增值到陣列裡面

步驟三: union 兩個物件 (for迴圈檢查每一個位置判斷 )

步驟四: 判斷兩個物件是否有同一個id 值



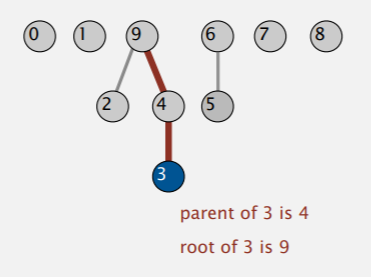
1. 第二種方法 Union find : 利用root 的方式 去串接每一個object

步驟一: 新增一個陣列

步驟二: 新增值到陣列裡面

步驟三: 利用root 方法找到parent 父層的id

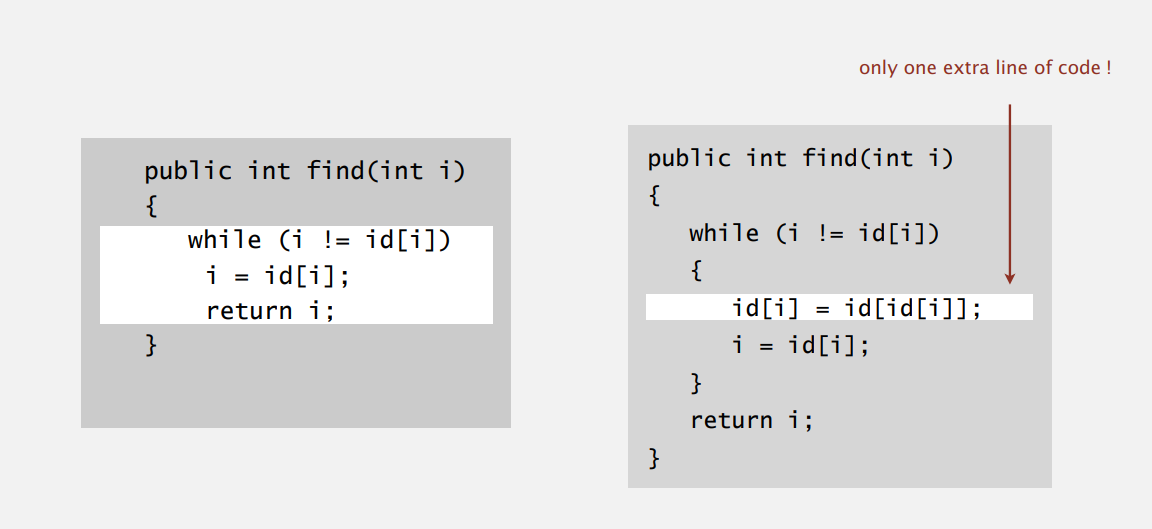
步驟四: p 跟q 有沒有相同的root 判斷有沒有connect

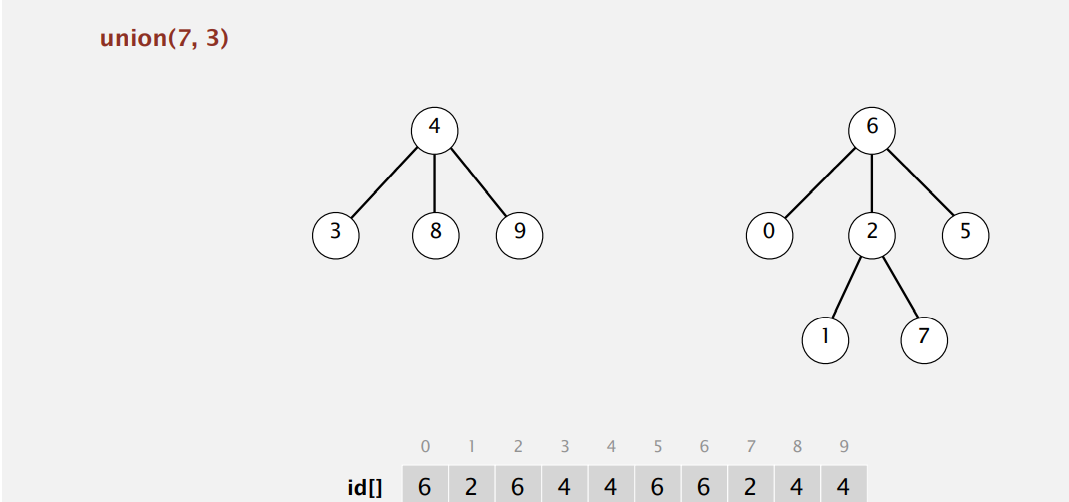


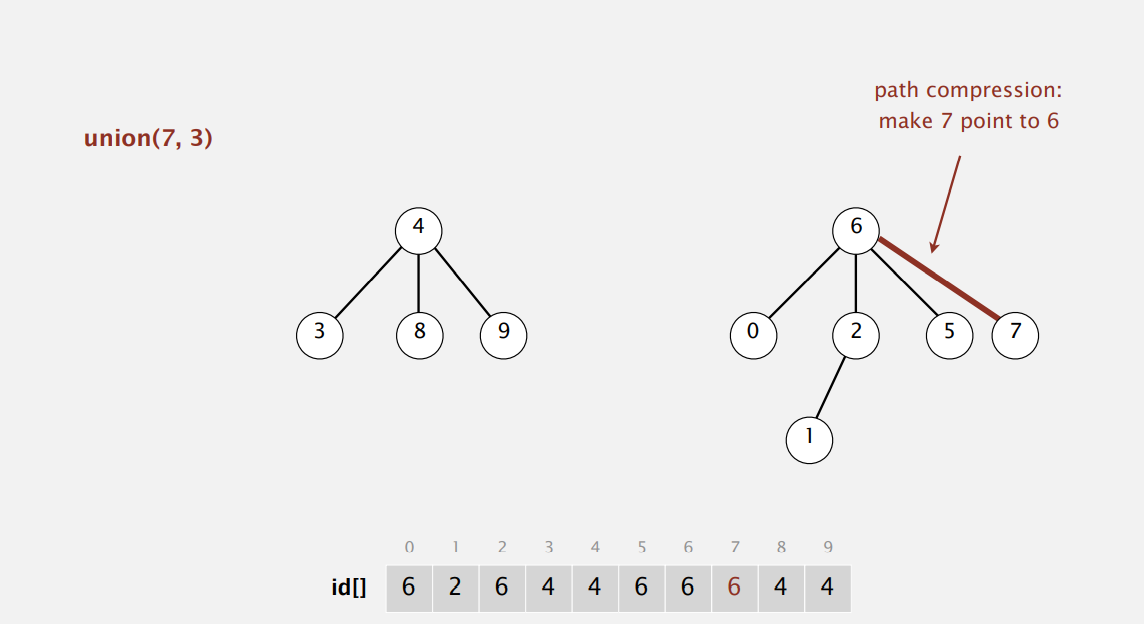
