Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Programming Assignment 1 (7.5%) CSI2110/CSI2510

Due: October 11, 11:59PM, 2025

Late assignment policy: ***1min-24hs late are accepted with 30% off; no assignments accepted after 24hs late.***

Problem Description

In this assignment, you will learn about the **partition Abstract Data Type** (also called union-find) and use it in an application. The first task will be to implement the **partition ADT** using the sequence implementation. The second task will be to solve a problem that involves keeping track of islands of land in a flood area.

1 Sequence Implementation of the partition ADT

The **partition ADT** has the following operations:

- makeCluster(E *x*): Creates a singleton cluster containing new element *x* and returns its position.
- union(*p*, *q*): Merges the clusters containing positions *p* and *q*.
- find(*p*): Returns the position of the leader of the cluster containing position *p*.

There are several ways to implement this data structure. In this assignment you must use the **sequence implementation** of the **partition ADT** described in the textbook in pages 672-673 (see these pages in appendix), which is implemented using several linked lists.

Using the **sequence implementation,** the running time of makeCluster(*x*), find(*p*), and union(*p*, *q*) and should be O(1), O(1) and O(min(n1,n2)), respectively, where n1 is the size of p's cluster and n2 is the size of q's cluster.

In addition to the usual methods above, the **partition** ADT should provide the following methods:

- element(p): returns the element that was stored at position p
- numberOfClusters(): returns the current number of different clusters
- clusterSize(p): returns the size of the cluster containing p
- clusterPositions(p): returns a list of all positions that belong to the same cluster as position p
- clusterSizes() returns a list of all the cluster sizes in decreasing order of size

The running time of the first 3 methods should be O(1), for clusterPositions(p) and clusterSizes(), the time should be O(s), where s is the size of the returned list (except that clusterSizes() may spend extra time related to sorting its returned list).

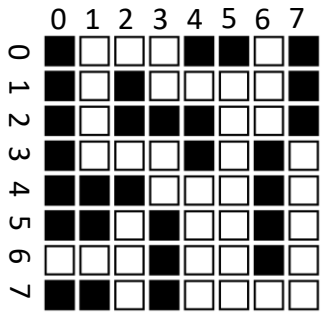
The partition must be implemented as a class using a generic element E: `class Partition<E>`
A "position" is a reference to a node that stores the element and any other necessary information, the node class should be a class using a generic element E: `class Node<E>`

2 Islands in a flood area

In this assignment, you should write a program to keep track of a region that has been subject to a flood, so the land surface has been divided into "islands" as a result of the flood.

PART 2A Surveying a changing flood landscape

You will read an initial map of size S rows and T columns, which contains points/pixels that are either 0 (white, representing water) or 1 (black, representing land). The following picture shows an example of an 8 by 8 region with 7 islands:



(credits: picture by Professor Jeff Ericson[1])

Note that two squares are part of the same island if they share one of their four sides with another square. Squares touching in a corner are not considered a land connection.

After the initial map is read and the islands are identified, you should provide a **Survey of the Land** which includes the following information:

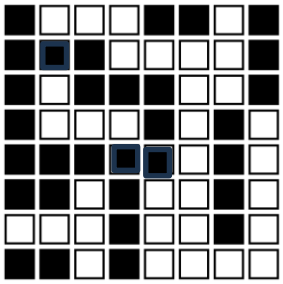
- Number of islands
- List of islands size (in decreasing order of size counted as number of black squares); if there are no islands the empty list is represented by -1
- Total area of islands (sum of the island sizes)

However, the flood level can decrease making some land reappear (i.e. some white squares become black) and effectively change the map landscape.

In each *phase* of flood recovery, we learn the coordinates of several new land positions (squares that were formerly white and became black). The map and associated data structures should be updated, and a new Survey of the Land should be printed.

[1] https://jeffe.cs.illinois.edu/teaching/algorithms/notes/11-unionfind.pdf

The following example shows the map after phase 1 where the positions (1,1), (4,3), (4,4) become land. Now there are only 5 islands of sizes 20, 4, 3, 2, 2.



The result showing the survey for the initial survey and the survey after phase 1 should be output as:
7
9 5 4 3 3 2 2
28

5
20 4 3 2 2
31

PART 2B Surveying islands with lake identification

The flood level has been slow moving and the people are adapting to their new life as residents of islands, some of which contain now several lakes. In the toy example of the previous page, you can see a lake is present in the largest island.

