【第三十八课】金融系统中的 RSA 算法 (二)

1、1477. 找两个和为目标值且不重叠的子数组

1. 计算dp过程中不断计算满足条件的两个长度之和
2. dp[i]表示到编号i（不包含i）的情况下满足target目标的最短子数组的长度
3. 初始化 较大值表示无效
4. 因为是连续子数组，使用前缀和来解决，并且基于双指针很去计算范围即可
5. d[i] 取决于两种情况：（1）能构成target，则min(dp[i-1], r-l+1)，取更短的子数组；（2）不能，则等于dp[i-1], 即没找到直接取上一个情况的值

```
/**
 * @param {number[]} arr
 * @param {number} target
 * @return {number}
 */
var minSumOfLengths = function(arr, target) {
    const n = arr.length;
    const dp = new Array(n + 1).fill(Infinity);
    let res = Infinity;
    for(let i = 0,j  = 0,win = 0;i < n;i++){
        win += arr[i];//集合的元素和值，因为是连续的子数组，所以可以用前缀和来计算
        while(win > target) win -= arr[j++];
        if(win === target){
            const len = i - j + 1;
            res = Math.min(res,dp[j] + len);
            dp[i + 1] = Math.min(len,dp[i]);
        }else{
            dp[i + 1] = dp[i];
        }
    }
    return res === Infinity ? -1 : res;
};
```

2、面试题 17.13. 恢复空格

1. 思路和 [132. 分割回文串 II]很像，拥有前 i 个字符的子串 [0,i - 1] 的最优解 取决于两部分：
    1. (1) 拥有前 j 个字符的子串 [0,j - 1] ---> 规模更小的子问题
    2. (2)[j,i - 1] 是否是字典里的单个单词

2. 我们用指针 j 去划分，[j,i-1] 区间从长度 0 逐渐到 i，考察它是否是字典的单词

```
/**
 * @param {string[]} dictionary
 * @param {string} sentence
 * @return {number}
 */
var respace = function(dictionary, sentence) {
    if(sentence.length == 0) return 0;
    let dp = new Array(sentence.length).fill(0);
    for(let i = 1;i <= sentence.length;i++){
        dp[i] = dp[i - 1] + 1;
        for(let j = 0;j < dictionary.length;j++){
            let word = dictionary[j].length;
            if(dictionary[j] === sentence.substring(i - word,i) && word <= i){
                dp[i] = Math.min(dp[i],dp[i - word]);
            }
        }
    }
    return dp[sentence.length];
};
```

3、2029. 石子游戏 IX

1. 讨论mod0数量为奇数个和偶数个的情况：
   （1）偶数个
      mod0数量为偶数时，无实质作用；
   （2）讨论mod1 mod2数量
      a.mod1 mod2同时存在:
        1)数量不同，先手选取数量少的，获胜；
        2)数量相等，先手任选，获胜；
      b.如果只存在一种：
        1)取到第三次时后手胜；
        2)取光后手胜；
   2.奇数个
     mod0为奇数时，可以起到翻转局面的作用；
     如果mod1 mod2数量相差超过2，则先手选择数量多的可以取胜；

```
/**
 * @param {number[]} stones
 * @return {boolean}
 */
var stoneGameIX = function(stones) {
    const s = [0,0,0]//存储 mod0,mod1,mod2
            len = stones.length;
```

```
    for(let i = 0;i < len;i++){
        s[stones[i] % 3]++;
    }
    if(s[0] %  2 === 0){//mod0数量是偶数
        // mod1,mod2同时存在,
        // 1、数量不同，先下手。，取数量小，获胜
        // 2、数量相同，先下手任意选，获胜
        return s[1] != 0 && s[2] != 0;
    }
    //mod0数量是奇数，发逆转局面
    //  mod1,mod2 数值相差大于 2，先下手，取数量大，获胜
    return s[2] > s[1] + 2 || s[1] > s[2] + 2;
};
```

4、284. 窥探迭代器

1. JavaScript 中自定义的 Iterator 接口支持 next 和 hasNext 操作，但是不支持 peek 操作。为了在顶端迭代器中支持 peek 操作，需要使用 nextElement 存储迭代器的下一个元素，各项操作的实现如下：
2. next：首先用 ret 存储 ElemnextElement 表示返回值，然后将 nextElement 向后移动一位，最后返回 ret；
3. hasNext：由于 nextElement 为迭代器的下一个元素，因此当 nextElement 不为空时返回 true，否则返回 false；
4. peek：由于 peek 操作不改变指针，因此返回 nextElement。

```
/**
 * // This is the Iterator's API interface.
 * // You should not implement it, or speculate about its implementation.
 * function Iterator() {
 *     @ return {number}
 *     this.next = function() { // return the next number of the iterator
 *         ...
 *     };
 *
 *     @return {boolean}
 *     this.hasNext = function() { // return true if it still has numbers
 *         ...
 *     };
 * };
 */

/**
 * @param {Iterator} iterator
 */
var PeekingIterator = function(iterator) {
    this.iterator = iterator;
```

```
    this.nextElement = this.iterator.next();
};

/**
 * @return {number}
 */
PeekingIterator.prototype.peek = function() {
    return this.nextElement;
};

/**
 * @return {number}
 */
PeekingIterator.prototype.next = function() {
    const ret = this.nextElement;
    this.nextElement = this.iterator.hasNext() ? this.iterator.next() : null;
    return ret;
};

/**
 * @return {boolean}
 */
PeekingIterator.prototype.hasNext = function() {
    return this.nextElement != null;
};

/**
 * Your PeekingIterator object will be instantiated and called as such:
 * var obj = new PeekingIterator(arr)
 * var param_1 = obj.peek()
 * var param_2 = obj.next()
 * var param_3 = obj.hasNext()
 */
```