> 给你输入一个无序的数组nums和一个正整数k，让你计算nums中第k大的元素。

解法一： 二叉堆解法，给定一个小根堆，保持其上只能有k个元素。 解法二： 快速选择：

首先复习快速排序：

```
#include <vector>

int partation(std::vector<int> &nums, int left, int right) {
  if (left == right) {
    return left;
  }

  int key = nums[left];            // 将起始位置作为需要比较的key
  int lo = left, hi = right + 1;   // 先--在处理
  while (true) {
    // 保证nums[left ... lo]都小于key
    while (nums[++lo] < key) {
      if (lo == right) {
        break;
      }
    }

    // 保证nums[hi...right]都大于key
    while (nums[--hi] > key) {
      if (hi == left) {
        break;
      }
    }

    if (lo >= hi) {
      break;
    }

    // 此处一定存在nums[lo] > key, nums[hi] < key
    // 交换两个元素即可
    std::swap(nums[lo], nums[hi]);
  }

  std::swap(nums[left], nums[hi]);  // 将key放知道正确位置
  return hi;
}

void quicksort(std::vector<int> &nums, int left, int right) {
  if (left >= right) {
    return;
  }
  int index = partation(nums, left, right);

  quicksort(nums, left, index - 1);
  quicksort(nums, index + 1, right);
}
```

快速选择有：

```
class Solution {
public:
  int findKthLargest(std::vector<int>& nums, int k) {
    int left = 0, right = nums.size() - 1;
    k--;
    while (left <= right) {
      int p = partation(nums, left, right);
      if (k == p) {
        return nums[p];
      } else if (p < k) {
        left = p + 1;
      } else {
        right = p - 1;
      }
    }

    return -1;
  }

private:
  int partation(std::vector<int>& nums, int left, int right) {
    if (left == right) {
      return left;
    }

    int key = nums[left];
    int low = left, high = right + 1;

    while (true) {
      while (nums[++low] > key) {
        if (low == right) {
          break;
        }
      }

      while (nums[--high] < key) {
        if (high == left) {
          break;
        }
      }

      if (low >= high) {
        break;
      }

      std::swap(nums[low], nums[high]);
```

```
        }

        std::swap(nums[left], nums[high]);

        return high;
    }
};
```