Introduction to Algorithms
Exercise #1

a. Environment

- OS：windows
- Compiler version：g++
- IDE：visual studio 2019

b. Methods or solutions

1. Header

```
1   #include<iostream>
2   #include<vector>
3   #include<fstream>
4   #include<sstream>
5   #include<algorithm>
6   #include<climits>
7
8
9   using namespace std;
```

- fstream/sstream：to read/write the file.
- iostream：standard input/output.

2. declare a constant matrix “tableau” and two files “file” and “file2”.

```
10
11    fstream file,file2;
12    int tableau[20000][20000];
13
```

3. functions

- insert

Inserts a value into the tableau at the specified position and maintains the tableau property by repeatedly swapping the current element with its adjacent elements until the tableau property is satisfied.

```
14  void insert(int m, int n, int temp)
15  {
16      int row=m-1;
17      int col=n-1;
18      tableau[row][col]=temp;
19      while(tableau[row][col]<tableau[row][col-1] || tableau[row][col]<tableau[row-1][col])
20      {
21
22          if(row==0 || col==0)
23          {
24              break;
25          }
26          //x-u>x and x-u ≥ x-l
27          if(tableau[row-1][col]<tableau[row][col-1])
28          {
29              swap(tableau[row][col],tableau[row][col-1]);
30              col--;
31          }
32          //x-l>x and x-l>x-u
33          else
34          {
35              swap(tableau[row][col],tableau[row-1][col]);
36              row--;
37          }
38      }
```

```
39      //when row=0
40      if(row==0&&col≠0)
41      {
42          while(tableau[row][col-1]>tableau[row][col])
43          {
44              swap(tableau[row][col-1],tableau[row][col]);
45              col--;
46              if(col==0)
47              {
48                  //reach to the [0][0]
49                  break;
50              }
51          }
52      }
53      //when col=0
54      if(row≠0&&col==0)
55      {
56          while(tableau[row-1][col]>tableau[row][col])
57          {
58              swap(tableau[row-1][col],tableau[row][col]);
59              row--;
60              if(row==0)
61              {
62                  break;
63              }
64          }
65      }
66  }
```