Some experts noticed that the data you provide in your program in part 1 was not exactly what they wanted, because you did not account for the formation of lakes as being part the islands. Well, they expect that the area for the islands with lakes must include the area of the lakes that they contain. Your new program for part B should do similar tasks as in part A, but your Survey of the Land should be updated to account for the presence of lakes and this new interpretation of the area of an island.

The **Survey of the Land** for PART 2B should include in this order:

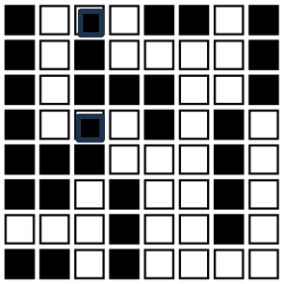
- Number of islands
- List of islands sizes (in decreasing order of size) – note that lakes inside an island should contribute to the area of the island; an empty list is represented by -1
- Total area of islands (sum of the island sizes)
- Total number of lakes (counting all lakes that exist in all islands)
- Total area of lakes

For the example in the previous page, the output would be:
7
9 5 4 3 3 2 2
28
0
0

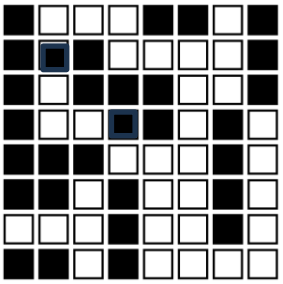
5
24 4 3 2 2
35
1
4

Note that a lake must be a set of white squares surrounded by black squares not only on the edges but also on the "corners". One island may have multiple lakes. One may have an island with water inside that does not form a lake. We show the following examples each of 6 islands and provide information about the **area of the largest island** and whether it has a lake:

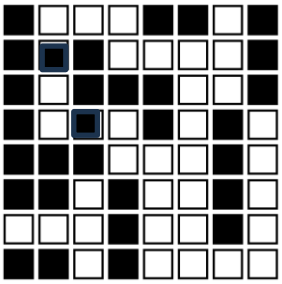
area=16, no lake



area=16, no lake



area=18 with 1 lake with area 2



3 Specification about input and output

INPUT format:
Both parts A and B will use the same format of input given in a text file as specified here. The first line contains two numbers, S and T, specifying that the map is a grid of S rows and T columns. For the following lines, each consist of a string of 0s and 1s specifying each row of the map in order. After that, there will be a sequence of zero or more phases. The first line will contain the number of phases F. Each phase is represented by lines in the following format: the first line for a phase contains the number L of new land squares, and the second line contains L pairs of coordinates for the L squares (in total 2L numbers). The coordinates come in the order i j, where 0<=i< S and 0<=j< T, representing position in the map, map[i,j], that should become a 1.

SAMPLE INPUT FILE 1(corresponds to map in the first picture)

```
8 8
10001101
10100001
10111001
10001010
11100010
11010010
00010010
11010000
0
```

SAMPLE INPUT FILE 2

```
8 8
10001101
10100001
10111001
10001010
11100010
11010010
00010010
11010000
1
3
1 1 4 3 4 4
```

SAMPLE INPUT FILE 3

```
3 4
0000
0000
0000
2
9
0 0 0 1 0 2 0 3 1 0 2 0 2 1 2 2 1 3
1
2 3
```

OUPUT format:

For both parts A and B, the output will provide the results for the various surveys separated by an empty line. The values should be given in order exactly as specified in the descriptions of the Survey of the Land for each part. Note that the surveys have different formats in parts A and B.

At the end of part 2A and part 2B, we already gave examples of outputs for an input that corresponds to **SAMPLE INPUT FILE 2.**

Here we show the output for **SAMPLE INPUT FILE 3**

SAMPLE OUPUT FILE 3 (FOR PART 2A)

```
0
-1
0

1
9
9

1
10
10
```

SAMPLE OUPUT FILE 3 (FOR PART 2B)

```
0
-1
0

1
9
9
0
0

1
12
12
1
2
```

4 Algorithms

Algorithms for part 2A

You must keep track of the islands by using the Partition ADT.

A sketch of the main steps for creating the initial partition in islands (before printing the first survey) are given next:

```
Create a S by T auxiliary array cluster to keep track of cluster
positions with all entries initialized to null
Create BP a new object of the class Partition
for each black grid point i,j:
     p = BP.makeCluster(info(i,j));
     cluster[i,j]=p

for each black grid point i,j:
     for each black grid point k,l adjacent to i,j:
          if  BP.find(cluster[i,j])!= BP.find(cluster[k,l]) then
                         BP.union(cluster[i,j],cluster[k,l])
```

Any subsequent phase will add a list of new black positions using a similar method. The new list of clusters can be used to update the islands as follows:

```
for each point i,j in the new list
     p = BP.makeCluster(info(i,j));
     cluster[i,j]=p
     change grid point i,j to black

for each point i,j in the new list
     for each black grid point k,l adjacent to i,j:
          if  BP.find(cluster[i,j])!= BP.find(cluster[k,l]) then
              BP.union(cluster[i,j],cluster[k,l])
```

The pieces of information that need to be printed for the Survey of the Land can be obtained by invoking specific methods of the class partition, provided that the element info(i,j) stores relevant info like i and j.

