Sets and Set Algorithms

Robert Horvick

SOFTWARE ENGINEER

@bubbafat   www.roberthorvick.com

Overview

Set Overview

Basic Set Implementation

Set Algorithms

- Union
- Intersection
- Difference
- Symmetric Difference

Demo: Explorer data using set algebra

Set

A data structure that stores unique values in an undetermined order.

Set Properties

Contains only distinct items

Items are iterated in an implementation-defined order

Set operations are O(log n)

Set Examples

Name	Values
Integers	… -4, -3, -2, -1, 0, 1, 2, 3, 4 …
Positives	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 …
Negatives	… -7, -8,-6, -5, -4, -3, -2, -1
Evens	0, 2, 4, 6, 8, 10, 12, 14, 16, …
Odds	1, 3, 5, 7, 9, 11, 13, 15, 17, …

Set Examples

Name	Values
Teams	“LA Galaxy”, “Portland Timbers”, …
Players	“Diego Valeri”, “Ike Opara”, …

Example Set

```
public interface ISet<T> : IEnumerable<T>
    where T: IComparable<T>
{
    bool Add(T value);
    bool Remove(T value);
    bool Contains(T value);
    int Count { get; }

    ISet<T> Union(ISet<T> other);
    ISet<T> Intersection(ISet<T> other);
    ISet<T> Difference(ISet<T> other);
    ISet<T> SymmetricDifference(ISet<T> other);
}
```

◀ Set interface is enumerable

◀ Set type must be comparable

◀ Basic container operations

◀ Set algebra operations

```
public class Set<T> : ISet<T>
    where T : IComparable<T>
{
    private readonly AVLTree<T> _store;
    public Set()
    {
        _store = new AVLTree<T>();
    }
    public Set(IEnumerable<T> values)
        : this()
    {
        AddRange(values);
    }
```

◄ Set type implements ISet interface

◄ AVL tree used as backing store

◄ Empty tree created in empty ctor

◄ Constructor for adding existing values

```
public bool Add(T value) {
    if (!_store.Contains(value)) {
        _store.Add(value);
        return true;
    }
    return false;
}

private void AddRange(IEnumerable<T>
values) {
    foreach (T value in values) {
        Add(value);
    }
}
```

- Items are only added if they are not already in the set.
- Returns true if the item was added
- False otherwise
- Private method to help add multiple items

```
public bool Contains(T value) {
    return _store.Contains(value);
}
public bool Remove(T value) {
    return _store.Remove(value);
}
public int Count {
    get {
        return _store.Count;
    }
}
```

◀ Checks if the value is in the set

◀ Removes a value from the set (if it exists)

◀ Returns the number of items in the set

```
public IEnumerator<T> GetEnumerator() {
    return _store.GetEnumerator();
}


IEnumerator IEnumerable.GetEnumerator() {
    return _store.GetEnumerator();
}
```

- IEnumerable<T> methods defer to the AVL tree for enumeration. This AVL tree defaults to in-order traversal.

Algebra of Sets

MLS Data

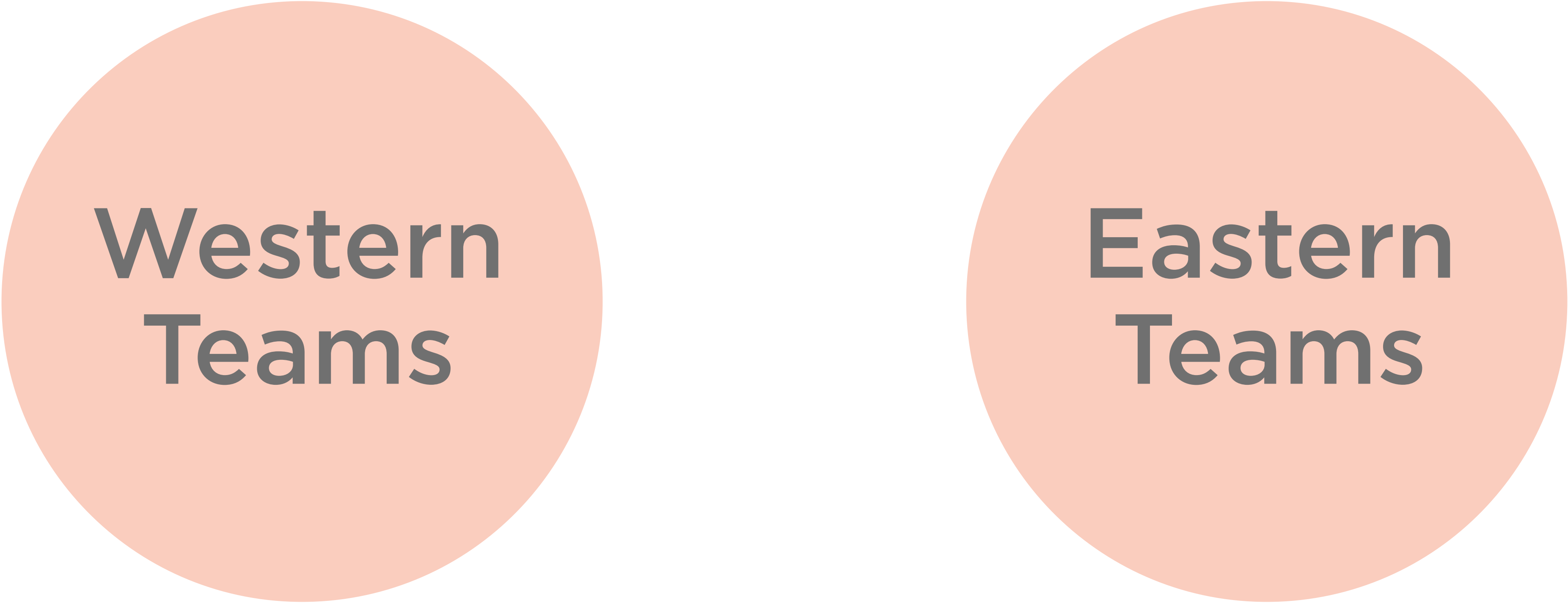
MLS Teams

```
Set<Team> teams = new Set<Team>();

teams.Add(new Team("LA Galaxy", “Western”));

teams.Add(new Team("Seattle Sounders FC", “Western”));

teams.Add(new Team("D.C. United", “Eastern”));

teams.Add(new Team("Toronto FC", “Eastern”));
```

