Assignment: Young tableaus, one of the famous applications of the heap properties

1. Draw 4×4 tableau containing the elements {9,16,3,2,4,8,5,14,12}

2	3	5	14
4	8	9	16
12	∞	∞	∞
∞	∞	∞	∞

2. Argue that an m×n Young tableau Y is empty if Y[1,1]=∞. Argue that Y is full (contains mn elements) if Y[m,n]<∞.

   Young Tableau has properties, elements increase from left to right and from top to bottom. Each element has two parent nodes (left and top) and two children (right and down) such that children have greater values than or equal to parents. With these properties Y[1,1] will have the smallest value as its neighboring elements Y[1,2] and Y[2,1] which are its children. Thus Y[1,1] ≤ Y[i,j] for all 1 ≤ i ≤ m and 1 ≤ j ≤ n. Since Y[1,1]=∞, all the elements in the Young Tableau should be greater than or equal to ∞ , which means all elements are ∞. As mentioned if an element has ∞ as value it is empty, therefore the Young Tableau is empty.

   Y[m,n] is the last element of the Young Tableau of size m×n. This last element will be the largest among all elements as Y[m-1, n] and Y[m,n-1] will be its parents and hence smaller than Y[m,n]. This if Y[m,n]<∞ that means all the elements in the Young Tableau are smaller than ∞, that is all the elements are non empty. Hence the Young Tableau is full.

3. Y[1,1] is the smallest element in the Young Tableau, hence we remove it and store it for returning. We replace this with an empty element that is an element with value ∞, which means Y[1,1] = ∞. This breaks the Young Tableau property and we need to perform a procedure similar to Max-Heapify to restore the properties.

   We recursively compare Y[i,j] with each of its neighbors and exchange it with the smallest. This restores the property of Y[i,j] but reduces the problem to either Y[i+1,j] or Y[i,j+1]. We terminate the recursion when Y[i,j] is smaller than or equal to its neighbors. Since Y has size m×n, with every recursion we reduce the problem to either m-1+n or m+n-1, since p=m+n, we reduce it to p-1. Thus we get the recurrence relation:

$$
\begin{aligned}
T(p) &= T(p-1) + O(1) \\
&= T(p-2)+O(1)+O(1) \\
&\ \ . \\
&\ \ . \\
&= T(p-k)+O(k) && \text{when } p-k = 1,\ k = p-1 \\
&= T(1)+O(p-1) \\
&= O(p) && \text{Since } p=m+n \\
&= O(m+n)
\end{aligned}
$$

   Algorithm for Extract_Min

```
Extract_Min(Y)
1.  minElement ← Y[1,1]
2.  Youngify (Y, 1, 1)
3.  return minElement
```

```
Youngify(Y, i, j)
1.  minRow← i
2.  minCol← j
3.  if i+1 ≤ m and Y[i,j] > Y[i+1,j]
4.      then minRow ← i+1
5.           minCol ← j
6.  if j+1 ≤ n and Y[minRow, minCol] > Y[i,j+1]
7.      then minRow ← i
8.           minCol ← j+1
9.  if minRow != i or minCol != j
10.     then swap Y[i,j] <–> Y[minRow, minCol]
11.          Youngify(Y, minRow, minCol)
```

4. The algorithm for inserting an element in the Young Tableau is similar to youngify, except that we start with the bottom right element of the tableau and move it upwards and leftwards to its correct position. The asymptotic analysis is the same, that is O(m+n). In insertion we first check if the Tableau is not full, then we insert this new value in the last cell Y [m, n], and percolating it up by the inverse of the youngify function until it is greater in value than both its parents. The inverse of the youngify function up will recursively exchange it with the greater of its two parents.
Algorithm for Insert(Y,i,j,key)

```
1.  if Y[i,j] < key
2.       then error -> the tableau is full
3.       return
4.  Y[i,j] ← key
5.  x ← i, y ← j
6.  max ← ∞
7.  while (i > 0 or j > 0) and (max > Y[i,j])
8.      swap Y[i,j] <–> Y[x,y]
9.      i ← x, j ← y
10.     if i - 1 >= 0 and Y[i,j] < Y[i-1,j]
11.          x ← i -1, y ← j
12.     if j -1 >= 0 and Y[x,y] < Y[i,j-1]
13.          x ← i, y ← j – 1
14.     max ← Y[x,y]
```

5. We can sort a given set of numbers using a Young Tableau by inserting all the elements one by one into the Young Tableau. That is we insert n^2 elements. According to the Young Tableau property the element at the [0][0] index will be the smallest. We can extract this element by using ExtractMin. This will remove the smallest element and maintain the Young Tableau properties again. This extracted minimum element can be stored at the smallest index in our sorted array. We call ExtractMin again giving the newly computed Young Tableau as input. We continue this until all elements are stored in the sorted array. Each insertion is $O(n+n)=O(n)$. The complexity is n^2 O(n) = O(n^3). ExtractingMin and storing it in the sorted array has the same complexity. In total, its O(n^3).
After calling the InsertKey fuction on each element we get a Young Tableau, which is given to sortArray as input.
SortArray(Y)

```
1.  i← 0
```

