Snippet:

if (m > n) {

return findMedianSortedArrays(nums2, nums1);

}

Snippet:

int partitionX = (low + high) / 2;

int partitionY = (m + n + 1) / 2 - partitionX;

Snippet:

int maxLeftX = (partitionX == 0) ? Integer.MIN\_VALUE : nums1[partitionX - 1];

int minRightX = (partitionX == m) ? Integer.MAX\_VALUE : nums1[partitionX];

int maxLeftY = (partitionY == 0) ? Integer.MIN\_VALUE : nums2[partitionY - 1];

int minRightY = (partitionY == n) ? Integer.MAX\_VALUE : nums2[partitionY];

Snippet:

if (maxLeftX <= minRightY && maxLeftY <= minRightX) {

if ((m + n) % 2 == 0) {

return (double)(Math.max(maxLeftX, maxLeftY) + Math.min(minRightX, minRightY)) / 2;

} else {

return (double)Math.max(maxLeftX, maxLeftY);

}

}

Snippet:

else if (maxLeftX > minRightY) {

high = partitionX - 1;

} else {

low = partitionX + 1;

}

Snippet:

throw new IllegalArgumentException();