# 算法分析和複雜性理論

干皓丞，2101212850, 信息工程學院

2022 年 5 月 1 日

## 1　作業目標與章節摘要

1. LeetCode 240. Search a 2D Matrix II 搜索二維矩陣 II
2. LeetCode 347. Top K Frequent Elements 前 K 個高頻元素
3. LeetCode 374. Guess Number Higher or Lower 二叉樹的所有路徑

## 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 240. Search a 2D Matrix II 搜索二□矩□ II

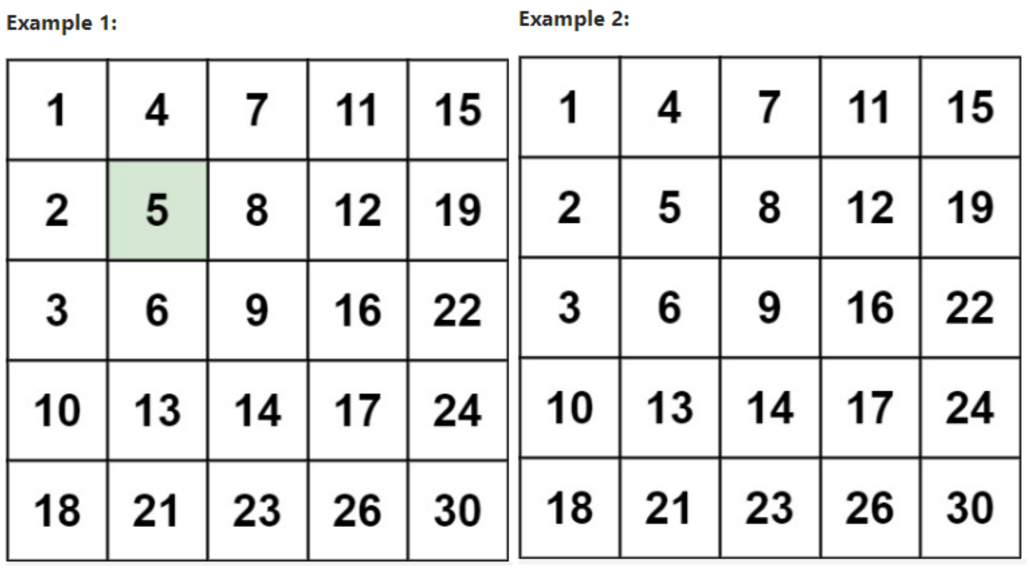
## 3.1 LeetCode 240. 題目

Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

編寫一個高效的算法來搜索 m x n 矩陣 matrix 中的一個目標值 target 。該矩陣具有以下特性：

- 每行的元素從左到右升序排列。
- 每列的元素從上到下升序排列。



**Example 1:**

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 11 | 15 |
| 2 | 5 | 8 | 12 | 19 |
| 3 | 6 | 9 | 16 | 22 |
| 10 | 13 | 14 | 17 | 24 |
| 18 | 21 | 23 | 26 | 30 |

**Example 2:**

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 11 | 15 |
| 2 | 5 | 8 | 12 | 19 |
| 3 | 6 | 9 | 16 | 22 |
| 10 | 13 | 14 | 17 | 24 |
| 18 | 21 | 23 | 26 | 30 |

Fig. 2. Example

Example 1:

```
Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 5
Output: true
```

Example 2:

```
Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 20
Output: false
```

Constraints:

1. m == matrix.length
2. n == matrix[i].length
3. 1 <= n, m <= 300
4. $-10^9 <= matrix[i][j] <= 10^9$
5. All the integers in each row are sorted in ascending order.
6. All the integers in each column are sorted in ascending order.
7. $-10^9 <= target <= 10^9$

