**技术笔试题**

要求：用自己熟悉的语言（包含但不限于C、C++、Objective-C、Java），编写代码，完成下面的题目。要求编写真实可运行的代码。

1. 字符串：把字符串中的空格转换为 %20。

例如：输入字符串 "how are you"，返回字符串"how%20are%20you"

方法1:stringByReplacingOccurrencesOfString

//调用示例

NSString \*resultStr = [self replaceStr:@"how are you" sourceStr:@"%20"];

- (NSString \*)replaceStr:(NSString \*)originStr sourceStr:(NSString \*)sourceStr {

originStr = [originStr stringByReplacingOccurrencesOfString:@" " withString:sourceStr];

NSLog(@"====%@====",originStr);

return originStr;

}

方式2:正则匹配

//调用示例

NSString \*resultStr = [self verifyRegex:@"how are you" sourceStr:@"%20"];

- (NSString \*)verifyRegex:(NSString \*)originStr sourceStr:(NSString \*)sourceStr {

NSError \*error = nil;

NSRegularExpression \*regex = [NSRegularExpression regularExpressionWithPattern:@"\\s" options:NSRegularExpressionCaseInsensitive error:&error];

NSArray \*arr = [regex matchesInString:originStr options:NSMatchingReportCompletion range:NSMakeRange(0, originStr.length)];

arr = [[arr reverseObjectEnumerator] allObjects];

for (NSTextCheckingResult \*str in arr) {

originStr = [originStr stringByReplacingCharactersInRange:[str range] withString:sourceStr];

}

NSLog(@"====%@====",originStr);

return originStr;

}

2. 链表：给定两个用链表表示的整数，每个结点包含一个数位。这些数位是反向存放的，也就是个位排在链表首部。编写函数对这两个整数求和并用链表形式返回结果。

例如： 364 表示为 4->6->3，789 表示为 9->8->7，

输入链表 4->6->3 和 9->8->7

返回链表 3->5->1->1 （即表示1153的链表）

3. 递归/动态规划：有个小孩正在上楼梯，楼梯有n阶台阶，小孩一次可以上1阶、2阶或3阶。实现一个算法，计算小孩有多少种上楼梯的方式。

输入n，返回一个整数。

调用示例:

[self verifyLoftSteps:@"50"];

// 验证递归算法

- (void)verifyLoftSteps:(NSString \*)inputStr {

int value = [inputStr intValue];

if (value > 77) {

NSLog(@"数据长度过大，会越界");

return;

}

dispatch\_async(dispatch\_get\_global\_queue(0, 0), ^{

//动态规划算法，缩短计算时间，降低成本

NSString \*results = [self newCacul:value];

//递归算法，如果处理的数据量过大，那么运算时会过于耗

//long long allmethod = [self cacul:value];

dispatch\_async(dispatch\_get\_main\_queue(), ^{

NSLog(@"result = %@",results);

//NSLog(@"all = %ld",(long)allmethod);

});

});

}

// 动态递归，动态规划算法

- (NSString \*)newCacul:(int)allSteps {

NSMutableArray \*resutlsArr = [NSMutableArray array];

resutlsArr[0] = @"0";

resutlsArr[1] = @"1";

resutlsArr[2] = @"2";

resutlsArr[3] = @"4";

for (int i = 4; i <= allSteps; i ++) {

resutlsArr[i] = [NSString stringWithFormat:@"%lli",([resutlsArr[i - 1] longLongValue] + [resutlsArr[i - 2] longLongValue] + [resutlsArr[i - 3] longLongValue])];

}

return resutlsArr[allSteps];

}

// 常规的递归方法, 处理的数据量过大，那么运算时会过于耗

- (long long)cacul:(int) allSteps {

if (allSteps < 0) {

return 0;

}

if (allSteps == 0 || allSteps == 1) {

return 1;

}

return [self cacul:(allSteps-1)]+[self cacul:(allSteps-2)] + [self cacul:(allSteps-3)];

}