Indian Statistical Institute

Semester-I 2024–2025

M.Tech.(CS) - First Year

Lab Test 2 (25 October, 2024)

Subject: Computing Laboratory

Total: 60 marks Duration: 4 hrs.

INSTRUCTIONS

1. You may consult or use slides / programs provided to you as course material, or programs that you have written yourself as part of classwork / homework for this course, but please **do not** consult or use material from other Internet sources, your classmates, or anyone else.
2. Unless otherwise specified, all programs should take the required inputs from stdin, and print the desired outputs to stdout. Please make sure that your programs adhere strictly to the specified input and output format. **Your program may not pass the test cases provided, if your program violates the input and output requirements.**
3. Submissions from different students having significant match will **not be evaluated**.
4. To avoid mismatches between your output and the provided output, please store all floating point numbers in `double` type variables.

Part I

Consider a database R of film ratings. The database is stored in the form of an $m \times n$ matrix, with each row representing a viewer, and each column representing a film. For $0 \leq i < m$, $0 \leq j < n$, $R[i][j]$ stores the numerical rating assigned by viewer i to film j.

A viewer assigns a higher numerical value to film j_1 than to film j_2 if she likes film j_1 more than film j_2. However, **not all viewers use the same numerical scale.** For example, one viewer may give ratings from $[0, 5]$, while another viewer's ratings are from $[1, 10]$, and yet another viewer's ratings are from $[-100, 100]$.

Example: Consider the following running example, where $m = 2$ and $n = 3$. The ratings given by the 2 viewers for the 3 films are as follows.

Film	0	1	2
Viewer 0	100	0	-100
Viewer 1	5	2	4

Q1. (10 marks)

Note that, by assigning ratings, each viewer implicitly ranks the films from 1 to n (with rank 1 being assigned to the viewer's favourite film). Write a program that takes R as input, and prints ρ, the corresponding matrix of viewer's ranks ($\rho[i][j]$ should contain the **rank** assigned by viewer i to film j). You may assume for simplicity that a viewer assigns **distinct** ratings to all films. Thus, there will be no ties in the ranks assigned by a viewer.

Input format: The input will consist of the integers m and n, followed by $m \times n$ floating point numbers corresponding to the entries of R in row-major order.

Output format: Your output should contain one line per viewer; the ranks assigned by the viewer should be separated by a single space.

Example: For our running example, the viewers have **ranked** the films as follows.

Film	0	1	2
Viewer 0	1	2	3
Viewer 1	1	3	2

Q2. (10 marks)

The *overall rank* of each film is determined as follows.

1. Each film gets an *overall score* equal to the sum of the ranks assigned by the m viewers to that film.
2. The films are then ranked in *increasing* order of the overall score.
3. The example below shows that there may be **ties** in this ranking. Suppose k films get the same score. Let their ranks be $i, i+1, \ldots, i+k-1$, when ties are broken arbitrarily. Then, you should assign the **average** of these ranks $(i, i+1, \ldots, i+k-1)$ to each of the k films that have the same score (as shown in the example given below).

Write a program to compute the overall rank of each film using the above method.

Input format: As above.

Output format: Your output should consist of a single line, containing n ranks, separated by spaces. Observe that the ranks will be of the form l or $l + 0.5$, where l is a positive integer. For uniformity, print each rank as $l.0$ or $l.5$.

Example: The films' overall scores in our example are 2, 5, and 5, resp., so their final ranks should be printed as `1.0 2.5 2.5`.

Q3. (5 marks)

For each viewer, we define a quantity called *Mean Absolute Rank Error* (abbreviated *MARE*), that is intended to indicate how different the viewer's opinions are from the overall opinions about the given set of films. Let r_j denote the overall rank of film j as calculated above, and let $r_j^{(i)}$ be the rank assigned to film j by viewer i. Then, $MARE(i)$ is defined as

$$MARE(i) \triangleq \frac{1}{n} \sum_{j=0}^{n} |r_j^{(i)} - r_j|$$

Write a program to compute and print the *MARE* for each viewer.

Input format: As above.

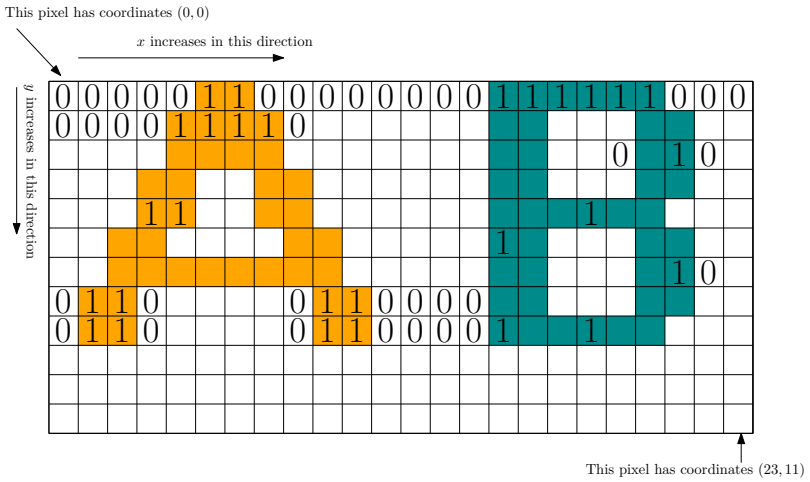
Output format: Your output should consist of a single line, containing m floating point values, correct to 6 decimal places, separated by spaces.

