Problem 1. The selection (Blum-Floyd-Rivest-Pratt-Tarjan) algorithm to find the $k$-th smallest value in a list, described in the class, uses columns of size 5. Assume you implement the same selection algorithm using columns of size 7, rather than 5.

1. Exactly how far from either end of the array is the median of medians guaranteed to be. Just give the high order term. (Recall that with columns of size 5 we got $\frac{3n}{10}$.)

2. Write down the recurrence for a Selection algorithm based on columns with 7 elements each, using (the full) bubble sort to find the median of each column. (You can ignore floors and ceilings, as we did in class.) You do not have to give the algorithm, but state where each of the terms in your recurrence comes from. (For example, you might say that the $n - 1$ term comes from partition.)

3. Solve the recurrence, and give the high order term exactly.

4. How does this value compare with what we got in class for columns of size 5?

Problem 2. In our analysis of the select algorithm (Blum-Floyd-Rivest-Pratt-Tarjan) for a block size of 5, we compared the median of the medians with every other element in the input array to partition it. However, that led to more comparisons than we should have done. We could reduce it since we know that at least $3n/10$ elements are less than or equal to the median of the medians and similar number of elements is at least greater than or equal to the median of the medians. Obtain a new upper bound on the worst-case number of comparisons using this piece of information. What is the value of the constant?

Problem 3. Let $G = (V, E)$ be a directed graph.

1. Assuming that $G$ is represented by a 2-dimensional adjacency matrix $A[1, \ldots, n, 1, \ldots, n]$, give a $\theta(n^2)$-time algorithm to compute the adjacency list representation of $G$, with $A[i, j]$ representing an edge between $i$ and $j$ vertices. (Represent the addition of an element (vertex), $v$, to an adjacency list, $l$, using pseudo-code, $l \leftarrow l \cup \{v\}$.)

2. Assuming that $G$ is represented by an adjacency list $Adj[1, \ldots, n]$, give a $\theta(n^2)$-time algorithm to compute the 2-dimensional adjacency matrix representation of $G$. 