1

point

1. Question 1

In this programming problem and the next you'll code up the clustering algorithm from lecture for computing a max-spacing k-clustering.
Download the text file below.

clustering1.txt

This file describes a distance function (equivalently, a complete graph with edge costs). It has the following format:

[number_of_nodes]

[edge 1 node 1] [edge 1 node 2] [edge 1 cost]

[edge 2 node 1] [edge 2 node 2] [edge 2 cost]

...

There is one edge (i,j) for each choice of $1 \leq i < j \leq n$, where n is the number of nodes. For example, the third line of the file is "1 3 5250", indicating that the distance between nodes 1 and 3 (equivalently, the cost of the edge (1,3)) is 5250. You can assume that distances are positive, but you should NOT assume that they are distinct.

Your task in this problem is to run the clustering algorithm from lecture on this data set, where the target number k of clusters is set to 4. What is the maximum spacing of a 4-clustering?
ADVICE: If you're not getting the correct answer, try debugging your algorithm using some small test cases. And then post them to the discussion forum!

1

point

2. Question 2

In this question your task is again to run the clustering algorithm from lecture, but on a MUCH bigger graph. So big, in fact, that the distances (i.e., edge costs) are only defined *implicitly*, rather than being provided as an explicit list.

The data set is below.

clustering_big.txt

The format is:

[# of nodes] [# of bits for each node's label]

[first bit of node 1] ... [last bit of node 1]

[first bit of node 2] ... [last bit of node 2]

...

For example, the third line of the file "0 1 1 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1" denotes the 24 bits associated with node #2.

The distance between two nodes u and v in this problem is defined as the *Hamming distance*--- the number of differing bits --- between the two nodes' labels. For example, the Hamming distance between the 24-bit label of node #2 above and the label "0 1 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1 0 1 0 0 1 0 1" is 3 (since they differ in the 3rd, 7th, and 21st bits).
The question is: what is the largest value of k such that there is a k-clustering with spacing at least 3? That is, how many clusters are needed to ensure that no pair of nodes with all but 2 bits in common get split into different clusters?
NOTE: The graph implicitly defined by the data file is so big that you probably can't write it out explicitly, let alone sort the edges by cost. So you will have to be a little creative to complete this part of the question. For example, is there some way you can identify the smallest distances without explicitly looking at every pair of nodes?