Sorting ~~Array~~ Linked List

- So far, our sorting algorithms work on arrays
- However, which sorting algorithm can work on **Linked List**?
  - Bubble Sort
  - Selection Sort
  - Insertion Sort
  - Merge Sort
  - Quick Sort

Sorting ~~Array~~ Linked List

- Discussion with your neighbors
- However, which sorting algorithm can work on **Linked List**?
  - Bubble Sort
  - Selection Sort
  - Insertion Sort
  - Merge Sort
  - Quick Sort

Discussion

Algorithm	Can work on LL?	Time Complexity	Extra Space	Comments
Bubble Sort	Yes			
Insertion Sort	Yes			
Selection Sort	Yes			
Mergesort	Yes			
Quicksort	Yes			

Compare to Array Version

Algorithm	Can work on LL?	Time Complexity	Extra Space	Comments
Bubble Sort	Yes	Same	Same	
Insertion Sort	Yes	Same	Same	
Selection Sort	Yes	Same	Same	
Mergesort	Yes	Same	O(1), (O (log n) for function stack)	
Quicksort	Yes	Same	Same	

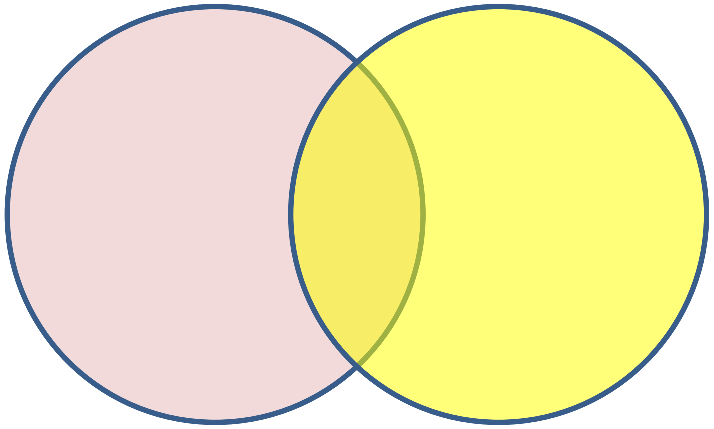
The ADT Sets

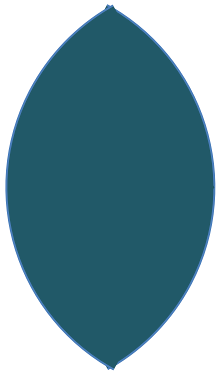
- In Mathematics, a set is a group of distinct items.
- E.g. the set of all positive integers
$$A = \{x \in N | x \geq 1\}$$
- E.g. the set of all even numbers
$$B = \{x \in N | x \bmod 2 = 0\}$$
- And we can perform set operations on them, e.g. The set of all positive even numbers:
$$C = A \cap B$$

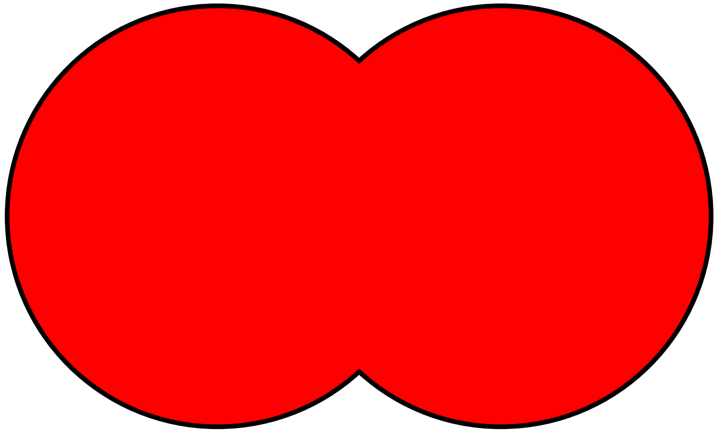
Set Operations

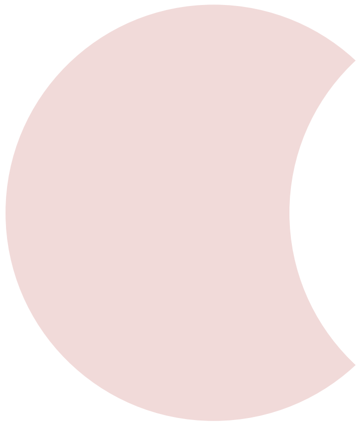
- A set X is a subset of Y, namely $X \subseteq Y$, if
  - $\forall x \in X, x \in Y$
  - E.g. the set of all male is a subset of all humans
- A set X is equal to Y if $X \subseteq Y$ and $Y \subseteq X$

Set Operations


Set A and Set B


Intersection


Union


A - B

ADT Set (of Integers)

```
template <class T>
class Set {
private:
  // what data structure(s)?
public:
  void add(T);                      // add an element
  void remove(T);                   // remove an element
  bool exist(T);                    // check if x exists
  bool isSubsetOf(Set<T>& Y);   // this subsets of Y
  bool isEqualTo(Set<T>& Y);    // this == Y
  Set<T> setUnion(Set<T>& Y);   // return this U Y
  Set<T> intersect(Set<T>& Y); // return this intersect Y
  Set<T> minus(Set<T>& Y);      // return this - Y
};
```

Discussions

- Presentation after 10 min discussion
- What Data Structures should we used to store the elements?
- How to implement the member functions?
  - What is the time/space complexity for each of them?

Example Operations

```
Set<int> A, B;
A.add(1);
B.add(2);
C = A.setUnion(B);
B.add(3);
```