DISJOINT-SET DATA STRUCTURE: A Short Introduction

Ramoni ADEOGUN, PhD
Assistant Professor – Machine Learning and AI for Wireless Communications
Department of Electronic Systems, Aalborg University, DK
Email: ra@es.aau.dk

Introduction

Consider a situation with 10 persons with specified relationships (fig)

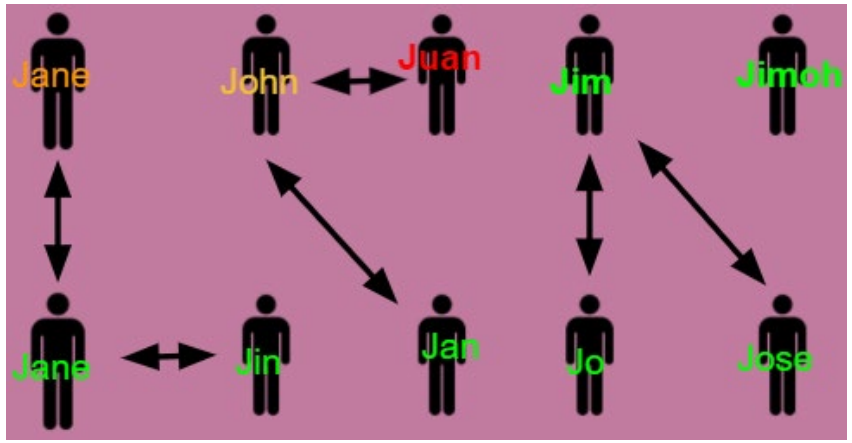
- Query: To find whether x and y belong to same group or not, i.e., to find if x and y are direct/indirect friends.
  - **Is Jack a friend of Jin?**

- Solution: Partitioning the individuals into different sets according to the groups in which they fall -> **disjoint set data structure**:

$$S_1 = \{Jack, Jane, Jin\}$$
$$S_2 = \{John, Juan, Jan\}$$
$$S_3 = \{Jo, Jose, Jim\}$$
$$S_4 = \{Jimoh\}$$

A **disjoint set data structure** maintains a collection $S = \{S_1, S_2, \dots, S_n\}$ of disjoint dynamic sets.

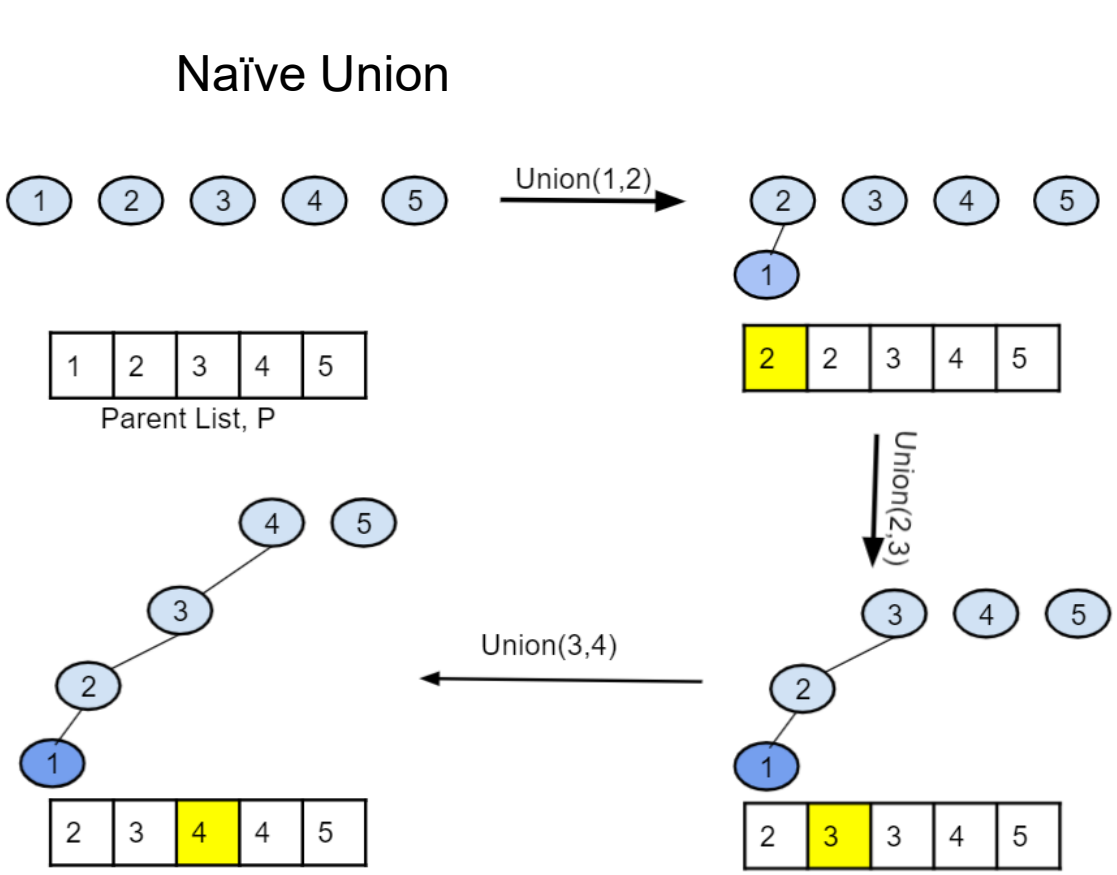
  - Each set has a representative -> a member of the set that identifies it, e.g., member with largest index
  - Each element is represented as an object
- Operations for an object Jo:
  - **FIND(Jo):** returns a pointer to the representative of the set having Jo
  - **UNION(Jo,Jan):** performs the union operation between set having Jo and that with Jan as member
  - **MAKE(x):** create a new set with a single member x

