CSE 373: Disjoint Sets

book 8 – 8.5

Pete Morcos
University of Washington
4/24/00

http://www.cs.washington.edu/education/courses/cse373/00sp

Equivalence

- Two integers A and B are either <, >, or ==
  - Only equal if they are the same
- Sometimes we care about a weaker condition than equality, called *equivalence*, represented by ~
- The equivalence operator obeys the following properties:
  - Reflexive: A ~ A
  - Symmetric: A ~ B means that B ~ A
  - Transitive: A ~ B and B ~ C means that A ~ C

Equivalence Classes

- Operator divides the universe into disjoint sets of "equivalent" items
- These sets are called *equivalence classes*
  - Electrically, A ~ B if there is a wire path between them
  - On a map, A ~ B if a road runs between them
  - Modulo-N divides the integers into N equivalence classes
    - Example: under modulo 5, 3 ~ 8 ~ 13 ~ 18 ~ 23
  - Genetically, A ~ B if they are blood-related
- Given a set of equivalent pairs, we want to figure out the equivalence classes
  - If no pairs are equivalent, there will be N classes, one per item
  - Minimum of one class

Disjoint Set ADT

- Stores N unique items
- Divides them into E classes, 1 < E <= N
  - Classes are assigned arbitrary names; e.g. "1" to "N"
- Two operations:
  - Find—given an item, return the name of its equivalence class
  - Union—given the names of two equivalence classes, merge them into one class (which may have a new, arbitrary name)

Tradeoffs and Naive Implementations

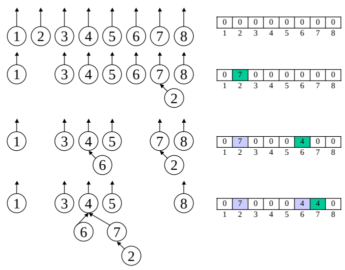
- Make Find fast, Union slow
  - Use array, with each element holding class name for that item
    - e.g., if 3 ~ 5, pick 5 as class name, and A[3] == A[5] == 5
  - Find is O(1), Union is O(N)
- Make Union fast, Find slow
  - Use linked lists, one for each class
    - Class name might be a pointer to head of list
  - Union is simple list append, O(1)
  - Find is a full scan of all lists, O(N)
- If we do N-1 unions (the max) and M finds, both are O(MN)
  - We'll find a way to be O(M + N) [sort of]

Data Structure

- Use a forest with one tree per equivalence class
  - Name of class is whatever item is at the root
- Unusual since we only need parent pointers
  - Find follows pointers to root
- Union simply makes one tree a subtree of the other—Finds will automatically find new root
- Since each node just has one pointer (to parent), can use an array where each array element is the index of the parent

Example

- Initally there are N trees
- Union (7,2)
  - i.e., 7 ~ 2
- Union (4,6)
- Union(4,7)

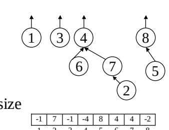


Analysis

- Union is obviously O(1)
- Find depends on prior sequence of unions
  - Worst case: 1~2, 2~3, 3~4, ... O(N)
  - Best case: 1~2, 1~3, 1~4, ... O(1)
- Average case is ambiguous; what's an average sequence of unions?
  - Any pair of classes equally likely
  - Any pair of elements equally likely
  - could think of others
- For M finds, N unions, worst case is O(MN)—quadratic time

Union-by-xxxx

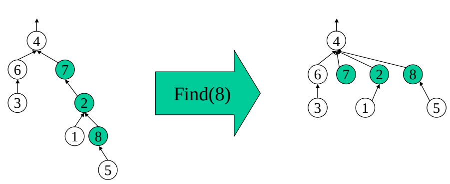
- We can keep depth of trees low by always making smaller tree a child of the larger
  - Thus, each time a tree becomes a child and increases depth by one, it at least doubles in size
  - At most log N doublings possible, so Find is O(log N)
- "Smaller" is ambiguous
  - Count of nodes: union-by-size
  - Height of trees: union-by-height
- Simple trick allows us to keep using array representation
  - Instead of storing 0 in all roots, store negative size
- Worst case O(M log N); average O(M+N)



Path Compression

- Common inputs still get worst case of O(M log N)
  - Union operation then creates tall, skinny trees
- Modify Find to have side effects
  - Make all nodes traversed on the way to the root point to the root
- In conjunction with union-by-xxxx, a sequence of M finds and N unions is O(M + N)
  - almost...slight math complication in book; don't worry about it

Path Compression Example



Amortized Analysis

A brief introduction

- A single Find could require log N steps, even with path compression
  - Naive analysis would say it's still O(M log N)
- However, future Finds will be faster
- Amortized analysis computes total cost for any sequence of operations, and averages out the total
  - Applied to Union/Find, works out to O(M + N)
- We may see more on amortized analysis later in the quarter