### 3.2 LeetCode 240. 思路總結

1. 給出一個二維矩陣，矩陣的特點是每一個行內，元素隨著下標增大而增大，每一列內，元素也是隨著下標增大而增大。但是相鄰兩行的元素並沒有大小關係。例如第一行最後一個元素就比第二行第一個元素要大。要求設計一個算法能在這個矩陣中高效的找到一個數，如果找到就輸出 true，找不到就輸出 false。

2. 這一題是第 74 題的加強版。第 74 題中的二維矩陣完全是一個有序的一維矩陣，但是這一題如果把它拍扁成一維，並不是有序的。首先每一個行或者每一列是有序的，那麼我們可以依次在每一行或者每一列中利用二分去搜索。這樣做時間複雜度為 O(n log n)。

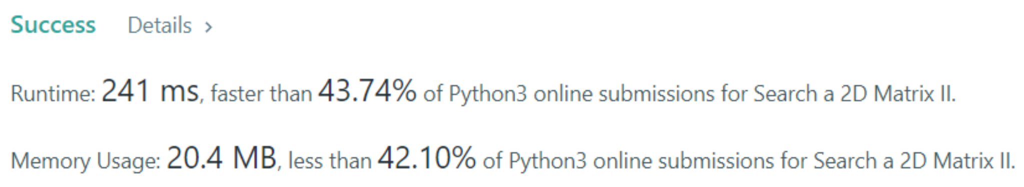
3. 還有一個模擬的解法。通過觀察，我們發現了這個矩陣的一個特點，最右邊一列的元素是本行中最大的元素，所以我們可以先從最右邊一列開始找到第一個比 target 元素大的元素，這個元素所在的行，是我們接著要搜索的。在行中搜索是從最右邊開始往左邊搜索，時間複雜度是 O(n)，算上一開始在最右邊一列中查找的時間複雜度是 O(m)，所以最終的時間複雜度為 O(m+n)。

### 3.3 LeetCode 240. Code 範例

```python
class Solution:
    def searchMatrix(self, matrix, target):
        """
        :type matrix: List[List[int]]
        :type target: int
        :rtype: bool
        """
        m = len(matrix)
        if m == 0:
            return False
        n = len(matrix[0])
        if n == 0:
            return False

        i = m - 1
        j = 0
        while i >= 0 and j < n:
            if matrix[i][j] == target:
                return True
            elif matrix[i][j] < target:
                j = j + 1
            else:
                i = i - 1
        return False
```

### 3.4 LeetCode 240. 結果



Success Details ›

Runtime: 241 ms, faster than 43.74% of Python3 online submissions for Search a 2D Matrix II.

Memory Usage: 20.4 MB, less than 42.10% of Python3 online submissions for Search a 2D Matrix II.

Fig. 3. LeetCode 240 結果

# 4 LeetCode 347. Top K Frequent Elements 前 K 个高⍰元素

## 4.1 LeetCode 347. 題目

Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.

給你一個整數數組 nums 和一個整數 k ，請你返回其中出現頻率前 k 高的元素。你可以按任意順序返回答案。

Example 1:

```
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
```

Example 2:

```
Input: nums = [1], k = 1
Output: [1]
```

Constraints:

- $1 <= nums.length <= 10^5$
- k is in the range [1, the number of unique elements in the array].
- It is guaranteed that the answer is unique.

