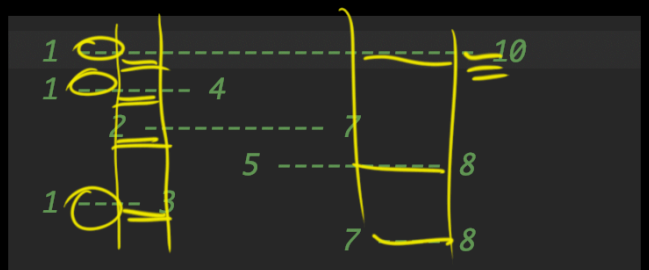
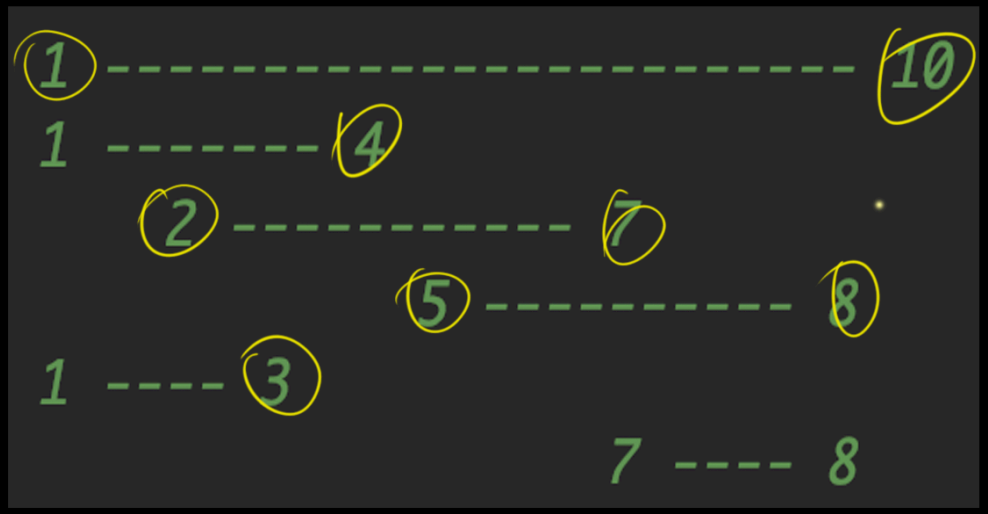


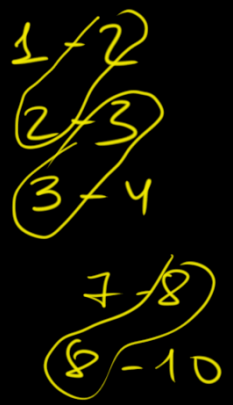
task: spit & count all sub-intervals

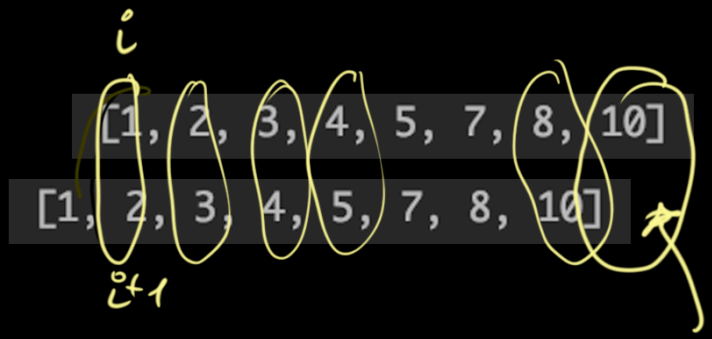
expected result:

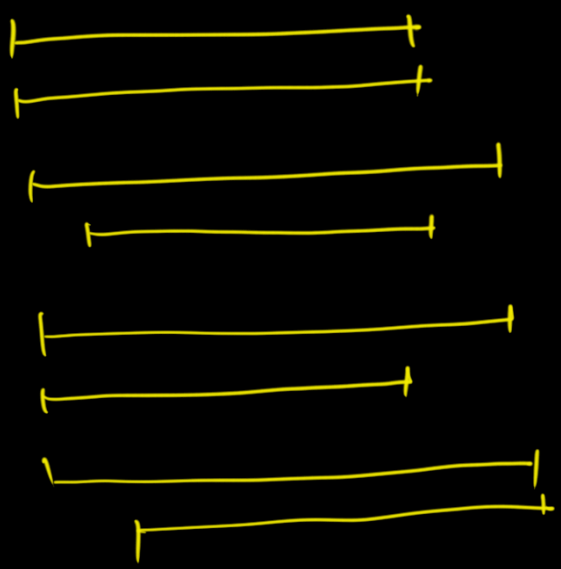
```
1 -- 2  -> 3
2 -- 3  -> 4
3 -- 4  -> 3
4 -- 5  -> 2
5 -- 7  -> 3
7 -- 8  -> 3
8 -- 10 -> 1
```

