P12

P12

【题目5】

kmp算法

【题目5】

一个矩阵中只有0和1两种值，每个位置都可以和自己的上、下、左、右四个位置相连，如果有一片1连在一起，这个部分叫做一个岛，求一个矩阵中有多少个岛?

【举例】

4

001010

111010

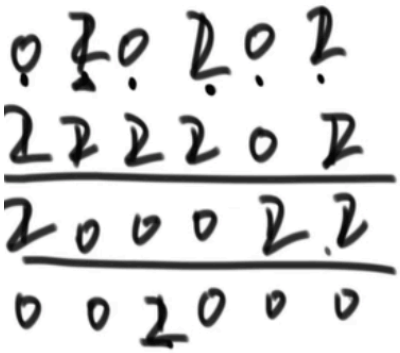
100100

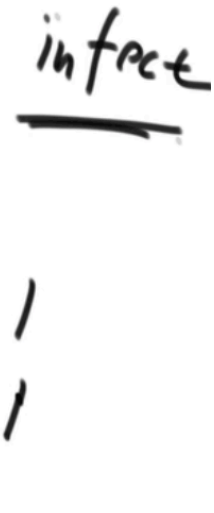
000000

这个矩阵中有三个岛

【进阶】

如何设计一个并行算法解决这个问题





```java
public class Code03_Islands {

    public static int countIslands(int[][] m) {
        if (m == null || m[0] == null) {
            return 0;
        }
        int N = m.length;
        int M = m[0].length;
        int res = 0;
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                if (m[i][j] == 1) {
                    res++;
                    infect(m, i, j, N, M);
                }
            }
        }
        return res;
    }

    public static void infect(int[][] m, int i, int j, int N, int M) {
        if (i < 0 || i >= N || j < 0 || j >= M || m[i][j] != 1) {
            return;
        }
        // i，j没越界，并且当前位置值是1
        m[i][j] = 2;
        infect(m, i + 1, j, N, M);
        infect(m, i - 1, j, N, M);
        infect(m, i, j + 1, N, M);
        infect(m, i, j - 1, N, M);
    }
```

并查集：

```java
// 样本进来会包一层，叫做元素
public static class Element<V> {
    public V value;

    public Element(V value) {
        this.value = value;
    }

}
```

```java
public static class UnionFindSet<V> {
    public HashMap<V, Element<V>> elementMap;
    // key  某个元素 value 该元素的父
    public HashMap<Element<V>, Element<V>> fatherMap;
    // key 某个集合的代表元素  value 该集合的大小
    public HashMap<Element<V>, Integer> sizeMap;

    public UnionFindSet(List<V> list) {
        elementMap = new HashMap<>();
        fatherMap = new HashMap<>();
        sizeMap = new HashMap<>();
        for (V value : list) {
            Element<V> element = new Element<V>(value);
            elementMap.put(value, element);
            fatherMap.put(element, element);
            sizeMap.put(element, 1);
        }
    }
```

```java
public boolean isSameSet(V a, V b) {
    if (elementMap.containsKey(a) && elementMap.containsKey(b)) {
        return findHead(elementMap.get(a)) == findHead(elementMap.get(b));
    }
    return false;
}
```

