Two's a Crowd, Three's Company

September 10, 2016

Logs from a web server include one line per access to the system (ordered by time of access) with a user ID (a string) on each line (plus other fields we don't care about). Unusual accesses may suggest security concerns. In this problem we are identifying the first user ID that only ever accessed the system once.

We will use n—the total number of entries in the log—to describe problem size. Note: **assume** that comparing two user IDs (strings) for equality or order takes constant time.

1 Brute Force

Consider this algorithm that attempts to solve the problem by brute force:

```
for each user ID s_i in order by indexes i do
    found ← false
    for each later user ID s_j in order do
        if s_i = s_j then
            found ← true
        end if
    end for
    if found is false then
        return i
    end if
end for
return None
```

Now, answer the following questions:

1. Give and **briefly** justify (perhaps including annotating the algorithm) a good asymptotic bound on the algorithm's worst-case runtime in terms of n.

2. If the algorithm is **correct**, briefly sketch (only the key elements/insights in) a proof of its correctness.

   If the algorithm is **incorrect**, illustrate the fact by giving the smallest possible example (in terms of n) on which it fails and explaining what the algorithm does and what **should** happen.

   Justification:

2 Tracking Uniqueness and Indexes

The following algorithm solves the same problem using self-balancing BSTs:

```
User ← an empty self-balancing BST (that will map indexes to user IDs)
Index ← an empty self-balancing BST (that will map user IDs to indexes)
for each user ID s_i in order by indexes i do
    if Index does not contain s_i then
        Index[s_i] ← i
```

```
        User[i] ← s_i
    else
        Delete Index[s_i] from User (if it is present)
    end if
end for
if User is empty then
    return None
else
    return the value (ID) of the minimum key (index) in User
end if
```

Now, answer the following questions:

1. Give and **briefly** justify (perhaps including annotating the algorithm) a good asymptotic bound on the algorithm's worst-case runtime in terms of n.

2. This algorithm **is correct**. **Briefly** sketch (only the key elements/insights in) a proof of its correctness. (E.g., you will want to justify that every ID that is in User after the loop ends appears exactly once in the log.)