Lec04-union-find

Tuesday, August 25, 2026 3:49 PM

Recall that Kruskal adds edges in increasing order avoiding cycle

How do we check in Kruskal if adding an edge (u,v) would create a cycle?

- Would create a cycle if u, v is same connected component already
- We start with a component for each node.
- Components merge when we add an edge
- Need way to check if u, v are in same component & to merge two components into one.

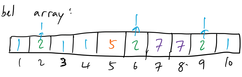
Union-Find Abstract Data Type (KT 4.6)

Operations:

- MakeUnionFind(S) – create data structure containing $|S|$ sets, each containing one item from S.
- Find(i) – return label of set containing i
- Union(a,b) – merge sets a, b into single set

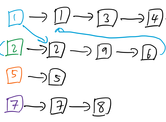
Array-based union-find

Set label array:



← says which set an item is in

Items list:



Sizes

MakeUnionFind(S): create data structure, initialize to $|S|$ sets

$O(|S|)$ time

Find(i): return UF.sets[i]

$O(1)$ time

Union(x,y): Use size array to determine which set is smaller

WLOG, assume $|x| < |y|$

Walk down elements i in set x, setting sets[i]=y

Set size[y] = size[y] + size[x]

Make y point to start of x list and last item of x list point to old start of y list

prepend smaller list to larger list

Let's compute runtime of array-based union-find

Thm Any sequence of k union operations on a collection of n items takes $O(k \log k)$ time

proof: After k unions, at most $2k$ items have been involved in a union

(each union can touch at most 2 new items if it was a union of 2 singleton sets, often only 0 or 1 new items touched)

For any item v, set[v] changes at most $\log_2(2k)$ times (total number of items possible in largest set)

because each time set[v] changes, $|set[v]|$ at least doubles

(since we update the smaller set)

At most $2k$ items updated at most $\log_2(2k)$ times each

$\Rightarrow 2k \log_2 2k$ work

$\Rightarrow O(k \log k)$ work

(size & item list update constant per update is $O(k)$) ▱

Runtime of Kruskal using arrays

Sorting edges $\approx m \log m$ for m edges

$m \le n^2$, so $\log m \le \log n^2 = 2 \log n$

$\approx m \log n$

At most $2m$ "find" ops = $2m$ time

(to check if u, v in same component)

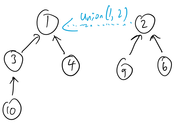
At most $n-1$ union ops $\approx n \log n$ time

$\Rightarrow$ total = $m \log n + 2m + n \log n$ $O(m \log n)$

largest term since $m \ge n$ if graph is connected & not a tree

---

Tree-based union-find



MakeUnionFind(S): create $|S|$ single-node trees $\Theta(|S|)$ time

Find(i): follow pointers from i to root of tree

Union(x,y): If $|x| < |y|$, make x point to y. $\Theta(1)$ time

Thm. Find(i) takes time $O(\log n)$

proof: set[i] is renamed at most $\log_2(n)$ times

because each rename doubles the size.

The depth of a tree is the max number of renamings of an item. ▱

Runtime of Kruskal using trees:

Sorting edges $O(m \log n)$

$\le 2m$ "find" ops: $\log n$ time each $\rightarrow 2m \log n$

$\le n-1$ union ops: $O(n)$ time

Total = $O(m \log n)$