355. Design Twitter
Medium

Design a simplified version of Twitter where users can post tweets, follow/unfollow another user, and is able to see the 10 most recent tweets in the user's news feed.

Implement the Twitter class:

- Twitter() Initializes your twitter object.
- void postTweet(int userId, int tweetId) Composes a new tweet with ID tweetId by the user userId. Each call to this function will be made with a unique tweetId.
- List<Integer> getNewsFeed(int userId) Retrieves the 10 most recent tweet IDs in the user's news feed. Each item in the news feed must be posted by users who the user followed or by the user themself. Tweets must be ordered from most recent to least recent.
- void follow(int followerId, int followeeId) The user with ID followerId started following the user with ID followeeId.
- void unfollow(int followerId, int followeeId) The user with ID followerId started unfollowing the user with ID followeeId.

```
class Twitter:
    def __init__(self):
        self.count = 0
        self.tweetMap = defaultdict(list)
        self.followMap = defaultdict(set)

    def postTweet(self, userId: int, tweetId: int) -> None:
        self.tweetMap[userId].append([self.count, tweetId])
        self.count -= 1

    def getNewsFeed(self, userId: int) -> List[int]:
        result = []
        minheap = []

        self.followMap[userId].add(userId)

        for followeeId in self.followMap[userId]:
            if followeeId in self.tweetMap:
                index = len(self.tweetMap[followeeId]) - 1
                count, tweetId = self.tweetMap[followeeId][index]
                minheap.append([count, tweetId, followeeId, index - 1])

        heapq.heapify(minheap)

        while minheap and len(result) < 10:
            count, tweetId, followeeId, index = heapq.heappop(minheap)
            result.append(tweetId)
            if index >= 0:
                count, tweetId = self.tweetMap[followeeId][index]
                heapq.heappush(minheap, [count, tweetId, followeeId, index - 1])
        return result

    def follow(self, followerId: int, followeeId: int) -> None:
        self.followMap[followerId].add(followeeId)

    def unfollow(self, followerId: int, followeeId: int) -> None:
        if followeeId in self.followMap[followerId]:
            self.followMap[followerId].remove(followeeId)
```

```
self.tweetMap = {
    userId : [[count, tweetId]]
}


self.followMap = {
    userId : set([followeeId])
}
```

refers to followeeId lists in self.followMap. Gets all of the followeeIds, and adds [count, tweetId, followeeId, index - 1] to a heap.

- looks for all of the posts by followeeId, by getting index
- gets the last post for followeeId, and destructures to count and tweetId
- appends [count, tweetId, followeeId, index - 1] into the minheap

- heapifies 'minheap'

while minheap contains items, and result is less than 10 items.
- destructure the minheap to [count, tweetId, followeeId, index - 1]
- appends the tweetId to the result
- then adds [count, tweetId, followeeId, index - 1] back to minheap

- return the result list