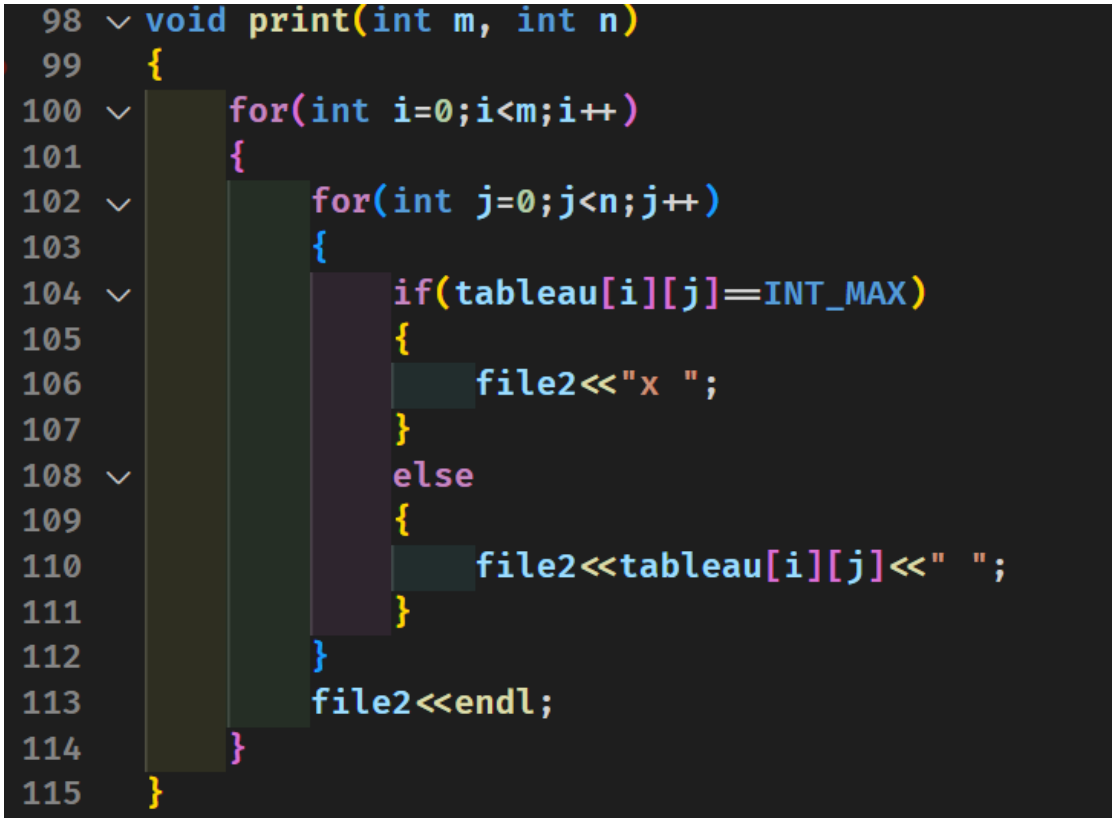
- extractMin

Extracts the minimum element from the tableau (assumes tableau is a min-heap).Recursively adjusts the tableau to maintain the min-heap property after extraction.

```
68  int extractMin(int m, int n)
69  {
70      int min=tableau[m][n];
71      if(tableau[m+1][n]==INT_MAX&&tableau[m][n+1]==INT_MAX)
72      {
73          tableau[m][n]=INT_MAX;
74          return min;
75      }
76      if(tableau[m+1][n]>tableau[m][n+1])
77      {
78          tableau[m][n]=tableau[m][n+1];
79          tableau[m][n+1]=min;
80          return extractMin(m,n+1);
81      }
82      else
83      {
84          tableau[m][n]=tableau[m+1][n];
85          tableau[m+1][n]=min;
86          return extractMin(m+1,n);
87      }
88  }
```

- Print

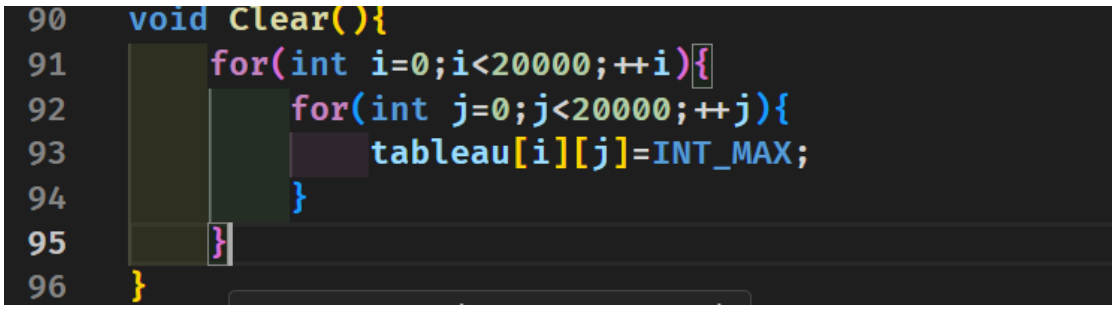
  To print the whole matrix.



```
void print(int m, int n)
{
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(tableau[i][j]==INT_MAX)
            {
                file2<<"x ";
            }
            else
            {
                file2<<tableau[i][j]<<" ";
            }
        }
        file2<<endl;
    }
}
```

- Clear

Initializes the tableau with INT_MAX values and clear the tableau before processing a new set of operations.



```
void Clear(){
    for(int i=0;i<20000;++i){
        for(int j=0;j<20000;++j){
            tableau[i][j]=INT_MAX;
        }
    }
}
```

4. Main function and File i/o

   Opens "input.txt" for reading and "output.txt" for writing.

```
117  int main()
118  {
119      file.open("input.txt",ios::in);
120      file2.open("output.txt",ios::out);
121      int num_tab;
122      file>>num_tab;
```

