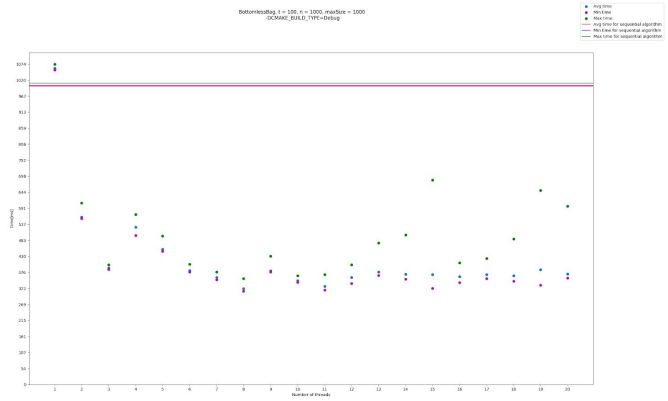
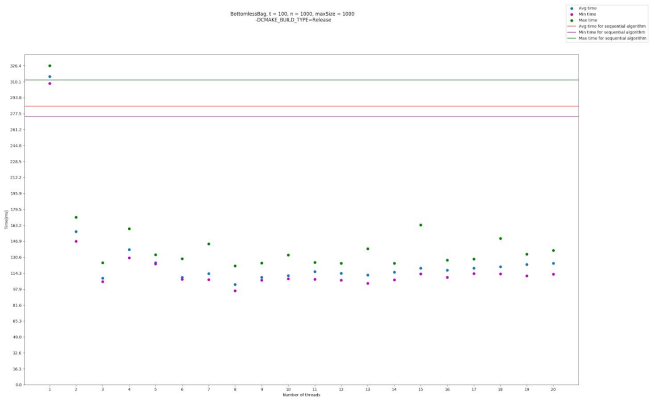
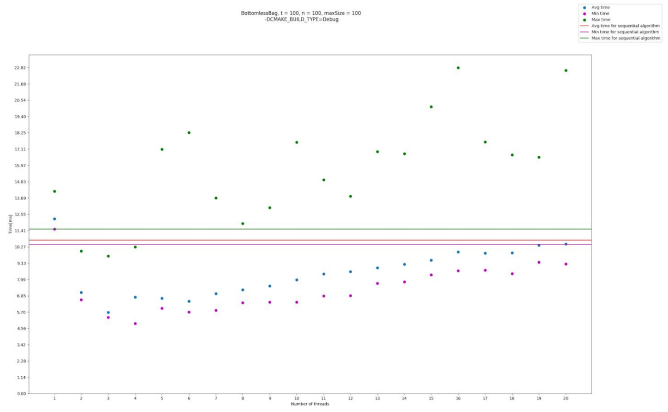
Rafał Czarnecki

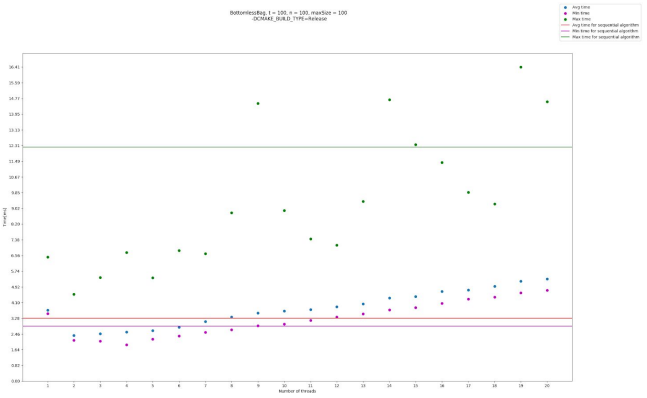
PACKING EGGS

The sequential solution is a standard 0-1 knapsack problem algorithm, where it iterates through all possible eggs, for each of them iterates through all possible maximal capacities and determines the best result that can use any eggs that have already been visited and has combined size lesser than currently chosen capacity. After computing the result, it iterates backwards through eggs, checks each of them whether they should be packed or not, and fills the bag. The parallel version works the same, except the iteration through all capacities is divided between all threads into equal-sized intervals.
On all graphs t denotes the number of input cases per test, n denotes the size of the input and maxSize the largest number that could appear both as a weight and a size.



