K^{th} smallest element

You are given an array of integers ***'ARR'*** of size ***'N'*** and another integer ***'K'***.

Your task is to find and return 'K'th smallest value present in the array.

Note: All the elements in the array are distinct.

Example

If 'N' is 5 and 'K' is 3 and the array is 7, 2, 6, 1, 9

Sorting the array we get 1, 2, 6, 7, 9

Hence the 3rd smallest number is 6.

Detailed explanation (Input/output format, Notes, Images)

Sample Input 1:

7 5
5 9 1 8 10 6 4

Sample Output 1:

8

Explanation of Input 1:

Sorted array will be 1 4 5 6 8 9 10, this shows that 8 is the fifth-smallest element in the array.

→ Approach 1:-

a. Sort the array.
b. Return the required index.

Time Complexity : O(nlogn)

→ Approach 2: Using min heap.

a. Convert the array to min heap.
b. Remove first (k-1) elements.
c. Return the first elements.

Time Complexity : O(nlogn)

```
void heapify(vector<int> &heap, int i, int n) {
  int smallest = i;

  int leftChild = 2 * i + 1;
  int rightChild = 2 * i + 2;
  if (leftChild < n && heap[leftChild] < heap[smallest])
    smallest = leftChild;
  if (rightChild < n && heap[rightChild] < heap[smallest])
    smallest = rightChild;

  if (smallest != i) {
    swap(heap[i], heap[smallest]);
    heapify(heap, smallest, n);
  }
}

void deleteHeap(vector<int> &heap) {
  if (heap.size() == 0)
    return;
  int size = heap.size();
  if (size == 1) {
    heap.erase(heap.begin());
    return;
  }
  heap[0] = heap[size - 1];
  heap.pop_back();
  heapify(heap, 0, size);
}

int kthSmallest(int n, int k, vector<int> Arr) {
  // convert to min heap
  for (int i = n / 2 - 1; i >= 0; i--) {
    heapify(Arr, i, n);
  }
  // remove first (k-1) elements
  k = k - 1;
  while (k--) {
    deleteHeap(Arr);
  }
  // return the first element
  return Arr[0];
}
```