Example: Creating a Set of Teams

We initialize the set type and add the teams to the set

MLS Data



Western Teams

Eastern Teams

```
Set<Team> westernTeams = teams.Where(t => t.Conference == "Western");

Set<Team> easternTeams = teams.Where(t => t.Conference == "Eastern");
```

Example: Creating Team Conference Sets

Create two new sets each containing the teams of the respective conference

MLS Data

MLS Players

MLS Data

LA Galaxy Players

Real Salt Lake Players

Chicago Fire FC Players

MLS Data

LA Galaxy Players

2016

LA Galaxy Players

2017

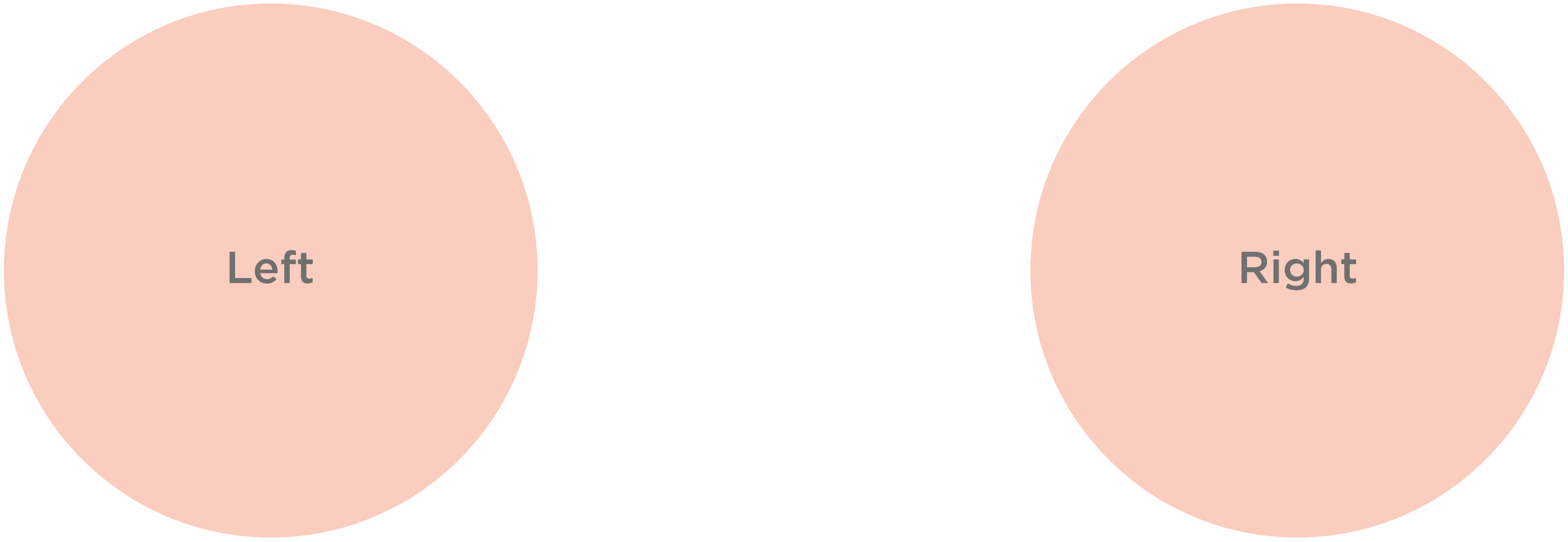
LA Galaxy Players

2018

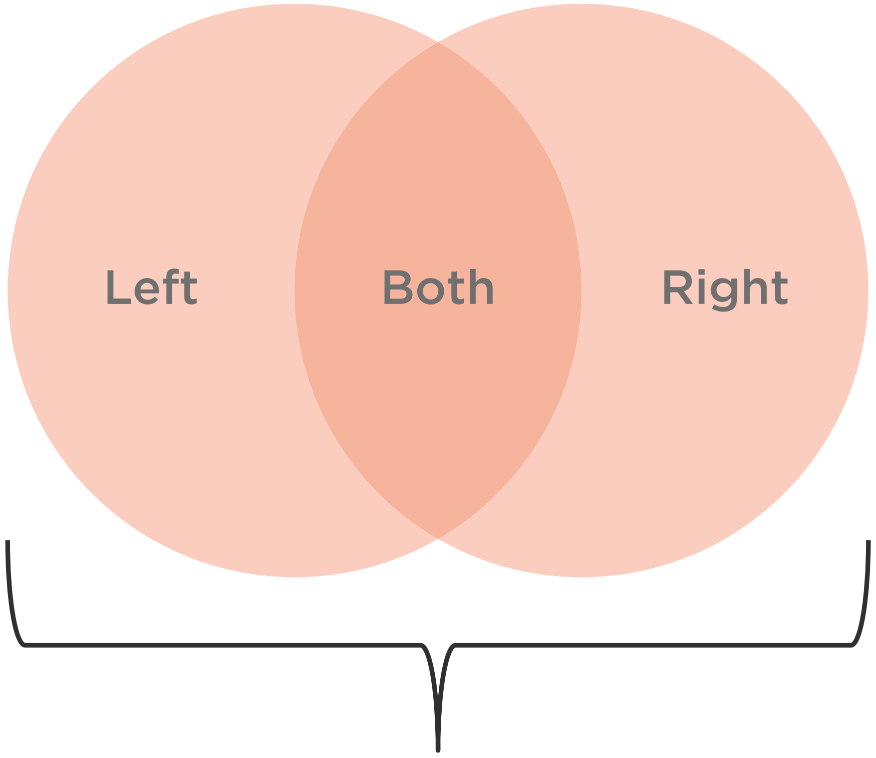
Union

The set of all distinct items that exist within any of the input sets

Union



Union



Union

Union answers “OR” questions

Union “or” Questions

Which players have played for the Sounders or and Galaxy?

