CSE 102 3-12-20

Two Heuristics for Disjoint Set forests

Union by rank

for each node x, maintain attribute rank[x] which is an upper bound for height(x):

$$height(x) \leq rank[x]$$

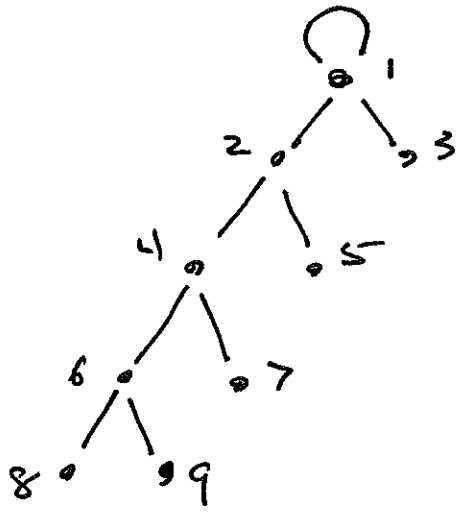
Modify Union(·,·) so root with smaller rank points to root with higher rank.

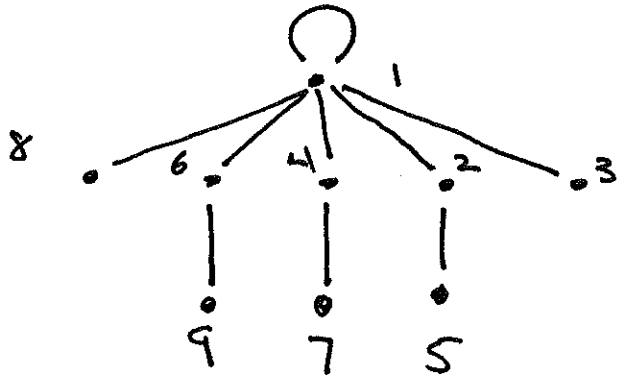
Path Compression

Modify FindSet( ) to be a two-pass method:

- up the Find-Path to find root
- down the Find-Path setting each node to have root as parent.

Ex.





FindSet(8)

MakeSet(x)

1. p[x] = x
2. rank[x] = 0

Union(x, y) (union-by-rank)

1. Link(FindSet(x), FindSet(y))

Link(x, y) (Pre: x, y both roots, $x \neq y$)

1. if rank[x] > rank[y]
2. p[y] = x
3. else
4. p[x] = y
5. if rank[x] == rank[y]
6. rank[y]++

Exercise

As long as only MakeSet and Link are called, rank[x] = height(x) for all x. Prove by induction on # of Link operations.

FindSet(x)

1. if $x \neq p[x]$
2. $p[x] = \text{FindSet}(p[x])$
3. return $p[x]$

Trace on previous example!

FindSet(8): 1. $8 \neq p[8]$
2. $p[8] = \text{FindSet}(6)$
3. return 1

FindSet(6): 1. $6 \neq P[6]$

2. $P[6] = \text{FindSet}(4)$

3. return 1

FindSet(4): 1. $4 \neq P[4]$

2. $P[4] = \text{FindSet}(2)$

3. return 1

FindSet(2): 1. $2 \neq P[2]$

2. $P[2] = \text{FindSet}(1)$

3. return 1

FindSet(1): 1. $1 \neq P[1]$ (false)

2. ———

3. return 1

Note FindSet(x) may reduce height(x), but it does not alter ranks. Thus

$$height(x) \leq rank[x]$$

for all nodes x.

Theorem

Using union-by-rank and Path-Compression heuristics, any sequence of m MakeSet, Union & FindSet operations (n of which are MakeSets) can be performed in time

$$O(m \cdot \alpha(n))$$

Here $\alpha(n)$ is an extremely slow growing function. In any conceivable application $\alpha(n) \leq 4$. so effectively, $\alpha(n) = \text{const.}$, and runtime $O(m)$.

Ammortizing over all m operations, gives run time

$$\frac{O(m)}{m} = O(1)$$

per operation.

BACK TO SEC. 23.2

RECALL THAT KRUSKAL'S ALGORITHM GROWS SEVERAL TREES SIMULTANEOUSLY UNTIL THEY MERGE INTO A SINGLE MWST. THE IMPLEMENTATION BELOW USES A DISJOINT SET DATA STRUCTURE TO MAINTAIN A COLLECTION OF VERTEX SETS. EACH SET IN THIS COLLECTION IS THE VERTEX SET OF ONE OF THE TREES UNDER CONSTRUCTION.

NOTE THE SIMILARITY OF THIS ALGORITHM TO THE ALGORITHM FOR DETERMINING CONNECTED COMPONENTS FROM 21.1.

KRUSKAL (G, w)

1.) $A = \emptyset$
2.) for each $v \in V$
3.) MakeSet(v)
4.) SORT E BY WEIGHT w (NON-DECREASING)
5.) for each $(u,v) \in E$ (IN SORTED ORDER)
6.) if FindSet$(u) \neq$ FindSet(v)
7.) $A = A \cup \{(u,v)\}$
8.) Union(u,v)
9.) return A

ON EACH ITERATION OF LOOP 5-8 WE PICK THE LIGHTEST EDGE JOINING TWO DISTINCT TREES AND ADD THAT TO A. i.e. AMONGST ALL EDGES WHOSE ADDITION TO A WOULD NOT CREATE A CYCLE, WE PICK ONE OF MINIMUM WEIGHT. THE RESULTING SET $A \subseteq E$ CONSTITUTES THE EDGES OF A MWST IN G.

Run Time

Assume we use the disjoint set forest implementation with union-by-rank and path compression heuristics, which is the asymptotically fastest known.

Initialization of A in line 1 costs $O(1)$ time. Loop 2-3 does $|V|$ MakeSet operations and loop 5-8 $O(|E|)$ FindSet and Union operations. Thus the disjoint set operations have total cost $O((|V|+|E|)\cdot\alpha(|V|))$.

Now since G is connected (otherwise there is no spanning tree) we have $|E| \geq |V|-1$, whence $|V| \leq |E|+1 = O(|E|)$, so the disjoint set operations cost $O(|E|\alpha(|V|))$

The sort on line 4 costs $O(|E|\lg|E|)$ time (assuming we use Mergesort or Heapsort, or a similarly efficient sort.)

Also note that $\alpha(|V|) = O(\lg|V|) = O(\lg|E|)$, so the run time of Kruskal is

$$O(|E|\lg|E|)$$

But also since G is simple, $|E| < |V|^2$ so $\lg|E| = O(\lg|V|)$, and the run time of Kruskal can be expressed as

$$O(|E|\lg|V|)$$

which is the same as Prim.