Problem 1

1. C Q S A X B T
   Min=A
   A| C Q S X B T
   Min=B
   A B| C Q S X T
   Min=C
   A B C| Q S X T
   Min=Q
   A B C Q|S X T
   Min=S
   A B C Q S|X T
   Min=T
   A B C Q S T X

   2.|C| Q S A X B T
   C
   C |Q| S A X B T
   Q>C
   C Q |S |A X B T
   S>Q, S>C
   C Q S |A| X B T
   A<S, A<Q, A<C
   A C Q S |X| B T
   X>S, X>Q, X>C, X>A
   A C Q S X|B| T
   B<X, B<S, B<Q, B<C, B>A
   A B C Q S X |T|
   T<X, T>S
   A B C Q S T X

   Problem2

   a. Initial the first unsorted array n[][]
      Initial the second unsorted array n2[][]
      for the i equal to 0 at begin and smaller than the length of n[] and i++
      and loop again for the j equal 0, j smaller than i and j++
      if n[i]==n2[j] that show TRUE
      else show break out

      big O: O(1+n^2) = O(n^2)
      best case: the first element is True
      worst case: search n^2 times
      if return false, time is constant

if True, time is the fastest.

b. Initial the first unsorted array n[][]
for the i equal to 0 at begin and smaller than the length of n[] and i++
and loop again for the j equal 0, j smaller than i and j++
if n.charAt[i] equal with n.charAt[j] that count is added to 1
and then show number of each letter from a to z, with letters.

big O= O(n^3)
best case: the first element is True
worst case: need to search n times
time is n^3 times

c. initial row=n
initial colum=m
for the i equal to length of n at begin and smaller than the length of n[] and i++
and loop again for the j equal length of m, j smaller than i and j++
use the 2 loops to check every element in the 2D array
if there are have one element equal to x that show True
else pass

big O: O (n*m)
best case: the all x is in beginning
the worst case: have to search n*m elements
time is constant