# Day10

## 1.思考题

服务器响应时间变慢，应该怎么进行排查这类问题？

服务器响应时间变慢，主要要从四个方面排查入手：CPU、内存、磁盘、网络

* uptime命令：此命令可以快速查看机器的负载情况，命令输出分别表示1分钟、5分钟、15分钟的平均负载情况。

文本

描述已自动生成

* top命令：可以查看机器总体的cpu、内存、磁盘、网络等使用情况，并且具体展示各进程使用cpu、内存等的情况，同时支持排序，可以按cpu使用率、内存使用等进行排序，方便快速找出cpu占用大或内存占用大的进程。

文本

描述已自动生成

* df命令：可以查看磁盘的使用率、剩余空间等。

图形用户界面, 文本

描述已自动生成

* 对于java应用服务器，使用jmap -heap PID命令查看JVM堆的详细信息，jstat命令用来监控JVM垃圾回收的情况
* 对于数据库服务器，检查数据库连接池参数设置是否合理，查看数据库日志，检查是否有慢sql，针对sql进行优化。

## 2.算法题

给定字符串J代表石头中宝石的类型，和字符串S代表你拥有的石头。S中每个字符代表了一种你拥有的石头的类型，你想知道你拥有的石头中有多少是宝石。

/\*\*

\* 解法1：

\* 思路：

\* 额外借助一个哈希表，key存宝石类型，value存宝石数量，

\* 因为j、s都值包含大小写字母，所以宝石最多52种，

\* 先遍历宝石字符串，找出所有目标宝石，

\* 再遍历石头字符串，记录各目标宝石出现的次数，

\* 最后累加所有目标宝石出现的次数，即是结果

\* 时间复杂度：O(n)，n为石头字符串的长度

\* 空间复杂度：使用常数级别的额外空间，O(1)

\* @param j

\* @param s

\* @return

\*/

public static int solution1(String j, String s) {

if (StringUtils.isBlank(j) || StringUtils.isBlank(s)) {

return 0;

}

// 找出所有目标宝石

Map<Character, Integer> map = new HashMap<>(26 \* 2);

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

map.put(j.charAt(i), 0);

}

// 遍历石头字符串，记录各目标宝石出现的次数

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

char item = s.charAt(i);

if (map.containsKey(item)) {

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

}

}

// 累加所有目标宝石出现的次数

AtomicInteger result = new AtomicInteger();

map.forEach((k, v) -> result.getAndAdd(v));

return result.intValue();

}

/\*\*

\* 解法2：

\* 思路：

\* 遍历石头字符串，每遍历到石头字符串的一个字符，

\* 都直接用宝石字符串进行字符串匹配，

\* 若此字符在宝石字符串中，结果 + 1，

\* 最后返回结果

\* 时间复杂度：

\* 遍历石头字符串O(n)，n为石头字符串长度，

\* 每一次字符串匹配，时间复杂度O(m)，m为宝石字符串长度，

\* 所以总时间复杂度：O(n \* m)

\* 空间复杂度：O(1)

\* @param j

\* @param s

\* @return

\*/

public static int solution2(String j, String s) {

if (StringUtils.isBlank(j) || StringUtils.isBlank(s)) {

return 0;

}

int result = 0;

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

char item = s.charAt(i);

if (j.contains(String.valueOf(item))) {

result++;

}

}

return result;

}