CSC263 Monday lecture Feb 12, 2018

Disjoint Sets: Union/Find // CLRS 21

- n distinct elements: $1, 2, \ldots, n$
- initially each in own set: $\{1\}, \{2\}, \ldots, \{n\}$
- each set has a representative element
  S_x: represented by element x

- UNION(S_x, S_y): replace S_x and S_y with $S_x \cup S_y = S_z$
  returns pointer to representative of S_z
- FIND(z): given pointer to z, finds set S that contains z

- Note: in textbook, elements are not in sets, and there is function MakeSet that is $O(1)$. We are abstracting this away.

Ex Let underline denote representative element.

	S_1	S_2	S_3	S_4	S_5	
	1	2	3	4	5	
$U(S_3, S_4)$	1	2	3 4	×	5	$F(4) = S_3$
$U(S_1, S_5)$	1 5	2	3 4		×	
$U(S_1, S_3)$	1 5 3 4	2	×			$F(4) = S_1, F(2) = S_2$
$U(S_1, S_2)$	1 5 3 4 2	×				

- Note: every union reduces # of sets by 1, ∴ max # unions = $n - 1$

Q
- Starting from n singleton sets, do sequence T where
  T: sequences of at most $n-1$ unions, and at least $m \geq n$ finds.

- Plain linked list: S_2: [1] → [5] → [3] → [4] → [2] // representative = head
  - Union - tail to tip pointer reassignment ∴ $O(1)$, called $(n-1)$ times in T
  - FIND - traverse list, find head, return ∴ $O(n)$, called m times in T
  - $T \in O(n-1+m \cdot n) \in O(n+m \cdot n) \in O(m \cdot n)$

AUGMENTED Linked List:

- at every element, store pointer to representative of set that contains it
- FIND: follow one pointer to representative ∴ $O(1)$
- UNION: have to reassign pointers of elements from one set to head of the other ∴ $O(n)$
  - Redefine union such that Larger ∪ Smaller, ∴ smallest number of reassignments possible

∴ $T \in O(m + n\log n)$

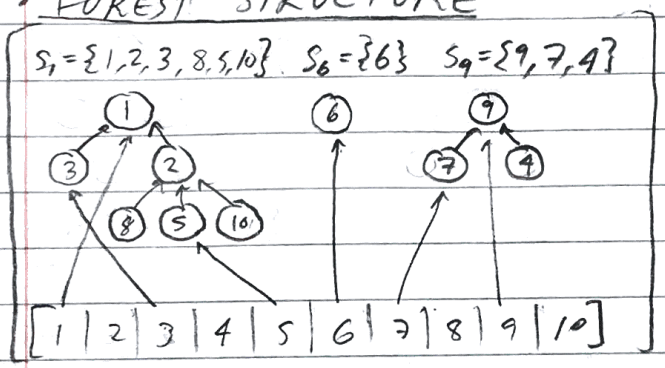
m: cost of m finds done in constant time

nlogn: cost of n-1 special unions

THIS WILL BE PROVED IN TUTORIAL

*

- FOREST STRUCTURE



- unstructured group of trees
- additional array of length n stores pointers in tree to all elements
- UNION: $O(1)$ // single new pointer

- FIND: must follow pointers from element to head

  path of pointers max length n-1, ∴ FIND $\in O(m \cdot n)$

  → If you follow weighted union rule, you never create tree of height > log n ∴ FIND $\in O(m \log n)$

* Will be proved on WEDNESDAY *