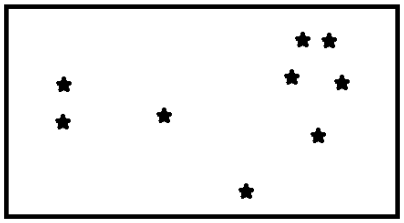
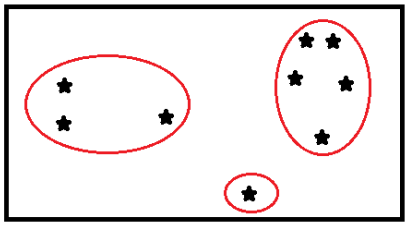
CS 2001 DATA STRUCTURES
ASSIGNMENT 3 – MINI-PROJECT
Fall 2025

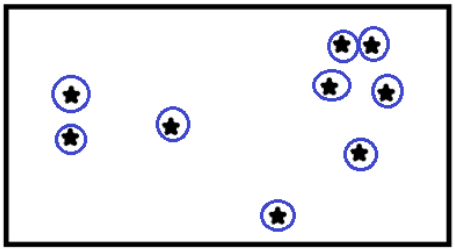
DUE: 28th Nov 2025
SUBMIT: Commented and well-written, structured code in C++ should be submitted to the classroom. Undocumented code will be assigned a zero. The name of your file should be your NUCES roll number.

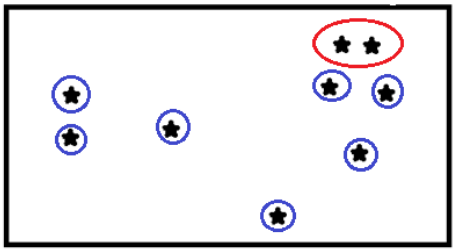
PROBLEM
We are designing a visual tool and want to group objects on the screen based on the distance between two objects. The input would be a text file containing (x,y) co-ordinates of objects and the number of groups to form.

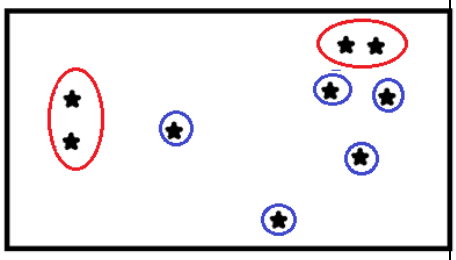
Input: N = 9 , M=3, (1,1) (1,1.5) (2.5,1.1) (3.8,0.1) (4.7,0.9) (4.5,1.6) (5,1.4) (4.6,2) (4.8,2)	***Visually (For understanding)*** Input 	***Visually (For understanding)*** Output 

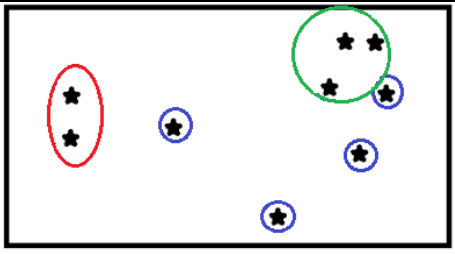
We take the number of groups to form, ***M***, as input from the user. We assume that initially each point lies in an individual group. We join the points into larger groups until we are left with ***M*** groups. We merge the groups on the base of the distance between them. *We merge the groups that are closest to one another*. ***More precisely, to join the groups, we find the pairwise distance between all the groups and merge the ones with the minimum distance between them.***

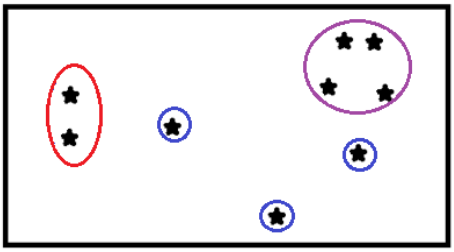
This is a visual representation of the problem (for understanding). The number of points N =9 and M =3

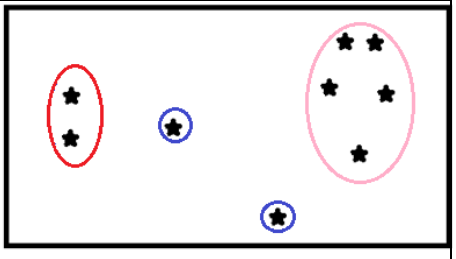

All points are in individual group initially, we have N groups

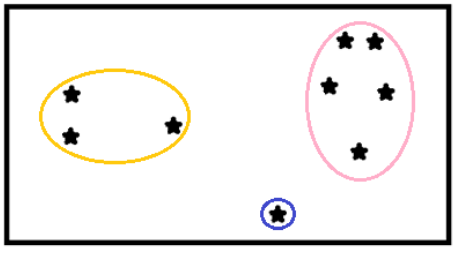

Combine two groups that have minimum distance between them


Now M =7
Continue joining groups


Continue joining groups M=6


Now M =5


Now M =4


M = 3, STOP. We merge groups with the minimum distance.

