Problem 1

We run BFS starting from node s. Let d be the layer in which node t is encountered; by assumption, we have $d > n/2$. We claim first that one of the layers $L_1, L_2, \ldots, L_{d-1}$ consists of a single node. Indeed, if each of these layers had size at least two, then this would account for at least $2(n/2) = n$ nodes; but G has only n nodes, and neither s nor t appears in these layers.

Thus, there is some layer L_i consisting of just the node v. We claim next that deleting v destroys all s-t paths. To see this, consider the set of nodes $X = \{s\} \cup L_1 \cup L_2 \cup \cdots \cup L_{i-1}$. Node s belongs to X but node t does not; and any edge out of X must lie in layer L_i, by the properties of BFS. Since any path from s to t must leave X at some point, it must contain a node in L_i; but v is the only node in L_i.

Problem 2

This can be accomplished directly using a convolution. Define one vector to be $a = (q_1, q_2, \ldots, q_n)$. Define the other vector to be $b = (n^{-2}, (n-1)^{-2}, ..., 1/4, 1, 0, -1, -1/4, ... - n^{-2})$. Now, for each j, the convolution of a and b will contain an entry of the form

$$\sum_{i<j} \frac{q_i}{(j-i)^2} + \sum_{i>j} \frac{-q_i}{(j-i)^2}.$$

From this term, we simply multiply by Cq_j to get the desired net force F_j.

The convolution can be computed in $O(n \log n)$ time, and reconstructing the terms F_j takes an additional $O(n)$ time.