Disjoint Sets

IE 496 Lecture 14

Reading for This Lecture

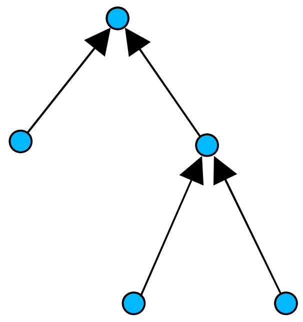
- Horowitz and Sahni, Chapter 2
- Kozen, Lecture 10-11

Data Structures for Disjoint Sets

- We have a set S and a partition $S_1, \ldots, S_n$ of S.
- We want a data structure that supports
  - union()
  - find()
- Applications
  - Constructing equivalence classes
  - Graph algorithms

Union-Find

- Represent each member of the partition as a rooted tree.
- Choose a designated "representative".
- All other elements are connected to the representative.



First implementation

- union()
  - Point root of set *A* to root of set *B*
- find()
  - Follow the path to the root.
- Analysis

A Tale of Two Heuristics

- How can we improve the complexity of `find()`?
- Heuristic 1

- Heuristic 2

Analysis

- Heuristic 1 guarantees that the depth of each tree is no more than $\lfloor log\ n \rfloor + 1$.
- The proof of this is by induction.
- This implies that `find()` can be performed in *O(log n)*
- Heuristic 2 allows us to perform `find()` in *almost* constant time (amortized).

Ackerman's Function

- Ackerman's function is an extremely fast growing function.
- Definition
  - $A_0(x) = x + 1$
  - $A_{k+1}(x) = A_k^x(x)$, where $A_k^{i+1}(x) = A_k(A_k^i(x))$
- $A_0(x) = x+1$, $A_1(x) = 2x$, $A_2(x) = x2^x$, $A_3(x) \geq 2 \uparrow x$
- $A_4(2)$ is greater than the number of particles in the known universe or the number of nanoseconds since the Big Bang (large number).

Inverse Ackerman's Function

- Define $A(k) = A_k(2)$.
- Now define $\alpha(n)$ = smallest k such that $A(k) \geq n$
- $\alpha(n)$ is the inverse Ackerman's function
- $\alpha(n)$ is 4 for all practical purposes.
- Let $T(m,n)$ be the running time of a sequence of $m \geq n$ `find()` operations and n-1 `union()` operations.
- $T(m, n) \in O(\alpha(n)(m+n))$