Disjoint Sets II
Chapter 8 in Weiss

CSE 326
Data Structures
Ruth Anderson

Today’s Outline

• Announcements
  – Project 3 partner selection due Mon Feb 22 by 11pm, DO NOT WAIT UNTIL THEN TO START!
  – Written Homework #6 due Friday 2/26

• Today’s Topics:
  – Disjoint Sets
  – Sorting

Weighted Union/Union by Size

• Weighted Union
  – Always point the smaller (total # of nodes) tree to the root of the larger tree

Example Again

Find(x) in tree T takes O(log n) time.
– Can we do better?

Analysis of Weighted Union

With weighted union an up-tree of height h has weight at least $2^h$.

• Proof by induction
  – Basis: $h = 0$. The up-tree has one node, $2^0 = 1$
  – Inductive step: Assume true for all $h' < h$.

Analysis of Weighted Union (cont)

Let T be an up-tree of weight n formed by weighted union. Let h be its height.

$$n \geq 2^h$$

$$\log_2 n \geq h$$

• Find(x) in tree T takes $O(\log n)$ time.
  – Can we do better?
Worst Case for Weighted Union

n/2 Weighted Unions

n/4 Weighted Unions

Example of Worst Cast (cont’)

After n/2 + n/4 + ... + 1 Weighted Unions:

Find

If there are n = 2^k nodes then the longest path from leaf to root has length k.

Array Implementation

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>up</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>weight</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Weighted Union

W-Union(i, j : index){
//i and j are roots
wi := weight[i];
wj := weight[j];
if wi < wj then
    up[i] := j;
    weight[j] := wi + wj;
else
    up[j] := i;
    weight[i] := wi + wj;
}

New runtime for Union():

New runtime for Find():

runtime for m finds and n-1 unions =

Nifty Storage Trick

• Use the same array representation as before
• Instead of storing -1 for the root, simply store -size

[Read section 8.4, page 276]

How about Union-by-height?

• Can still guarantee O(log n) worst case depth

Left as an exercise!

• Problem: Union-by-height doesn’t combine very well with the new find optimization technique we’ll see next
Path Compression

- On a Find operation point all the nodes on the search path directly to the root.

```
int Find(Object x) {
    // x had better be in // the set!
    int xID = hTable[x];
    int i = xID;
    // Get the root for // this set
    while(up[xID] != -1) {
        xID = up[xID];
    }
    // Change the parent for // all nodes along the path
    while(up[i] != -1) {
        temp = up[i];
        up[i] = xID;
        i = temp;
    }
    return xID;
}
```

(New?) runtime for Find:

Interlude: A Really Slow Function

**Ackermann's function** is a really big function $A(x, y)$ with inverse $\alpha(x, y)$ which is really small

How fast does $\alpha(x, y)$ grow?

$\alpha(x, y) = 4$ for $x$ far larger than the number of atoms in the universe ($2^{100}$)

$\alpha$ shows up in:
- Computation Geometry (surface complexity)
- Combinatorics of sequences
A More Comprehensible Slow Function

\[ \log^* x = \text{number of times you need to compute } \log \text{ to bring value down to at most 1} \]

E.g. \( \log^* 2 = 1 \)
\( \log^* 4 = \log^* 2^2 = 2 \)
\( \log^* 16 = \log^* 2^4 = 3 \quad (\log \log 16 = 1) \)
\( \log^* 65536 = \log^* 2^{2^8} = 4 \quad (\log \log \log 65536 = 1) \)
\( \log^* 2^{2^{5536}} = \ldots \ldots \ldots = 5 \)

Take this: \( \alpha(m,n) \) grows even slower than \( \log^* n \)!!

Disjoint Union / Find with Weighted Union and PC

- Worst case time complexity for a W-Union is \( O(1) \) and for a PC-Find is \( O(\log n) \).
- Time complexity for \( m \geq n \) operations on \( n \) elements is \( O(m \log^* n) \) where \( \log^* n \) is a very slow growing function.
  - \( \log^* n < 7 \) for all reasonable \( n \). Essentially constant time per operation!
- Using “ranked union” gives an even better bound theoretically.

Amortized Complexity

- For disjoint union / find with weighted union and path compression.
  - average time per operation is essentially a constant.
  - worst case time for a PC-Find is \( O(\log n) \).
- An individual operation can be costly, but over time the average cost per operation is not.