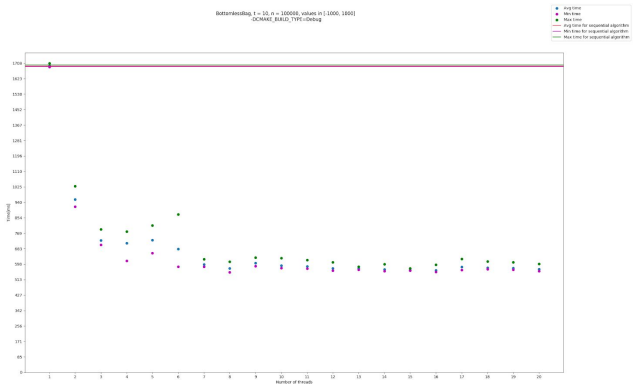


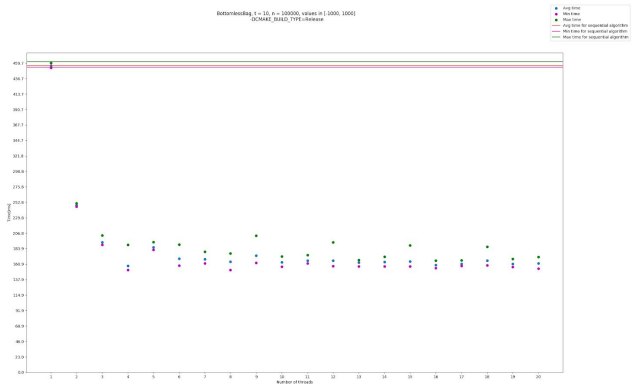


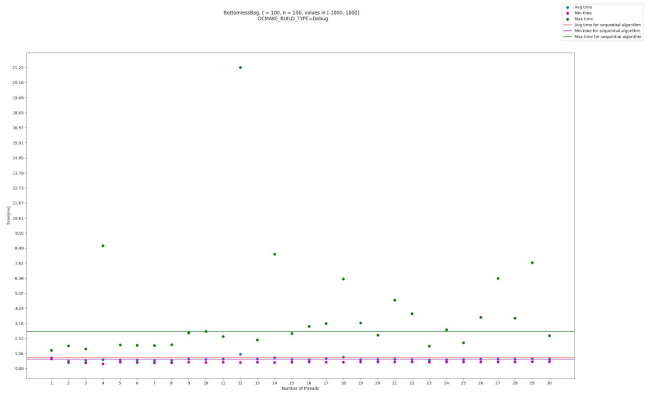


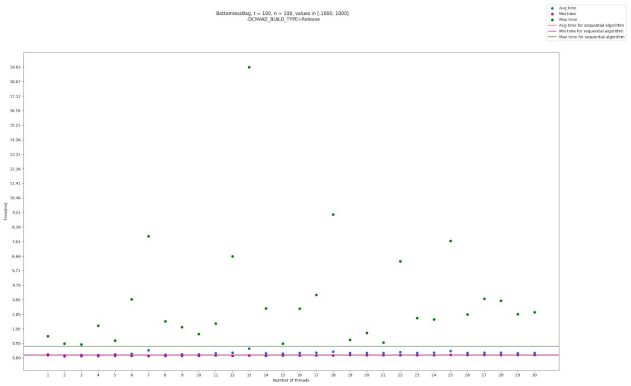
SORTING GRAINS OF SAND

I have chosen to implement the merge sort algorithm. The parallel version divides the input into equal-sized intervals, which are then sorted by the separate threads using merge sort. When all threads are done, all intervals are merged by recursively assigning a new thread to merge the left half of the currently processed segment of intervals and continuing to merge the right one. When finished, both left and right parts are merged together.
On all graphs t denotes the number of input cases per test, and n denotes the size of the input.