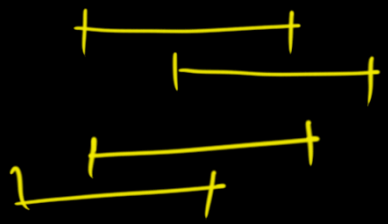


Set

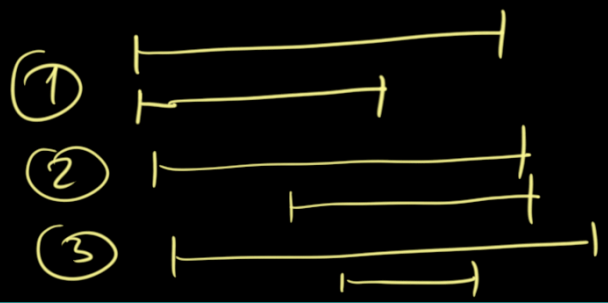








a lot of 'if' statements

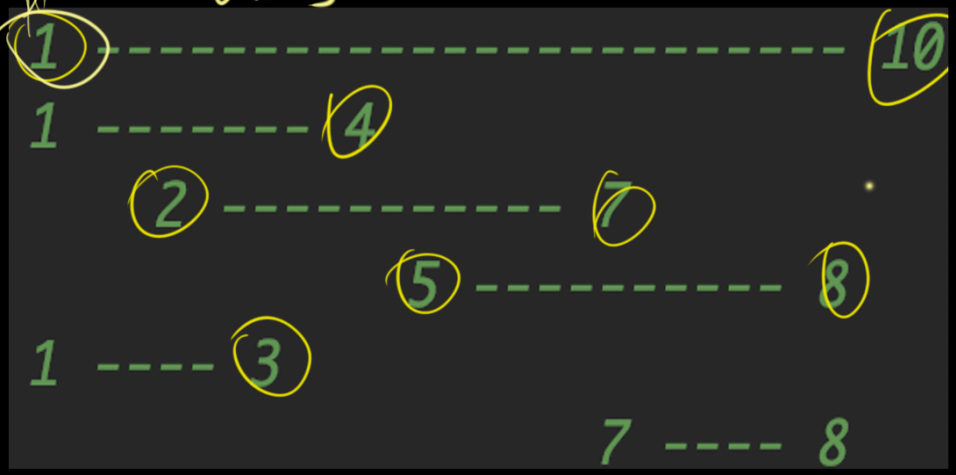


sub

```
1 -- 2   -> 3
2 -- 3   -> 4
3 -- 4   -> 3
4 -- 5   -> 2
5 -- 7   -> 3
7 -- 8   -> 3
8 -- 10  -> 1
```

```
sub: [1 - 2] => 3
sub: [2 - 3] => 4
sub: [3 - 4] => 3
sub: [4 - 5] => 2
sub: [5 - 7] => 3
sub: [7 - 8] => 3
sub: [8 - 10] => 1
```

intervals




```java
subIntervals.forEach(sub -> {
  int[] c = {0};
  data.forEach(interval -> {
    if (interval.contains(sub)) c[0]++;
  });
  System.out.printf("sub: %s => %d\n", sub, c[0]);
});
```

