题目描述

排序规则：

1. 单词中字母比较不区分大小写，两个单词先以第一个字母作为排序的基准，如果第一个字母相同，就用第二个字母为基准，如果第二个字母相同就以第三个字母为基准。依此类推，如果到某个字母不相同，字母顺序在前的那个单词顺序在前。
2. 当一个短单词和一个长单词的开头部分都相同（即短单词是长单词从首字母开始的一部分），短单词顺序在前。
3. 字母大小写不同的相同单词，只输出一次。

输入描述

无

输出描述

无

用例

输入	Hello hello world
输出	Hello world
说明	无

输入	i LOVE Cc I love CC Hello Hel Hellow
输出	Cc Hel Hello Hellow i LOVE
说明	无

题目解析

这题有两部分逻辑，一个是排序，一个是去重。

我这里是先排序后去重，因为我想基于栈结构去重，即排序后，不区分大小写的两个字符串一定是紧挨着的，因此后入栈的字符串如果和栈顶字符串相同（不区分大小写），则不入栈。

排序规则是其实就是 字典序。但是需要注意的是，不区分大小写排序，因此我们不能直接使用Array.prototype.sort的原生字典序排序，而是需要自定义不区分大小写的字典序排序。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const arr = line.split(" ");

  console.log(sortStr(arr));
});

function sortStr(arr) {
  // 排序
  arr.sort((a, b) => {
    a = a.toLowerCase();
    b = b.toLowerCase();

    return a === b ? 0 : a > b ? 1 : -1;
  });

  const stack = [arr.shift()];

  // 去重
  for (let str of arr) {
    const top = stack.at(-1).toLowerCase();
    const add = str.toLowerCase();
    if (top === add) continue;
    stack.push(str);
  }

  return stack.join(" ");
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
  // 输入获取
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] arr = sc.nextLine().split(" ");

    System.out.println(getResult(arr));
  }

  // 算法入口
  public static String getResult(String[] arr) {
    Arrays.sort(arr, (a, b) -> a.toLowerCase().compareTo(b.toLowerCase()));

    LinkedList<String> stack = new LinkedList<>();
    stack.add(arr[0]);

    for (int i = 1; i < arr.length; i++) {
      String s = arr[i];
      String top = stack.getLast().toLowerCase();
      String add = s.toLowerCase();
      if (top.equals(add)) continue;
      stack.add(s);
    }

    StringJoiner sj = new StringJoiner(" ");
    for (String s : stack) sj.add(s);
    return sj.toString();
  }
}
```

Python算法源码

```python
# 输入获取
arr = input().split()


# 算法入口
def getResult():
    arr.sort(key=lambda x: x.lower())

    stack = [arr[0]]

    for i in range(1, len(arr)):
        s = arr[i]
        top = stack[-1].lower()
        add = s.lower()
        if top == add:
            continue
        stack.append(s)

    return " ".join(stack)


# 算法调用
print(getResult())
```