Homework 2#

Date	@September 7, 2023
Type	Homework

Question 1

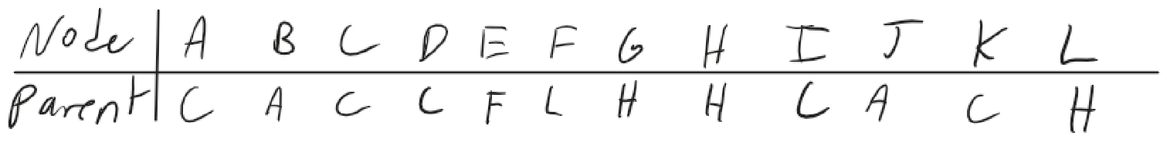
Union: $O(n)$

This happens if all nodes except one are in the same tree (like a linked list), and union is executed on the bottommost item with the other node. It will take $n-1$ operations to get to the highest parent of the leaf

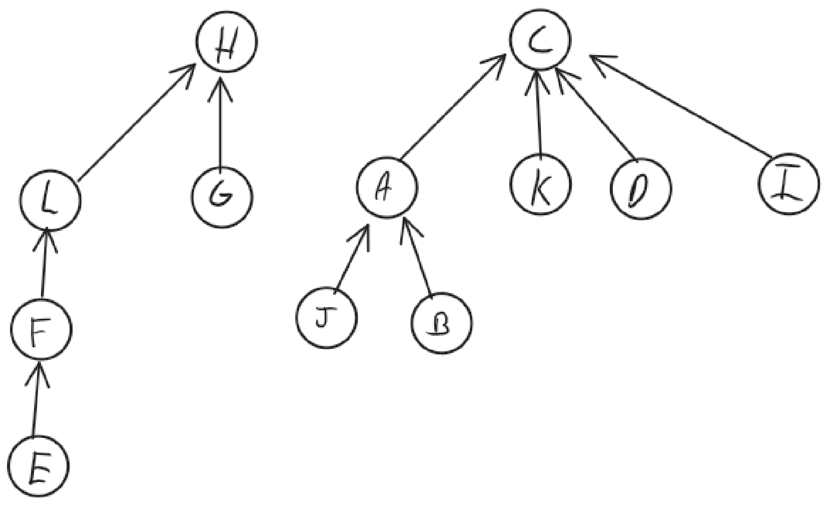
Find: $O(n)$

This happens if all nodes are in a linked list-like tree and find is executed on the bottom one

Question 2

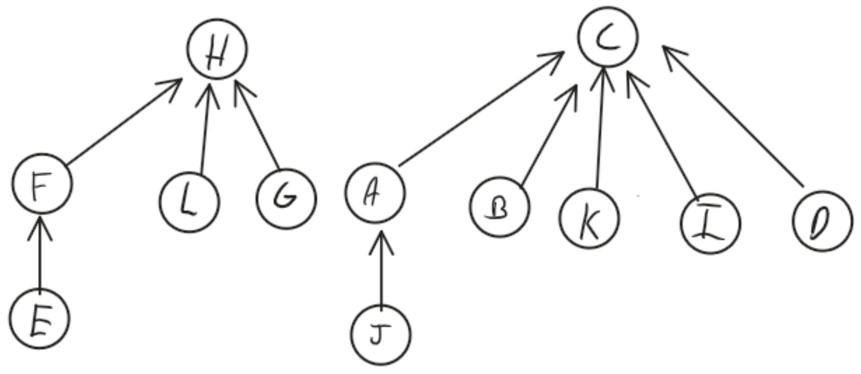


Node	A	B	C	D	E	F	G	H	I	J	K	L
Parent	C	A	C	C	F	L	H	H	C	A	C	H



Question 3

Node	A	B	C	D	E	F	G	H	I	J	K	L
Parent	C	C	K	C	F	H	H	H	C	A	K	H



Question 4

Prim's algorithm makes use of an Indexed Priority Queue. An IPQ supports `add`, `popmin`, and `reduce_key` operations.

A few implementations of this abstract idea, in order of increasing complexity, are Linked Lists, Binary Heaps, Binomial Heaps, and Fibonacci Heaps

Questions 5

The `homework2.py` file is attached. Here's the notable class, for your convenience:

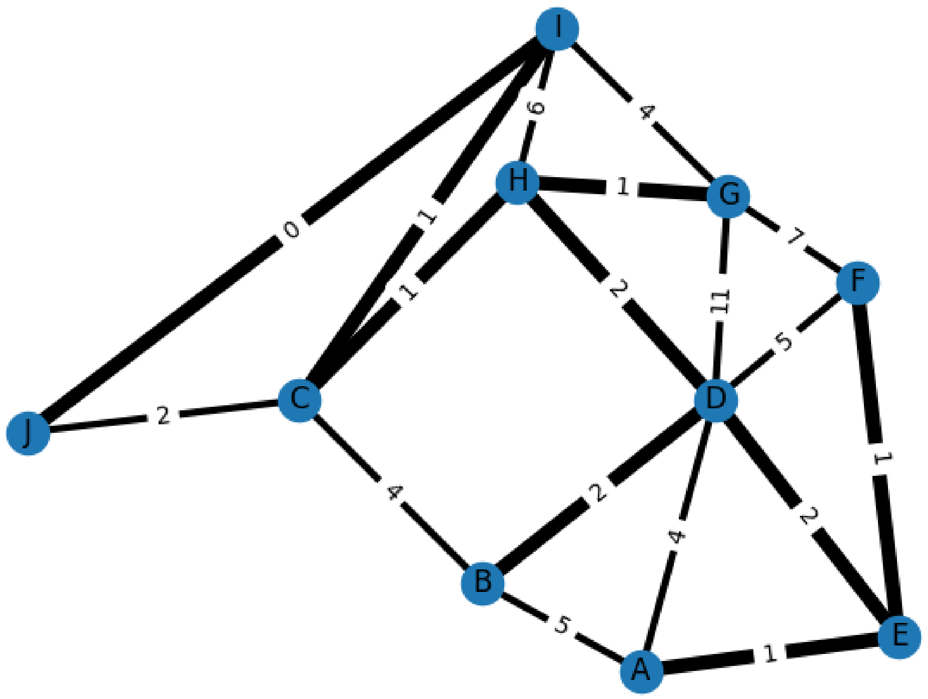
```
class DisjointSet:
    # data structure to back the disjoint set here (you can use an array, a dict, or you could use a graph)

    def __init__(self):        self.parents = {}

    def makeset(self, x):
        self.parents[x] = x

    def find(self, x):
        if self.parents[x] == x:
            return x
        self.parents[x] = self.find(self.parents[x])
        return self.parents[x]

    def union(self, x, y):
        px, py = map(self.find, (x,y))
        self.parents[py] = px if px != py else Non
```

Question 6

The `homework2.py` file is attached. Here's the notable function, for your convenience:

```
def kruskal(G):
    ds = DisjointSet()
    sets_count = len(G.nodes)
    mst = []
    for n in G.nodes:
        ds.makeset(n)
    edges = sorted(G.edges(data=True), key=lambda x: x[2]['weight'])
    for e in edges:
        s,t, _ = e
        ps, pt = map(ds.find, (s,t))
        if ps == pt:
            #components already connected, skip
            continue
        ds.union(s,t)
        mst.append(e)
        sets_count -= 1
        if sets_count == 1:
            return mst
    print("graph is not connected, failing")
    return
```

Here's the MST it produced (also attached separately)



I
H
G
F
J
C
D
B
A
E
0
1
1
6
4
1
7
2
11
5
2
4
2
4
2
1
5
1