CS 758/858: Algorithms

`http://www.cs.unh.edu/~ruml/cs758`

Topological Sorting

The Problem

Given a set of pairwise orderings $a \prec b$, find an ordering of all the elements that respects them or detect that no such ordering is possible.

How long does this take?

Break

- asst 8

Union-Find

Connected Components

Problem: find components in an undirected graph and answer membership queries

Two cases: static vs dynamic

How can we identify components in the static case?

Now let's do the dynamic case...

Union-Find ADT

$\textsc{Make-Set}(x)$ makes new set containing x

$\textsc{Union}(x, y)$ combine the set containing x with the set containing y

$\textsc{Find-Set}(x)$ return a representative of the set containing x

Connected Components Algorithm

find-components

```
foreach vertex $v$
    $\textsc{Make-Set}(v)$
for each edge $(u, v)$
    $\textsc{Union}(u,v)$
```

in-same-component?(u,v)

```
is $\textsc{Find-Set}(u) = \textsc{Find-Set}(v)$?
```

Disjoint Sets

set is a tree rooted at representative

How to implement make, union, find?

Speed-Ups

union by rank track approximate height, put shorter under taller

path compression after Find-Set, ensure touched nodes point directly to root

Pseudo-code

```
MAKE-SET($x$)
$x.p \leftarrow x$
$x.$rank $\leftarrow 0$

UNION($x$, $y$)
$x \leftarrow$ FIND-SET($x$)
$y \leftarrow$ FIND-SET($y$)
if $x.rank > y.rank$
    $y.p \leftarrow x$
else
    $x.p \leftarrow y$
    if $x.rank = y.rank$
        increment $y.rank$
```

More Pseudo-code

```
Find-Set($x$)
1. if $x \neq x.p$
2.     $x.p \leftarrow$ Find-Set($x.p$)
3. return $x.p$
```

For m operations on n sets, worst-case time is $O(m\alpha(n))$.

$\alpha(n)$ is inverse of Ackermann's function. It is ≤ 4 if $n \leq 2^{2048} = 16^{512}$.

Strongly-Connected Components

$G^T = G$ but with reversed arcs

1. DFS(G), recording finishing times.
2. DFS(G^T), starting from vertices with higher finishing times first (in outer loop)
3. each tree in second DFS is a SCC

let $f(C)$ be max of any finishing time in C

- G and G^T have same SSCs.
- If G has an arc from some $u \in C_i$ to some $v \in C_j$, $f(C_i) > f(C_j)$.
- If G has an arc from C_i to C_j, G^T can't have such an arc.
- If there is an arc in G^T from C_j to C_i, then according to first DFS, $f(C_i) > f(C_j)$.
- When the second DFS is processing C_j in G^T, all vertices in C_i will already be finished.

EOLQs

For example:

- What's still confusing?
- What question didn't you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.
Thanks!