5. Processing the input

   - Use a while loop to read the matrices

```
123      while(num_tab--)
```

- Insert the number into the tableau.
  - Initialize the tableau matrix first.
  - Read means and numbers want to insert from the “input.txt”.
  - Transform the string a single number.
  - Record the number in the vector ”temp”.

```
125         Clear();
126         int means;
127         file>>means;
128         stringstream ss;
129         string s;
130         if(means==1)
131         {
132             getline(file,s);  /*clean buffer*/
133             getline(file,s);
134             ss<<s;//take from ss
135             int val;
136             vector<int> temp;
137             while(ss>>val)
138             {
139                 temp.push_back(val);
140             }
141             file2<<"Insert ";
142             for(int i=0;i<temp.size();++i)
143             {
144                 file2<<temp[i]<<" ";
145             }
146             file2<<endl;
147             ss.clear();
148             s.clear();
149             int m=0,n=0;
150             char ch,pre_ch;
```

- Recursively read the matrix from “input.txt”.

```
151         while(1)
152         {
153             file.get(ch);
154             if(ch≥48&&ch≤57)
155             {
156                 s+=ch;
157             }
158             if(ch=='x')
159             {
160                 n++;
161             }
162             if((ch==' '||ch=='\n')&&(pre_ch≥48&&pre_ch≤57))
163             {
164                 ss<<s;//write
165                 ss>>val;//read
166                 temp.push_back(val);
167                 s.clear();
168                 ss.clear();
169                 n++;
170             }
171             if(ch=='\n')
172             {
173                 m++;
174             }
175             if(ch=='\n'&&pre_ch=='\n')
176             {
177                 m--;
178                 break;
179             }
180             pre_ch=ch;
181         }
182         n/=m;
```

- Output the result in “output.txt”.

```
183             for(int i=0;i<temp.size();++i)
184             {
185                 insert(m,n,temp[i]);
186             }
187             print(m,n);
188             file2<<endl;
189         }
```

- Same process recursively read matrix from “input.txt”.
  - Extract_min the tableau.

```
190     else
191     {
192         getline(file,s);
193         vector<int> temp;
194         int m=0,n=0;
195         int val;
196         char ch,pre_ch;
197         while(1){
198             file.get(ch);
199             if(ch>=48&&ch<=57)
200             {
201                 s+=ch;
202             }
203             if(ch=='x')
204             {
205                 n++;
206             }
207             if((ch==' '||ch=='\n')&&(pre_ch>=48&&pre_ch<=57))
208             {
209                 ss<<s;
210                 ss>>val;
211                 temp.push_back(val);
212                 s.clear();
213                 ss.clear();
214                 n++;
215             }
216             if(ch=='\n')
217             {
218                 m++;
219             }
```

```
220             if(ch=='\n'&&pre_ch=='\n')
221             {
222                 m--;
223                 break;
224             }
225             pre_ch=ch;
226         }
227         n/=m;
228         for(int i=0;i<temp.size();++i)
229         {
230             insert(m,n,temp[i]);
231         }
232         int e=extractMin(0,0);
233         file2<<"Extract-min "<<e<<endl;
234         print(m,n);
235         file2<<endl;
236     }
237 }
```

- Close the file.

```
238     file.close();
239     return 0;
240 }
```

- Result-output.txt

```
Insert 6 7
2 3 6 7
4 8 12 14
5 9 16 x
x x x x

Insert 13
1 3 4
2 5 9
6 7 13
11 12 14

Extract-min 2
3 5 12 14
4 8 16 x
9 x x x
x x x x

Extract-min 1
2 3 5
4 7 14
6 9 x
11 12 x
```

c. Analyze the running time of your algorithm

- insert function:

The worst-case time complexity of the insertion operation is O(m + n), where m is the number of rows and n is the number of columns in the tableau.

- extractMin function:

The worst-case time complexity of the extraction operation is O(m + n), where m is the number of rows and n is the number of columns in the tableau.

- Clear function:

The time complexity of the initialization is O(m * n), where m is the number of rows and n is the number of columns in the tableau.

- print function:

The time complexity of printing is O(m * n), where m is the number of rows and n is the number of columns in the tableau.

- Main function

The main loop iterates through each test case, and for each test case, it performs either an insertion or an extraction operation. Let k be the number of test cases. The overall time complexity of the main loop is O(k * (m + n)).

In summary, the overall time complexity of the algorithm is approximately O(k * (m + n))

d. Anything you want to share

In this time, I use the way to read the file “input.txt” and write “output.txt”. Next time, I will choose the standard way to cin in value and cout the results in the terminal.