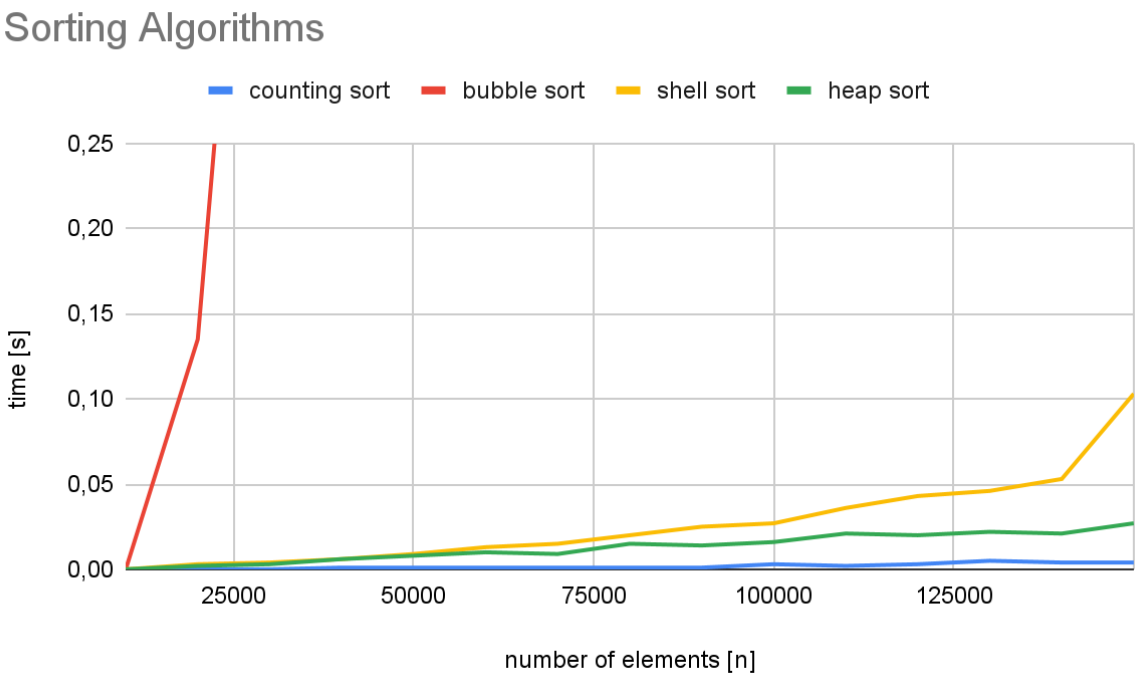
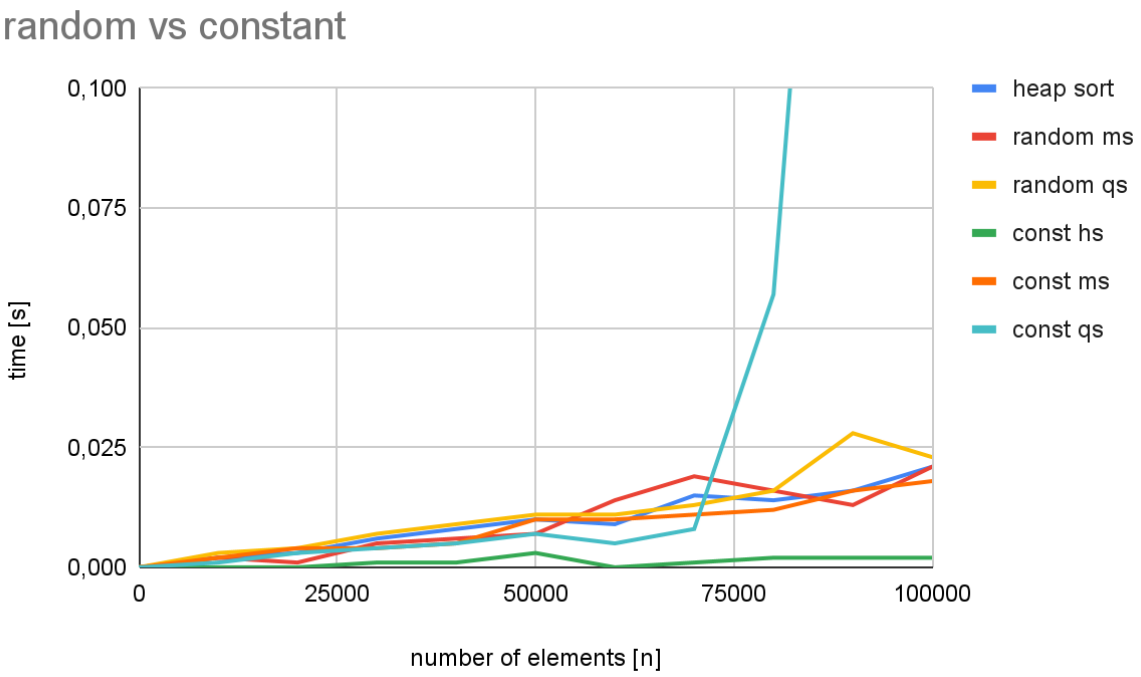
Małgorzata Gierdewicz ID: 148264
Algorithms and data structures
Sorting algorithms

