UNIVERSITÀ DEGLI STUDI FIRENZE

Parallel Programming Mid-term assignment

Lucia Giorgi

12/01/24

UNIVERSITÀ DEGLI STUDI FIRENZE

Index

Random Maze Solver

Find the exit from a maze using the random movement of a particle.

- Start a large number of particles, move them randomly bouncing on the walls
- Backtrack the first particle to get out of the maze to find the exit

Implementation

- Take an image of a maze from the Internet
- Extract information about the maze geometry and load it into an appropriate data structure
- Move the particles randomly from the starting point of the maze until it reaches the exit.

Cell

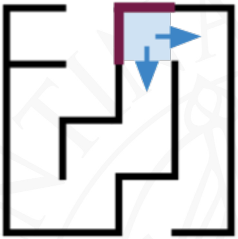


Figure: A cell with two possible directions and two walls

- (x, y) coordinates

Maze

- Cells array
- Move from a cell in a direction
- Start cell
- Load maze from image
- Save image with solution

Maze

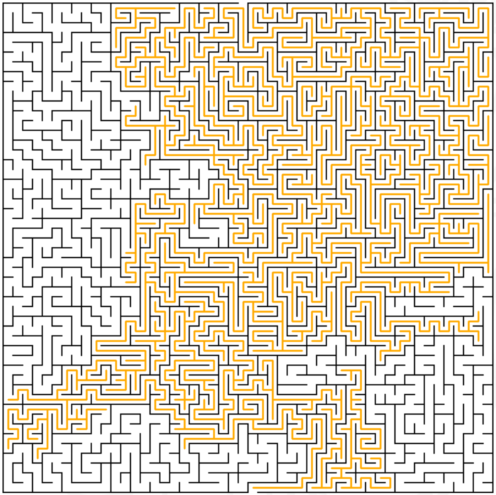


Figure: A 50 $\times$ 50 mazes with the solution found by the particle

Maze

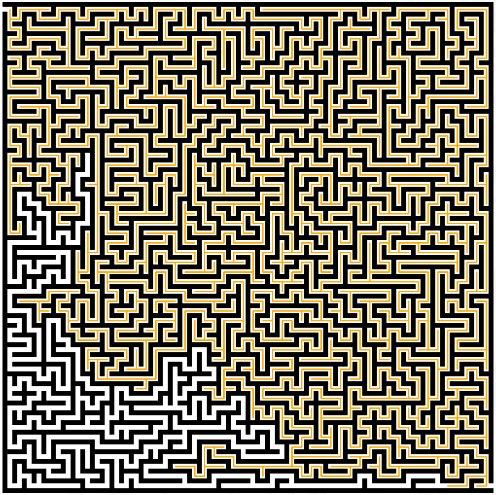


Figure: A 50 $\times$ 50 mazes with the solution found by the particle

Maze

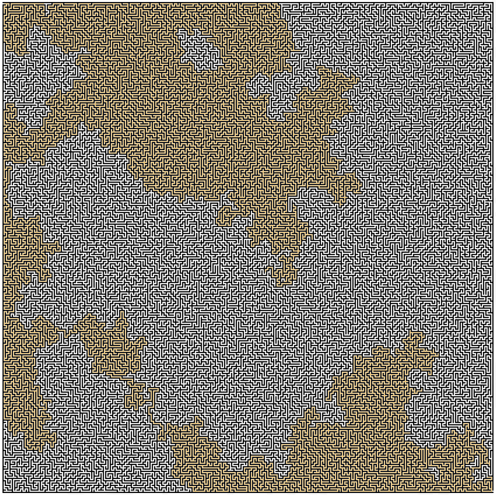


Figure: A 210 $\times$ 210 maze with the solution found by the particle

Main function

Four loops:

- Move particle to next cell as long as `solution_found==false`
- Loop over all particles
- Loop over all runs
- Loop over all images

Main function

Four loops:

- Move particle to next cell as long as `solution_found==false`
- Loop over all particles
- Loop over all runs
- Loop over all images

Command line arguments: particles number, thread numbers, runs, save solution

Parallelization

OpenMP

- `#pragma omp parallel for` on the particles loop
- `solution_found` is `shared`

Parallelization

```
while (!solution_found) {
// steps for moving the particle
// ...
     catch (OutOfMazeException &e) {
        solution_found = true;
        endTime = std::chrono::high_resolution_clock::now();
        bool is_first;
        #ifdef _OPENMP
        omp_set_lock(&solution_found_write);
        #endif
        if (!solution_found_locked) {
            solution_found_locked = true;
            is_first = true;
        }
        #ifdef _OPENMP
        omp_unset_lock(&solution_found_write);
        #endif
        if (is_first) {
            // log the execution time
            // ...
        }
    }
}
```