Algorithms for part 2B

The same algorithms as in part 2A must be used to keep track of the current partition BP of black points into islands.
You need to devise your own algorithm that, after BP has been created, it can identify the white components/clusters and decide whether each of them forms a lake and for which island, accounting for the size change that must be accounted for the corresponding island.

Hint: One option is to create a new object WP of the class Partition to keep track of the white components/clusters, but change the concept of cluster **for white points** to put in the same cluster white points that only touch at a corner. Then for each cluster of white points, check that their sides do not touch more than one island nor an outside edge of the map; if true then this is a lake of the unique island that is touched by some of its points.

5 Assignment Specifications and Mark Breakdown (60 marks)

The input MUST be read from the **standard input** and the output should be sent to the **standard output**. For reading from an input file and writing to an output file, we will redirect the input and output via the command line as shown below. If you don't know how this works, ask a TA or the professor.

Your code will be tested by us via the command line Part 2A as follows

```
javac Sequence.java Node.java IslandSurvey.java
java IslandSurvey < map1.txt > map1Output.txt
```

And for Part 2B as follows:

```
javac Sequence.java Node.java IslandLakeSurvey.java
java IslandLakeSurvey < map1.txt > map1Output.txt
```

You must hand in the following files without any subdirectories

1 all classes needed to run your code (*.java), namely:
`Partition.java` (as described is section 1)
`Node.java` (as described is section 1)
`IslandSurvey.java` (class that implements part2A; it contains a main method)
`IslandLakeSurvey.java` (class that implements part2B; it contains a main method)

2 two sample inputs that you design to verify the correctness of your code

3 A short report in pdf format with name: `report.pdf` (1-3 pages)
a short report containing the following sections:
1) Status of Code: explaining the status of the submitted code (working, partially working, known bugs, and anything that can help marking)
2) Testing: describe your testing strategy with your chosen outputs and results
3) Program
Include any relevant information about your Partition class implementation, implementation in Part 2A and implementation in Part2B
4) Resources
Give reference to all resources used to complete your assignment.

4 Include all input files provided by the professor and the output produced by your program adding the suffix "Output" to the file name. If an input file is called `map1.txt` the output file should be saved in `map1Output.txt`
In addition, include the outputs for your two sample inputs.

Marking Scheme (out of 60 points) - **efficiency counts!**

- **PART 1 Sequence implementation of the Partition ADT (20 points)** **efficiency counts!**
  makeCluster - 1 point
  union - 5 points
  find - 4 points
  element(p) - 1 point
  numberOfClusters() - 1point
  clusterSize(p) - 1 point
  clusterPositions(p) - 2 point
  clusterSizes() - 4 points
  general (other parts) - 1 point
  If the complexity of operations has a larger big_Oh than specified, discounts can be applied up to 10 points.
  If you use a different implementation of the Partition ADT which is not the sequence implementation, no marks will be given, but you will be marked normally for part 2A and 2B.
- **PART 2A Surveying a changing flood landscape (15 points)** **efficiency counts!**
  Correct test output: 5 points (errors detected can lead to more discounts)
  10 points will be distributed for correctness of specific parts, and for clarity and efficiency
- **PART 2B Surveying islands with lake identification (15 points)** **efficiency counts!**
  Correct test output: 5 points (errors detected can lead to more discounts)
  10 points will be distributed for correctness of specific parts, and for clarity and efficiency
- **General for all parts: documentation (10 points)**
  report
  quality of code and clear comments added throughout the code to explain its parts and specific commands
- **Discounts if specifications are not followed:** (up to -15 points if specifications are not followed even if the program is correct)
  - Input and output are done using the standard I/O; command line with redirection works
  - The format of the output is exactly as specified (this means an automatic program can test the correctness of your output without adjustments)
  - Java classes are 4 java classes with exact names as specified
  - Two sample inputs have been included
  - Provided input files and their output have been included, or a justification in the report that certain input files produce error or wrong output.
  - Report has been included in pdf format with name specified.

Appendix:

See attached textbook pages.