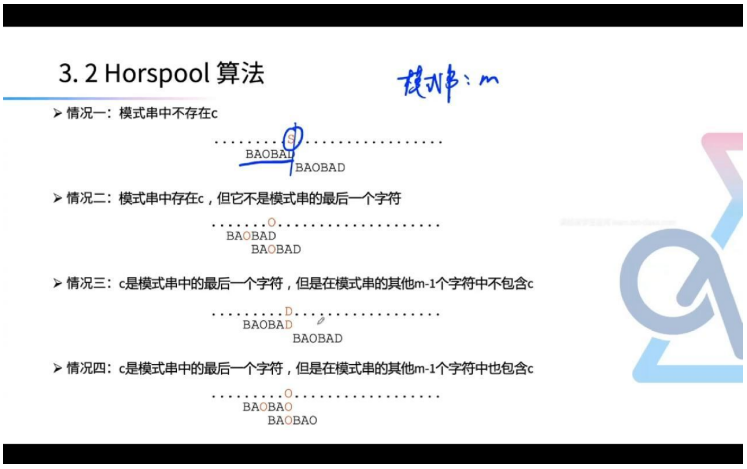
# INT102W6_计数排序和Horspool算法

## 定义：

1、计数排序是一种不涉及比较的排序。他使用的是累计数组通过空间的比较来得到对应的序列，具体操作流程可以看做是对应一个数组，统计每个数出现的次数，然后存储在累加数组里面，在这之后逐项累加，之后对应累计数组的值减一就可以得到他本来应该处于的位置。

2、Horspool是一种依赖时间和空间进行查找的算法，某种程度上和滑动窗口问题有相似之处，它主要是依靠主串（待查找项目）和模式串（目标），从模式串最后一个字符开始向前比较，若全部匹配成功那么我们就成功找到了一个子串，反之则后移且幅度尽可能的大，如下是他的四种情况，我们通常使用移动表来存储每次移动的距离




应用场景：计数排序通常有如下应用：

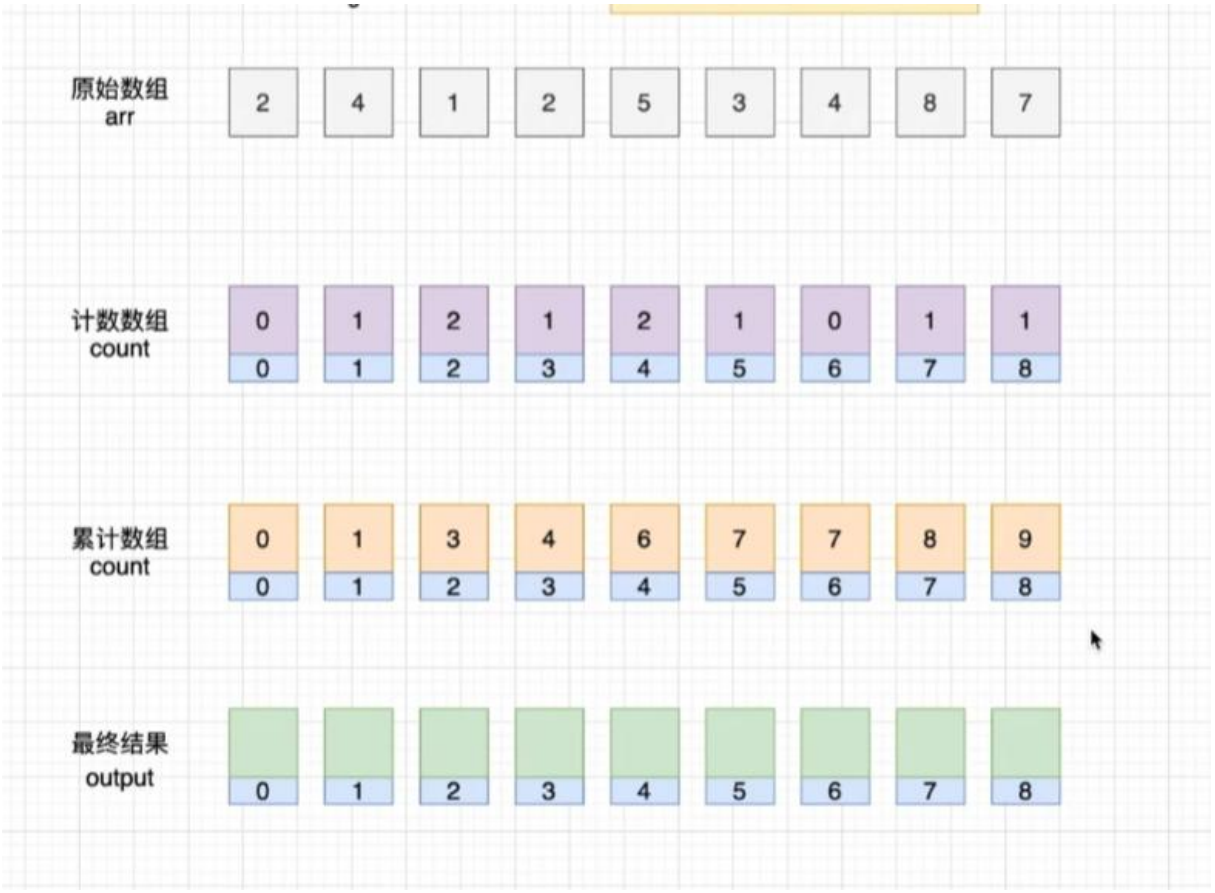
1、整数排序：计数排序特别适合于非负整数排序，尤其是当数字范围不大时。

- 2、 统计频率：计数排序可以用来统计一个数字序列中每个数字出现的次数。
- 而horspool则一般用于字符串匹配和文本搜索

优点： 计数的优点在于时间复杂度低，精确度高，稳定。而horspool则是精确而稳定。

缺点：计数排序不能用在范围特别大的情况下的排序，同时无法处理浮点数，horspool无法处理除开字符外的数据，而且需要预处理移动表和模式串

例子：horspool就只是一个字符串查找，例子我暂时没找到，不过计数我有一个比较好的例子，如下图所示




我们这里首先是找到他的最大值，然后按照他的最大值设定计数数据组和累计数据组的长度，在这基础上对应每个数计算他们的出现次数，在累计数组中逐项相加，然后我们就可以得到对应有多少个数小于等于累计数组中的索引，也就是对应索引所应该处于的位置，然后索累计数组中的数据减一，把对应的索引填入到最终结果中。代码实现如下

```
public class CountingSortExample {
    public static void countingSort(int[] array) {
        int size = array.length;
        int max = getMaxValue(array); // 找到最大的值

        // 创建计数数组和输出数组
        int[] count = new int[max + 1];
        int[] output = new int[size];

        //将每个元素的计数存储在计数数组中（array[i] 表示当前位置 i 上的元素值。假设某个元素值为 val，那么 count[val] 就表示元素值为 val 的个数。）
        for (int i = 0; i < size; i++) {
            count[array[i]]++;
        }

        // 存储累计计数
        for (int i = 1; i <= max; i++) {
            count[i] += count[i - 1];
```

```
    }

    // 将元素放在输出数组中
    for (int i = size - 1; i >= 0; i--) {
        output[count[array[i]] - 1] = array[i];
        count[array[i]]--;
    }

}
```