1 Selection problem and complexity.

The selection problem consists of finding the k-th smallest element in an array. Write the following methods to solve the selection problem using a randomly-generated array of integers as input:

1. Sort the array using bubble sort, then return the k-th element of the sorted array
2. Sort the array using quicksort, then return the k-th element of the sorted array
3. Use a modification of quicksort to solve the problem. Observe that quicksort uses a pivot to partition the array into a subarray that has elements that are less than or equal to the pivot and a subarray that has elements that are larger than the pivot. By looking at the size of the first subarray, you can figure out if the k-th smallest element is in the first or second subarray, and make a recursive call to search in that subarray. Thus you will need just one recursive call instead of the two quicksort makes.

Run experiments with various values of array size, comparing the execution times of the three methods.

2 Sparse matrices.

A matrix consists of $m$ rows and $n$ columns of numbers. Matrices are usually represented as 2D arrays. A matrix is sparse if most of its elements are zero. A sparse matrix can be represented efficiently if we only encode the positions and values of non-zero elements.

For example, the matrix

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 & 5 \\ 0 & 3 & 0 & -1 & 0 \\ 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Can be represented by an array with three columns as follows:

$$S = \begin{bmatrix} 4 & 5 & 6 \\ 0 & 0 & 2 \\ 0 & 4 & 5 \\ 1 & 1 & 3 \\ 1 & 3 & -1 \\ 2 & 1 & -4 \\ 3 & 2 & 1 \end{bmatrix}$$

Where $A[0][0]$, represents the number of rows in the array, $A[0][1]$ represents the number of columns, and $A[0][2]$ represents the number of non-zero elements. For $i > 0$, $A[i][0]$, $A[i][1]$, and $A[i][2]$ represent the row, column, and value of each of the non-zero elements.

Write methods to do the following:
1. Generate random matrices of integers, which may be sparse. Your method should receive the desired number of rows and columns and the proportion of non-zero elements in the matrix (known as the matrix density). Write a method to store matrix as a 2D array and another to store it as a sparse matrix, as explained above.

2. Convert a matrix from regular to sparse representation.

3. Convert a matrix from sparse to regular representation.

4. For each representation, write methods for:
   
   (a) Finding the largest element in the matrix.
   
   (b) Finding the smallest element in the matrix.
   
   (c) Finding the sum of the elements in the matrix.
   
   (d) Finding the mean of the elements in the matrix.
   
   (e) Finding the variance of the elements in the matrix.
   
   (f) Adding two matrices (notice that they must have the same number of rows and columns).

Run experiments for all of your methods with various matrix sizes and densities. We are interested in your observations about the relationship between the density of a matrix and the running times depending on the chosen representation. Notice that each element stored in a sparse matrix requires the storage of three integers (row, column, and value), so for densities greater than $\frac{1}{3}$ a sparse representation should not be considered. Thus limit your experiments to densities of at most that value.