Experiments

- Sequential version
- Parallel version
  - 2, 4, 6, ... 30 threads
  - Particles number $\geq$ threads number

Experiments

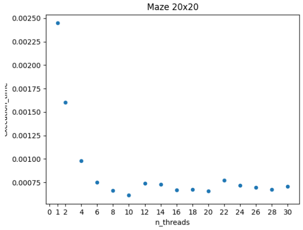
- Sequential version
- Parallel version
  - 2, 4, 6, ... 30 threads
  - Particles number $\geq$ threads number

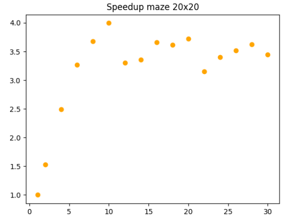
100 runs

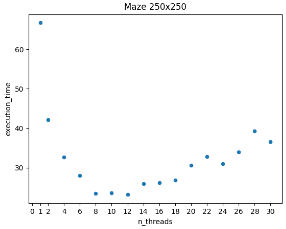
No solution

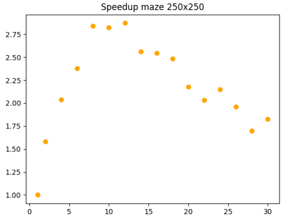
UNIVERSITÀ DEGLI STUDI FIRENZE

Results



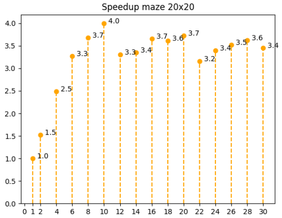


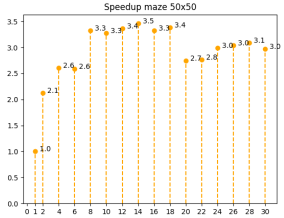


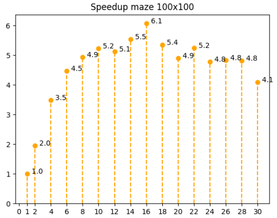


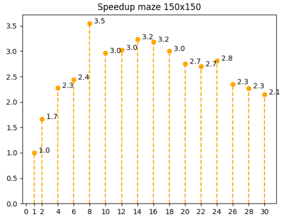
UNIVERSITÀ DEGLI STUDI FIRENZE

Results



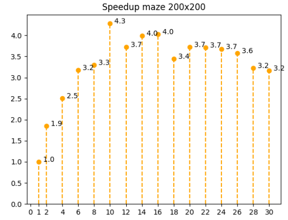


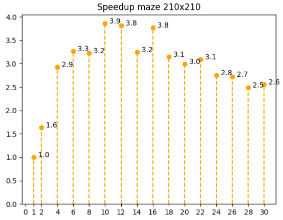


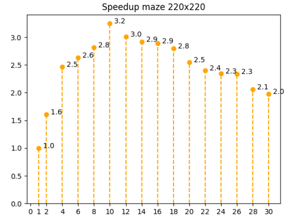


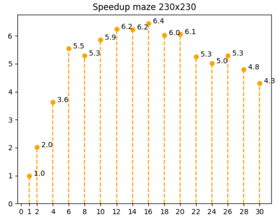
UNIVERSITÀ DEGLI STUDI FIRENZE

Results

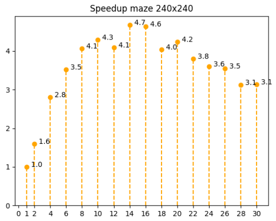




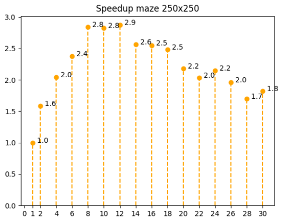




Results



Speedup maze 240x240
1.0
1.6
2.8
3.5
4.1
4.3
4.1
4.7
4.6
4.0
4.2
3.8
3.6
3.5
3.1
3.1



Speedup maze 250x250
1.0
1.6
2.0
2.4
2.8
2.8
2.9
2.6
2.5
2.5
2.2
2.0
2.2
2.0
1.7
1.8

Profiling

Gperftools

```
Total: 81970 samples
   16134  19.7%  19.7%    46927  57.2% std::generate_canonical
   14829  18.1%  37.8%    29644  36.2% std::mersenne_twister_engine::operator
   14815  18.1%  55.8%    14815  18.1% std::mersenne_twister_engine::_M_gen_rand
    7671   9.4%  65.2%    81929  99.9% main._omp_fn.0
    6443   7.9%  73.1%    58316  71.1% std::uniform_real_distribution::operator
    5338   6.5%  79.6%    14167  17.3% Maze::move
    3895   4.8%  84.3%     4888   6.0% Maze::getCell
    2127   2.6%  86.9%     2127   2.6% Cell::getY
    ... other lines
```