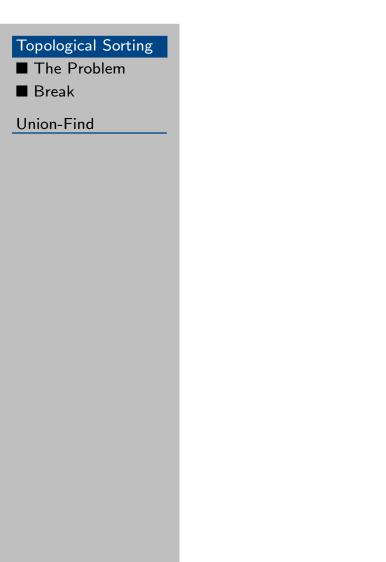
CS 758/858: Algorithms

Topological Sorting

Union-Find

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



Class 14, CS 758 – 2 / 14

The Problem

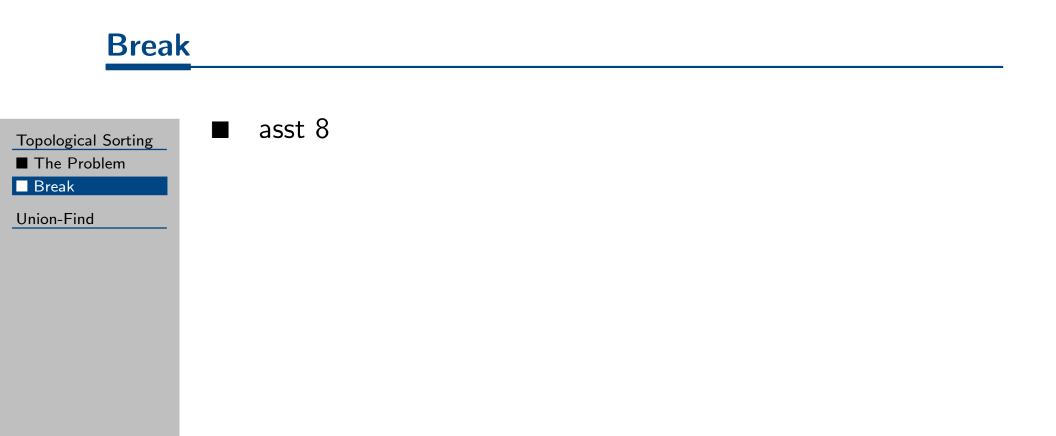
Topological Sorting
The Problem

Break

Union-Find

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?



Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets
- Speed-Ups
- Pseudo-code
- More Pseudo-code
- Strongly
- EOLQs

Union-Find

Topological	Sorting
-------------	---------

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets
- Speed-Ups
- Pseudo-code
- More Pseudo-code
- Strongly
- EOLQs

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

Topological	Sorting
-------------	---------

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets
- Speed-Ups
- Pseudo-code
- More Pseudo-code
- Strongly
- EOLQs

MAKE-SET(x) makes new set containing x

 $\mathrm{UNION}(x,y)$ combine the set containing x with the set containing y

FIND-SET(x) return a representative of the set containing x

■ Union-Find ADT

Union-Find

■ Components

Algorithm

■ Speed-Ups

■ Strongly

■ EOLQs

■ Disjoint Sets

Pseudo-code

■ More Pseudo-code

find-components

- 1. foreach vertex v
- 2. Make-Set(v)
- 3. for each edge (u, v)
- 4. UNION(u,v)

in-same-component?(u,v)5. is FIND-SET(u) = FIND-SET(v)?

Disjoint Sets

Topological Sorting

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets
- Speed-Ups
- Pseudo-code
- More Pseudo-code
- Strongly
- EOLQs

set is a tree rooted at representative

How to implement make, union, find?

Speed-Ups

Topological Sorting

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets

Speed-Ups

- Pseudo-code
- More Pseudo-code
- Strongly
- EOLQs

union by rank track approximate height, put shorter under taller

path compression find FIND-SET, ensure touched nodes point directly to root

Pseudo-code

Topological Sorting

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets
- Speed-Ups
- Pseudo-code
- More Pseudo-code
- Strongly
- EOLQs

Make-Set(x)

- 1. $x.p \leftarrow x$
- 2. x.rank $\leftarrow 0$
- 3. UNION(x, y)
- 4. $x \leftarrow \text{FIND-Set}(x)$
- 5. $y \leftarrow \text{Find-Set}(y)$
- 6. if x.rank > y.rank
- 7. $y.p \leftarrow x$
- 8. else

9.
$$x.p \leftarrow y$$

- 10. if x.rank = y.rank
- 11. increment y.rank

More Pseudo-code

Topological Sorting

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets

Speed-Ups

Pseudo-code

More Pseudo-code

■ Strongly

EOLQs

```
FIND-SET(x)
1. if x \neq x.p
```

- 2. $x.p \leftarrow \text{FIND-Set}(x.p)$
- 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 $n \leq 2^{2048} = 16^{512}.$

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets
- Speed-Ups
- Pseudo-code
- More Pseudo-code
- Strongly
- EOLQs

- $G^T = G$ 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

Topological Sorting

Union-Find

- Components
- Union-Find ADT
- Algorithm
- Disjoint Sets
- Speed-Ups
- Pseudo-code
- More Pseudo-code
- Strongly
- 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!