```
                        i = parent;
                } else {
                        break;
                }
        }
}

public void heapify(char[] chas, int i, int size) {
        int left = i * 2 + 1;
        int right = i * 2 + 2;
        int largest = i;
        while (left < size) {
                if (chas[left] > chas[i]) {
                        largest = left;
                }
                if (right < size && chas[right] > chas[largest]) {
                        largest = right;
                }
                if (largest != i) {
                        swap(chas, largest, i);
                } else {
                        break;
                }
                i = largest;
                left = i * 2 + 1;
                right = i * 2 + 2;
        }
}

public void swap(char[] chas, int index1, int index2) {
        char tmp = chas[index1];
        chas[index1] = chas[index2];
        chas[index2] = tmp;
}
```

在有序但含有空的数组中查找字符串

【题目】

给定一个字符串数组 strs[]，在 strs 中有些位置为 null，但在不为 null 的位置上，其字符串是按照字典顺序由小到大依次出现的。再给定一个字符串 str，请返回 str 在 strs 中出现的最左的位置。

【举例】

strs=[null,"a",null,"a",null,"b",null,"c"]，str="a"，返回 1。

strs=[null,"a",null,"a",null,"b",null,"c"]，str=null，只要 str 为 null，就返回-1。

strs=[null,"a",null,"a",null,"b",null,"c"]，str="d"，返回-1。

【难度】

尉 ★★☆☆

【解答】

本题的解法尽可能多地使用了二分查找，具体过程如下：

1．假设在 strs[left..right]上进行查找的过程，全局整型变量 res 表示字符串 str 在 strs 中最左的位置。初始时，left=0，right=strs.length-1，res=-1。

2．令 mid=(left+right)/2，则 strs[mid]为 strs[left..right]中间位置的字符串。

3．如果字符串 strs[mid]与 str 一样，说明找到了 str，令 res=mid。但要找的是最左的位置，所以还要在左半区寻找，看有没有更左的 str 出现，所以令 right=mid-1，然后重复步骤 2。

4．如果字符串 strs[mid]与 str 不一样，并且 strs[mid]!=null，此时可以比较 strs[mid]和 str，如果 strs[mid]的字典顺序比 str 小，说明整个左半区不会出现 str，需要在右半区寻找，所以令 left=mid+1，然后重复步骤 2。

5．如果字符串 strs[mid]与 str 不一样，并且 strs[mid]==null，此时从 mid 开始，从右到左遍历左半区（即 strs[left..mid]）。如果整个左半区都为 null，那么继续用二分的方式在右半区上查找（即令 left=mid+1），然后重复步骤 2。如果整个左半区不都为 null，假设从右到左遍历 strs[left..mid]时，发现第一个不为 null 的位置是 i，那么把 str 和 strs[i]进行比较。如果 strs[i]字典顺序小于 str，同样说明整个左半区没有 str，令 left=mid+1，然后重复步骤 2。如果 strs[i]字典顺序等于 str，说明找到 str，令 res=mid，但要找的是最左的位置，所以还要在 strs[left..i-1]上寻找，看有没有更左的 str 出现，所以令 right=i-1，然后重复步骤 2。如果 strs[i]字典顺序大于 str，说明 strs[i..right]上都没有 str，需要在 strs[left..i-1]上，所以令 right=i-1，然后重复步骤 2。

具体过程请参看如下代码中的 getIndex 方法。

```
public int getIndex(String[] strs, String str) {
        if (strs == null || strs.length == 0 || str == null) {
                return -1;
        }
        int res = -1;
        int left = 0;
        int right = strs.length - 1;
```

```
        int mid = 0;
        int i = 0;
        while (left <= right) {
            mid = (left + right) / 2;
            if (strs[mid] != null && strs[mid].equals(str)) {
                res = mid;
                right = mid - 1;
            } else if (strs[mid] != null) {
                if (strs[mid].compareTo(str) < 0) {
                    left = mid + 1;
                } else {
                    right = mid - 1;
                }
            } else {
                i = mid;
                while (strs[i] == null && --i >= left)
                    ;
                if (i < left || strs[i].compareTo(str) < 0) {
                    left = mid + 1;
                } else {
                    res = strs[i].equals(str) ? i : res;
                    right = i - 1;
                }
            }
        }
        return res;
    }
```

字符串的调整与替换

【题目】

给定一个字符类型的数组 chas[]，chas 右半区全是空字符，左半区不含有空字符。现在想将左半区中所有的空格字符替换成"%20"，假设 chas 右半区足够大，可以满足替换所需要的空间，请完成替换函数。

【举例】

如果把 chas 的左半区看作字符串，为"a b c"，假设 chas 的右半区足够大。替换后，chas 的左半区为"a%20b%20%20c"。

【要求】

替换函数的时间复杂度为 $O(N)$，额外空间复杂度为 $O(1)$。