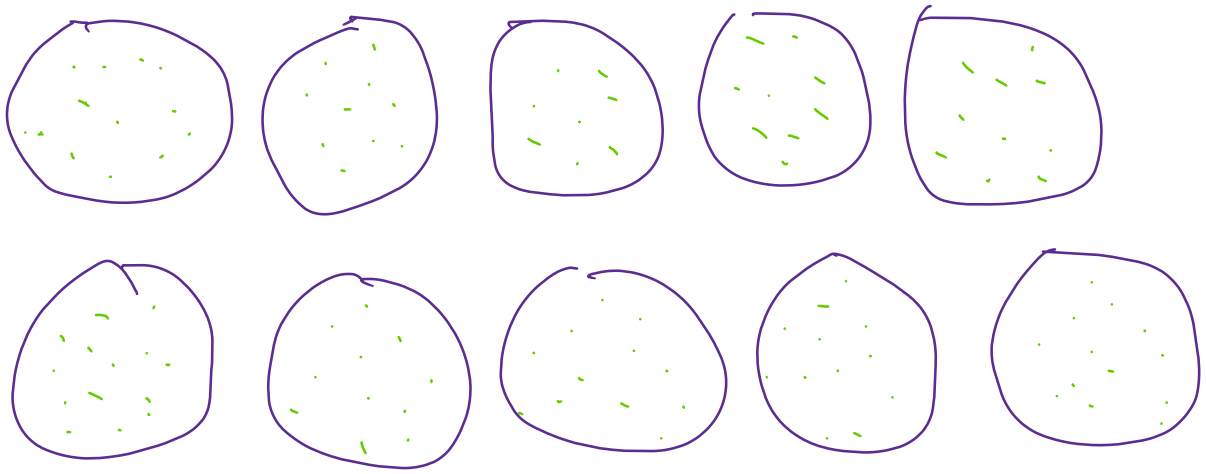
Disjoint-set

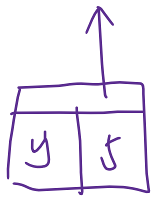
- **Disjoint-set** is a data structure that us used to handle a collection of disjoint sets. For example, in a middle school there are 10 classes in the eighth grade, we want to find whether two students belong to the same class quickly. Disjoint-set will be a great data structure for such situation.

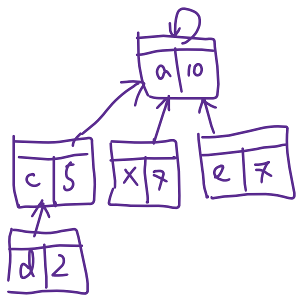


- Disjoint-set is designed as follows:
  In each set, we choose a representative element that represent this set. Other elements in the set can quickly find their representative, and we can check whether two elements belong to the same set by comparing their representative.

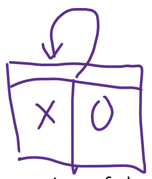
- As a data structure, disjoint-set provide the following methods:
  - Make-set (x): create a disjoint-set that contains only element x, and x is the representative of this set.
  - Union (x, y): merge the set that contains elements x and the set that contains y into a new set. The representative of the new set is one the representatives in two old sets.
  - Find-set (x): return the representative in the set that contains x.

- Since the size of a set can change dramatically, it is straightforward to use a Linked structure to implement a disjoint-set.
  - Other than the element we want to store, each node also contains an integer called its **rank**, and the pointer that points to the "**parent**" of this node. The only vertex whose parent is itself is the representative of that set.

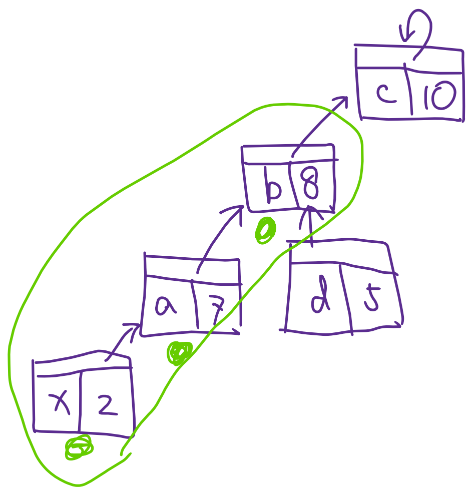


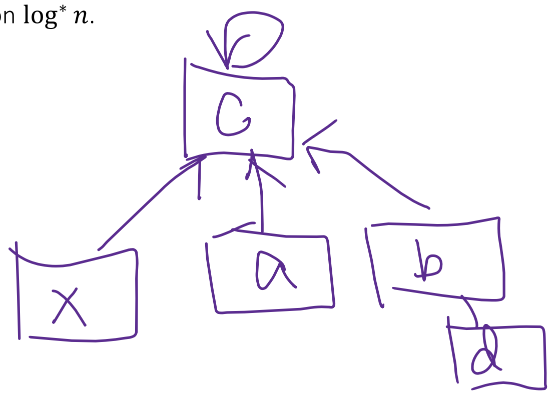


- Make-set (x): create a new set with one node containing elements x, rank $= 0$ and pointing to itself. It is easy to see that the time complexity of this this operation is $O(1)$.

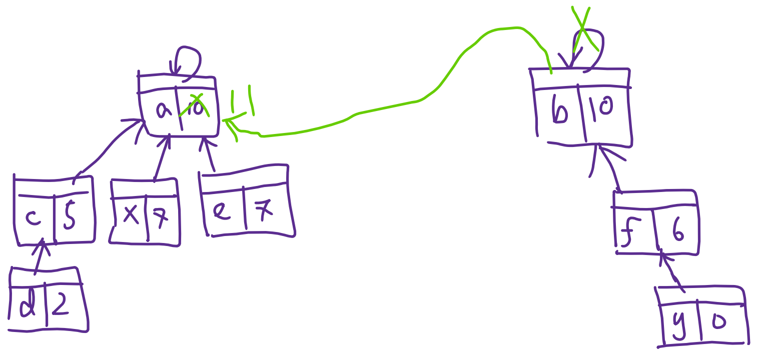


- Find-set (x): starting from x, following the pointers to find the representative of the set. To save running time for future operations, we reconnect all the nodes on the path to the representative. With path compression, the amortized time complexity of this this operation is $O(A^{-1}(n))$, where $A^{-1}(n)$ is the inverse of Ackermann function which grows slower than function $\log^* n$.





- Union (x, y):
  - Call Find-set (x) and Find-set (y) to find the representatives of these two sets (say they are a and b respectively).
  - Compare the ranks of a and b. Link the node with lower rank to the node with higher rank. If a and b have same rank, choose either one as the representative and increase its rank by 1.



a 10
11
c 5
x 7
e 7
d 2
b 10
f 6
y 0