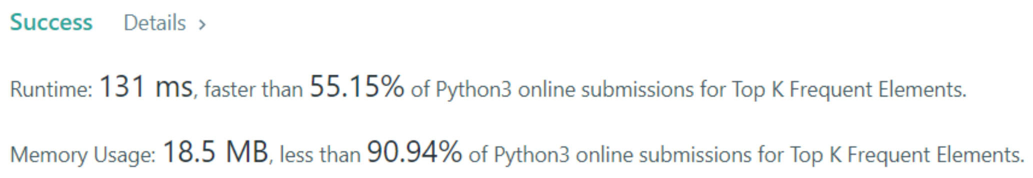
## 4.2 LeetCode 347. 思路總結

把數組構造成一個優先隊列，輸出前 K 個即可。

## 4.3 LeetCode 347. Code 範例

```python
from typing import List
import heapq
class Solution:
    def topKFrequent(self, nums: List[int], k: int) -> List[int]:
        map_ = {}
        for i in range(len(nums)):
            map_[nums[i]] = map_.get(nums[i], 0) + 1
        pri_que = []
        for key, freq in map_.items():
            heapq.heappush(pri_que, (freq, key))
            if len(pri_que) > k:
                heapq.heappop(pri_que)
        result = [0] * k
        for i in range(k-1, -1, -1):
            result[i] = heapq.heappop(pri_que)[1]
        return result
```

## 4.4 LeetCode 347. 結果



Runtime: 131 ms, faster than 55.15% of Python3 online submissions for Top K Frequent Elements.

Memory Usage: 18.5 MB, less than 90.94% of Python3 online submissions for Top K Frequent Elements.

Fig. 4. LeetCode 347 結果

# 5 LeetCode 374. Guess Number Higher or Lower 二叉⊠的所有路⊠

## 5.1 LeetCode 374. 題目

We are playing the Guess Game. The game is as follows:

I pick a number from 1 to n. You have to guess which number I picked.

Every time you guess wrong, I will tell you whether the number I picked is higher or lower than your guess.

You call a pre-defined API int guess(int num), which returns three possible results:

- -1: Your guess is higher than the number I picked (i.e. num > pick).

- 1: Your guess is lower than the number I picked (i.e. num < pick).

- 0: your guess is equal to the number I picked (i.e. num == pick).

Return the number that I picked.

猜數字遊戲的規則如下：

每輪遊戲，我都會從 1 到 n 隨機選擇一個數字。請你猜選出的是哪個數字。如果你猜錯了，我會告訴你，你猜測的數字比我選出的數字是大了還是小了。你可以通過調用一個預先定義好的接口 int guess(int num) 來獲取猜測結果，返回值一共有 3 種可能的情況（-1，1 或 0）：

-1：我選出的數字比你猜的數字小 pick < num 1：我選出的數字比你猜的數字大 pick > num 0：我選出的數字和你猜的數字一樣。恭喜！你猜對了！pick == num

返回我選出的數字。

Example 1:

```
Input: n = 10, pick = 6
Output: 6
```

Example 2:

```
Input: n = 1, pick = 1
Output: 1
```

Example 3:

```
Input: n = 2, pick = 1
Output: 1
```

Constraints:

- 1 <= n <= $2^3$1 - 1

- 1 <= pick <= n

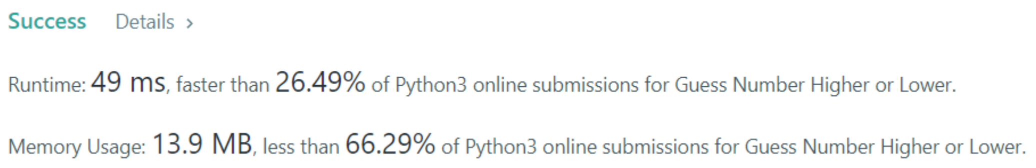
## 5.2 LeetCode 374. 思路總結

二分查找

## 5.3 LeetCode 374. Code 範例

```python
class Solution:
    def guessNumber(self, n: int) -> int:
        left ,right = 1,n
        while left <= right:
            mid = (left + right) // 2
            if guess(mid) == 1:
                left = mid + 1
            elif guess(mid) == -1:
                right = mid - 1
            else :
                return mid
```

## 5.4 LeetCode 374. 結果


Success Details ›

Runtime: 49 ms, faster than 26.49% of Python3 online submissions for Guess Number Higher or Lower.

Memory Usage: 13.9 MB, less than 66.29% of Python3 online submissions for Guess Number Higher or Lower.

Fig. 5. LeetCode 374 結果