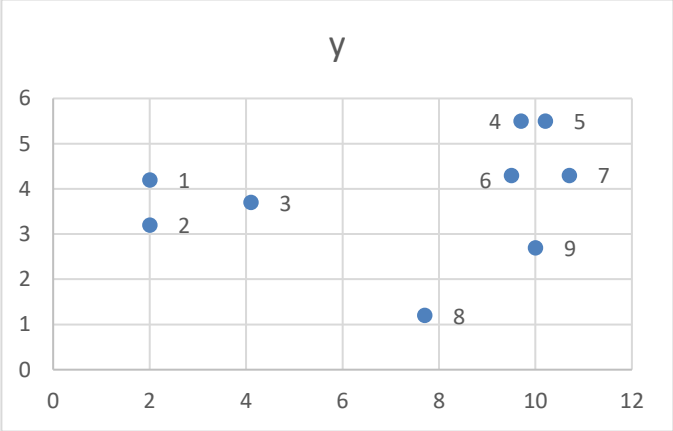
QUESTION 1: Find groups using the following BASIC Algorithm FOR GROUPING

Algorithm1 for grouping points:

1. Input N (the number of points) and M (the number of required groups) from the input file.
2. Read two-dimensional points from a file. Assume each point forms an individual group. Let the groups be numbered from 1 to N.
3. Compute the distance between each pair of points. Save these pairwise distances in a 2D Matrix of size NxN and call it the distance matrix.
4. Repeat till you are left with M groups of points.
    a. Find the minimum distance in the Distance Matrix. Find the groups that correspond to this minimum distance, let's call them Gp and Gq
        i. Merge Gp and Gq to form a Gpq. To merge two groups, you will have to update the 2D distance matrix (as described below)

What is the running time of the above algorithm?
Note that the basic idea is a bit inefficient, and later we will add a heap to make it efficient.

Consider the following example to understand the above basic algorithm for grouping points.

Input file	**The visual representation of the input file (just for your understanding)**
9 #3 2 4.2 2 3.2 4.1 3.7 9.7 5.5 10.2 5.5 9.5 4.3 10.7 4.3 7.7 1.2 10 2.7	

Initial 2D Distance Matrix of size 9x9 created using points in Input file

	1	2	3	4	5	6	7	8	9
1	0.0	1.0	2.2	7.8	8.3	7.5	8.7	6.4	8.1
2	1.0	0.0	2.2	8.0	8.5	7.6	8.8	6.0	8.0
3	2.2	2.2	0.0	5.9	6.4	5.4	6.6	4.4	6.0
4	7.8	8.0	5.9	0.0	0.5	1.2	1.6	4.7	2.8
5	8.3	8.5	6.4	0.5	0.0	1.4	1.3	5.0	2.8
6	7.5	7.6	5.4	1.2	1.4	0.0	1.2	3.6	1.7
7	8.7	8.8	6.6	1.6	1.3	1.2	0.0	4.3	1.7
8	6.4	6.0	4.4	4.7	5.0	3.6	4.3	0.0	2.7
9	8.1	8.0	6.0	2.8	2.8	1.7	1.7	2.7	0.0

How to compute the distance?

The distance between two points is computed using the Euclidean distance formula

For example:
Distance between P1 (2, 4.2) & P2 (2,3.2)
$= ((2-2)^2 + (4.2-3.2)^2)^{1/2} = 1.0$

Notice that the upper and lower triangular parts of the matrix are the same. So, we will only consider the upper triangular part.

