CS 218 DATA STRUCTURES
ASSIGNMENT 4
SECTION A, B, C and D
Fall 2019

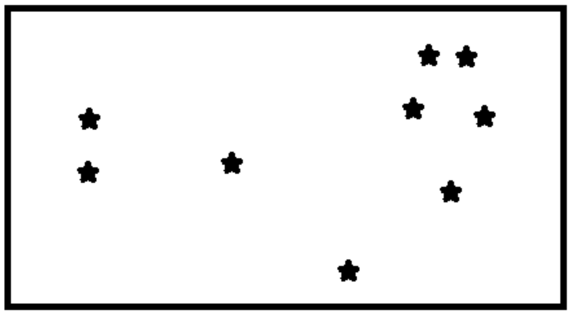
DUE: 20th November 2019 (Late submissions will not be accepted)

SUBMIT: Commented and well written structured code in C++ should be submitted on classroom. Undocumented code will be assigned a zero. The name of your file should be your NUCES roll number.

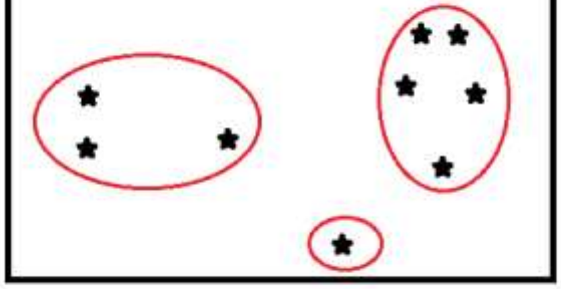
PROBLEM BACKGROUND

We are designing a software for astronomers. We have information regarding some stars scattered in space. We wish to find star assemblies; we can do so by grouping the stars.

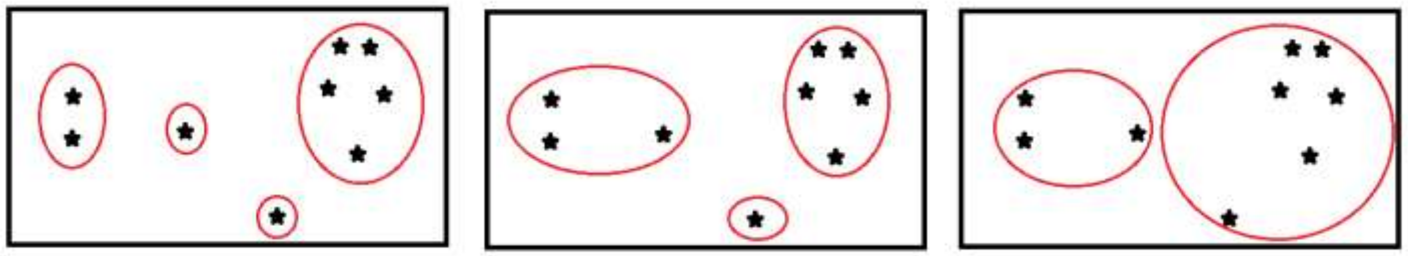
How can we group the stars? Now, this is a bit complicated and confusing task. We have mapped the information about some important stars to a two-dimensional plane.



We decided to group the stars based on the distance between them. The stars which are close to one another should be in one group.



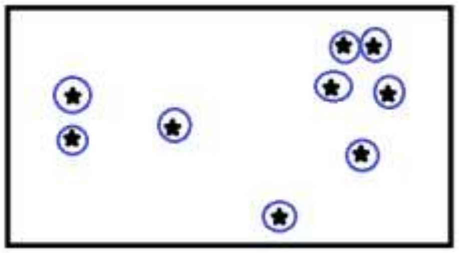
We don't know how many groups to form.



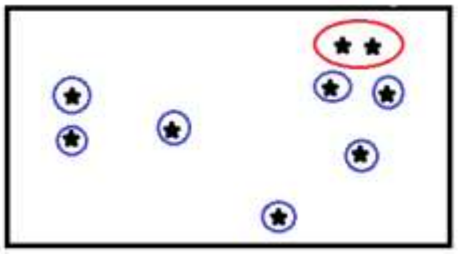
How many groups to form?
4 ? 3 ? 2 ?