Which students are in Algebra or Biology?

What books are available at the library or the bookstore?

What animals are birds or mammals?

```
Set<Player> sounders = players.Where(p => p.Team == ”Sounders");

Set<Player> galaxy = players.Where(p => p.Team == ”Galaxy");


Set<Player> either = galaxy.Union(sounders);
```

Example: Sounders or Galaxy Players

Create a set of the players who have played for either the Sounders or the Galaxy

```
public ISet<T> Union(ISet<T> other) {
    Set<T> result = new Set<T>(other);
    result.AddRange(_store);


    return result;
}
```

Union Algorithm

Create an output set that contains the distinct items from both input sets.

Intersection

The of items that exist within both input sets

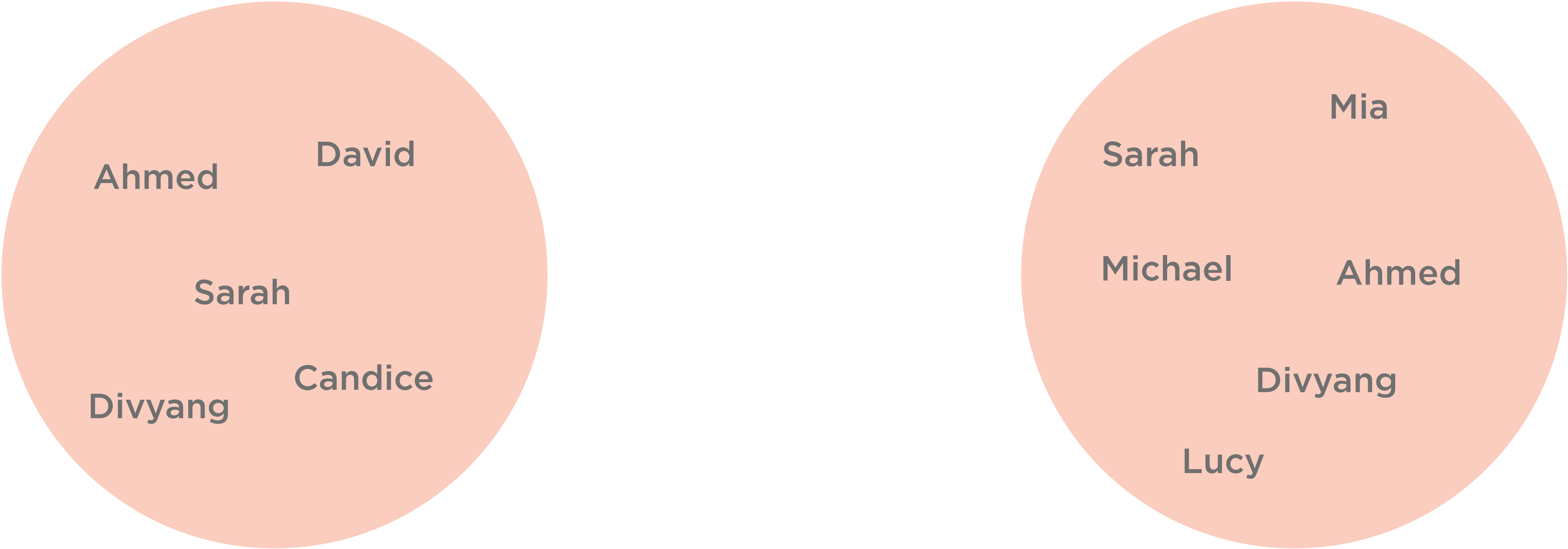
Intersection

Biology	Algebra
Ahmed	Lucy
Sarah	Michael
David	Sarah
Divyang	Mia
Candice	Divyang
	Ahmed