Initial Metrix

	1	2	3	4	5	6	7	8	9
1	0.0	1.0	2.2	7.8	8.3	7.5	8.7	6.4	8.1
2		0.0	2.2	8.0	8.5	7.6	8.8	6.0	8.0
3			0.0	5.9	6.4	5.4	6.6	4.4	6.0
4				0.0	**0.5**	1.2	1.6	4.7	2.8
5					0.0	1.4	1.3	5.0	2.8
6						0.0	1.2	3.6	1.7
7							0.0	4.3	1.7
8								0.0	2.7
9									0.0

Find the minimum in the above Matrix. It will take $O(n^2)$.

The min distance = 0.5
It is between P4 and P5
Merge P4 and P5 to form a group G4,5. And update the distance matrix

Iteration 1

	1	2	3	4,5	6	7	8	9
1	0.0	1.0	2.2		7.5	8.7	6.4	8.1
2		0.0	2.2		7.6	8.8	6.0	8.0
3			0.0		5.4	6.6	4.4	6.0
4,5				0.0				
6					0.0	1.2	3.6	1.7
7						0.0	4.3	1.7
8							0.0	2.7
9								0.0

The values in the green cell need to be recalculated.
Matrix[1][4,5] = minimum(distance(1,4) , distance(1,5))
Matrix[1][4,5] = minimum(7.8, 8.3) = 7.8

Matrix[2][4,5] = minimum(distance(2,4) , distance(2,5))
Matrix[1][4,5] = minimum(8.0, 8.5) = 8.0

	1	2	3	4,5	6	7	8	9
1	0.0	**1.0**	2.2	7.8	7.5	8.7	6.4	8.1
2		0.0	2.2	8.0	7.6	8.8	6.0	8.0
3			0.0	5.9	5.4	6.6	4.4	6.0
4,5				0.0	1.2	1.3	4.7	2.8
6					0.0	1.2	3.6	1.7
7						0.0	4.3	1.7
8							0.0	2.7
9								0.0

Iteration 2
The minimum = 1.0 between P1 and P2. The distance matrix after the merge.

	1,2	3	4,5	6	7	8	9
1,2	0.0	2.2	7.8	7.5	8.7	6	8
3		0.0	5.9	5.4	6.6	4.4	6.0
4,5			0.0	**1.2**	1.3	4.7	2.8
6				0.0	1.2	3.6	1.7
7					0.0	4.3	1.7
8						0.0	2.7
9							0.0

Iteration 3
After merging G4,5 and G6.

	1,2	3	4,5,6	7	8	9
1,2	0.0	2.2	7.5	8.7	6	8
3		0.0	5.4	6.6	4.4	6.0
4,5,6			0.0	1.2	3.6	1.7
7				0.0	4.3	1.7
8					0.0	2.7
9						0.0

Iteration 4
After merging G4,5,6 and G7.

	1,2	3	4,5,6,7	8	9
1,2	0.0	2.2	7.5	6	8
3		0.0	5.4	4.4	6.0
4,5,6,7			0.0	3.6	1.7
8				0.0	2.7
9					0.0

Iteration 5
Now G4,5,6,7 and G9 has minimum distance so merge them and update matrix.

	1,2	3	4,5,6,7,9	8
1,2	0.0	2.2	7.5	6
3		0.0	5.4	4.4
4,5,6,7,9			0.0	2.7
8				0.0

Iteration 6
After merging G1,2 and G3.

	1,2,3	4,5,6,7,9	8
1,2,3	0.0	5.4	4.4
4,5,6,7,9		0.0	2.7
8			0.0

Now we have three groups and M=3, so we will stop here.

Output

The output is the points in each group

Group 1: (2,4.2), (2,3.2), (4.1,3.7)
Group 2: (9.7,5.5), (10.2,5.5), (9.5,4.3), (10.7,4.3), (10,2.7)
Group 3: (7.7,1.2)

To keep track of the points in each group, we can maintain a vector of linked lists. Where each list represents a group. When two groups are merged, we can merge the corresponding linked list in O(1) time.

QUESTION 2:

The matrix-based implementation of the above algorithm is not efficient. In this question, you will provide the efficient implementation of the above idea using HEAPS.

Precisely, you will implement a program that inputs N points and groups them into M groups based on the minimum distance between them. You will use HEAPS to efficiently find the minimum.

INPUT

The input file will have **N, the number of points, and M, the number of required groups. From the second line onwards, the input file will consist of** x, y coordinates of points (separated by tab) where each point is the location of a star in a 2-dimensional plane.