To remedy this problem, we take number of groups, ***M***, as input from the user. We assume that initially each star lies in an individual group. We join the groups of stars to form larger groups till 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 pairwise distance between all the groups and merge the ones with the minimum distance between them.***

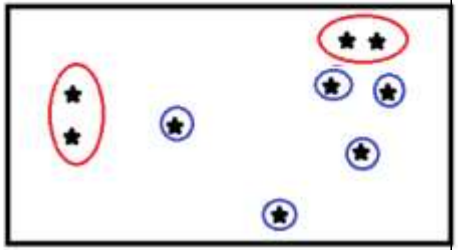
This is visual representation of the problem just for your understanding. The number of points N =9 and M =3



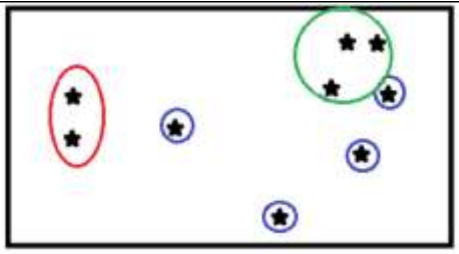
All points are in individual group initially we have N groups



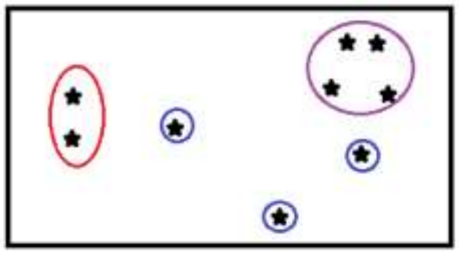
Combine two groups that have minimum distance between them



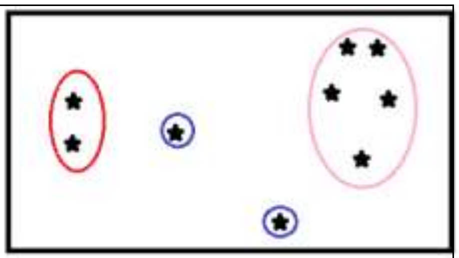
Now M =7
Continue joining groups



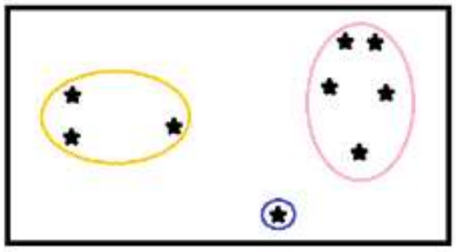
Continue joining groups M=6



Now M =5



Now M =4



M =3, STOP

We merge the groups with the minimum distance between them.
The minimum should ring a bell. Yes, we will be using Heaps to efficiently find the minimum distance. The distance between the two points will be calculated using Euclidean distance formula.

BASIC IDEA FOR GROUPING THE POINTS

Now we describe the basic idea. Note that it is bit inefficient and later we will add heap to make it efficient.

Algorithm for grouping points using 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. Save these pairwise distances in a 2D Matrix of size NxN and call it 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, lets 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)

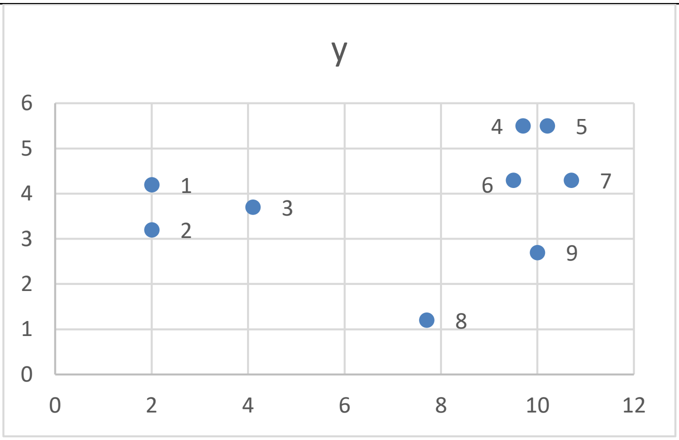
What is the running time of the above algorithm?

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

Input file

```
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
```

The visual representation of the input file (just for your understanding)



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 Euclidean distance formula

For example: Distance between pt1 (2,4.2) & pt2 (2,3.2)
$= ((2-2)^2 + (4.2-3.2)^2)^{1/2} = 1.0$

Notice that the upper and lower triangular part of matrix is same. So, we will only consider 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 minimum in the above Matrix.
It will take O(n^2).

