Worksheet 8: Clustering Aggregation

Clustering Aggregation

Point	C	P
A	0	a
B	0	b
C	2	b
D	1	c
E	1	d

a. Fill in the following table where for each pair of points determine whether C and P agree or disagree on how to cluster that pair.

Pair	Disagreement
A B	N
A C	Y
A D	Y
A E	Y
B C	Y
B D	Y
B E	Y
C D	Y
C E	Y
D E	N

b. Given N points, what is the formula for the number of unique pairs of points one can create?

$$\binom{N}{2} = \frac{N(N-1)}{2}$$

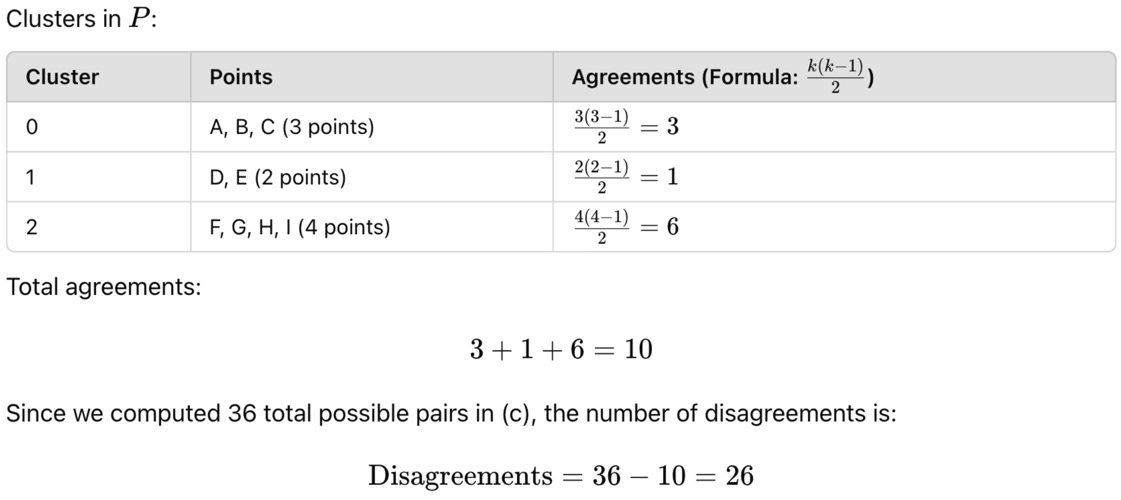
c. Assume that clustering C clusters all points in the same cluster and clustering P clusters points as such:

Point	P
A	0
B	0
C	0
D	1
E	1
F	2
G	2
H	2
I	2

What is the maximum number of disagreements there could be for a dataset of this size? (use the formula from b)?

$$\binom{9}{2} = \frac{9 \times 8}{2} = 36$$

d. If we look at cluster 0. There are (3 × 2) / 2 = 3 pairs that agree with C (since all points in C are in the same cluster). For each cluster, determine how many agreements there are. How many total agreements are there? How many disagreements does that mean there are between C and P?



Clusters in P:

Cluster	Points	Agreements (Formula: $\frac{k(k-1)}{2}$)
0	A, B, C (3 points)	$\frac{3(3-1)}{2} = 3$
1	D, E (2 points)	$\frac{2(2-1)}{2} = 1$
2	F, G, H, I (4 points)	$\frac{4(4-1)}{2} = 6$

Total agreements:

$$3 + 1 + 6 = 10$$

Since we computed 36 total possible pairs in (c), the number of disagreements is:

$$\text{Disagreements} = 36 - 10 = 26$$

e. Assuming that filtering the dataset by cluster number is a computationally easy operation, describe an algorithm inspired by the above process that can efficiently compute disagreement distances on large datasets.

Given the computational challenge of checking all $O(N^2)$ pairs, an efficient algorithm inspired by clustering would:

1. **Preprocess Clusters**: Sort or group points by their cluster in P and C.
2. **Compute Agreements Efficiently**:
    - For each cluster in P, count agreements using $\frac{k(k-1)}{2}$.
    - For each cluster in C, do the same.
3. **Count Total Pairs**: Use the formula $\frac{N(N-1)}{2}$ to get the total possible pairs.
4. **Compute Disagreements Efficiently**:
    - Subtract total agreements from total pairs.

Time Complexity Improvement:

- Instead of checking all $O(N^2)$ pairs, this approach runs in $O(N)$ **time**, assuming clustering is **sorted or preprocessed** efficiently.
- Grouping points into clusters can be done in $O(N)$ **time** using hash maps or sorted lists.