Example: It is easy to see that the *MARE* for both viewers in our example is $(0+0.5+0.5)/3 = 0.333333$.

Part II

Q4. (20 marks)

You are given an $m \times n$ matrix P representing an image. Each element of P corresponds to a *pixel*, and has the value 0 (if it represents a background pixel) or 1 (if it represents a foreground pixel). For example, the figure above shows a 12×24 matrix; each 0 pixel represents the white background, each 1 pixel represents the coloured foreground.[1] The foreground pixels together make up the two letters A and B.



Definition 1. A pixel $p = P[i][j]$ istb[2] a *neighbour* of another pixel $p' = P[i'][j']$ if $|i - i'| \leq 1$ **and** $|j - j'| \leq 1$, i.e., if they are adjacent to each other in either the horizontal or vertical or diagonal direction. Other than the pixels at the edge of the image, each pixel has 8 neighbours.

Definition 2. A pixel p' istb *reachable* from a pixel p if at least one of the following conditions holds: (a) $p' = p$; (b) p' is a neighbour of p; (c) there exists a pixel p'' such that p'' is a neighbour of p, and p' is reachable from p''. Reachability is thus an equivalence relation.

Definition 3. The *connected component* of a pixel p is defined as the set of all pixels reachable from p, i.e., it is the equivalence class of p under the reachability relation.

Write a program to identify the connected components in a given image / matrix P.

While writing your program, you may find ideas from the following function useful.

```
static void print_image(PIXEL **im, unsigned int rows, unsigned int cols) {
    for (unsigned int i = 0; i < rows; i++) {
        for (unsigned int j = 0; j < cols; j++)
            fputc(im[i][j].value ? '@' : '.', stderr);
        fputc('\n', stderr);
    }
    return;
}
```

Input format: The input will consist of the integers m and n, followed by the $m \times n$ entries of P in row-major order, separated by one or more spaces. You may assume that $m \times n$ is small enough that P can be stored in memory.

[1] The figure displays the value of only some selected pixels.

[2] is said to be

Output format: For each connected component present in the image, your program should print 1 line, containing 3 space-separated integers x, y, n, where x and y correspond to the coordinates of the left-most pixel of the top-most row of the component[3], and n is the number of pixels in the component.

Test case: Test case 0 corresponds to the figure in the example above. Your output for this case should be

```
5 0 38
15 0 42.
```

Q5. (15 marks)

Write a program that finds the binary trees present in a "forest" (a collection of binary trees), computes their preorder traversals, and the number of nodes in each tree that have less than two children.

Input format: The input will consist of a positive integer N, followed by N lines. Each of these lines will correspond to a node of a binary tree, and will consist of 2 integers: the index of its left child, and the index of its right child of the node[4]. If a node does not have a left/right child, the corresponding index will have a value of -1. Valid indices have values from $\{0, 1, 2, \ldots, N-1\}$.

In general, the N nodes will **NOT** belong to a single binary tree.

Output format: Your program should find the binary trees present in the given forest. If these trees are denoted as T_0, T_1, ..., T_{n-1}, then your output should print one line for each T_i. The line should consist of the indices of the nodes of T_i printed in preorder, followed by the number of nodes in T_i that have less than 2 children.

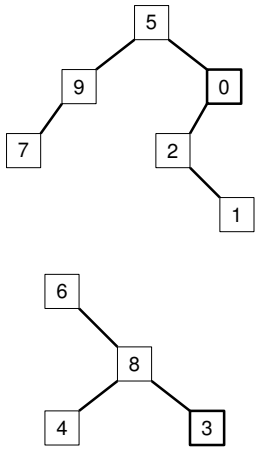
NOTE: For any T_i and T_j $(0 \leq i, j \leq n-1)$, if the index of the root node of T_i is less than that of the root node of T_j, then the output for T_i should appear before the output for T_j.

[3]This is the first pixel of the component to be reached if the matrix elements are traversed in row-major order.

[4]The data field is absent. Thus, this format is similar, but *not identical*, to the format that is accepted by the function `read_tree()` discussed in class.

Sample input 1:

0	2	-1
1	-1	-1
2	-1	1
3	-1	-1
4	-1	-1
5	9	0
6	-1	8
7	-1	-1
8	4	3
9	7	-1



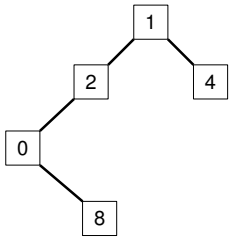
Sample output 1:

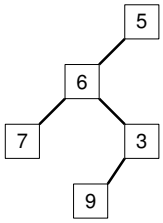
5 9 7 0 2 1 5

6 8 4 3 3

Sample input 2:

0	-1	8
1	2	4
2	0	-1
3	9	-1
4	-1	-1
5	6	-1
6	7	3
7	-1	-1
8	-1	-1
9	-1	-1





Sample output 1:

1 2 0 8 4 4

5 6 7 3 9 4

Explanation 1: The black digits in the output constitute the preorder traversal of the tree. The last red number is the number of nodes with less than 2 children.

Note that the indices are only shown for your convenience. They will not be part of the input.