**7205 HW2**

**Name: Xuebao Zhao NUID: 002108354**

**Q1：**The program implements the MERGE-SORT and MERGE algorithms. It first asks for an integer n. The n is the amount of the element in the array. Then fill the array using random values. Sort the array using merge algorithms. Then print the contents of the array before and after calling sort on it.

The modify is to compare the A[p] and A[p+1] at the beginning of the function Merge. Because the two subarrays are both orderly, If A[p] which is the greatest one of the precious array is smaller than A[p+1] which is the smallest one of the other subarray, the array A is orderly.

result：

**文本

描述已自动生成**

**文本

描述已自动生成**

**Q2:**

MERGE-SORT(A, p, r)

if p < r

q = ⌊ (p + r) / 2 ⌋

MERGE-SORT(A, p, q)

MERGE-SORT(A, q + 1, r)

MERGE(A, p, q, r)

MERGE(A, p, q, r)

If A[q] ≤ A[q + 1]

exit the function

else

n1 = q - p + 1

n2 = r – q

Let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays

for i = 1 to n1

L[i] = A[p + i - 1]

for j = 1 to n2

R[j] = A[q + j]

i = 1

j = 1

for k = p to r

if i ＞ n1

A[k] = R[j++]

else if j > n2

A[k] = L[i++]

else if L[i] ≤ R[j]

A[k] = L[i]

i = i + 1

else

A[k] = R[j]

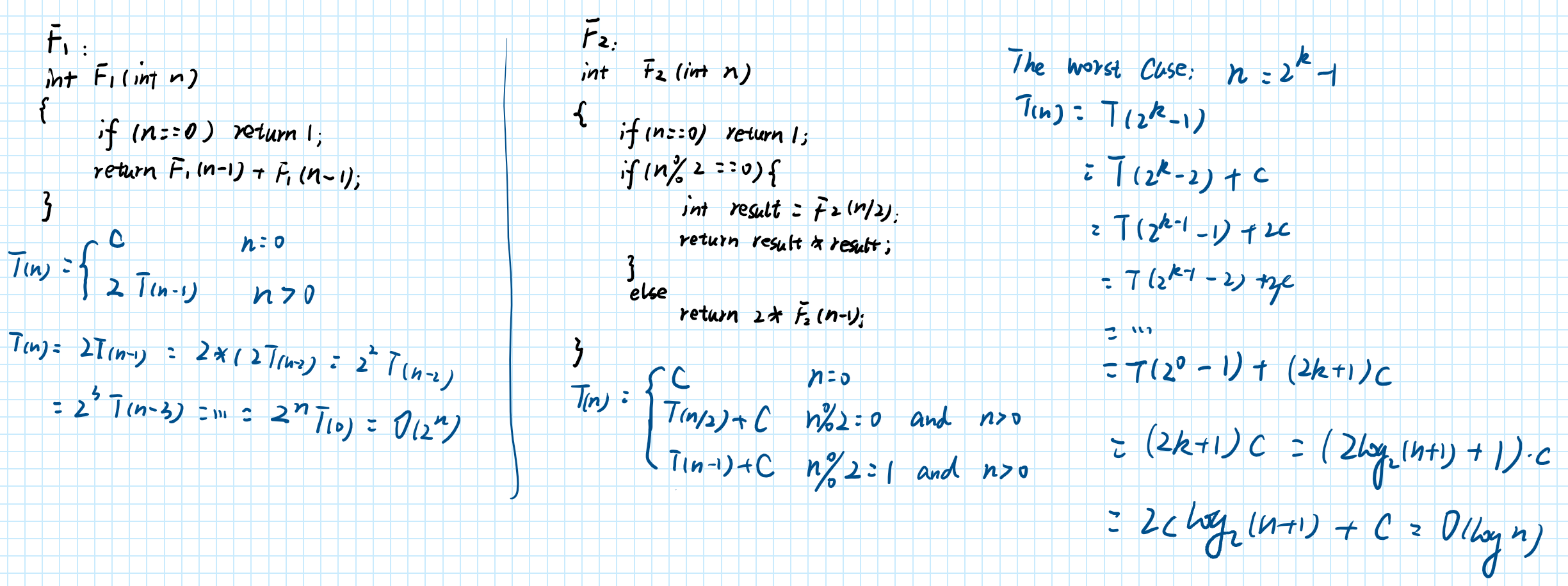
j = j + 1

**Q3:**

1. Both functions are the nth power of 2.
2. Function F2 is faster, the following image is the time of the two functions when n is 30. F1 used about 34 seconds while F2 used about 0.001 seconds.

