Disjoint-set data structure

A ***disjoint-set data structure*** maintains a collection of dis- joint dynamic sets. We identify each set by a ***representative***, which is some member of the set.

MAKE-SET(x) makes a singleton set with x as the only element

UNION(x,y) combines set containing x with set containing y

FIND-SET(x) find the representative of the set whose element is x

n, the number of MAKE-SET operations,

m, the total number of MAKE-SET, UNION, and FIND-SET operations.

The **number of UNION** operations is thus **at most n-1.**

since the MAKE-SET operations are included in the total number of operations m, we have **m>= n**

With this **linked-list representation**, both **MAKE-SET and FIND-SET** are easy, requiring **O(1)** time.

To carry out MAKE-SET(x), we create a new linked list whose only object is x.

For FIND-SET(x), we just follow the pointer from x back to its set object and then return the member in the object that *head* points to.

A sequence of **2n-1 operations on n objects that takes theta(n^2) time, or theta(n) time per operation on average**, using the linked-list set representation and the simple implementation of UNION.

In the worst case, the above implementation of the **UNION procedure requires an average of theta(n)** time per call because we may be appending a longer list onto a shorter list

With this simple ***weighted-union heuristic***, a single UNION operation can still take omega(n) time if both sets have omega(n) members

a sequence of m MAKE-SET, UNION, and FIND-SET operations, n of which are MAKE-SET operations, takes **O(m + n lg n)** time.

“union by rank” and “path compression”

Alone, **union by rank yields** a running time of **O(m lg n)** (see Exercise 21.4-4), and this bound is tight

for a sequence of n MAKE-SET operations (and hence at most n-1 UNION operations) and f FIND-SET operations, the **path-compression heuristic alone** gives a worst-case running time of **theta(n + f (1 + $\log_{2+f/n}$ n)).**

When we use both **union by rank and path compression**, the worst-case running time **is O(m alpha(n))**, where alpha(n) is a *very* slowly growing function

In any conceivable application of a disjoint-set data structure, **alpha(n)<= 4**; thus, we can view the running time as linear in **m** in all practical situations