Intersection

Biology	Algebra
Ahmed	Lucy
Sarah	Michael
David	Sarah
Divyang	Mia
Candice	Divyang
	Ahmed

Intersection



David

Ahmed

Sarah

Candice

Divyang

Mia

Sarah

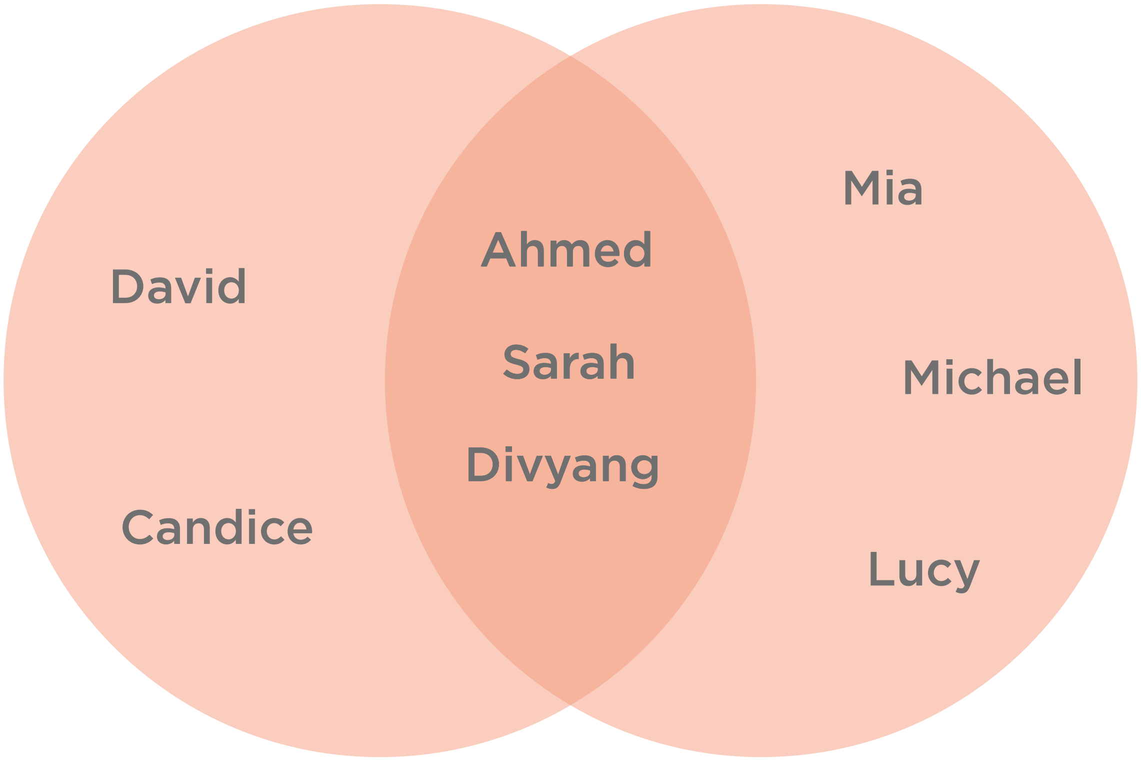
Michael

Ahmed

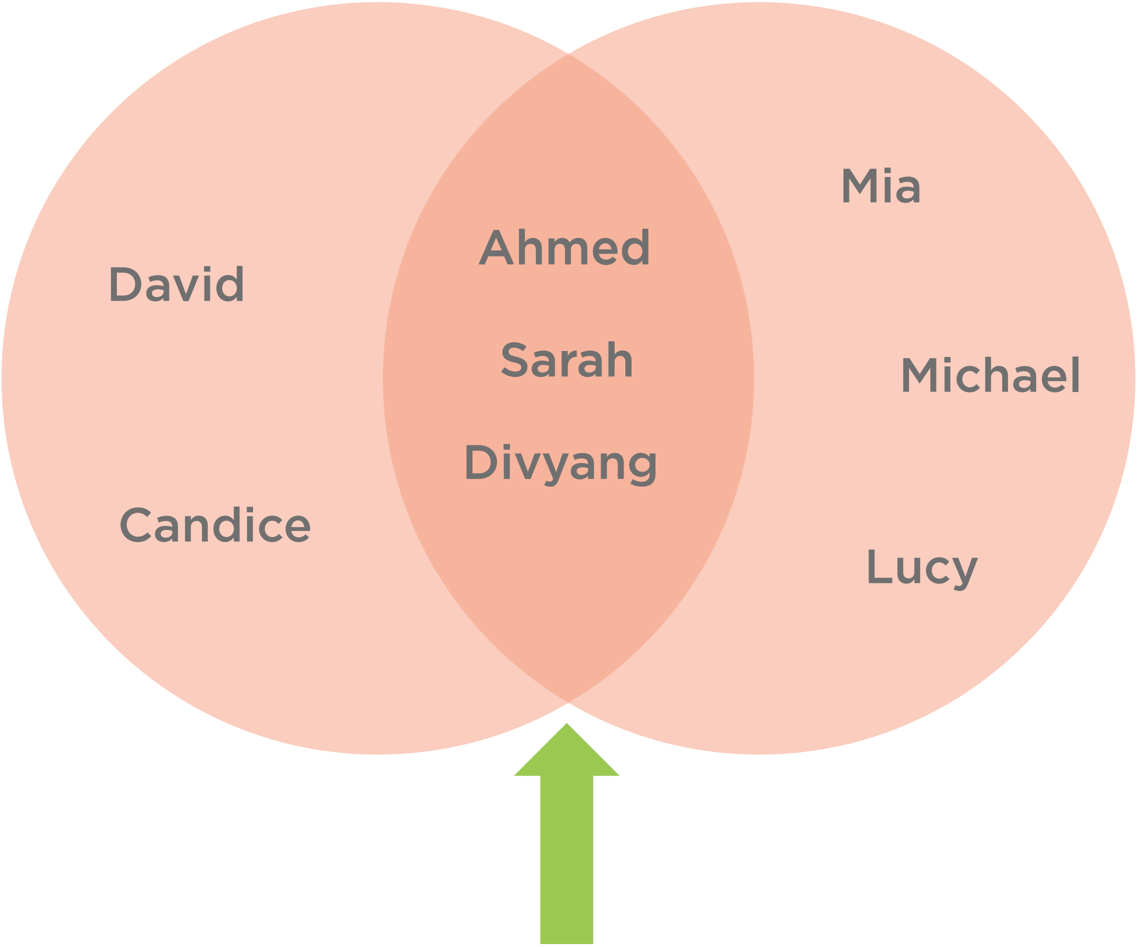
Divyang

Lucy

Intersection



Intersection



Intersection answers “AND” questions

Intersection “and” Questions

Which players have played for the Sounders and Galaxy?

Which students are in Algebra and Biology?

What books are available at the library and the bookstore?

What animals are birds and mammals?

