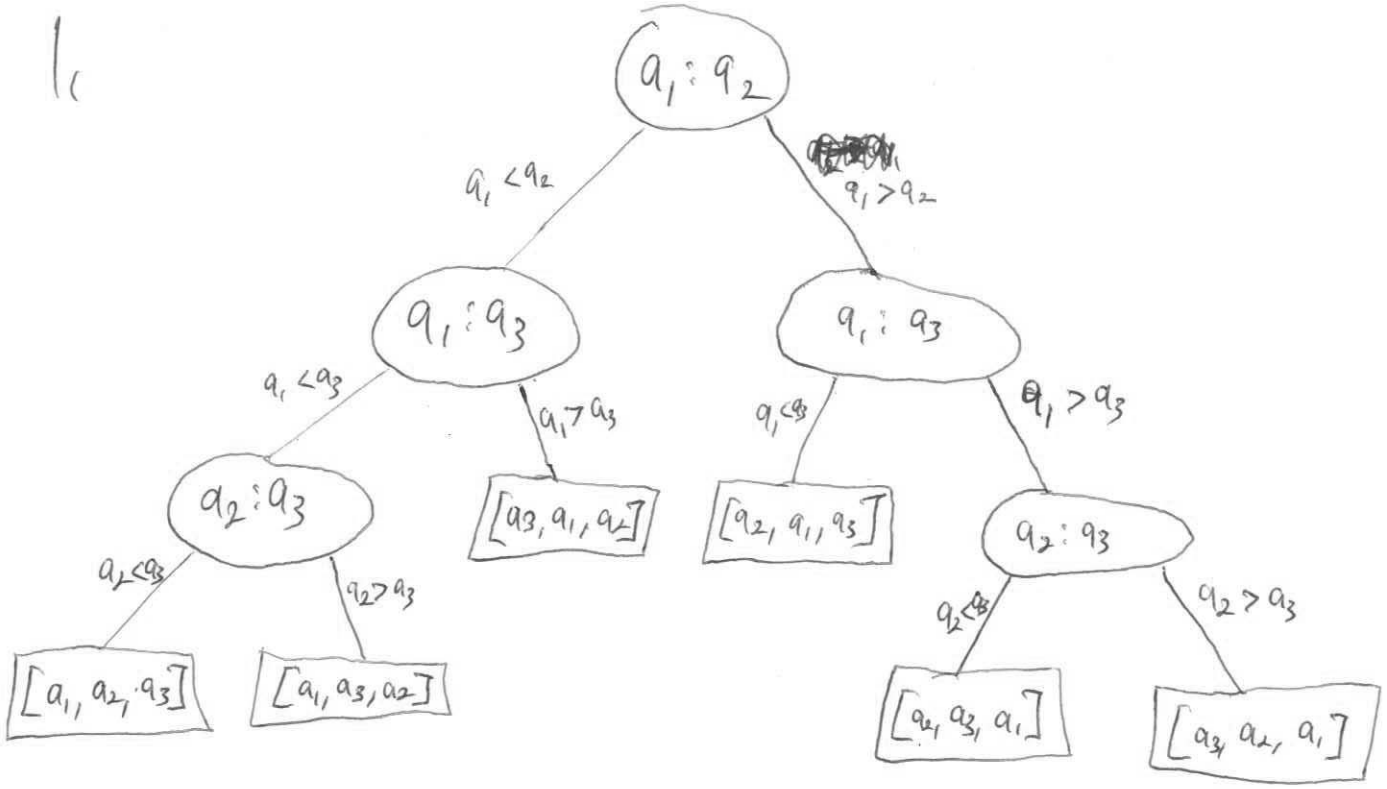
1c


$a_1 : a_2$
$a_1 < a_2$
$a_1 > a_2$
$a_1 : a_3$
$a_1 < a_3$
$a_1 > a_3$
$a_2 : a_3$
$a_2 < a_3$
$a_2 > a_3$
$[a_1, a_2, a_3]$
$[a_1, a_3, a_2]$
$[a_3, a_1, a_2]$
$a_1 : a_3$
$a_1 < a_3$
$a_1 > a_3$
$[a_2, a_1, a_3]$
$a_2 : a_3$
$a_2 < a_3$
$a_2 > a_3$
$[a_2, a_3, a_1]$
$[a_3, a_2, a_1]$

3. 646		196		619		920
920		167		920		541
619		541		541		661
853		582	→	646	→	582
864		646		853		853
541	→	619		167		864
196		678		661		646
582		661		864		196
167		853		678		167
678		864		582		678
661		920		196		619
↑		↑		↑		

Clearly these are not sorted. To fix, partition the numbers into "bins" by their digits, and only sort less significant digits from numbers in the same bin.

646		196 167		167
920				196
619		541 582		541
853				582
864	→	646 619 678 661	→	619
541				646
196				661
582				678
167		853 864		853
678				864
661		920		920
↑				

Each bin has 1 element after sorting 2 digits so we are done.

3. The best comparison-based algorithm will have O(nlogn) complexity.

Since the numbers are in the range of [0, 2n], we can use counting sort, and the time complexity will be O(n + 2n) = O(n)