[] j=0, k=0 int] S):

Talk
Example
Brute force
Optimize
Walk Through
Implement
Test

TALK

- input: S array of ints
- a, b, c in S such that a+b+c=0?
- return ALL unique triplets in S that give sum 0 (array of arrays)
- duplicates in original array
- return empty array or null if no soln?

EXAMPLE

SAMPLE INPUT	CLASS	OUTPUT
[-1,0,1,2,-1,-4]	array w/ a soln	[[-1,0,1], [-1,-1,2]]
[0,1,2,3]	array w/ no soln	null
[1,2,3]	array w/ only positive numbers	null
[]	empty array	null
[-1,-2,-3]	array w/ only negative #s	null
null	arr==null	null

BRUTE FORCE

```
threeSum (Array [int] S):                         [-1, 0, 1, 2, -1, -4]
                                                            ↓
1    if S == null || len(S) < 3:                  [-4, -1, -1, 0, 1, 2]
2        return null
3    else:
4        sorted = S.sort(), endArr = []
5        for i in range(len(sorted)):
6            curr = sorted[i]
7            j = i+1
8            while j < len(S)
9                next = sorted[j]
10               wanted = -curr - next
11               if wanted in remaining S and [curr, next, wanted] not in endArr:
12                   endArr.append([curr, next, wanted])
13               else:
14                   j++
15       return endArr
```

OPTIMIZE

```
- change line 8 to: while j < len(s)-1
- add to line 4: if sorted[0] >= 0:
                     return null
change line 5 to: while i < len(s)-2 and add i++ after line 14
```

WALK THROUGH

Input Validation : - check if array S is null or empty or has less than 3 elements

↓

Sort array

↓

iterate over array : in each iteration, iterate through rest of ~~[illegible]~~ array then find the needed c to make sum 0

↓

no solution : if end array is empty, return null
else return endArray

Implementation:

```
Arraylist< ArrayList<int>> public static threeSum (Arraylist<int> S){
  if (S == null || S.size() < 3){ return null;}

  ~~[illegible]~~
  ArrayList<int> sorted = Collections.sort(S);
  ArrayList<ArrayList<int>> endArr = new ArrayList ();
  int i = 0;
  while (i < sorted.size() - 2){
    int curr = sorted.get(i);
    int j = i+1;
    while (j < sorted.size()-1){
      int next = sorted.get(j);
      int wanted = -curr - next;
      if ( sorted.contains (wanted) && !endArr.contains({curr,next,wanted})){
        ~~[illegible]~~
        endArr.add ({curr,next,wanted});
      } else { j++; }
      i++;
      ~~[illegible]~~
  if (sorted.get(0) >= 0){ return null;}
  return endArr;
```

```
if s == null || len(s) < 3:
```

TEST

INPUT	OUTPUT
null	null
[0]	null
[1, 2, 3]	null
[1, 0, -1]	[[-1, 0, 1]]
[-1, 1, 2]	null