

A)

1. 定义一个方法 `getLowerCaseWords()` 从文本中解析出单词，并转为小写。

```
public static List<String> getLowerCaseWords(File file) {
    Scanner scanner = null;
    Pattern pattern = Pattern.compile("[a-zA-Z]+");
    String text = "";
    List<String> words = new ArrayList<>();
    try {
        scanner = new Scanner(file);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    if(scanner!=null){
        while(scanner.hasNextLine()){
            text+= " "+scanner.nextLine();
        }
        scanner.close();
    }

    Matcher matcher = pattern.matcher(text);
    while (matcher.find()){
        words.add(matcher.group().toLowerCase());
    }
    return words;
}
```

2. 定义一个方法，从一个 list 里面找到包含另一个 list 的元素

```
public static List<String> getMatchedWords(List<String>
source ,List<String> target){
    return
source.stream().filter(x->target.contains(x)).distinct().collect(Collectors.toList());
}
```

3. 写一个测试方法

```
public static void main(String[] args) {
    // read an_article.txt
    File articleFile = new
File("E:\\workspace\\thread\\src\\main\\resources\\an_article.txt");
    // read google-10000-english-no-swears.txt
    File googleFile = new
File("E:\\workspace\\thread\\src\\main\\resources\\google-10000-
```

```

english-no-swears.txt");
// get words from an_article.txt
List<String> articleWords=getLowerCaseWords(articleFile);
// get words from google-10000-english-no-swears.txt
List<String> googleWords=getLowerCaseWords(googleFile);
// match with google-10000-english-no-swears, get valid words
List<String> result = getMatchedWords(articleWords, googleWords);
// print result
System.out.println(result.size());
System.out.println(result.toString());
}

```

运行结果如图所示：

```

PartOneA
F:\java\jdk1.8.0_101\bin\java.exe ...
1094
[selecting, critical, patterns, based, on, local, and, statistical, information, li, abstract, pattern, selection, methods, have, been, developed, with,
a, specific, in, contrast, this, paper, presents, method, that, deemed, to, carry, essential, applicable, train, those, types, of, which, require,
spatial, the, training, include, edge, define, boundary, border, separate, classes, proposed, from, new, perspective, primarily, their, location,
input, space, it, determines, class, assistance, surface, also, identifies, between, using, probability, is, evaluated, benchmark, problems, popular,
including, basis, functions, support, vector, machines, nearest, neighbors, approach, compared, four, state, art, approaches, shown, provide, similar,
but, more, consistent, accuracy, reduced, data, set, experimental, results, demonstrate, sufficient, represent, preserve, decision, index, terms,
reduction, x, ieee, authors, are, school, computing, intelligent, systems, university, bt, united, kingdom, e, mail, y, ac, uk, l, received, insert,
date, submission, if, desired, introduction, g, do, n, aim, select, ds, ns, can, sufficiently, original, some, specified, criteria, usually, designed,
expected, maintain, performance, entire, benefit, thus, obvious, when, an, large, provided, for, design, instance, such, as, all, instances, lot,
memory, computation, resource, therefore, smaller, requires, less, faster, application, other, neural, networks, leads, moreover, often, needs, be,
conducted, many, times, order, find, optimal, parameters, control, s, learning, translate, into, even, speed, up, process, further, intensive,
necessary, standard, computer, one, additional, may, achieve, better, classification, due, removal, noise, there, vast, number, different, applications,
example, great, efforts, put, see, these, range, however, work, prefer, describe, two, perspectives, category, consider, certain, properties, then,
selected, geometry, hit, miss, calls, used, counts, most, analyze, group, around, comprehensive, review, papers, available, literature, background,
area, well, known, levels, generally, described, according, convenience, fig, uses, dimensional, relationship, problem, three, interior, mixing, each,
because, common, features, locations, etc, sit, region, i, near, free, how, far, extends, within, formed, by, note, no, only, needed, relationships,
introduce, considerable, difficulty, primary, source, error, reason, algorithms, minimize, kind, sense, learned, passes, through, focus, refer,
definition, trained, outside, will, drive, form, alone, not, good, particular, novelty, detection, where, extent, must, eliminate, mix, novel, seen,
during, included, rarely, studied, considering, both, retained, occupied, falls, observation, detected, plane, or, at, remainder, organized, follows,
next, section, first, provides, discussion, feature, reports, neighbor, discusses, potential, construction, considered, limited, continuous,
continuously, sub, regions, extreme, so, enclosed, tight, external, point, drawn, cross, its, shape, side, small, ratio, b, c, majority, depend,
illustration, shapes, solid, line, indicates, straight, identification, previous, illustrated, sitting, passing, task, becomes, indeed, defined,
normal, relative, respect, determined, fact, nor, alternative, solve, given, vs, density, derived, assume, distribution, subject, function, p, we, now,
estimate, technique, points, suppose, d, sphere, radius, r, centered, constructed, has, volume, contains, k, estimated, coverage, gives, conditional,
expansion, taylor, coriolis, coriolis, formula, wave, direction, required, plus, rotation, concerned, actual, value, relevant, above, equation, above.]

```

B).

- 1.定义一个 count，用来统计移动次数
- 2.定义一个 merge 方法，将最小分组的组内进行比较，并移动。

```

public static void merge(String[] strs, int low, int mid, int high) {
    count++;
    String[] temp = new String[high - low + 1];
    int i = low;// left point
    int j = mid + 1;// right point
    int k = 0;

    // Move the smaller element into the new array first
    while (i <= mid && j <= high) {
        if (strs[i].compareTo(strs[j]) <= 0) {
            temp[k++] = strs[i++];
        } else {
            temp[k++] = strs[j++];
        }
    }
}

```

```

    }

    // Move the rest of the elements on the left into the array
    while (i <= mid) {
        temp[k++] = strs[i++];
    }

    // Move the rest of the elements on the right into the array
    while (j <= high) {
        temp[k++] = strs[j++];
    }

    // Overwrite the elements of the new array over the strs array
    for (int k2 = 0; k2 < temp.length; k2++) {
        strs[k2 + low] = temp[k2];
    }
}

```

3. 定义一个 mergeSort 方法，使用递归，将待排序的数组分隔成最小的组，再调用 merge 方法，返回排好序的数组

```

public static String[] mergeSort(String[] strs, int low, int high) {

    int mid = (low + high) / 2;
    if (low < high) {
        // left
        mergeSort(strs, low, mid);
        // right
        mergeSort(strs, mid + 1, high);
        // merge left and right
        merge(strs, low, mid, high);
    }

    return strs;
}

```

最后写驱动方法运行截图如下