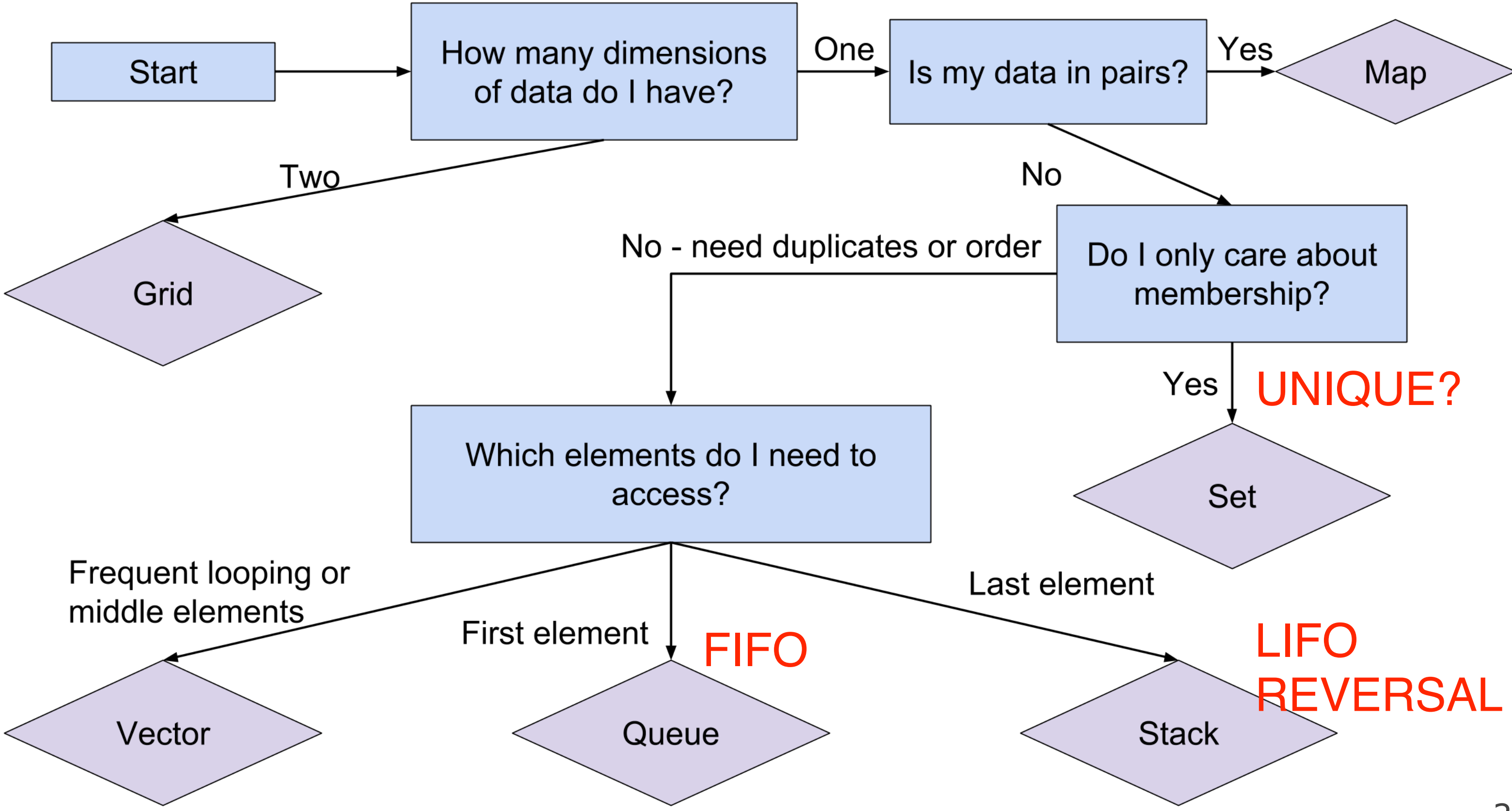
Abstract data types (ADTs)

- **abstract data type (ADT)**: A specification of a collection of data and the operations that can be performed on it.
  - Describes *what* a collection can do, not *how* it does it.
  - We could say that both `Grid` and `SparseGrid` implement the operations of the *abstract data type* called "**grid**".
    - other examples of ADTs: stack, queue, set, map, graph

- We don't always know exactly how a given collection is implemented internally, and we don't need to.
  - We just need to understand the idea of the collection and what operations it can perform.

ADT Soup


Start
How many dimensions of data do I have?
One
Is my data in pairs?
Yes
Map
Two
Grid
No
Do I only care about membership?
No - need duplicates or order
Yes
UNIQUE?
Set
Which elements do I need to access?
Frequent looping or middle elements
First element
Last element
FIFO
LIFO
REVERSAL
Vector
Queue
Stack

Closing Remarks

- Sets/Maps do extend functionality past the vector unlike what we saw with stack/queue. If stack/queue didn't extend functionality, why do we care about them?

- Example counting words in books using a vector and vec.contains(…). Really slow. Now switch vector to set and goes much faster. Why?

- Stack/queue does NOT have for-each loop. That would violate our rule of only being able to see the “next” element.