CS2040S: Data Structures and Algorithms

Tutorial Problems for Week 8: Union-Find

For: 4 Oct 2021, Tutorial 6

Problem 1. Largest set

Given a UFDS initialised with n disjoint sets, what is the maximum possible rank h that can be obtained from calling any combination of `unionSet(i, j)` and/or `findSet(i)` operations? Assume that both the path-compression and union-by-rank heuristics are used. Justify your answer.

Problem 2. Intergalatic wars

n intergalactic warlords scattered throughout the galaxy have been vying to conquer each other and become the intergalactic overlord. Each of them start out being only the overlord of his/her own world. A conquered warlord will have all the worlds under him/her added to the worlds of the victorious warlord. Each starting world may be represented by a positive integer from 0 to $n-1$. Each warlord is also represented by an integer from 0 to $n-1$. At the beginning warlord x will only be the conqueror of world x.

Problem 2.a. Since a warlord x can have many worlds under him/her, describe the data structure(s) and associated operations you need so that the warlord can

i) Add worlds of a conquered warlord y to his/her domain in $< O(\log n)$ time.

ii) Answer the query of whether a world is under him/her in $< O(\log n)$ time.

Problem 2.b. Make modifications/additional operations to the data structure you described in 2(a) so that a warlord can answer in $< O(\log n)$ time the most important question - Am I now the intergalactic overlord (meaning have I conquered all the N worlds)? Justify your answer.

Problem 3. Number candies (Adapted from AY19/20 Sem 1 Final Exam)
The current rage among children are number candies. As the name suggest, these candies are given a number from 1 to n, where n is an extremely huge number. The ultimate goal is to collect all n candies. This is very difficult as the candies carried by any store is random (and can have a lot of repeats). To make it easier, the manufacturer has a competition to gather all sequential candies from 1 to m, where m is much smaller than n. The winner will be given all the n candies.

One particularly rich kid has hired h helpers, where $m < h < n$. Each helper will buy one candy from each of the h available stores. He hopes that doing so will allow him to get all m candies.

However, he does not know how to figure out if he has all candies from 1 to m.

To help him solve the problem, his butler suggests him to answer the following query every time one of the helpers return with a candy: **What is the largest numbered candy he has that is in an unbroken sequence from candy 1?** (If greater than m, just return m. If he does not have candy 1, return -1.)

Design an algorithm that makes use of UFDS such that each of the h queries can be answered in $O(\alpha(h))$ time.

Problem 4. Funfair Ride

You are operating a ride at a funfair. The ride contains n seats numbered from 1 to n arranged **clockwise** in a circle (i.e. seat i is followed (clockwise) by seat $i+1$ for $1 \leq i < n$, and seat n is followed (clockwise) by seat 1).

m children ($m \leq n$) numbered 1 to m arrive at the ride in sequence (from 1 to m). Each child wants a certain seat in the ride: the child numbered i wants seat s_i. However, there are some seats that are wanted by more than one child, so you have decided to allocate the seats in the following way:

When the ith child arrives:

- If seat s_i is not taken, allocate seat s_i to child i.
- Otherwise, get child i to walk clockwise starting from seat s_i, and allocate the first empty seat encountered to child i.

There are many children coming for the ride, so this process of getting each child to manually find another seat if the seat he wants is taken is slow. Fortunately, you have been given the seat preferences for all the children (all the s_i). Using a UFDS, give an algorithm to figure out the seat that a child i will eventually occupy. State the time complexity of your algorithm.