Documentation Lab 6 (Parallelizing techniques) - Mos Daniele 935

Algorithm:

- **Hamiltonian Path** in an undirected graph is a path that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian Path such that there is an edge (in the graph) from the last vertex to the first vertex of the Hamiltonian Path. Determine whether a given graph contains Hamiltonian Cycle or not. If it contains, then prints the path.
- When we find the first Hamiltonian cycle we stop searching from other nodes
- We use a Thread Pool, in which we submit tasks to
- Each task consists of different starting position (vertex). So we split the search between the threads

Synchronization:

- We are using ReentrantLock for synchronization
- A reentrant mutual exclusion `Lock` with the same basic behavior and semantics as the implicit monitor lock accessed using `synchronized` methods and statements, but with extended capabilities.
- A `ReentrantLock` is *owned* by the thread last successfully locking, but not yet unlocking it. A thread invoking `lock` will return, successfully acquiring the lock, when the lock is not owned by another thread

Tests:

- Test were done using graphs with 1000 nodes
- ! Results may depend on the graph that is randomly generated !

Threads	Time
5	92

10	35
50	43
100	55
500	161
1000	287