Disjoint Sets II – Using Algs4 Union–Find

January 8, 2019

Maintain a family of disjoint sets under repeated union operations, using a Union–Find dta structure.

This exercise has the following secondary aims: (i) To ensure that you have access to and and can use the course library `algs4`. (ii) In conjunction with the previous exercise *Disjoint Sets I*, to make clear that there is a difference between "How do I do X in idiomatic Python/Java?" and "How do I do X efficiently?" This exercises is about the second question, where we use an *existing* library implemtnation of an efficient data structure. In a later exercise *Disjoint Sets III* we will revisit the problem and write a *tailor-made* data structure.

Description

Exactly as Disjoint Sets I, except that the upper bound on the instance size is relaxed to $n, m \leq 10^7$.

Deliverable

Solution based on Union–Find. Solve the problem in Python or Java using a Union–find data structure from the `algs4` library. Hand this in as `Ds2.java` or `ds2.py`.

Report. Hand in a very brief report as a PDF, extrapolating the running time of your program to instances larger than 10^7, preferably by just finishing the skeleton in the appendix.

You need to find and install the `algs4` library on your machine. You need to make a decision about *which* Union–Find class from the library you want to use—there are several in `algs4.fundamentals.uf`. This solution should pass all tests on the Code Judge.

For the report, you will need to estimate how how much time your two programs would need to solve even larger instances. For this, you are supposed to run your program on a few instances of various sizes that you made yourself and then extrapolate from that using your good judgement. For example, an instance that consists of ten million lines of "`q 0 1`" will not give a good estimate of the running time. Instead, the generator could pick pairs of random elements, and choose at random whether to union or query. (If we union too often, the sets will quickly just become one big set $\{0, \ldots, n-1\}$.) There are many other ways of constructing good instances – your mindset should be to make your own program look *bad*.

Do *not* hand in the source code of the Union–find data structure from the library, nor the instance generator source code.

Disjoint Sets II Report

by Alice Cooper[1]

[1] Complete the report by filling in your real name, filling in the parts marked [...] and changing other parts wherever necessary. (For instance, the numbers is the tables are nonsense right now.) Remove the sidenotes in your final hand-in.

Results

I solved this exercise in Python 3 / Java.

Program `ds2` uses the `UF` class from `algs4` and passes all tests on the CodeJudge. I chose that class because the others were slower / I didn't have time to try the others / it didn't make any difference. [2]

[2] Replace by whichever library class you actually use, what your reason was (if any), and whichever errors you generated.

The following table gives a rough estimate on the running time of my program for large instances, based on extrapolating observed behaviour for smaller instances on my machine.[3]

[3] In the table, replace 'year' by second, minute, hour, day, week, month, century, etc. as appropriate. If you *want* to write a longer report with more experiments, you may.

Size	Time for `ds2`
$n, m = 10^5$	ca 1 year
$n, m = 10^6$	ca 1 year
$n, m = 10^7$	ca 1 year
$n, m = 10^8$	ca 1 year (extrapolated)
$n, m = 10^9$	ca 1 year (extrapolated)