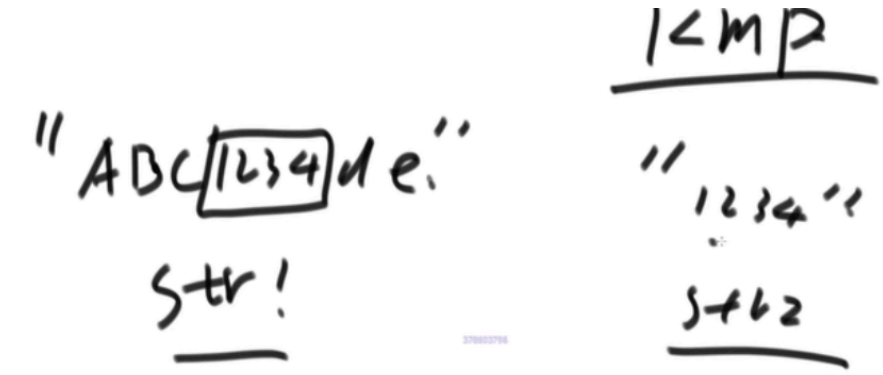
```java
// 给定一个ele，往上一直找，把代表元素返回
private Element<V> findHead(Element<V> element) {
    Stack<Element<V>> path = new Stack<>();
    while (element != fatherMap.get(element)) {
        path.push(element);
        element = fatherMap.get(element);
    }
    while (!path.isEmpty()) {
        fatherMap.put(path.pop(), element);
    }
    return element;
}
```

```java
public void union(V a, V b) {
    if (elementMap.containsKey(a) && elementMap.containsKey(b)) {
        Element<V> aF = findHead(elementMap.get(a));
        Element<V> bF = findHead(elementMap.get(b));
        if (aF != bF) {
            Element<V> big = sizeMap.get(aF) >= sizeMap.get(bF) ? aF : bF;
            Element<V> small = big == aF ? bF : aF;
            fatherMap.put(small, big);
            sizeMap.put(big, sizeMap.get(aF) + sizeMap.get(bF));
            sizeMap.remove(small);
        }
    }
}
```

---

kmp算法

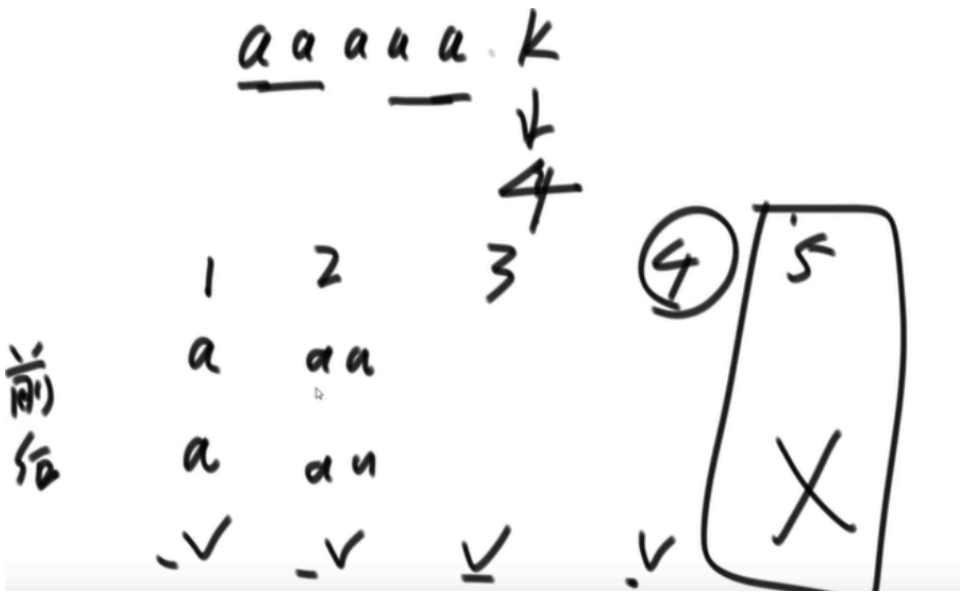
字符串str1和str2，str1是否包含str2，如果包含返回str2在str1中开始的位置。
如何做到时间复杂度0（N）完成?