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成



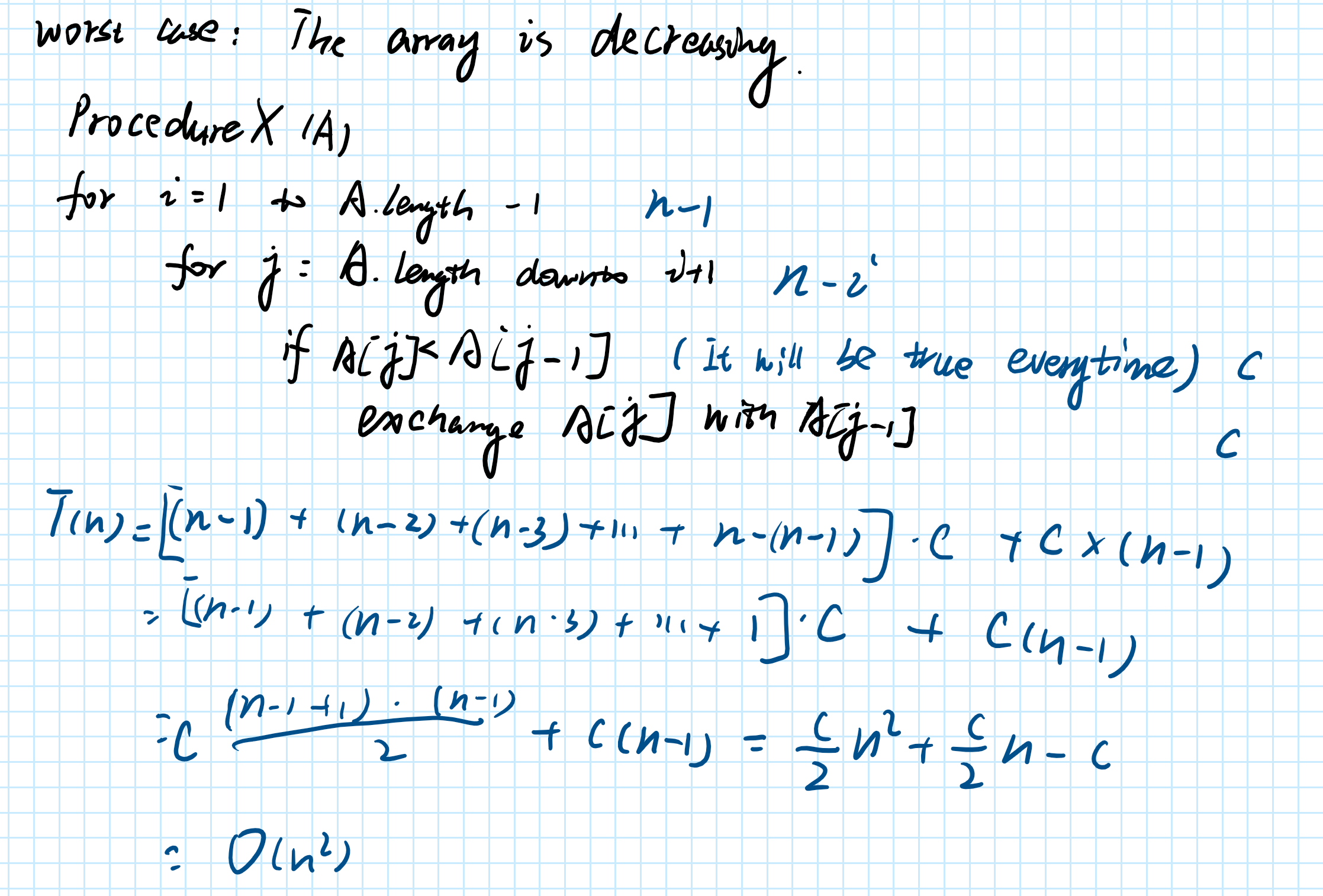
The worst case of function F2 is much better than F1. So, F2 is faster.

**Q4:**

1. Function procedureX is bubble sort.

It compares the adjacent elements from the end of the array. Exchange the adjacent elements if the previous number is greater than the latter number. So, the first element will be the smallest number. Repeat the above steps to move the second small number to the second position of the array. Repeat until the array is orderly.

b)



**Q5:**

Insertionsort(A, n)

{

if n > 1

Insertionsort(A, n-1)

REC-INSERTION(A, n)

}

