**Median of two sorted arrays**

double median(vector<int>& arr1, vector<int>& arr2) {

int n = arr1.size();

int m = arr2.size();

if (n > m) {

return median(arr2, arr1); // Ensure arr1 is the smaller array

}

int low = 0;

int high = n;

while (low <= high) {

int partitionX = (low + high) / 2;

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

int maxLeftX = (partitionX == 0) ? INT\_MIN : arr1[partitionX - 1];

int minRightX = (partitionX == n) ? INT\_MAX : arr1[partitionX];

int maxLeftY = (partitionY == 0) ? INT\_MIN : arr2[partitionY - 1];

int minRightY = (partitionY == m) ? INT\_MAX : arr2[partitionY];

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

// Found the correct partitions

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

// Even number of elements

return (max(maxLeftX, maxLeftY) + min(minRightX, minRightY)) / 2.0;

} else {

// Odd number of elements

return max(maxLeftX, maxLeftY);

}

} else if (maxLeftX > minRightY) {

// Too far to the right in arr1, move partitionX to the left

high = partitionX - 1;

} else {

// Too far to the left in arr1, move partitionX to the right

low = partitionX + 1;

}

}

return -1.0; // Median not found

}