Problem 1.

a. Create a graph where the nodes are actors. Add an edge (A,B) if A and B act in a movie together, for all actors A and B. The Bacon number of an actor is then the unweighted shortest path between that actor and Kevin Bacon. We can compute the shortest paths from all actors to Kevin Bacon using BFS.

b. Since we computed all the Bacon numbers in a single BFS run, we can just take the maximum.

c. To find the minimum number of links between any two actors A and B, run BFS using A as the source. The minimum number of links is then the value assigned to B.

Problem 2.

a. This algorithm does not work for graphs with cycles. Consider a simple graph with 3 nodes: A, B, and C and edges of weight 1 from A->B, A->C, B->A, B->C. Since A and B have edges to each other, they form a cycle. To compute shortest(A,C) we need to compute shortest(B,C). To compute shortest(B,C) we need shortest(A,C). Thus the program goes into an infinite recursive loop.

b. In an acyclic graph, once you take an edge from any node A, you can never get back to A (otherwise it would be a cycle). Therefore, we will only compute shortest(A,B) once for each A and B. Since the graph has a finite number of nodes, there is finite number of pairs (A,B) so it terminates eventually.

c. The worst case running time is $O(n^2)$. The first node A could have an edge to every other node (n-1 edges) so we'd have to compute n-1 shortest(A,v). The next node can have an edge to every node except A (to not form a cycle), thus it can have n-2 edges. This continues until the nth node has 0 edges. For each of those edges we had to run compute(s,t) once and there are $O(n^2)$ edges.