```
public class Solution {
    /**
     * @param source:
     * @param target:
     * @return: return the index
     */
    public int strStr(String source, String target) {
        // Write your code here
        final int BASE = 1000000;

        if(source == null || target == null){
            return -1;
        }
        int n = target.length();
        if(n == 0) {
            return 0;
        }

        int pow = 1;
        for(int i = 0; i < n; i++){
            pow = (pow * 31) % BASE;
        }
        int targetCode = 0;
        for(int i = 0; i < target.length(); i++){
            targetCode = (targetCode * 31 + target.charAt(i)) % BASE;
        }
        int hashCode = 0;
        for(int i = 0; i < source.length(); i++){
            hashCode = (hashCode * 31 + source.charAt(i)) % BASE;
            if(i < n - 1) {
                continue;
            }
            if(i >= n){
                hashCode = hashCode - (source.charAt(i - n) * pow) % BASE;
                if(hashCode < 0) {
                    hashCode += BASE;
                }
            }

            if(hashCode == targetCode) {
                if(source.substring(i - n + 1, i + 1).equals(target)) {
                    return i - n + 1;
                }
            }
        }
        return -1;
    }
}
```

```
public class Solution {
    /**
     * @param A: an integer array
     * @return: nothing
     */
    public void sortIntegers2(int[] A) {
        // write your code here
        int start = 0;
        int end = A.length - 1;
        quickSort(A, start, end);
    }

    public void quickSort(int[] A, int start, int end) {
        if(start >= end) return;

        int pivot = A[start + (end - start) / 2];
        int left = start;
        int right = end;

        while(left <= right) {
            while(left <= right && A[left] < pivot) {
                left++;
            }
            while(left <= right && A[right] > pivot) {
                right--;
            }
            if(left <= right){
                int temp = A[left];
                A[left] = A[right];
                A[right] = temp;
                left++;
                right--;
            }
        }

        quickSort(A, start, right);
        quickSort(A, left, end);
    }
}
```