14 最长公共前缀

Label：分制 二分查找

编写一个函数来查找字符串数组中的最长公共前缀。
如果不存在公共前缀，返回空字符串 ""。

```
输入：strs = ["flower","flow","flight"]
输出："fl"
输入：strs = ["dog","racecar","car"]
输出：""
解释：输入不存在公共前缀。
```

- 纵向扫描

```
class Solution {
    public String longestCommonPrefix(String[] strs) {
        if (strs.length <= 0) return "";
        else if (strs.length == 1) return strs[0];
        // 有空串直接返回结果
        for (String str: strs) {
            if (str == null || str.length()== 0)
                return "";
        }

        int index = 0;
        char curr = strs[0].charAt(0);

        while (true) {
            // 循环匹配
            for (String str: strs) {
                if (str.length() < index) {
                    return str.substring(0, index);
                }
                if (str.charAt(index) != curr) {
                    return str.substring(0, index);  // 遇到不匹配的直接返回就行
                }

            }
            index++;
            // 判断所有长度
            for (String str: strs) {
                if (index >= str.length()) {
                    return str.substring(0, index); // 只要有一个超长就直接返回
                }
            }
             // 赋值新一轮的 字符
            if (strs[0].length() > index)
                curr = strs[0].charAt(index);
        }
    }
}
```

- 横向扫描

```
class Solution {
    public String longestCommonPrefix(String[] strs) return "";
        String prefix = strs[0];

        for (int i = 1; i < strs.length; i++) {
            prefix = longestCommonPrefix(prefix, strs[i]);  // 重复使用 prefix
            if (prefix.length() == 0) {
                break;
            }
        }
        return prefix;
    }

    public String longestCommonPrefix(String str1, String str2) {
        int length = Math.min(str1.length(), str2.length());
        int index = 0;
        while (index < length && str1.charAt(index) == str2.charAt(index)) {
            index++;
        }
        return str1.substring(0, index);
    }
}
```

- 分制

```
class Solution {
    public String longestCommonPrefix(String[] strs) {
        if (strs == null || strs.length == 0) return "";
        else return longestCommonPrefix(strs, 0, strs.length - 1);
    }

    public String longestCommonPrefix(String[] strs, int start, int end) {
        if (start == end) {
            return strs[start];
        } else {
            int mid = (end - start) / 2 + start;
            String lcpLeft = longestCommonPrefix(strs, start, mid);
            String lcpRight = longestCommonPrefix(strs, mid + 1, end);
            return commonPrefix(lcpLeft, lcpRight);
        }
    }

    public String commonPrefix(String lcpLeft, String lcpRight) {
        int minLength = Math.min(lcpLeft.length(), lcpRight.length());
        for (int i = 0; i < minLength; i++) {
            if (lcpLeft.charAt(i) != lcpRight.charAt(i)) {
                return lcpLeft.substring(0, i);
            }
        }
        return lcpLeft.substring(0, minLength);
    }
}
```

- 二分查找

```
class Solution {
    public String longestCommonPrefix(String[] strs) {
        if (strs == null || strs.length == 0) {
            return "";
        }
        int minLength = Integer.MAX_VALUE;
        for (String str : strs) {
            minLength = Math.min(minLength, str.length());
        }

        int low = 0, high = minLength;  // 只用二分查找最短串就行

        while (low < high) {
            int mid = (high - low + 1) / 2 + low;
            if (isCommonPrefix(strs, mid)) {  // 如果是公共子串，那么 low 就设置为上一
轮的 mid
                low = mid;
            } else { // 如果没有子串的,则将 high 设置为 mid - 1
                high = mid - 1;
            }
        }
        return strs[0].substring(0, low);
    }

    public boolean isCommonPrefix(String[] strs, int length) {
        String str0 = strs[0].substring(0, length);
        for (int i = 1; i < strs.length; i++) {
            String str = strs[i];
            for (int j = 0; j < length; j++) {
                if (str0.charAt(j) != str.charAt(j)) {
                    return false;
                }
            }
        }
        return true;
    }
}
```