Disjoint Set

In [ ]:

Set = collection of unique elements
Disjoint set= collection of Sets which are mutually exclusive

s = (1,2,3,4,5)

In [ ]:

In [ ]:

ds = ((1,2), (3), (4,5)) # disjoint set
ds = ((1,2), (3,1), (4,5)) # not a disjoint set

s1 = (1,2,3), s2 = (3,4,5)
s1 = (1,2,3), s2 = (4,5)

In [ ]:

Set

- add
- check
- remove/pop/delete

In [ ]:

In [ ]:

In [ ]:

```
Disjoint Set: set of pre defined elements, where each element is in it's own s

ds = [(1), (2), (3), (4), (5)]
- union(e1, e2): merge 2 sets
- find(e1): find the set which e1 belongs to
   check if two elements are part of the same set
- count_ds() = no. of sets in the disjoint set


ds = [(1), (2), (3), (4), (5)]
find(1) == find(2) => False
union(1, 2)

ds = [(1, 2), (3), (4), (5)]
find(1) == find(2) => True

union(3,5)
union(3,5)
ds = [(1, 2), (3,5), (4)]
find(1) == find(3) => False
find(5) == find(3) => True

union(1,5)
ds = [(1, 2, 3, 5), (4)]
find(1) == find(3) => True
find(5) == find(3) => True
```

In [ ]:

In [17]:

```
class DisjointSet:
    def __init__(self, elements):
        self.__data = {}
        for e in elements:
            self.__data[e] = e

    def print(self):
        print()
        print(self.__data)

    def find(self, v):
        p = self.__data[v]
        if p == v:
            return p
        tp = self.find(p)
        self.__data[v] = tp # use path compression
        return tp

    def union(self, v1, v2):
        # make v2 parent of v1
        p1 = self.find(v1)
        self.__data[p1] = v2

d = DisjointSet([1,2,3,4,5])
d.print()
print(d.find(1))
print(d.find(2))

d.union(1, 2) # (1,2), 3, 4, 5
d.print()
print(d.find(1))
print(d.find(2))
print(d.find(3))
print(d.find(4))


d.union(1, 3) # (1,2, 3) , 4, 5
d.print()
print(d.find(1))
d.print()
print(d.find(2))
print(d.find(3))
print(d.find(4))
```

```
{1: 1, 2: 2, 3: 3, 4: 4, 5: 5}
1
2

{1: 2, 2: 2, 3: 3, 4: 4, 5: 5}
2
2
3
4

{1: 2, 2: 3, 3: 3, 4: 4, 5: 5}
3

{1: 3, 2: 3, 3: 3, 4: 4, 5: 5}
3
3
4
```