In exercise 1 there were two types of sorting algorithms to be compared: in place algorithms (Bubble Sort, Shell Sort and Heap Sort) and divide and conquer algorithms (Count Sort). Those algorithms differ in the way the data is stored while sorting: in place algorithms don't need additional arrays to store data, while divide and conquer algorithms do. Using the in place algorithms is great when we have very little place to store the data we want to operate on, however the time complexity of those algorithms is much higher than the time complexity of divide and conquer algorithms.
All in place algorithms in this exercise have the space complexity equal to O(1), but time complexity of Bubble Sort is O(n^2), while average time complexity of Shell Sort is somewhere between O(nlog(2)n) and O(n^2). For Heap Sort the time complexity is the lower of all three algorithms and is O(nlogn)
Counting sort use additional arrays to store the data, therefore the space complexity is higher (O(n+k), where n is number of elements in the array and k is range of those elements), but the time complexity is much lower when compared to other algorithms (O(n+k), so the complexity is linear).

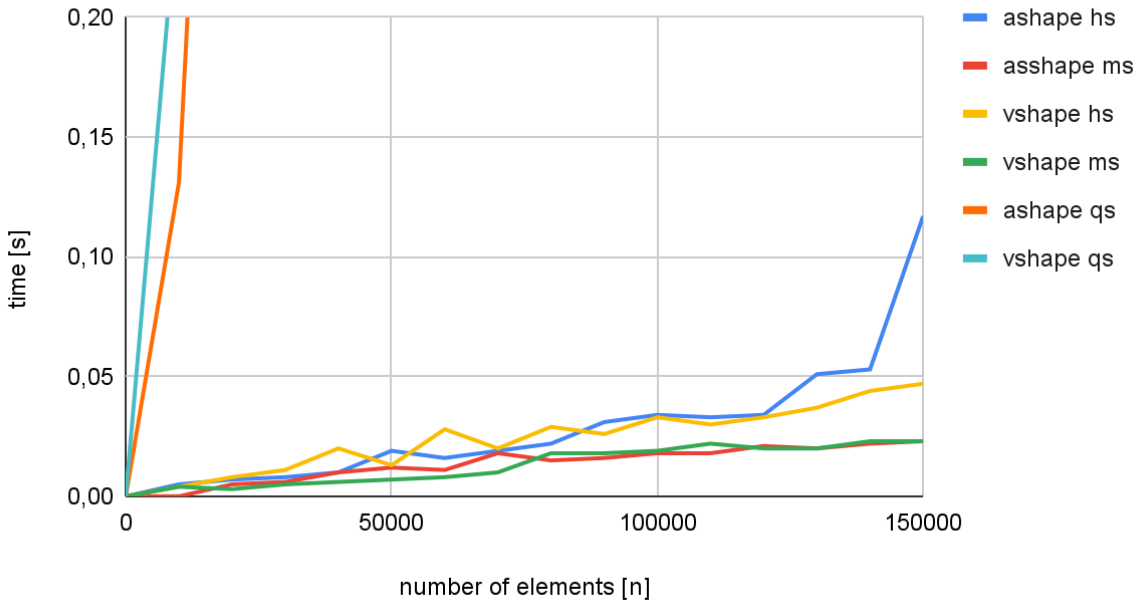


In exercise 2 the impact of different data placement was tested on in place sorting algorithm (HS) and divide and conquer sorting algorithms, such as merge sort and quick sort. In most cases, heap sort wasn't the fastest algorithm, however changes in data organization didn't affect it as much as quick sort. Quick sort in most cases has the time complexity of O(nlogn) which is the same as merge sort, but has lower space complexity (O(logn) vs O(n) in merge sort). In the worst case scenario, quick sort has the complexity of O(n^2), so when using an

algorithm to sort the data it is important to know how the data is arranged. In most cases quick sort will be more efficient than merge sort, due to lower space complexity.



A-shape vs V-shape



increment vs decrement