Homework 2

Task 3, to find palindromes and Semordnilaps

The solution to this task was to first, read the file with the words and put them in a double pointer char array. When the file was implemented to the program, the methods for finding palindromes and semordnilaps were done. The palindromes were found through a comparison with the reversed word and the semordnilaps was found through a binary search through the whole word list with the reversed version of the word. These two parts was done inside the parallel part of the program with the #pragma OpenMP declaration. The found palindromes and semordnilaps were put in a global double pointer array to later be put in a new text file with only these words.

The performance was measured with two timestamps, one before the parallel part and one after. The time was measured with OpenMP time function, which measures the time for each thread. To know the speedup we tried different wordlist sizes and different number of threads

As we can see in the table below there is a speedup in the execution time of all sizes of the wordlist when more threads are used to search through the wordlist. In wordlist 1, and (wordlist 2/2) the speedup slows down and the execution time increases again when using 3 and 4 threads respectively. The reason that the execution time increases for these two scenarios is reasonable since the wordlists we are handling, 25 and 50~ thousand entries, are not big arrays. When creating threads there is an overhead cost, so that when searching through small arrays, the net gain from using multiple threads will decrease at a certain number of threads.

Wordlist 2 is ~100 thousand elements so increasing the amount of threads from 1 to 4 only causes a net positive in execution time. For this array if threads >4 at some point execution time would start increasing again.

T = time in ms. Wordlist 1 = 25 000, Wordlist 2= 104 000

Number of Threads	1	2	3	4
T (Wordlist 1)	8,1	7,6	8,7	20
T (Wordlist 2)	18	13	10	7,6
T ((Wordlist 2)/2)	8,6	6,2	4,1	5,3

Table 1: the execution time of the parallel part of the program with different sizes and threads. Median of 5 executions

T = speedup. Wordlist 1 = 25 000, Wordlist 2= 104 000

Number of Threads	2	3	4

T (Wordlist 1)	8,1 / 7,6= 1.07	8,1 / 8,7 = 0.93	8,1/20 = 0.405
T (Wordlist 2)	18/13=1.38	18/10=1,8	18/7,6=2.37
T ((Wordlist 2)/2)	8,6/6,2=1,39	8,6/4,1=2,1	8,6/5,3=1.62

Table 2: The speedup of using more threads, sequential time/parallel time.