Ugeopgave 4

Mathias Bramming   Philip Meulengracht
Rasmus Borgsmidt

Task 1

We use a disjoint-set forest, identical to the one described in CLRS section 21.3, e.g. a tree structure using *union by rank* and *path compression*. The `LINK` operation, works by finding the root of each set, and appending the tree with the smallest rank, to the biggest. In case the roots have the same ranks, one is incremented. It uses a helper method, `FIND-SET` to find the root of each tree.

```
LINK(i,j)
        r1 = FIND-SET(i)
        r2 = FIND-SET(j)
        if r1.rank > r2.rank
                r2.p = r1
        else    r1.p = r2
                if r1.rank == r2.rank
                        r2.rank = r2.rank+1
```

Since all breweries with links are connected to the same tree, any given brewery x and y *must* have the same root if they are connected, which means that `QUERY` will only have to check if x and y have the same root:

```
QUERY(i,j)
        return FIND-SET(i) == FIND-SET(j)
```

We use the `FIND-SET` operation from CLRS (two-pass method) as a helper function to find the root of a given node:

```
FIND-SET(x)
        if x != x.p
                x.p = FIND-SET(x.p)
        return x.p
```

According to CLRS, *Union by rank* has a tight running-time bound of $O(m \lg n)$, which means that our `LINK` operation has the same running time. The running time of `QUERY` has a best case of $O(1)$ and a worst case running time of $O(n)$: The first time the operation is used, it's possible that it has to traverse the entire tree (up to n size) to find the root. The second pass, however, is in constant time, since all nodes now point directly to the root.

Task 2

```
DECREMENT(G, U)
  // Runs in O(m + a(n)) due to union by rank and path
      compression, CLRS p. 571
  comp_count = COUNT-COMPONENTS(G)

  // Runs in O(m + a(n)) due to union by rank and path
      compression, CLRS p. 571
  for each unlink (u, v) in U
    UNLINK(FIND-SET(u), FIND-SET(v))
    if not QUERY(u, v)
      increment comp_count
    print comp_count
```

```
COUNT-COMPONENTS(G)
  let R be an empty list of root vertices

  // Runs in O(m a(n)) due to union by rank and path
      compression, CLRS p. 571
  for each link (u, v) in G.E
    u.linked = true
    v.linked = true
    r = FIND-SET(u)
    r.marked = true
    R = [r | R]

  // Runs in O(m) as there can be at most m roots
  root_count = 0
  for each vertex r in R
    if r.marked
      increment root_count
      r.marked = false

  // Runs in O(m)
  linked_count = 0
  for each link (u, v) in G.E
    if u.linked
      increment linked_count
      u.linked = false
    if v.linked
      increment linked_count
      v.linked = false

  return n - linked_count + root_count
```

```
QUERY(u, v)
  FIND-SET(u) == FIND-SET(v)
```

```
UNLINK(r1, r2)
  r1.p = r1
  r2.p = r2
```

Task 3