10/03/2022

Implementing Kruskal's Algo

→ Union-Find data Structure.

Find(u) and Find(v)

if u and v blongs to the same set

then Find(u) = Find(v)

Otherwise

Find(u) ≠ Find(v).

In our case, we will name the set by one of the elements from the set.

A simple implementation of the data Structure

$S = \{1, 2, \ldots, n\}$

Maintain an array "Component"

Component[i] = name of the set containing i.

Initially, set Component[i] = i.

Find(u) = Component[u]

so O(1) time.

Union(A, B) = go over every element in A and B and change their name.

so it can take O(n) time.

Optimization.

→ Maintain an array of list that gives for every set the elements it contain.

Set $[1, \cdots, n]$

→ Change the name of one of the sets.

In particular,

Change the name of the smaller set to the larger set.

Maintain an array "size" $[1, \cdots, n]$.

How much time a single Union will take in the worst case?

$|A| = \frac{n}{3}$ $\quad$ $|B| = \frac{n}{3}$

→ $O(n)$ time.

How much time a sequence of k union operations take?

Lemma:- Consider the Union-Find data structure for a set of size n where the naming convention keeps the name of larger set. Then

Find Operation takes $O(1)$ time

Make UnionFind ——— $O(n)$ time.

and

a sequence of k Union operations take $O(k \log k)$ time.

Proof:

How many elements will you touch in a sequence of k Union operations?

In the beginning everything is a singleton set.

A total of at most 2K elements are touched in a sequence of K Union operations.

Let's Consider a particular element u.

How many times will you update component[u] ?

Start Size 1

a sequence of K Union operation

Size. 2K

every time you update component[u] the size of set containing u doubles.

$1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16$

times component[u] will get updated $\leq \log 2k$

How many elements are there for which you will possibly update component ? $\leq 2k$.

∴ Overall time taken to perform k union operations.

$O(k \log k)$.

$(|E|=m, |V|=n)$

Implementing Kruskal's Algo.

→ Maintain disjoint components using Union-Find.

→ Consider the edges in increasing order $O(m \log m) = O(m \log n)$.

→ For each edge $e = (u, v)$

Find (u) , Find (v)

at most $2m$ times you will call Find Operation.

if Find $(u) \neq$ Find (v)

then Union (Find (u), Find (v)).

How many times will you call Union operations?

$\leq n-1$ times.

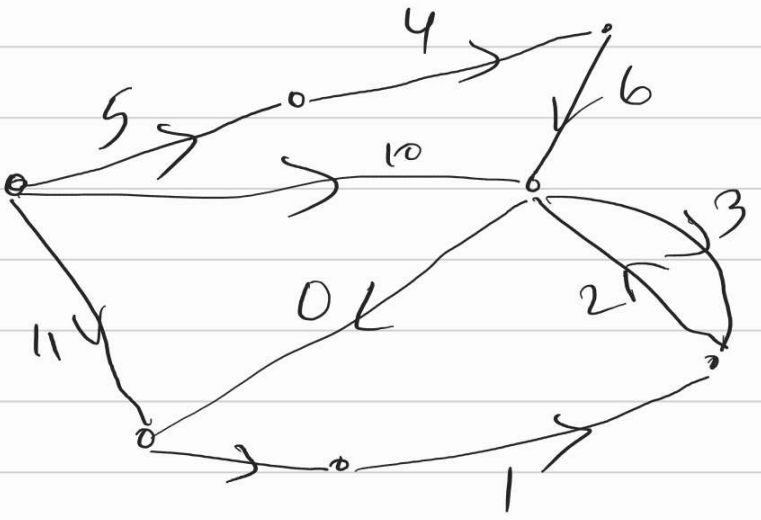
total time taken by Union operations

$O(n \log n)$.

$O(m \log n + m + n \log n)$

Shortest Path: DIJKSTRA's Algo

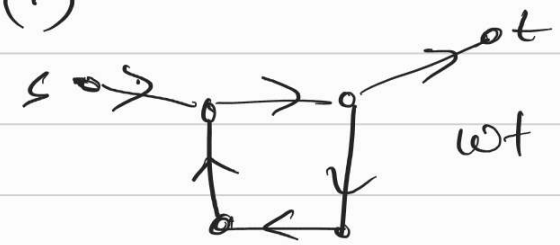
Q: Want to find shortest paths in a weighted graph?



$\ell(e)$:= length/ weight of edge e.

$$\ell : E \to \mathbb{R}^{\geq 0}$$

(1)



wt of this cycle is $= -1$

When we handle negative wts. we

will assume no negative cycle in the graphs.

(2)