算法分析和複雜性理論

干皓丞，2101212850, 信息工程學院

2022 年 4 月 12 日

1　作業目標與章節摘要

1. LeetCode 85. Maximal Rectangle 最大矩形
2. LeetCode 152. Maximum Product Subarray 乘積最大子數組

2　作業內容概述

作業可以從 GitHub 下的 kancheng/kan-cs-report-in-2022 專案找到，作業程式碼與文件目錄為 kan-cs-report-in-2022/AATCC/lab-report/。實際執行的環境與實驗設備為 Google 的 Colab 、MacBook Pro (Retina, 15-inch, Mid 2014) 、Acer Aspire R7 與 HP Victus (Nvidia GeForce RTX 3060)。

本作業 GitHub 專案為 kancheng/kan-cs-report-in-2022 下的 AATCC' 的目錄。程式碼可以從 code 目錄下可以找到 *.pynb，內容包含上次課堂練習、LeetCode 範例思路整理與作業。

https://github.com/kancheng/kan-cs-report-in-2022/tree/main/AATCC

Fig. 1. 作業專案位置

1. LeetCode : https://leetcode.com/
2. LeetCode CN : https://leetcode-cn.com/
3. OnlineGDB : https://www.onlinegdb.com/

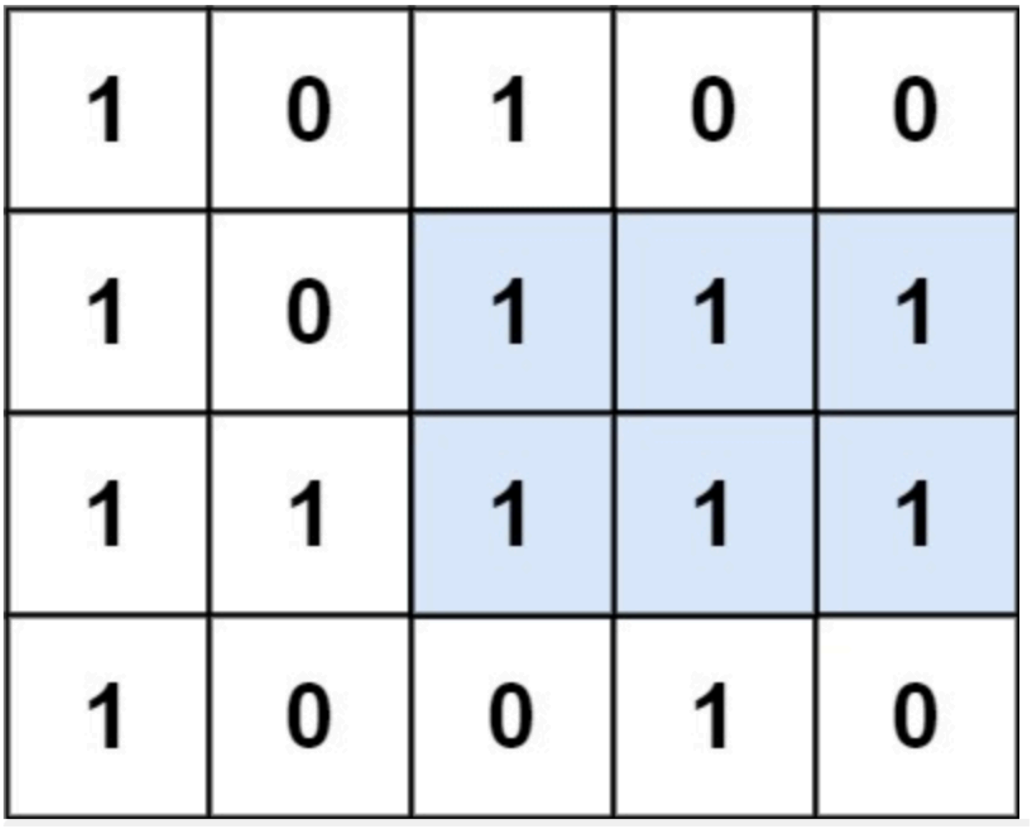
LeetCode 的平台部分，CN 的平台有針對簡體中文使用者進行處理，包含中英文切換等功能。OnlineGDB 則可線上進行簡易的環境測試，其程式碼涵蓋 C, C++, C#, Java, Python, JS, Rust, Go。

3 LeetCode 85. Maximal Rectangle 最大矩形

3.1 LeetCode 85. 題目

Given a rows x cols binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its area.

給定一個僅包含 0 和 1 、大小為 rows x cols 的二維二進制矩陣，找出只包含 1 的最大矩形，並返回其面積。



1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

Fig. 2. Example

Example 1:

```
Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]
Output: 6
Explanation: The maximal rectangle is shown in the above picture.
```

最大矩形如上圖所示。

Example 2:

```
Input: matrix = []
Output: 0
```

Example 3:

```
Input: matrix = [["0"]]
Output: 0
```

Example 4:

```
Input: matrix = [["1"]]
Output: 1
```

Example 5:

```
Input: matrix = [["0","0"]]
Output: 0
```

Constraints:

1. rows == matrix.length
2. cols == matrix[i].length
3. 1 <= row, cols <= 200
4. matrix[i][j] is '0' or '1'.

