

**面向对象程序设计上机实验报告**

实验题目2：Set/Map/List及文件处理

**学院名称 智能与计算学部**

**专 业 软件工程**

**学生姓名 王杨森**

**学 号 3020244116**

**年 级 2020级**

**班 级 软件工程3班**

**时 间 2022年 4月7日**

1. **实验目的**

1) 熟悉Eclipse/MyEclipse；

2) 熟悉Java中Set/Map/List File的使用；

1. **实验内容**
2. detail.txt文件是一个超市的历次购物小票的合计，每行表示一次购物，每个商品之间用半角逗号分开。请开发程序完成如下功能:
3. 找出哪两个商品总被同时购买的频率最高(不考虑商品的先后顺序)；
4. 找出哪N个商品(N=2)总被同时购买的频率最高。
5. **实验步骤**

1) 文件的读取。使用java中的File I/O操作读取detail.txt；

2) 将每行数据以一个对象的方式存储。（提示：使用;拆分字符串）

3) 统计两两出现的频率（如何解决不考虑先后顺序问题）。（提示：两两出现的商品为一个对象，自己构造equals 方法，利用Set中不重复的特性）

4) 取得频率最高的一对。（提示：排序，或者依次比较）

5) 解决N个商品的问题。（提示：N个出现的商品为一个对象，重构对象中的equals 方法）。

1. **程序实现**

下面这段程序解决哪两个商品总被同时购买的频率最高，用两两物品组成的字符串作为HashMap的key，出现次数作为value，

**public** String getFrequentItem(String content) {

//HashMap的键设置为每两个物品组成的字符串

HashMap<String, Integer> map = **new** HashMap<>();

String res = "";

String[] str = content.split(";");

**int** length = str.length;

**for**(**int** i = 0; i < length; i++) {

String[] s1 = str[i].split(",");

**for**(**int** j = 0; j < s1.length - 1; j++) {

**for**(**int** k = j + 1; k < s1.length; k++) {

String comb = s1[j] + "," + s1[k];//两两进行组合

**if**(!map.containsKey(comb)) {

map.put(comb, 1);

} **else** {

map.put(comb, map.get(comb) + 1);

}

}

}

}

Iterator it = map.keySet().iterator();//遍历这个map

**int** max = 0;

**while**(it.hasNext()) {

String temp = (String) it.next();

**if**(max < map.get(temp)) {

max = map.get(temp);

res = temp;

}//找到其中最大的那个

}

**return** res;//返回

}

而为了解决n个商品的问题，在这里每一次购物都设置了一个Set<Set<String>>，存放由n个商品的组合（即这些商品中取n个商品的所有组合形式），对于每次购物，遍历这个set，将Set<String>作为key，出现次数作为value添加到HashMap中，然后就转化为了2个商品的问题。再找出最大的那个组合即可。

在获取n个商品时，这里采取了深度优先搜索的算法。

/\*\*

\* **@param** items, 单次购物小票

\* **@param** set 当前选中的集合；

\* **@param** n 需要的个数

\* **@param** now 当前选中的个数

\* **@param** idx 当前搜索下标

\* 用到了深度优先搜索

\*/

**public** **void** getNItems(String[] items, Set<String> set, **int** n, **int** now, **int** idx, Set<Set<String>> comb) {

**if**(now == n) {

Set<String> newSet = **new** HashSet<>(set);

comb.add(newSet);

**return**;

}

**for**(**int** i = idx; i < items.length; i++) {

set.add(items[i]);

getNItems(items, set, n, now + 1, i + 1, comb);

set.remove(items[i]);

}

}

寻找频率最高的商品组合代码如下：

/\*\*

\* 字符串content是一个超市的历次购物小票的合计，每次购物的明细之间用分号分割，每个商品之间用半角逗号分开

\* 请找出 哪n(n>=1)个商品被同时购买的频率最高，将这n个商品名称的集合（set)返回

\*

\* **@param** content，历次购物的明细，例如：炸鸡,可乐,啤酒;薯片,啤酒,炸鸡;啤酒,雪碧,炸鸡

\* **@param** n

\* **@return** 哪n个商品被同时购买的频率最高，将这n个商品名称的集合（set)返回

\* 不会出现并列的情况

\*/

**public** Set<String> getFrequentItem(String content,**int** n) {

Set<String> res = **new** HashSet<>();

HashMap<Set<String>, Integer> map = **new** HashMap<>();

String[] receipt = content.split(";");

**for**(**int** i = 0; i < receipt.length; i++) {

Set<Set<String>> comb = **new** HashSet<>();

String[] item = receipt[i].split(",");

getNItems(item, **new** HashSet<>(), n, 0, 0, comb);

Iterator it = comb.iterator();

**while**(it.hasNext()) {

Set<String> temp = (Set<String>) it.next();

**if**(!map.containsKey(temp)) {

map.put(temp, 1);

} **else** {

map.put(temp, map.get(temp) + 1);

}

}

}

**int** max = 0;

Iterator it = map.keySet().iterator();

**while**(it.hasNext()) {

Set<String> temp = (Set<String>) it.next();

**if**(max < map.get(temp)) {

max = map.get(temp);

res = temp;

}

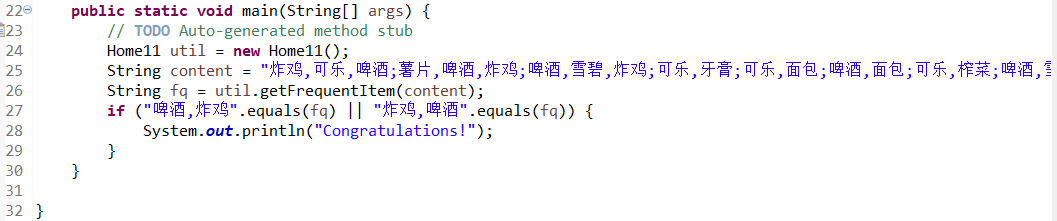
}

**return** res;

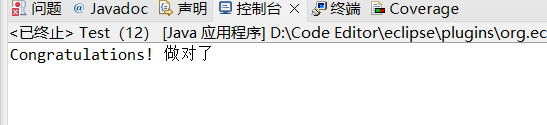
}

1. **实验结果**

对于2个商品的情况，编写测试用例并进行验证：



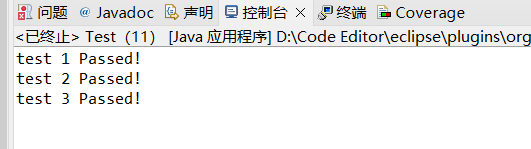
程序运行结果如下图所示：



对于n个商品的情况，编写测试用例并进行验证：



程序运行结果如下图所示：



1. **实验中遇到的问题及解决方法**

在这次实验中，我加深了对于Java中Set,Map，List等常用数据结构的理解，Map可以高效的统计每一种组合出现的次数，因此适合本题。同时还学会了一些其他的Java库函数，如split函数可以按照特定字符将String进行分割。

在统计n个商品时，由于并没有时间复杂度的限制，排列组合的解法很自然的想到了使用深度优先搜索进行遍历，通过DFS搜索的方式，对于每一次的购物，进行一次DFS，每搜索到一种组合，将其放入当前的Set，搜索结束后对set进行遍历，添加到map中，便可以统计n个商品的组合次数。最初由于错将comb这个set设置为了全局变量，导致只能通过第一组测试用例，无法多组测试。通过逐行分析代码，从堆和栈的角度去理解程序发现并解决了这个问题。