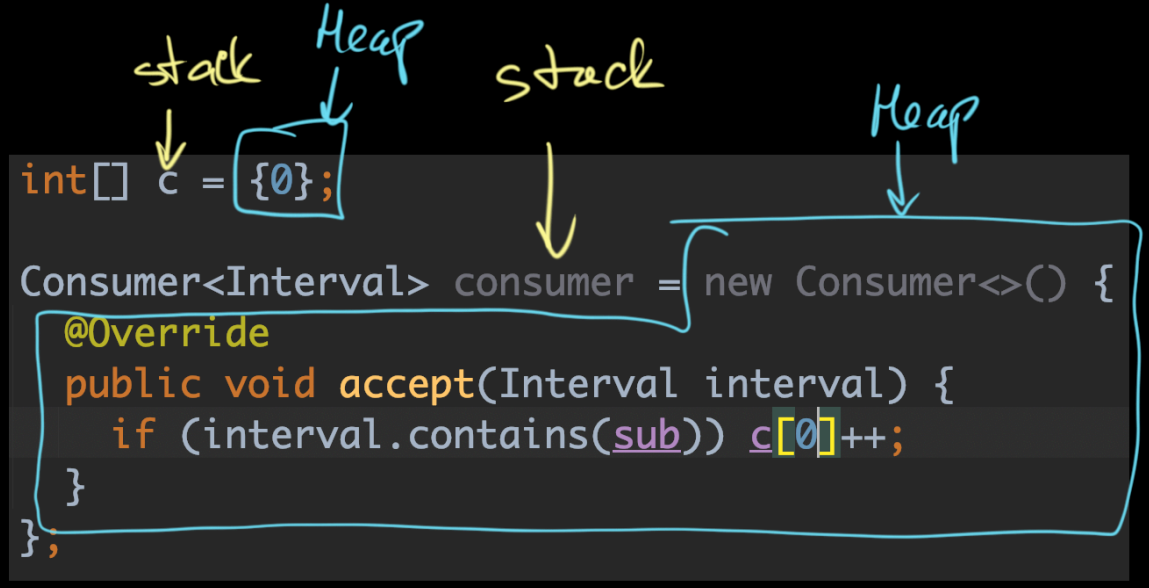
```
subIntervals.forEach(sub -> {
  int c = 0;
  Consumer<Interval> consumer = new Consumer<>() {
    @Override
    public void accept(Interval interval) {
      if (interval.contains(sub)) c++;
    }
  };
```

stack

stack

Heap

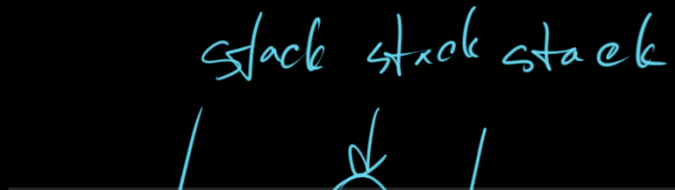
Heap & Class Loader dont have access to stack



Heap can access to Heap

Stack → Heap

Heap ✗ Stack




```java
for (Interval si: subIntervals) {
  int c = 0;
  for (Interval in: subIntervals) {
    if (in.contains(si)) c++;
  }
}
```

```java
subIntervals.forEach(sub -> {
  int[] c = {0};
  data.forEach(interval -> {
    if (interval.contains(sub)) c[0]++;
  });
  System.out.printf("sub: %s => %d\n", sub, c);
});
```

```java
HashMap<Interval, Integer> res = new HashMap<>();
subIntervals.forEach(sub -> {
  int[] c = {0};
  data.forEach(interval -> {
    if (interval.contains(sub)) c[0]++;
  });
  res.put(sub, c[0]);
});
return res;
```

```
Iterator<Integer> it = as.iterator();
while (it.hasNext()) {
  int x = it.next();
  System.out.println(x);
}
```

```
for (Integer a: as) {
  System.out.println(a);
}
```

```
for (int row = 0; row < height; row++)
```

too verbose

a lot of places to make a mistake

```
data.forEach(x -> {
  if (x < 0) result.add(x);
});
```

do smth w/ this element

declarative way

```java
HashSet<Integer> unique = new HashSet<>(); // TreeSet
data.forEach(x -> { unique.add(x.low); unique.add(x.hi); });
ArrayList<Integer> points = new ArrayList<>(unique);
points.sort(Comparator.comparingInt(x -> x));
ArrayList<Interval> subIntervals = new ArrayList<>();
for (int i = 0; i < points.size()-1; i++) {
  subIntervals.add(Interval.of(points.get(i), points.get(i+1)));
}
HashMap<Interval, Integer> res = new HashMap<>();
subIntervals.forEach(sub -> {
  int[] c = {0};
  data.forEach(interval -> {
    if (interval.contains(sub)) c[0]++;
  });
  res.put(sub, c[0]);
});
return res;
```

1

2

3

4

5

Java8

Java8

Java5