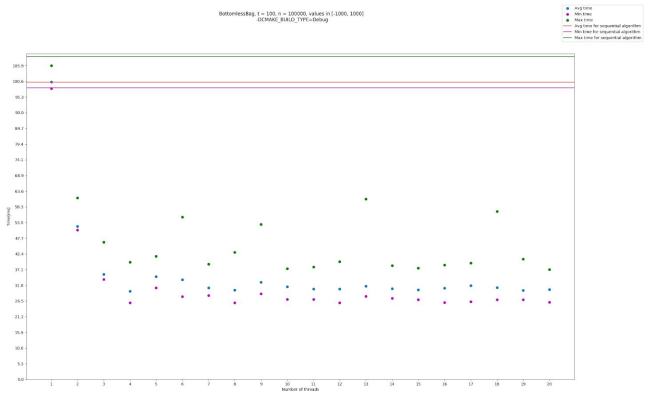


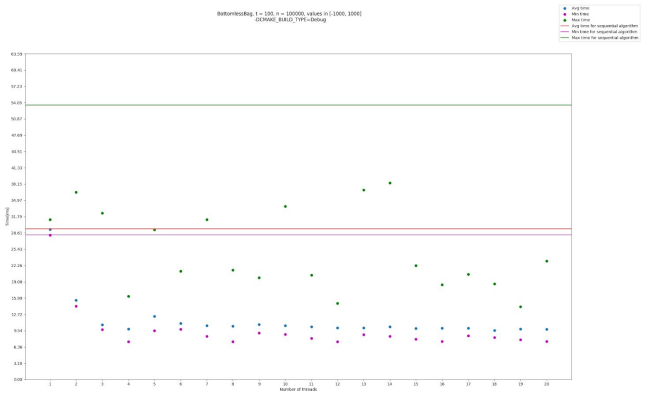


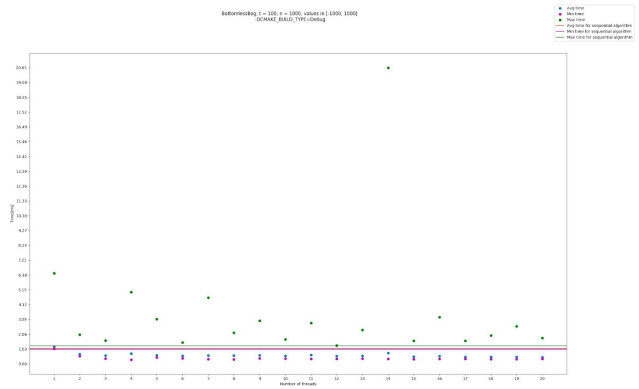


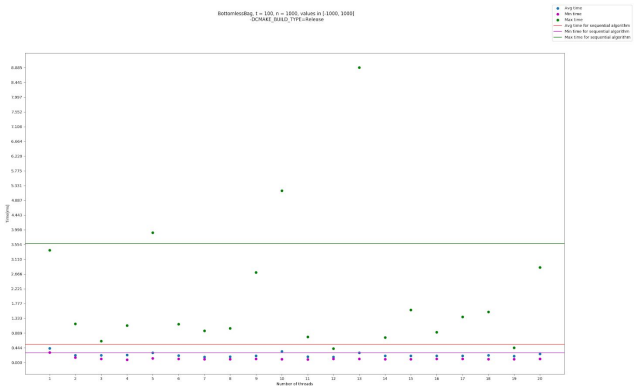
SELECTING THE BEST CRYSTAL

The sequential algorithm iterates through all the crystals and if it finds a crystal that is better than the previous best crystal, it sets the current crystal as the best one. The parallel version does the same, but each thread only iterates through the assigned interval. Updating the global result is done by using a mutex by every thread at the end of its work.
On all graphs t denotes the number of input cases per test, and n denotes the size of the input.






Bottomless8q, t = 100, n = 1000, values in [-1000, 1000]
-DCMAKE_BUILD_TYPE=Debug
Avg time
Min time
Max time
Avg time for sequential algorithm
Min time for sequential algorithm
Max time for sequential algorithm
Time[ms]
Number of threads


Bottomless8q, t = 100, n = 1000, values in [-1000, 1000]
-DCMAKE_BUILD_TYPE=Release
Avg time
Min time
Max time
Avg time for sequential algorithm
Min time for sequential algorithm
Max time for sequential algorithm
Time[ms]
Number of threads