```
Set<Player> sounders = players.Where(p => p.Team == ”Sounders");

Set<Player> galaxy = players.Where(p => p.Team == ”Galaxy");


Set<Player> both = galaxy.Intersection(sounders);
```

Example: Sounders and Galaxy Players

Create a set of the players who have played for the Sounders and the Galaxy

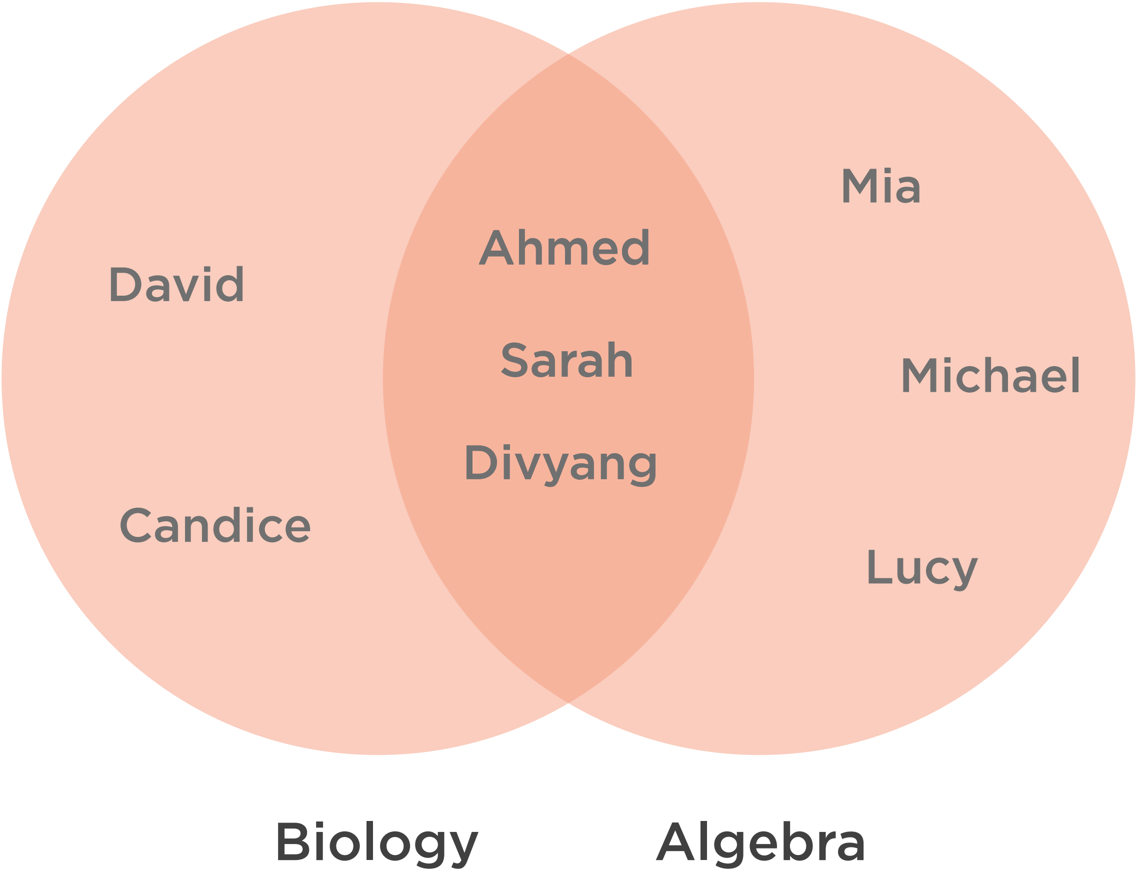
```
ISet<T> Intersection(ISet<T> other) {
    ISet<T> result = new Set<T>();

    foreach (T item in other) {
        if (Contains(item)) {
            result.Add(item);
        }
    }

    return result;
}
```