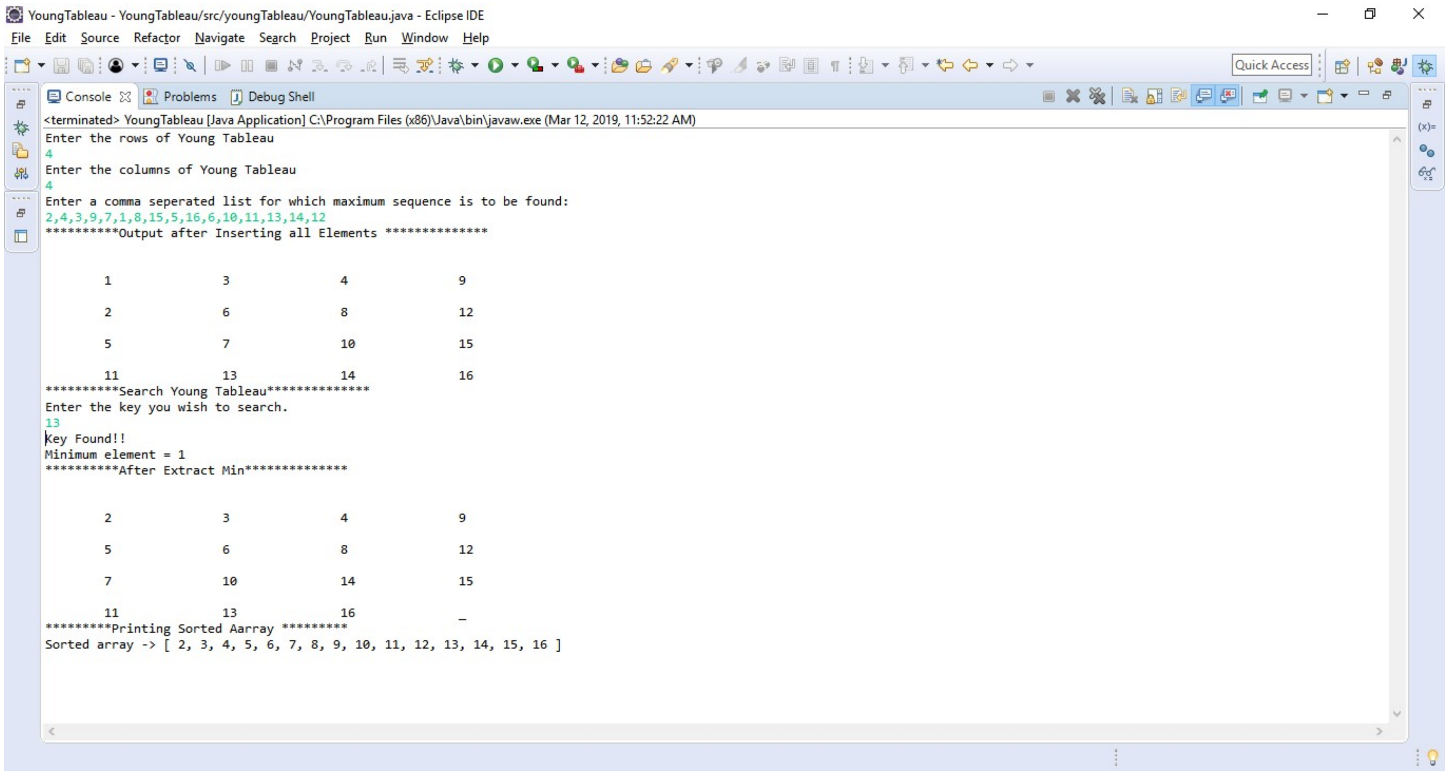
```
2.  While Y[1][1] is not ∞
3.      sortedArray[i] ← extractMin(Y[][], maxRow, maxColumn)
4.      i←i+1
```

6. To find whether a given number is present in the Young Tableau, we start with comparing the key with the upper right corner if it is less than the element we decrease the column by one, if it is greater we increase the row by one. We then compare the key with the elements at these modified indices. Repeating this until we reach the last row and first column. If on the way the key matches we return true. When we reach the end we return false.

   searchKey(Y, key, m, n)

```
1.  row←0, column ← n
2.  while row >= 0 and column < n
3.          if row>m or column<0
4.                  return false
5.          if Y[row,column] == key
6.                  return true
7.          if Y[row,column]  > key
8.                  column← column -1
9.          else row← row+1
10. return false
```

Console Output: An unsorted array is given as input. A young Tableau is created using insert key on all the elements. An element is searched. Extracting the minimum element. The tableau is used so as to get a sorted array as output.



```
<terminated> YoungTableau [Java Application] C:\Program Files (x86)\Java\bin\javaw.exe (Mar 12, 2019, 11:52:22 AM)
Enter the rows of Young Tableau
4
Enter the columns of Young Tableau
4
Enter a comma seperated list for which maximum sequence is to be found:
2,4,3,9,7,1,8,15,5,16,6,10,11,13,14,12
**********Output after Inserting all Elements **************


        1               3               4               9

        2               6               8               12

        5               7               10              15

        11              13              14              16
**********Search Young Tableau**************
Enter the key you wish to search.
13
Key Found!!
Minimum element = 1
**********After Extract Min**************


        2               3               4               9

        5               6               8               12

        7               10              14              15

        11              13              16              _
*********Printing Sorted Aarray *********
Sorted array -> [ 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 ]
```