Today’s Outline

- Admin
  - Project 3 will be out Thursday
  - Introduction in tomorrow’s section
- Midterm statistics
  - Sample solutions
  - B-Tree clarification

- Disjoint Set ADT
  - Union-Find implementation

Get ready for Project 3!

- Find a partner and send me an email!
  - someone you haven’t worked with yet

- Save your work from project 2
  - your team ID may change
  - may not be able to access old shared directory

Midterm Statistics

<table>
<thead>
<tr>
<th>Total points</th>
<th>Max</th>
<th>Min</th>
<th>Average</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>70</td>
<td>31</td>
<td>57.8</td>
<td>57</td>
</tr>
</tbody>
</table>

Good job!

Histogram of Midterm Scores

B-Tree Clarification

- Homework 2, problem 1, parts c and d
- Sample solutions split internal node differently than what we did in class
- M is # pointers, not # keys
Our Last Data Structure!

- Implementation of the Disjoint Set ADT
- Uses Union-Find Algorithm

Hm… so what will we do next?

- Sorting algorithms, Graph based algorithms, …
- Use data structures learned to make these efficient!

Maze Construction Algorithm

- Given:
  - A collection of rooms V
  - Walls/doors between the rooms (initially no doors) E
  - We want to build a collection of walls to knock down, \( E' \subset E \), such that one unique path connects every two rooms

```plaintext
While edges remain in E {
  \( (x, y) = \text{RemoveRandomWall}() \)
  if( \( x \) and \( y \) are not connected so far ) {
    Add \( (x, y) \) to \( E' \)
    Mark \( x \) and \( y \) as connected
  }
}
```

connected = have a direct or indirect path

Motivation: What’s a Good Maze?

The Problem, Formally

- “If \( x \) and \( y \) have not yet been connected”
  - Are two elements in the same set?
- “Mark \( x \) and \( y \) as connected”
  - Form the union of two sets

Disjoint Set ADT

Data: elements (no priority, not necessarily comparable)

Operations
1. Find(x)
   - Returns set identifier
   - Find(x) = Find(y) if x and y are in the same set
2. Union(A, B)
   - Arguments are set identifiers
3. MakeNewSet(item)
   - Create a new set containing only item

Disjoint Set: Properties

- Equivalence property
  - Every element of a DS belongs to exactly one set

- Dynamic equivalence property
  - The set of an element can change after execution of a union

Note: Underlined elements are set IDs
Modified Maze Construction Algorithm

While edges remain in $E$

$(A, B) = \text{RemoveRandomWall}()$

if( $\text{Find}(A) \neq \text{Find}(B) )$

$E' = E' \cup \{A, B\}$

$\text{Union( Find(A), Find(B) )}$

Maze Construction Example

Construct this maze!

Initially (the identifier of each set is underlined):

\{a\} \{b\} \{c\} \{d\} \{e\} \{f\} \{g\} \{h\} \{i\}

Order of edges in blue

Example, continued

\{a\} \{b\} \{c\} \{d\} \{e\} \{f\} \{g\} \{h\} \{i\}

find(b) $\Rightarrow$ b

find(c) $\Rightarrow$ c

find(b) $\neq$ find(c) so:
add 1 to $E'$
union(b, c)

Result:

Order of edges in blue

Implementing the DS ADT

- $n$ elements, $m$ finds, $\leq n-1$ unions \(\text{can there be more unions?}\)

- Target complexity: $\Theta(m+n)$
  \(\text{i.e. } \Theta(1) \text{ amortized}\)

- $\Theta(1)$ worst-case for find as well as union would be great, but...

  Known result: both find and union cannot be done in worst-case $\Theta(1)$ time

Attempt #1

- Hash elements to a hashtable
- Store set identifier for each element as data

  runtime for find:
  
  runtime for union:
  
  runtime for $m$ finds, $n-1$ unions:

Attempt #2

- Hash elements to a hashtable
- Store set identifier for each element as data
  \(\text{Link all elements in the same set together}\)

  runtime for find:
  
  runtime for union:
  
  runtime for $m$ finds, $n-1$ unions:
Attempt #3
• Hash elements to a hashtable
• Store set identifier for each element as data
• Link all elements in the same set together
• Always update identifiers of smaller set
  runtime for find:
  runtime for union:
  runtime for m finds, n-1 unions:

[Read section 8.2]

DS ADT Tree Representation
• Maintain a forest of up-trees
• Each set is a tree
• What’s a natural set identifier?

Find Implementation
Find(x)
  Traverse parents of x to the root

Runtime:

Union Implementation
Union(A, B)
  Join the two trees
  Since A and B are already the roots of a tree, this is easy!

Runtime:

More of the Example
union(b,e)

(extra space)
The Final Maze

[Diagram of the Final Maze]

Ooh... scary!
Such a hard maze!

Nifty storage trick

- A forest of up-trees can easily be stored in an array
- Use hashtable to map node names to array indices

Implementation

```java
int Find(Object x) {
    int xID = hTable[x];
    while(up[xID] != -1) {
        xID = up[xID];
    }
    return xID;
}

void Union(int x, int y) {
    up[y] = x;
}
```

Now this doesn’t look good L

Can we do better? Yes!

1. Improve union so that find only takes \(\Theta(\log n)\)
   - Union-by-size
   - Reduces complexity to \(\Theta(m \log n + n)\)

2. Improve find so that it becomes even better!
   - Path compression
   - Reduces complexity to almost \(\Theta(m + n)\)

To Do

- Find partner for Project 3
  - Send me email
- Read Chapter 8