- Accepts the set to intersect with
- Create an empty result set
- If an item is in the current set and in the other set
- Add it to the result set
- Return the set of intersecting items

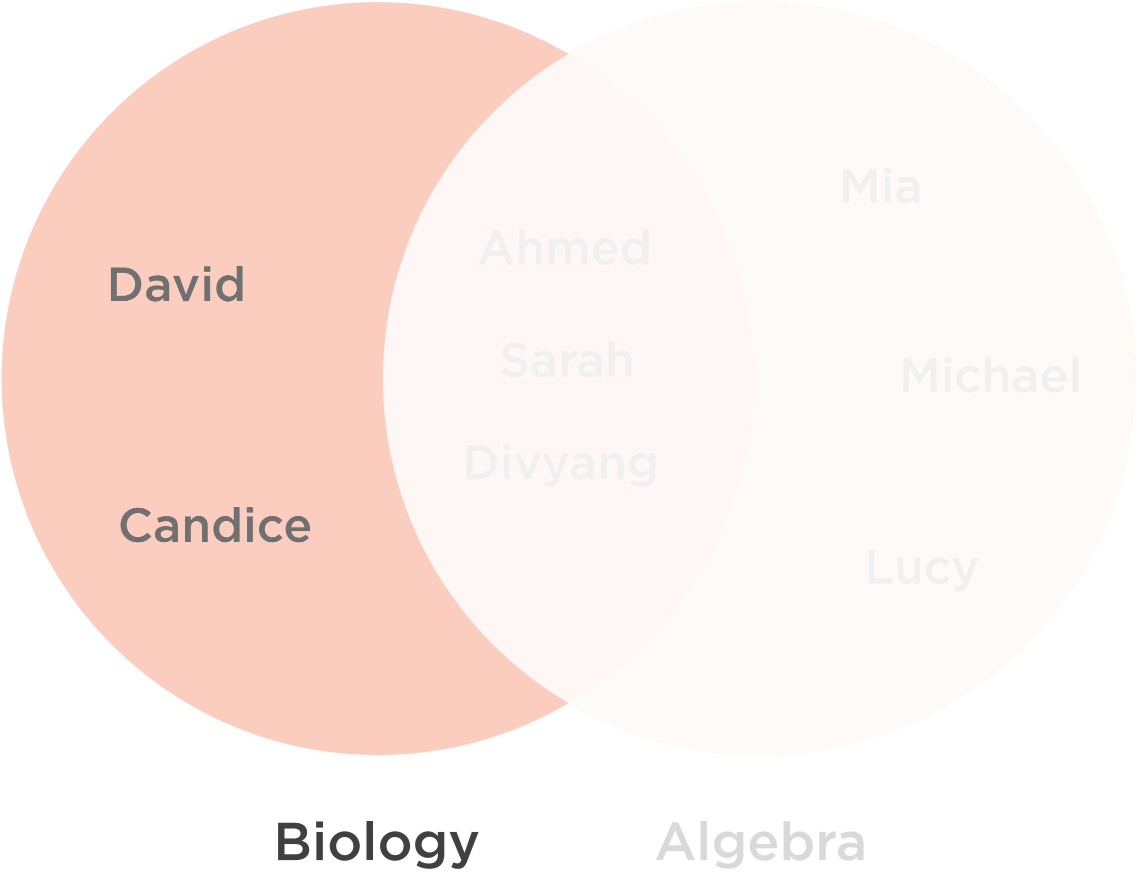
Difference

The set of items which exist in one set which do not exist in the other.