Input file
3 , 9 3 4.5 9.6 2.6 …

Pseudocode of the Efficient Algorithm for grouping points (USES Heap):

1. Input N (the number of points) and M (the number of required groups) from the input file.
2. Read two-dimensional points from a file. Assume each point forms an individual group. Let the groups are numbered from 1 to N.
3. Compute the distance between each pair of points. Create a MINHEAP of N(N-1)/2 pairwise distances in $O(N^2)$.
4. Repeat till you are left with M groups of points.
    a. Extract the Minimum distance from the above Min-Heap. Also extract the groups that correspond to this minimum distance; let's call them Gp and Gq.
    b. Merge Gp and Gq to form a Gpq. To merge two groups, you will have to update the distances in the Minheap. For this we maintain a matrix to track where the distances between different groups exist in the heap.

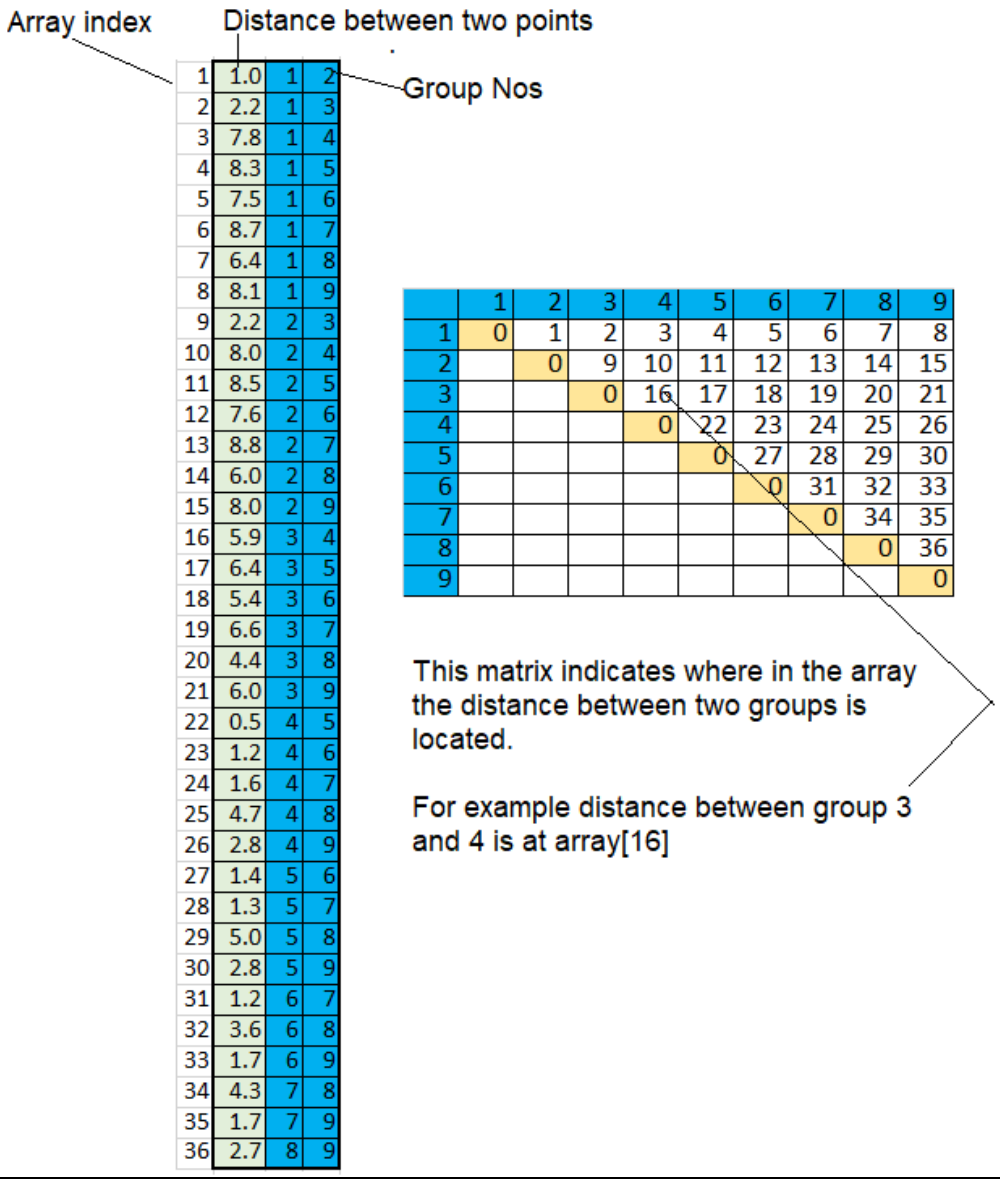
REQUIRED OUTPUT

The required output is the points in each group

Group 1: (2,4.2), (2,3.2), (4.1,3.7)
Group 2: (9.7,5.5), (10.2,5.5), (9.5,4.3), (10.7,4.3), (10,2.7)
Group 3: (7.7,1.2)

INITIALIZATION FOR THE ABOVE INPUT FILE:

Compute N(N-1)/2 pairwise distances. For N points, there will be 36 pairwise distances.