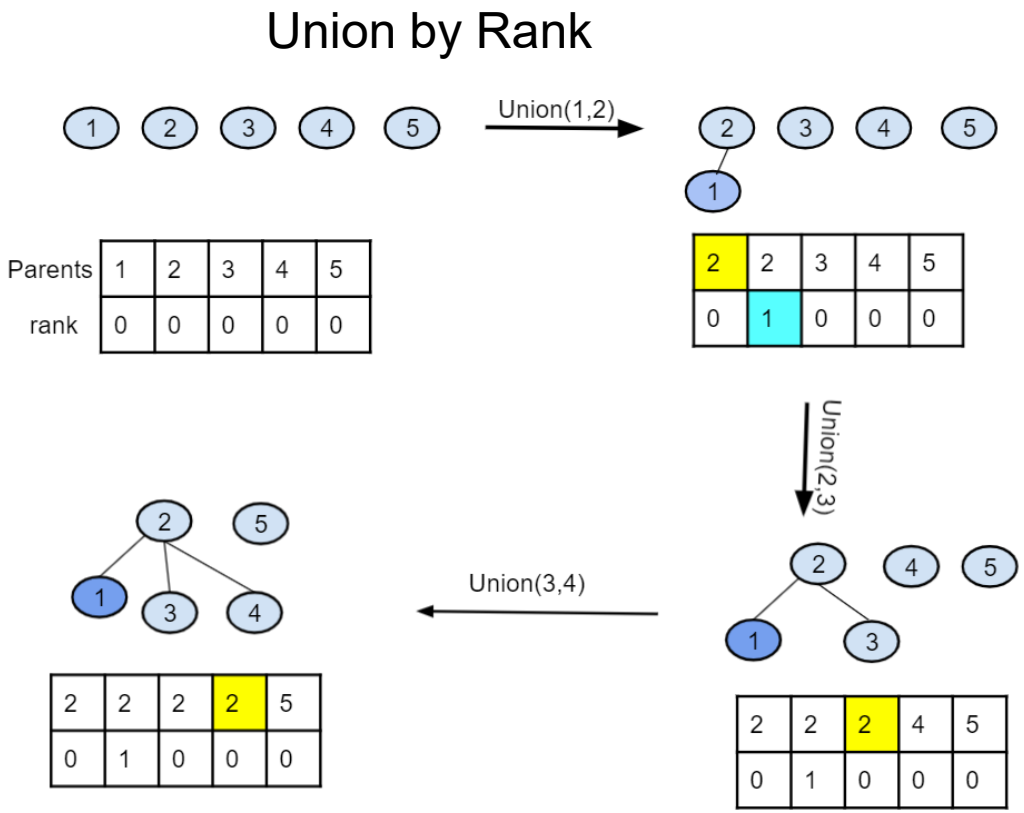


Is Jack a friend of Jin?

- If representatives of the sets containing Jack and Jin are same, then they are friends.