The min distance = 0.5
It is between pt 4 and 5
Merge pt4 and pt5 in a group. 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 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 2. The distance matrix after 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

As 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 have 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.

ASSIGNMENT DETAILS

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

Precisely, you will implement a program that inputs N points and group them into M groups based on 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 on first line separated by #. From 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-dimentional plane.

Input file
9 # 3 3 4.5 9.6 2.6 … … …

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

5. Input N (the number of points) and M (the number of required groups) from the input file.
6. Read two dimensional points from a file. Assume each point forms an individual group. Let the groups are numbered from 1 to N.
7. Compute the distance between each pair of points. Create a MINHEAP of N(N-1)/2 pairwise distances in $O(N^2)$.
8. 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 exists in the heap.

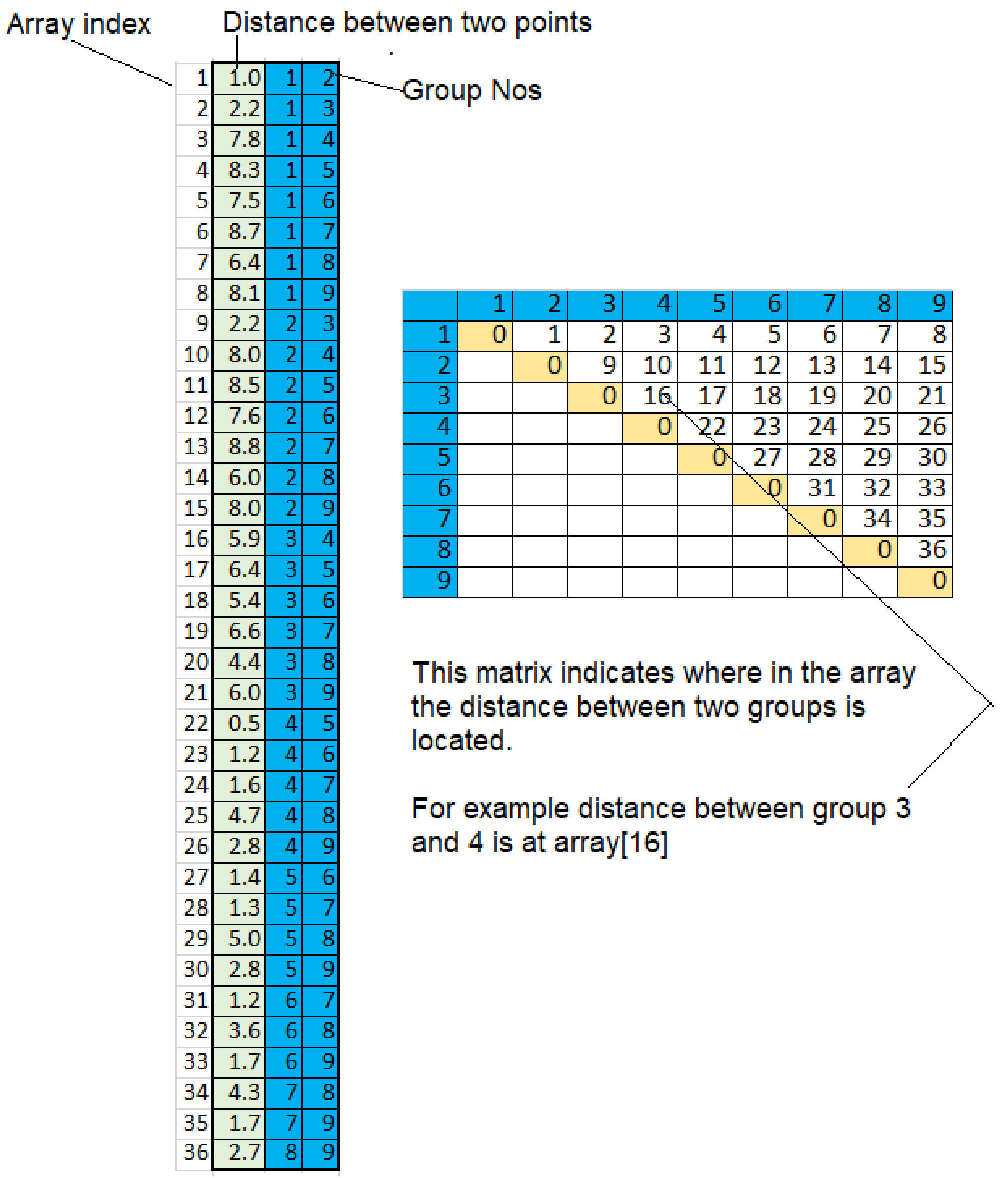
REQUIRED OUTPUT

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 ABOVE INPUT FILE:

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



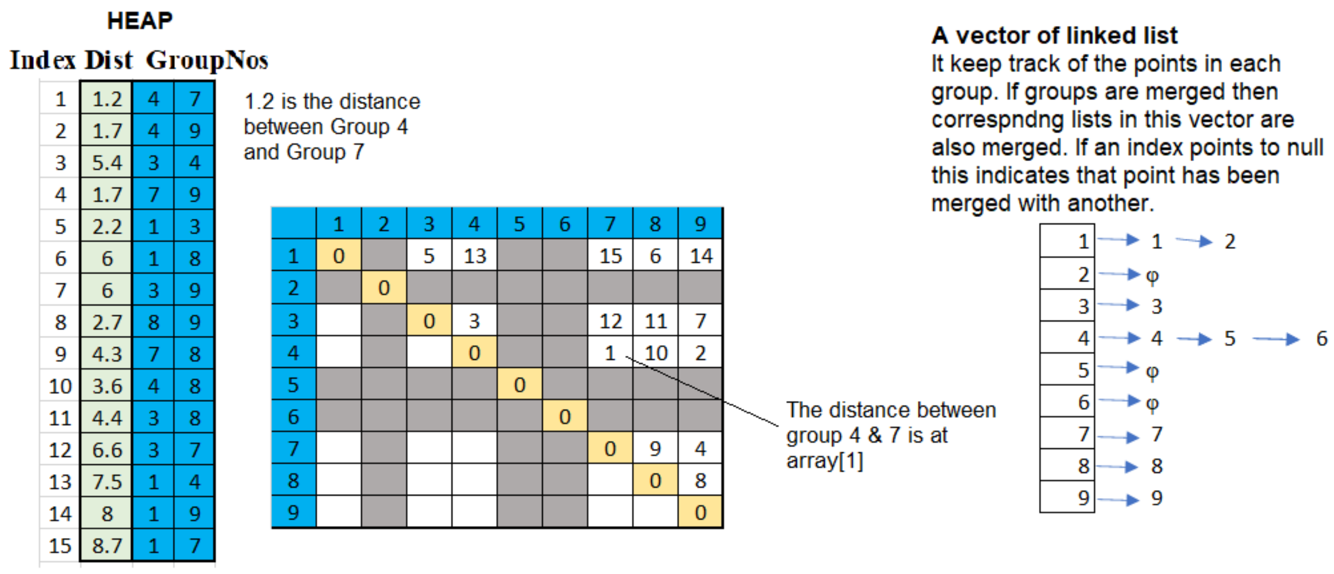
1	1.0	1	2
2	2.2	1	3
3	7.8	1	4
4	8.3	1	5
5	7.5	1	6
6	8.7	1	7
7	6.4	1	8
8	8.1	1	9
9	2.2	2	3
10	8.0	2	4
11	8.5	2	5
12	7.6	2	6
13	8.8	2	7
14	6.0	2	8
15	8.0	2	9
16	5.9	3	4
17	6.4	3	5
18	5.4	3	6
19	6.6	3	7
20	4.4	3	8
21	6.0	3	9
22	0.5	4	5
23	1.2	4	6
24	1.6	4	7
25	4.7	4	8
26	2.8	4	9
27	1.4	5	6
28	1.3	5	7
29	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

This matrix indicates where in the array the distance between two groups is located.

For example distance between group 3 and 4 is at array[16]

After 3rd Iteration of the above algorithm. The point 1 &2 would be merged and point 4, 5,6 would be merged in a group. The heap, matrix and vector of linked list would 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.

1	→ 1 → 2
2	→ φ
3	→ 3
4	→ 4 → 5 → 6
5	→ φ
6	→ φ
7	→ 7
8	→ 8
9	→ 9

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 heap tell that the minimum distance is 1.5 and it is between group 4 and 5. Implement modified heap functions. When two groups are merged. The new distances are computed between the new merged group and other existing group. 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 cluster use 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. 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^2logn) 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.