要连续

暴力方法复杂度：

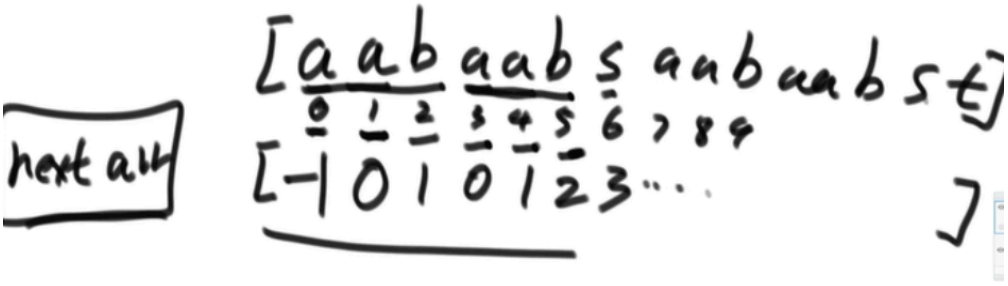
相等的最大长度：3

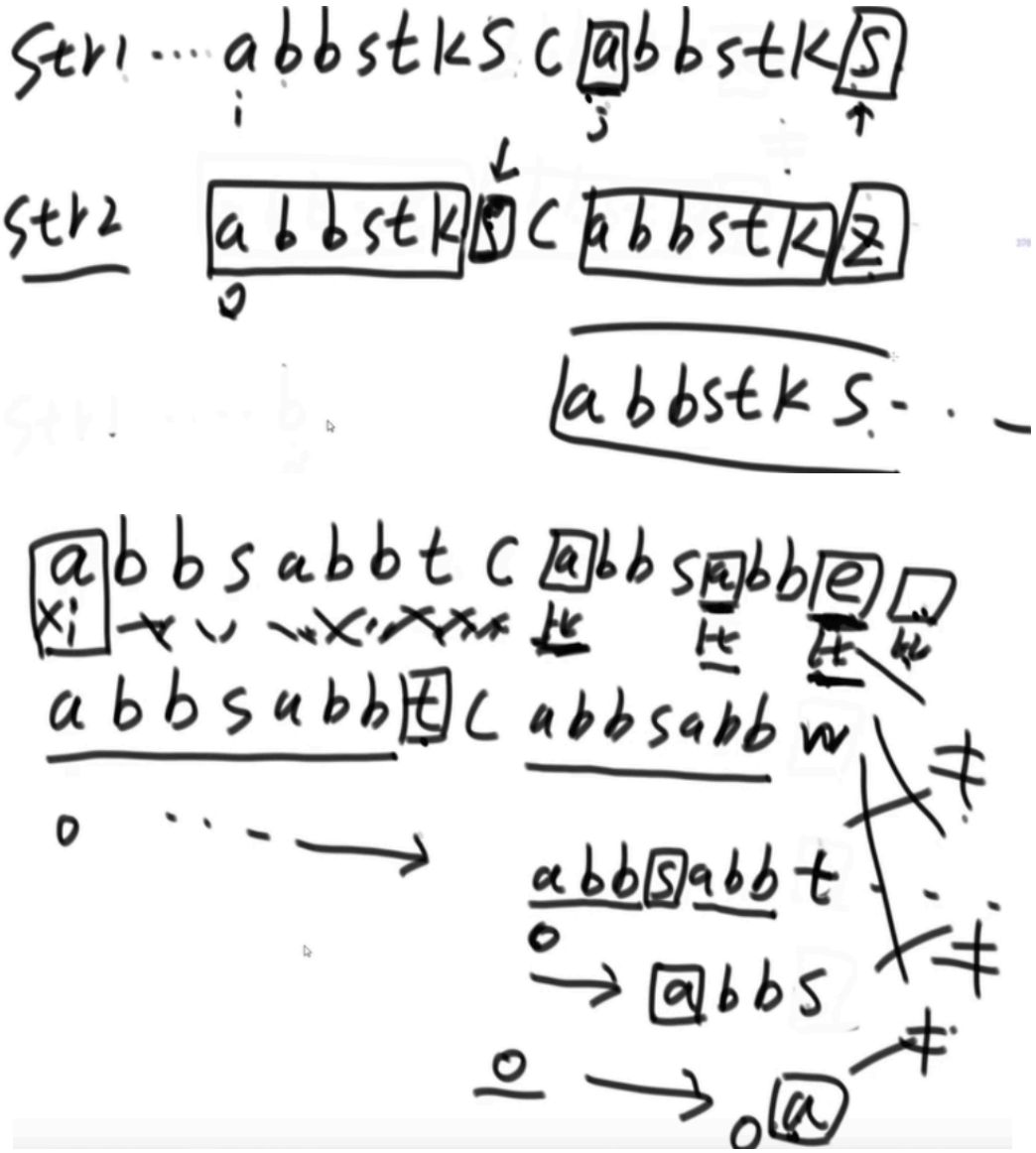


相等的最大长度：4

（不能取到整体）

