國立臺北科技大學
National Taipei University of Technology

資料結構

Data Structure

Lab 1

姓名：曾致嘉

學號：113AB0014

Lab01-Ex1. (題目編號)

Add operation counts to this code as comments

Code

```
int findMax(const vector<int>& arr) {
    // TODO: Add count = 2 for initialization
    int max = arr[0];

    // TODO: Add 1 + 5n for loop operations
    for (int i = 1; i < arr.size(); i++) {
        //init count = 1

        // TODO: Add 3 for comparison and assignment
        if (arr[i] > max) {
        max = arr[i];
      }
    }
    // TODO: Add count = 1 for return
    return max;
}
```

Lab01-Ex2. (題目編號)
Integrate an actual count variable into the code
Code

```
void printPairs(const vector<int>& arr) {
	int count = 0;

	//for outer loop initialization
	count++;
	for (int i = 0; i < arr.size(); i++) {
		//for comparison
		count++;

		//for inner Loop initialization
		count++;
		for (int j = i + 1; j < arr.size(); j++) {
			//for comparison
			count++;

			//for Array access
			count += 2;
			cout << arr[i] << "," << arr[j] << endl;

			//for inner Loop increment
			count++;
		}

		//for outer Loop increment
		count++;
	}
}
```

Discussion Section

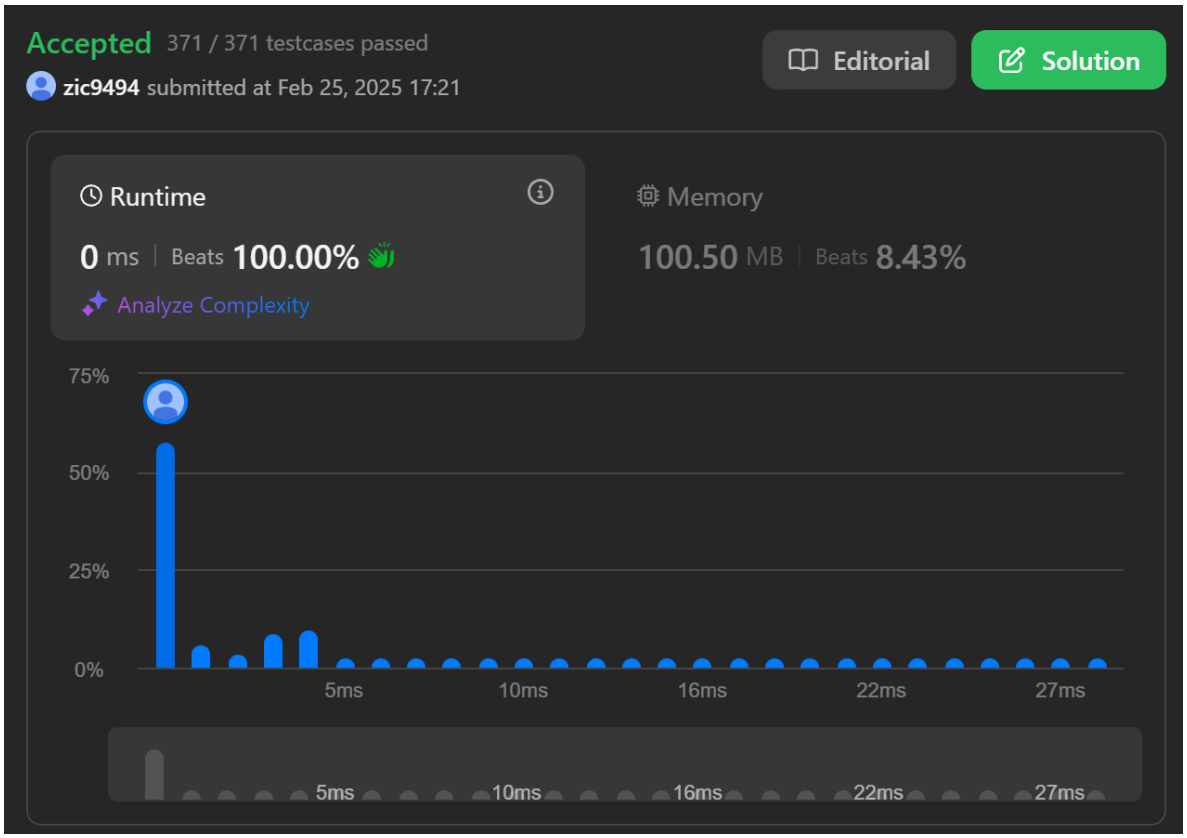
1. Why don't we count operations like arr.size()?
   Because it is constant time access.
2. Why do we drop constants in Big-O notation?
   - If n is large, the constant term becomes negligible.
   - Can help us measure the effectiveness of code.
3. Looking at the examples above:
   - sumArray() more, (5n + 3) > (4n + 3)
   - No, it's the same O(n), both grow linearly.

Lab01-Q1. 896
Check a list is or not Monotone.
Code

```
class Solution {
    public:
        bool isMonotonic(vector<int>& nums) {
            int count = 0;

            count++;
            //  排除只有一個值的情況
            if (nums.size() == 1){
                count++;
                return true;
            }

            count += 2 ;
            //判斷數列上升或下降，排除相同值的情況
            int in_decrease = 1;
            int p = 1;

            //第一次 while 判斷
            count += 3;
            while (nums[0] == nums[p]){

                //if 判斷
                count+=2;
                if ( p == (nums.size()-1) ){
                    count++;
                    return true;
                }

                //p 遞增
                count++;
                p++;
                //追加 while 判斷
                count += 3;
            }
```

```
            //if 判斷
            count+=3;
            if (nums[p]-nums[0]<0){
                count++;
                in_decrease = -1;
            }

            count++;//for 初始化
            //判斷是否為單調數列
            for(int i = 0; i < nums.size()-1; i++){
                //for 判斷
                count+=2;

                //if 判斷
                count+=6;
                if ( (in_decrease * ( nums[i+1] - nums[i])) < 0 ){
                    count++;
                    return false;
                }

                //for 遞增
                count++;
            }

            return true;
        }
};
```

Result



Discussion

3. The worse condition(all different and Descend), f(n) = 11n + 10

4. O(n)

Lab01-Q#. (題目編號)		
Briefly describe the problem. (題目簡述)		
Code		
Paste your code here. Please use Pygments to highlight your code. (請使用 Pygments 傳換程式碼格式後再貼上)	Paste your code here. Please use Pygments to highlight your code. (請使用 Pygments 傳換程式碼格式後再貼上)	
result		
(a)	(b)	(c)
	Show your result in tables or screenshots. (請使用表格或是截圖呈現結果。)	
Discussion		
Discuss and conclude your results, or answer questions. (實驗討論、結論或回答問題。)		

[EXAMPLE] Q1. Factorial Function

The factorial function n! has value 1 when n<=1 and value n*(n-1)! when n>1.

Please write both a recursive and an iterative C function. Indicate the input, output, and give proper comments. Test your codes and log the results.

Code

<iterative version>

```c
#include<stdio.h>
#include <stdlib.h>
int main()
{
int n,answer;
char i;
printf("Factorial function n!,n=");
scanf("%d",&n); //輸入要查詢的 n
if(n<=1)
{
printf("Answer=1"); //小於 1 時答案為 1
}
else //大於 1 時
{
answer=n; //計算 (n>12 時超出 int 儲存上限)
for(i=(n-1);i>1;i--)
{
answer=answer*i;
}
printf("Answer=%d\n",answer);
}
system("pause");
return 0;
}
```

<recursive version>

...

result

```
Factorial function n!,n=7
Answer=5040
```

Discussion