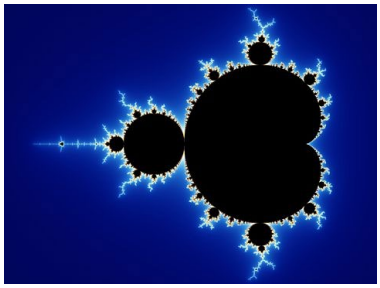
1 Matrix multiplication

Implement a parallel version of a matrix multiplication function (assume matrices with fixed sizes) with explicit handling of the threads in a parallel region.

Experiment with different numbers of threads.

2 Area of Mandelbrot Set



Source: wikipedia

The Mandelbrot Set is the set of complex numbers c for which the iteration $z = z^2 + c$ does not diverge. To determine (approximately) whether a point c belongs in the set, a finite number of iterations are performed, and if the condition $|z| > 2$ is satisfied then the point is considered to be outside the Set.

There is no known theoretical value to calculate the area of the set, but approaches exist to calculate estimates. The supplied program performs a sequential algorithm for this estimation. The method generates a grid of points in a box containing the upper half of the (symmetric) Mandelbrot Set. Then each point is iterated using the equation above a finite number of times. If within that number of iterations the threshold condition $|z| > 2$ is satisfied then that point is considered to be outside of the Mandelbrot Set. Then counting the number of points within the Set and those outside will give an estimate of the area of the Set.

Parallelize the code with explicit handling of the threads in a parallel region. Experiment with different numbers of threads.

3 Parallel loops

Implement the two previous exercises with parallel loops, instead of explicit handling of threads.

a. Experiment with different placement of parallel loops and the collapse clause.
b. Experiment with different scheduling approaches, and compare with the previous implementations.
c. Can you conclude on the most efficient (and scalable) approach?

4 Quick Sort

Quick Sort is a sorting algorithm which uses a divide and conquer approach. The algorithm sorts the numbers by first dividing the list into two sublists, so that all the numbers in one sublist are smaller than all the numbers in the second sublist. This is done by selecting one number, the pivot, against which all other numbers are compared to. Then the algorithm is applied recursively to the two sublists.

Write a OpenMP program that sorts an array using quick sort. The size of the array can be fixed, and the array global. The decision on the pivot is irrelevant (e.g. take the first).

Determine the time it takes to perform the sorting, and compare it to a sequential implementation. Use also different sizes of the array. What can you conclude?

5 Bubble Sort

Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted. Since it compares adjacent elements, loop iterations are dependent. However, a simple solution is to divide the inner loop in a odd-even pair.

Write a OpenMP program that sorts an array using buuble sort. The size of the array can be fixed, and the array global.

Determine the time it takes to perform the sorting, and compare it to a sequential implementation. Use also different sizes of the array. What can you conclude?

6 Linked List

Implement a thread safe linked list library that supports applications parallelized with OpenMP. Implement the insert (ordered by key), delete and search functions. To make it more efficient, do not lock the full list.
Node data is a variable of the type:

```
typedef struct data{
        int key;
        char data[SIZE];
}data;
```

Implement also a program that performs parallel insertion, deletion and search of data in the list.

Credits:

- Version 1.0, Luis Miguel Pinho, with inputs from:
  - An Introduction to Parallel Programming, P Pacheco, M. Malensek, University of San Francisco, 2021, available at https://www.cs.usfca.edu/~mmalensek/cs521/schedule/materials/pthreads.pdf
  - A "Hands-on" Introduction to OpenMP, Tim Mattson, https://www.openmp.org/wp-content/uploads/Intro_To_OpenMP_Mattson.pdf
  - OpenMP Online Course, http://www.archer.ac.uk/training/courses/2019/11/openmp_online/