1. weigthted quick uinion O (logn)
2. path comprassion

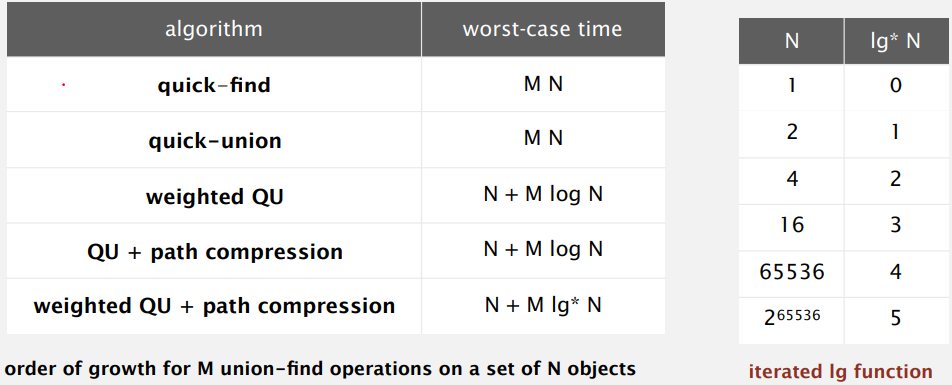
any other possible to speed up

id[i] = id[id[i]];多加這行程式碼能有效找到root









演算法注重在 worst case 和 average case

Theory of algorithms

目標:

證明一個演算法是最佳的

方法: 注重在worst case

Upper bound:

Lower bound: 沒有演算法比這個還好

當 upper bound = lower bound 就是最佳解 (optimal algorithm)