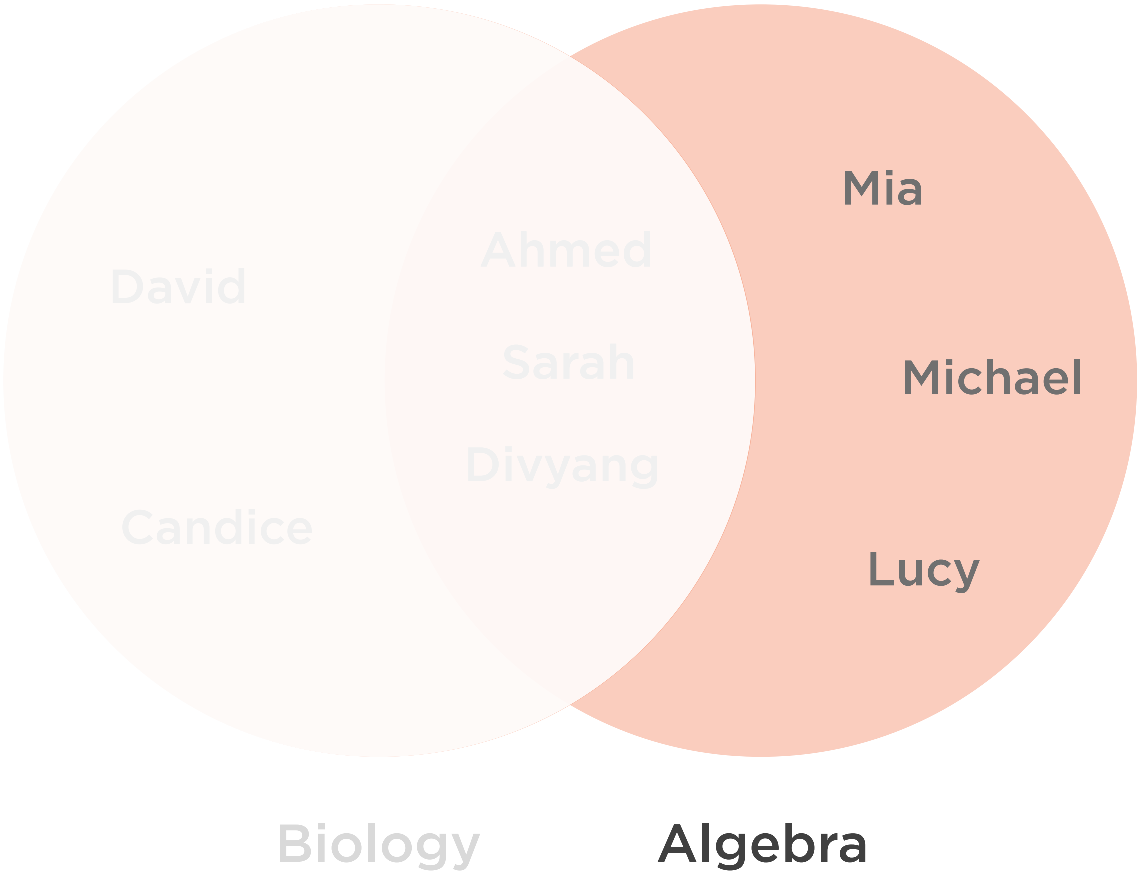
Difference


David
Candice
Ahmed
Sarah
Divyang
Mia
Michael
Lucy
Biology
Algebra

Difference


David
Candice
Ahmed
Sarah
Divyang
Mia
Michael
Lucy
Biology
Algebra

Difference


David
Candice
Ahmed
Sarah
Divyang
Mia
Michael
Lucy
Biology
Algebra

Difference answers “BUT NOT” questions for a single input set

Difference “but not” Questions

Which players have played for the Sounders but not Galaxy?

Which students are in Algebra but not Biology?

What books are available at the library but not the bookstore?

What animals are birds but not mammals?

```
Set<Player> sounders = players.Where(p => p.Team == ”Sounders");

Set<Player> galaxy = players.Where(p => p.Team == ”Galaxy");


Set<Player> soundersOnly = sounders.Difference(galaxy);
```

Example: Sounders But Not Galaxy Players

Create a set of the players who have played for the Sounders but have not played for the Galaxy

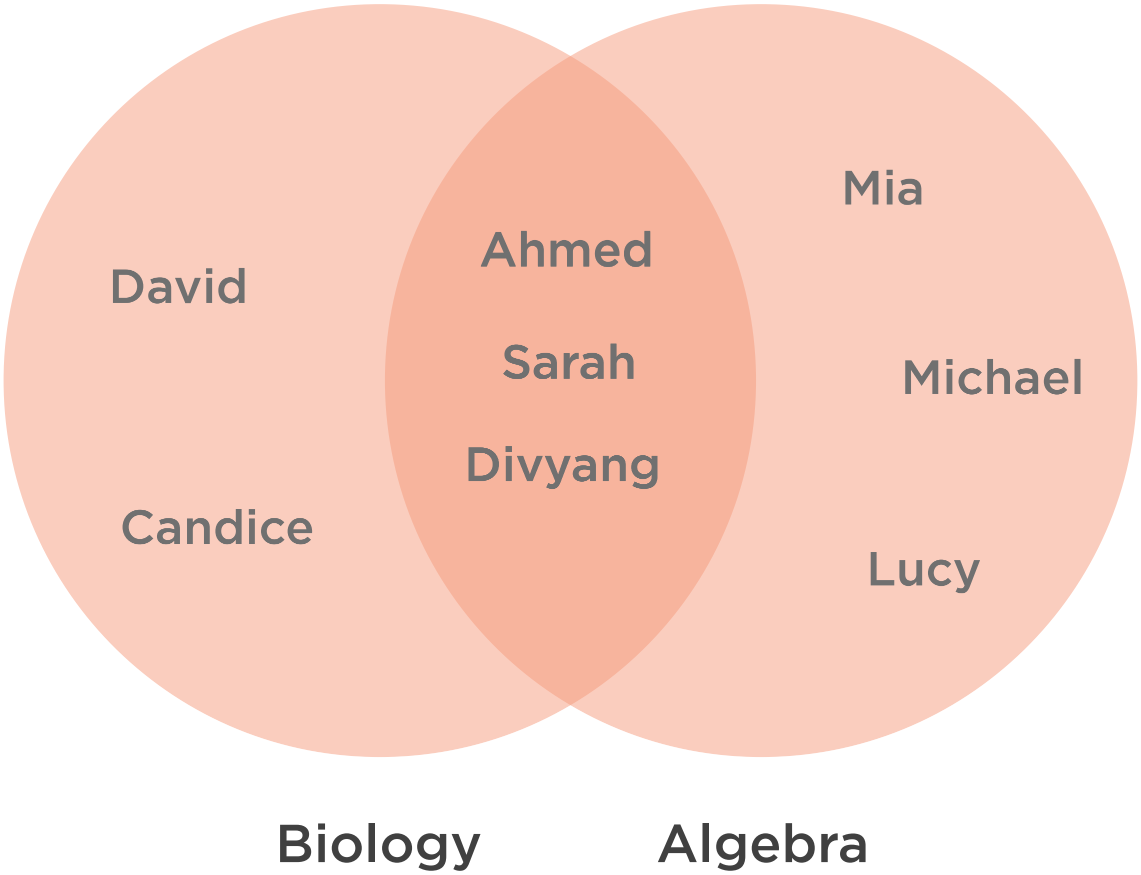
```
ISet<T> Difference(ISet<T> other) {

    ISet<T> result = new
Set<T>(_store);


    foreach (T item in other)

    {

        result.Remove(item);

    }


    return result;

}
```