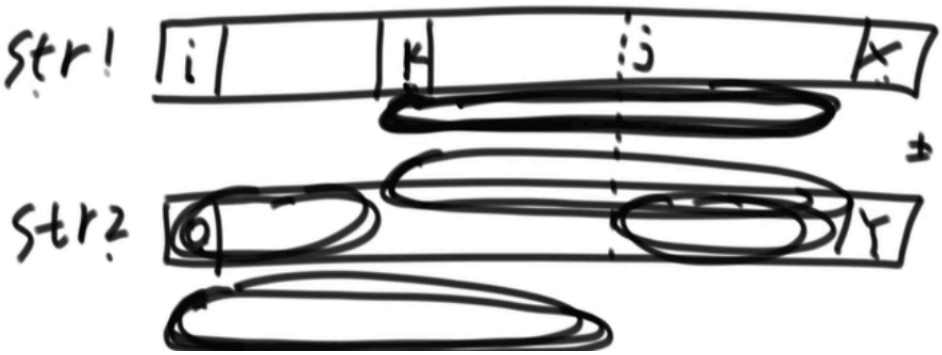
构建nextarr： 针对str2

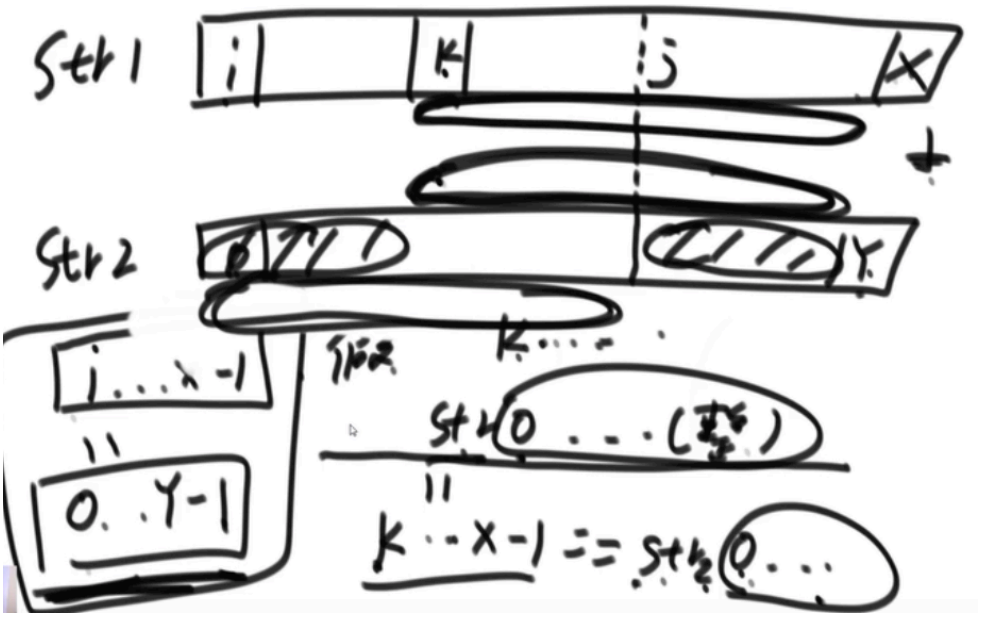



str1 ··· abbstkS C abbstkS
i
j
str2 abbstk C abbstkZ
0
abbstkS...