	Distance between two points	Group Nos	
1	1.0	1	2
2	2.2	1	3
3	7.8	1	4
4	8.3	1	5
5	7.5	1	6
6	8.7	1	7
7	6.4	1	8
8	8.1	1	9
9	2.2	2	3
10	8.0	2	4
11	8.5	2	5
12	7.6	2	6
13	8.8	2	7
14	6.0	2	8
15	8.0	2	9
16	5.9	3	4
17	6.4	3	5
18	5.4	3	6
19	6.6	3	7
20	4.4	3	8
21	6.0	3	9
22	0.5	4	5
23	1.2	4	6
24	1.6	4	7
25	4.7	4	8
26	2.8	4	9
27	1.4	5	6
28	1.3	5	7
29	5.0	5	8
30	2.8	5	9
31	1.2	6	7
32	3.6	6	8
33	1.7	6	9
34	4.3	7	8
35	1.7	7	9
36	2.7	8	9

	1	2	3	4	5	6	7	8	9
1	0	1	2	3	4	5	6	7	8
2		0	9	10	11	12	13	14	15
3			0	16	17	18	19	20	21
4				0	22	23	24	25	26
5					0	27	28	29	30
6						0	31	32	33
7							0	34	35
8								0	36
9									0

After the 3rd iteration of the above algorithm. We extract the minimum from the heap. We get 4 and 7, we look up 4 and 7 in the vector of linked list and merge G4, 5, 6 with G7. The heap, matrix, and vector of the linked list would be as follows:

HEAP

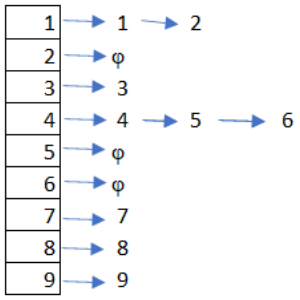
Index	Dist	GroupNos	
1	1.2	4	7
2	1.7	4	9
3	5.4	3	4
4	1.7	7	9
5	2.2	1	3
6	6	1	8
7	6	3	9
8	2.7	8	9
9	4.3	7	8
10	3.6	4	8
11	4.4	3	8
12	6.6	3	7
13	7.5	1	4
14	8	1	9
15	8.7	1	7

1.2 is the distance between Group 4 and Group 7

	1	2	3	4	5	6	7	8	9
1	0		5	13			15	6	14
2		0							
3			0	3			12	11	7
4				0			1	10	2
5					0				
6						0			
7							0	9	4
8								0	8
9									0

The distance between group 4 & 7 is at array[1]

A vector of linked list
It keep track of the points in each group. If groups are merged then correspndng lists in this vector are also merged. If an index points to null this indicates that point has been merged with another.



Note: It is not efficient to shrink a 2D matrix, so we didn't shrink the matrix. The rows and columns in grey are no longer needed and will not be accessed.

REQUIREMENTs:

1. **Implement the heap class using arrays. You cannot use the STL heap class. The heap should hold the distances between groups of points along with the group Nos.** In the above example, the root of the heap tells that the minimum distance is 1.2 and it is between group G4 and G7. Implement modified heap functions. When two groups are merged. The new distances are computed between the new merged group and other existing groups. The distances in the heap should be updated accordingly. The matrix will help to locate the distances (that need to be changed) in the heap array.

2. To keep track of the points in each group, use a **vector of linked lists**; where each list represents a group. When two groups are merged, we can merge the corresponding linked lists in O (1) time. Use the built-in STL linked list or vector. You cannot use your own code of linked list.

3. **Ensure your algorithm runs in $O(n^2 \log n)$ time.**

VERY IMPORTANT

- Academic integrity is expected of all the students. Plagiarism or cheating in any assessment will result in negative marking or an **F** grade in the course, and possibly more severe penalties.