110612117 張仲瑜 演算法概論 Exercise#1 Report

1. **Environment**

   OS: windows 10, compile version: gnu++ 14, IDE: VSCode

   Compile and Run in VScode Terminal:

   g++ Exercise1_110612117.cpp -o Exercise1_110612117 ; .\Exercise1_110612117.exe

2. **Results**

   a. **Method (Core)**

      i. **Insertion**

         1. Use a vector to store the numbers to be inserted.
         2. Read and build the existing Young Tableau
         3. Replace the last (right-down) with the first number to be inserted.
         4. Maintain the tableau property by recursively exchanging the inserting number with the largest number among itself, its top neighbor and its left neighbor not until the inserting number is the largest or equals to among these three numbers.

      ii. **Extraction**

         1. Read and build the existing Young Tableau
         2. Output then replace the first (left-top) numbers with INF(x)
         3. Maintain the tableau property by recursively exchanging the first number with the smallest number among itself, its down neighbor and its right neighbor not until the INF number is the smallest or equals to among these three numbers.

   b. **Running time**

      i. **The running time of Inserting k numbers in a m*n Young Tableau is O(k(m + n))**

      The worst case happens when we always insert the smallest number to the table, and the number is compared 2 * (m + n - 2) times with its neighbor, and k numbers are inserted in total, so **O(2k(m + n - 2)) = O(k(m + n))**

      ii. **The running time of Extracting the minimum numbers in a m*n Young Tableau is O(m + n)**

      The worst case happens when the table is full, which means that we have to compare 2 * (m + n - 2) times with its neighbor to bubble the empty spot all the way to the right-down corner, so **O(2(m + n - 2)) = O(m + n)**

c. **Further discussion**

Maintain every time or Maintain in once.

If we extract all the existing number and sort it with the inserting number, and rebuild the table, the time complexity is

O(m * n) (Extract all the existing numbers) + O((m * n)lg(m * n)) (sort) + O(m * n) (put the in order number from left to right, from up to down)

= O(2 * (m * n) + lg(m * n)(m * n))

= O((m * n)lg(m * n))

This can be more efficient than O(k(m + n)) = O((m * n)(m + n)) when m,n becomes larger.

d. **Screen Shot**

input.txt

```
5
1
2147483647 123 213 324 -124 4234 9324 -2
1 3 x x
7 x x x
x x x x
x x x x

2
2 20 35 39 59 70 90 100
23 43 55 123 235 600 1000 20001

2
-100 -34 x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x

1
1 2 3 4 5 6 7 8 9 10
0 12
1 x
3 x
x x
x x
x x
x x
x x
x x
x x

1
-2147483648
1 20 39 46
2 40 213 589
90 55 895 10002
1023 2034 213123 x
```

output.txt

```
Insert 2147483647 123 213 324 -124 4234 9324 -2
-124 1 3 123
-2 213 324 2147483647
7 4234 9324 x
x x x x

Extract-min 2
20 35 39 59 70 90 100 20001
23 43 55 123 235 600 1000 x

Extract-min -100
-34 x x x x x x x x x
x x x x x x x x x x
x x x x x x x x x x

Insert 1 2 3 4 5 6 7 8 9 10
0 1
1 2
3 4
3 6
5 8
7 10
9 12
x x
x x
x x

Insert -2147483648
-2147483648 20 39 46
1 40 213 589
2 55 895 10002
90 1023 2034 213123
```