- ◀ Accepts the set to difference with
- ◀ Create a result set with the current set’s items
- ◀ For each item in the other set
- ◀ Remove it from the result if it exists
- ◀ Return the set of differing items

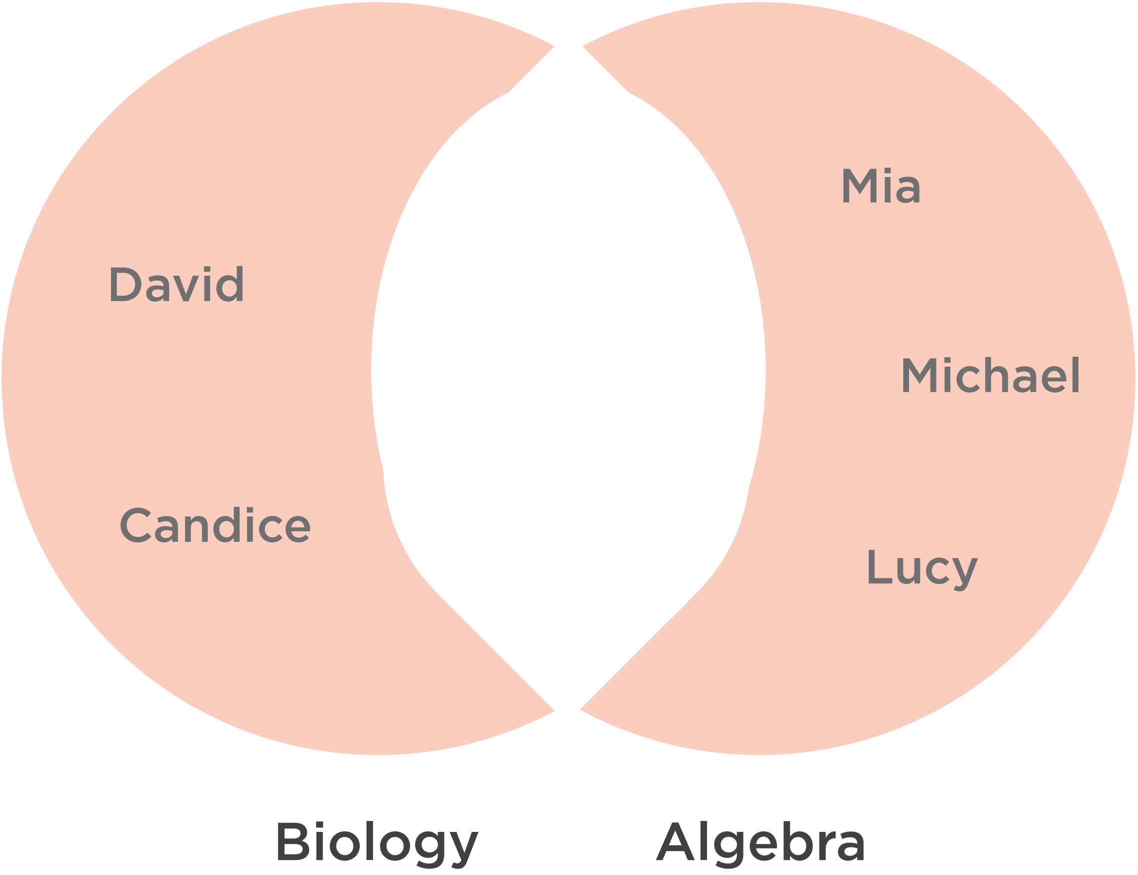
Symmetric Difference

The set of items which exist in either of the two input sets, but which are not in their intersection.

Symmetric Difference



Symmetric Difference



Symmetric Difference answers “OR … BUT NOT BOTH” questions

Symmetric Difference Questions

Which players have played for the Sounders or Galaxy but not both?

Which students are in Algebra or Biology but not both?

What books are available at the library or the bookstore but not both?

What animals are birds or mammals but not both?

```
Set<Player> sounders = players.Where(p => p.Team == ”Sounders");
Set<Player> galaxy = players.Where(p => p.Team == ”Galaxy");


Set<Player> both = galaxy.SymmetricDifference(sounders);
```

Example: Sounders or Galaxy But Not Both

Create a set of the players who have played for the Sounders or the Galaxy but have not played for both teams

```
ISet<T> SymmetricDifference(ISet<T>
other) {

    ISet<T> ntr = Intersection(other);

    ISet<T> union = Union(other);


    return union.Difference(ntr);

}
```

- Accepts the set to symmetric difference
- Intersects with the input set
- Union with the other set
- Returns the difference of the union and the intersection

Demo

Explore MLS team and player data

Answer various questions using set algebra