# Prob1:

1. Sometime it works, but not guarantee because there is no way to make sure the re-arrange will sort the array ever.
2. Best case: array already sorted,
3. running time O(1)
4. Non-stop
5. No

# Prob2:

static int[] sort(int[] a) {

int c0 = 0, c1 = 0;

for (int i = 0; i < a.length; i++) {

if (a[i] == 0)

c0++;

else if (a[i] == 1)

c1++;

}

c1 += c0;

for (int i = 0; i < a.length; i++) {

if (i >= c1)

a[i] = 2;

else if (i >= c0)

a[i] = 1;

else

a[i] = 0;

}

return a;

}

# Prob3:

a.

private void bubbleSort1() {

int len = arr.length;

for (int i = 0; i < len; ++i) {

boolean hasSwap = false;

for (int j = 0; j < len - 1; ++j) {

if (arr[j] > arr[j + 1]) {

swap(j, j + 1);

hasSwap = true;

}

}

if (!hasSwap)

return;

}

}

b.

private void bubbleSort2() {

int len = arr.length;

for (int i = 0; i < len; ++i) {

for (int j = 0; j < len - (i + 1); ++j) {

if (arr[j] > arr[j + 1]) {

swap(j, j + 1);

}

}

}

}

c.Result:

1. 403 ms -> InsertionSort
2. 728 ms -> SelectionSort
3. 3314 ms -> BubbleSort2
4. 4395 ms -> BubbleSort1
5. 4516 ms -> BubbleSort

# Prob4:

static int count(int[] a, int from, int to) {

if (from > to)

return 0;

int mid = (from + to) / 2;

int v = a[mid];

if (v == 1)

return count(a, from, mid - 1);

// v=0

if (from == to)

return mid + 1;

if (a[mid + 1] == 1)

return mid + 1;

return count(a, mid + 1, to);

}