abbsabbt C abbsabbe
xi
abbsabbt C abbsabbw
0
abbsabbt
0
abbs
0
a
0

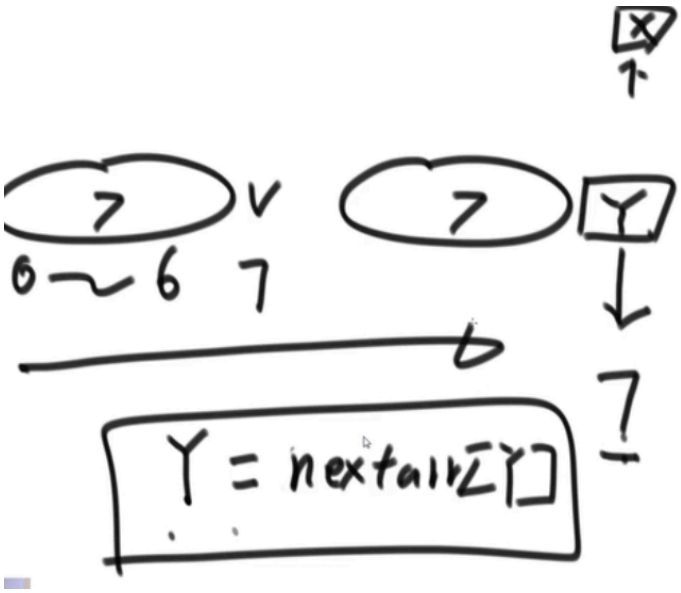
证明

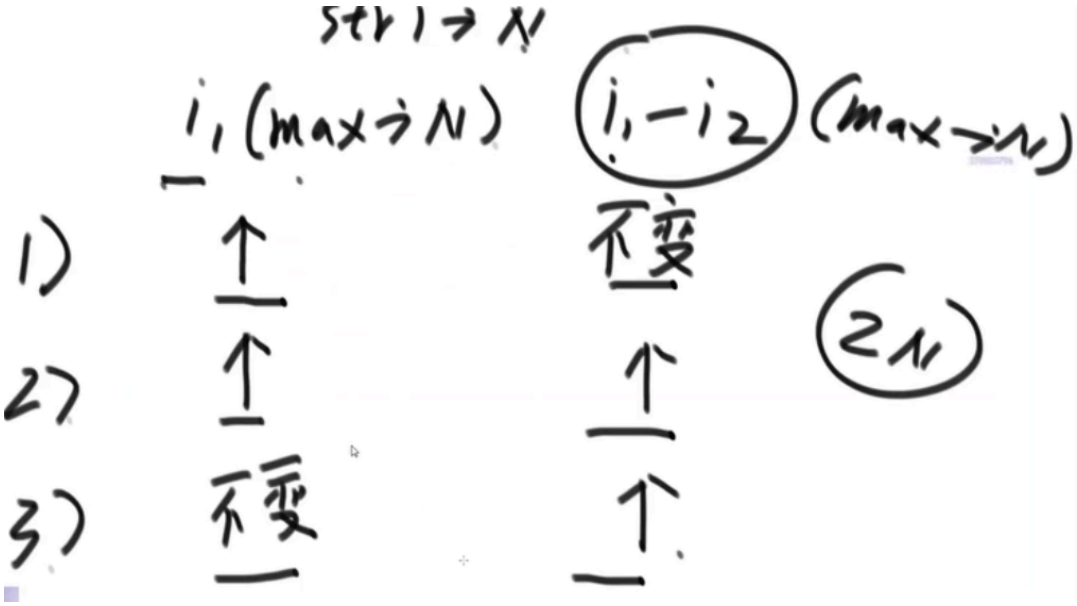

str1 i
str2 0



实现：

```java
// N >= M
public static int getIndexOf(String s, String m) {
    if (s == null || m == null || m.length() < 1 || s.length() < m.length()) {
        return -1;
    }
    char[] str1 = s.toCharArray();
    char[] str2 = m.toCharArray();
    int i1 = 0;
    int i2 = 0;
    int[] next = getNextArray(str2); // O (M)
    // O(N)
    while (i1 < str1.length && i2 < str2.length) {
        if (str1[i1] == str2[i2]) {
            i1++;
            i2++;
        } else if (next[i2] == -1) {
            i1++;
        } else {
            i2 = next[i2];
        }
    }
    // i1 越界 或者 i2越界了
    return i2 == str2.length ? i1 - i2 : -1;
}
```





while循环的复杂度是线性的

```
public static int[] getNextArray(char[] ms) {
    if (ms.length == 1) {
        return new int[] { -1 };
    }
    int[] next = new int[ms.length];
    next[0] = -1;
    next[1] = 0;
    int i = 2; // next数组的位置
    int cn = 0;
    while (i < next.length) {
        if (ms[i - 1] == ms[cn]) {
            next[i++] = ++cn;
        }
        // 当前跳到cn位置的字符，和i-1位置的字符配不上
        else if (cn > 0) { //
            cn = next[cn];
        } else {
            next[i++] = 0;
        }
    }
    return next;
}
```

1) next[i][j] = cnt+1

2) i++

3) cnt++