3.2 LeetCode 85. Code 範例

LeetCode 85. Python 1

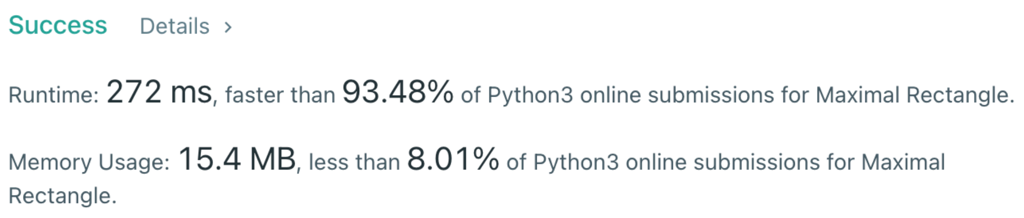
```python
from typing import List
class Solution:
    def maximalRectangle(self, matrix: List[List[str]]) -> int:
        if not matrix or not matrix[0]:
            return 0
        nums = [int(''.join(row), base=2) for row in matrix]
        ans, N = 0, len(nums)
        for i in range(N):
            j, num = i, nums[i]
            while j < N:
                num = num & nums[j]
                # print('num=',bin(num))
                if not num:
                    break
                width, curnum = 0, num
                while curnum:
                    width += 1
                    curnum = curnum & (curnum >> 1)
                ans = max(ans, width * (j-i+1))
                j += 1
        return ans
```

LeetCode 85. Python 2

```python
class Solution:
    def maximalRectangle(self, matrix) -> int:
        if len(matrix) == 0:
            return 0
        res = 0
        m, n = len(matrix), len(matrix[0])
        heights = [0] * n
        for i in range(m):
            for j in range(n):
                if matrix[i][j] == '0':
                    heights[j] = 0
                else:
                    heights[j] = heights[j] + 1
            res = max(res, self.largestRectangleArea(heights))
        return res

    def largestRectangleArea(self, heights):
        heights.append(0)
        stack = []
        res = 0
        for i in range(len(heights)):
            while stack and heights[i] < heights[stack[-1]]:
                s = stack.pop()
                res = max(res, heights[s] * ((i - stack[-1] - 1) if stack else i
                    ))
            stack.append(i)
        return res
```

3.3 LeetCode 85. 結果


Success Details ›

Runtime: 272 ms, faster than 93.48% of Python3 online submissions for Maximal Rectangle.

Memory Usage: 15.4 MB, less than 8.01% of Python3 online submissions for Maximal Rectangle.

Fig. 3. LeetCode 85 結果

4 LeetCode 152. Maximum Product Subarray 乘積最大子數組

4.1 LeetCode 152. 題目

Given an integer array nums, find a contiguous non-empty subarray within the array that has the largest product, and return the product.

The test cases are generated so that the answer will fit in a 32-bit integer.

A subarray is a contiguous subsequence of the array.

給你一個整數數組 nums，請你找出數組中乘積最大的非空連續子數組（該子數組中至少包含一個數字），並返回該子數組所對應的乘積。

測試用例的答案是一個 32-位整數。

子數組是數組的連續子序列。

Example 1:

```
Input: nums = [2,3,-2,4]
Output: 6
Explanation: [2,3] has the largest product 6.
子數組 [2,3] 有最大乘積 6。
```

Example 2:

```
Input: nums = [-2,0,-1]
Output: 0
Explanation: The result cannot be 2, because [-2,-1] is not a subarray.
結果不能為 2, 因為 [-2,-1] 不是子數組。
```

Constraints:

1. $1 <= nums.length <= 2 * 10^4$

2. -10 <= nums[i] <= 10

3. The product of any prefix or suffix of nums is guaranteed to fit in a 32-bit integer.

nums 的任何前綴或後綴的乘積都保證是一個 32-位整數

4.2 LeetCode 152. 思路總結

1. 給定一個整數數組 nums ，找出一個序列中乘積最大的連續子序列（該序列至少包含一個數）。

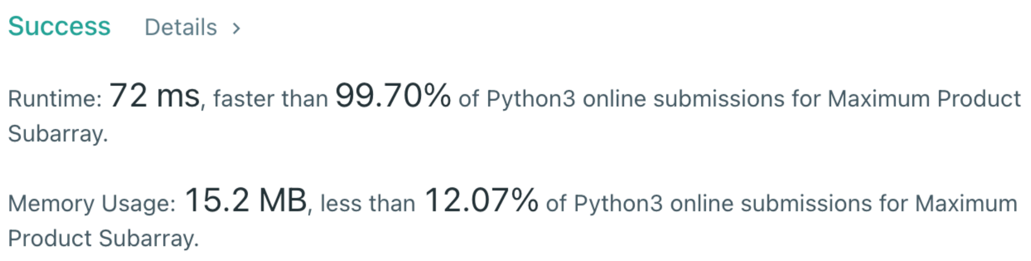
2. 給出一個數組，要求找出這個數組中連續元素乘積最大的值。

3. 這一題是 DP 的題，狀態轉移方程是：最大值是 Max(f(n)) = Max(Max(f(n-1)) * n, Min(f(n-1)) * n)；最小值是 Min(f(n)) = Min(Max(f(n-1)) * n, Min(f(n-1)) * n)。只要動態維護這兩個值，如果最後一個數是負數，最大值就在負數 * 最小值中產生，如果最後一個數是正數，最大值就在正數 * 最大值中產生。

4.3 LeetCode 152. Code 範例

```python
class Solution:
    def maxProduct(self, A):
        B = A[::-1]
        for i in range(1, len(A)):
            A[i] *= A[i - 1] or 1
            B[i] *= B[i - 1] or 1
        return max(max(A),max(B))
```

4.4 LeetCode 152. 結果


Success Details ›

Runtime: 72 ms, faster than 99.70% of Python3 online submissions for Maximum Product Subarray.

Memory Usage: 15.2 MB, less than 12.07% of Python3 online submissions for Maximum Product Subarray.

Fig. 4. LeetCode 152 結果