Hey

[acbdbcdcc] - n

bcd - Any permutation of the pattern exists in the string

Bcdd

Dcbd - m

1. Occurrence of character : there could be duplicate char <Map <Char, Integer> >
2. Length: L(pattern) == L(substring)
3. There should equal number of char occ.

---

1 min [4,4,4,4,4,4,2,2,4,5,5] => [4:7, 2:2, 5:2]
2 min [5,5,7,7,7] => [5:2, 7:3]
 [4:7, 2:2, 5:4, 7:3]

n…. Aggregate map populated. Lifetime value map.
Top 10 products which have been ordered. Thousands of products.

1:00 -> Top 10 products
1:01 -> Getting new batch. Upsert the batch. Refresh the top 10.

HashMap, List, Heap[size=10]

```
Class Product implement Comparable<Product> {
        String productId; //int
        Long freq;  //this guy for compare
        Boolean isInHeap;

        Equals, hashcode
}
```

Algo:

1. Take the list of current minute order : List<Order> order : cluttered this list
2. Parse this one by one,

2.1 see this order id (product id) is present in my Map or not

If yes; increase the count

If no: insert a new product with key as product Id and value Product with freq = 1

2.2 current Min heap size <=10

For new item

If Heap has space : put this new item

New item

If Heap don't have space, compare with root element : take the adjustment [root.freq < current.element freq , update the root, heapify

3. At any time, request for top "x" items, flush the return it to client

Top 10 products over last 1 hour duration.
1:00 - 2:00 => Top 10 products
1:01 - 2:01 =>
1:00 , 1:01, ….1:20 [20 set of data]
1:21

1. Sliding window of 1 hr

```
Class Product implement Comparable<Product> {
        String productId; //int
        Long freq;  //this guy for compare
        Boolean isInHeap;
        TimeStamp time;

        Equals, hashcode
}
```

1 min [4,4,4,4,4,4,2,2,4,5,5] => [4:7, 2:2, 5:2]
2 min [5,5,7,7,7] => [5:2, 7:3]
3 [5,5,6, 7,7,7, 6]

[5:6: , 7: 3: 2:2, , 4:7, 6:1]

Top 10 : min heap
1 hr window:
    Map (just like 1st solution)
    *Queue-> Double Linked List

1. Queue (use front as least time stamp, rear as current timestamp)
2. Each queue would contain a list of elements with there count at that point

4 <-> 5 <->

2 3 4

Queue<Map<PId, Int>>