Set Operations– Union by Rank

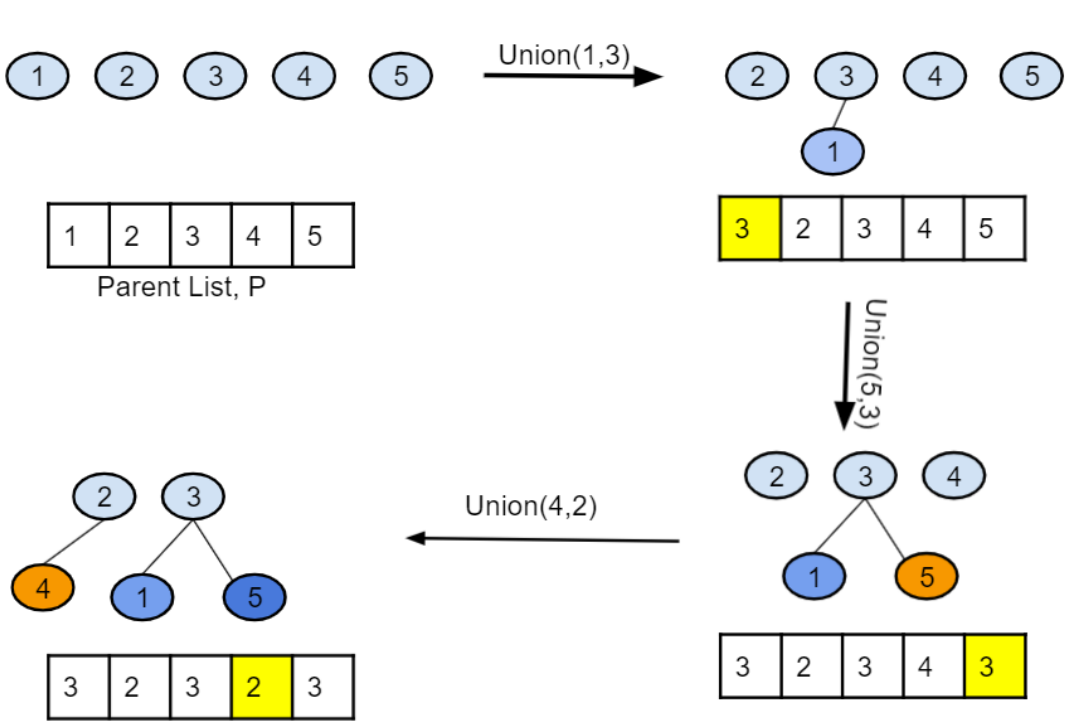


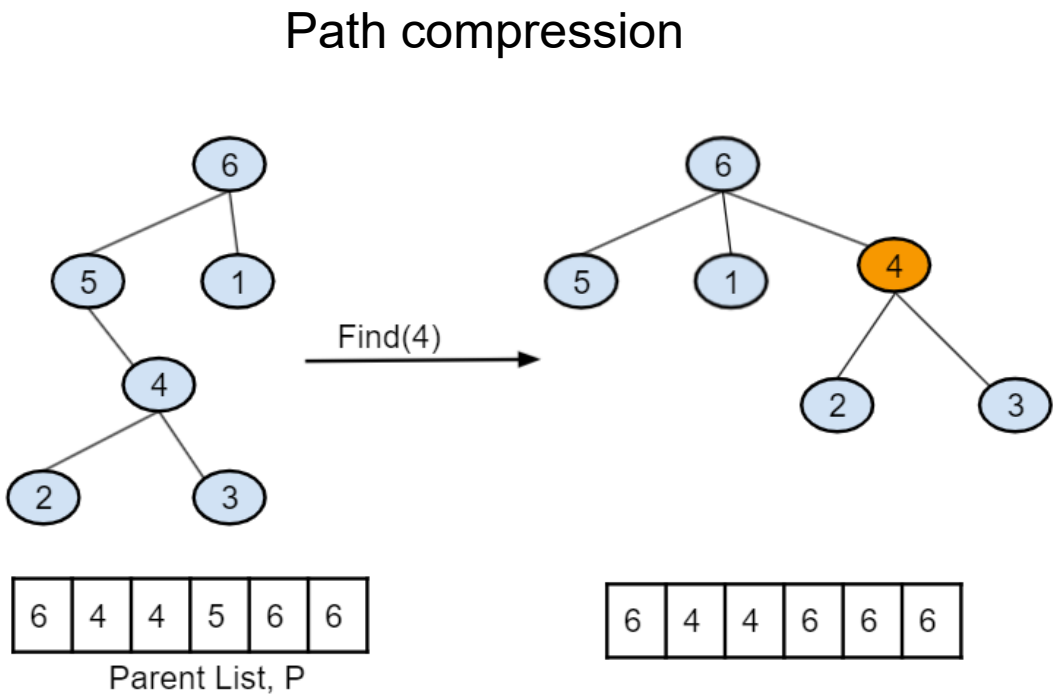


What happens when we call find(1)?

How about calling find(1) on this?

Set Operations – Union & Path Compression





Conclusion

- Disjoint set data structures are essential in several computer applications
- Disjoint sets models non-overlapping partitioning of items into groups
- Three basic operations are required for disjoint set data structures:
  - FIND -> which set does an item belong to?
  - UNION -> combine the items in two sets having specified members
  - MAKE -> Represent the sets as objects
- Efficient techniques on disjoint set data structure include:
  - Union by rank
  - Path compression