CompSci 161
Winter 2023 Lecture 21:
Greedy Algorithms:
Kruskal’s Algorithm, Union-Find Data Structure
Finding a MST (Kruskal)

1st attempt rejected
Disjoint Sets

**Goal:** Quickly determine \(a, b\) connected?

**Answer:** Disjoint-set data structure

- Maintain a collection of disjoint dynamic sets
- Identify each set by a representative
- Support the following operations:
  - **Make-Set:** constructor
  - Given a vertex, in which tree?
    - \(x, y\) same tree? Same answer.
  - Two vertices, merge
    - Different trees now connected
An Application: Counting Disjoint Sets

Edges | Collection
--- | ---
initial | \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}
(d, e) | \{a\}, \{b\}, \{c\}, \{d, e\}, \{f\}, \{g\}, \{h\}, \{i\}, \{j\}
(e, f) | \{a\}, \{b\}, \{c\}, \{d, e, f\}, \{g\}, \{h\}, \{i\}, \{j\}
Linked List Representation

- Objects represented by linked list
  - Each set has head, tail
  - Each element has member, next, pointer back
- Implementation of Union($x, y$)
  - Append $y$’s list onto end of $x$’s
  - Convenient use of tail pointer

- How long does a series of operations take?
Disjoint-set Forests

- Represent sets by rooted trees
- Each tree is one set
- Each node has one member

Support data structure

Vertices w/ parent ptr
Continuing the structure

- Edge $BE$?
- Edge $CG$?
- Edge $AD$?
Continuing the structure

- Edge $EI$?
- Edge $AC$? **rejected**
- Edge $IL$? **rejected**
- Edge $DII$?
Running time for Operations

Union(A, B)

\[ X \leftarrow \text{find}(A) \]
\[ Y \leftarrow \text{find}(B) \]

\[ \text{if } X \neq Y \text{ then} \]
\[ X \cdot \text{parent} \leftarrow Y \]

\[ \text{if } X \cdot \text{count} + = Y \cdot \text{count} \]
\[ \text{if } A \cdot \text{parent} \neq \text{nullptr} \]
\[ \text{then} \]

\[ \text{return } \text{find}(A \cdot \text{parent}) \]
\[ \text{return } A \]

\[ 2 \times \text{find} + \mathcal{O}(1) \]

- If there are \( n \) elements, times?
- Can we improve the worst-case for one?

Union by rank.
Example of Union by Rank

This is the result of *Union by Rank*

- Ties are broken alphabetically
  - Earlier letter $\rightarrow$ later letter (when tied)

- Running time for operations? $O(\log n)$
Running time for Operations

Union(A,B)

\[
X \leftarrow \text{find}(A) \\
Y \leftarrow \text{find}(B) \\
\text{if } X \neq Y \text{ then} \\
\quad \text{if } X.\text{count} > Y.\text{count} \text{ then} \\
\quad \quad \text{Swap } X \text{ and } Y \\
\quad X.\text{parent} \leftarrow Y \\
\quad Y.\text{count} += X.\text{count}
\]

find(A)

\[
\text{if } A.\text{parent} \neq \text{nullptr} \text{ then} \\
\quad \text{return } \text{find}(A.\text{parent}) \\
\text{return } A
\]

- If there are \( n \) elements, find is \( O(\log n) \)
- Union takes time \( 2 \times \text{find} + O(1) \)
Improving Find

Can we improve `find` further?

```cpp
find(const Key &k) const

find(A)
    if A.parent != nullptr then
        A.parent ← find(A.parent)
    return A.parent

return A
```

Path compression
Path Compression

- Path compression change worst-case of find?

- Does path compression improve over time?

- Suppose we have $m$ union and $f$ find operations