HW1: Sokoban

Due: Mon, 2020/3/30 23:59

Problem Description

> *"Sokoban (倉庫番, sōko-ban, "warehouse keeper") is a puzzle video game in which the player pushes crates or boxes around in a warehouse, trying to get them to storage locations." – Wikipedia*

You are asked to implement a solver for Sokoban and parallelize it with threads (either pthread or OpenMP; you can use `std::thread` as a pthread wrapper).

Input Format

Input is given in a ASCII text file. The filename is given from the command line in `argv[1]`.

Every line ends with a newline character `\n`. Each line represents a row of tiles; every character in the line represents a tile in the game.

The tiles are:

- `o`: The player stepping on a regular tile.
- `O`: The player stepping on a target tile.
- `x`: A box on a regular tile.
- `X`: A box on a target tile.

- ` ` (space): Nothing on a regular tile.
- `.`: Nothing on a target tile.
- `#`: Wall.

You program only need to deal with valid inputs. It is guranteed that:

- There is only one player, stepping on either a regular tile or a target tile.
- The number of target tiles are equal to the number of boxes.
- All tiles other than wall tiles are within an enclosed area formed by wall tiles.
- There is at least one solution.

Output Format

You program should output a sequence of actions that pushes all the boxes to the target tiles, to `stdout`. (Do not output anything else to `stdout`, otherwise your output may be considered invalid. For debugging purposes, please output to `stderr`. However, it is advised that you remove the debug output from your submission as they may harm performance)

The output sequence should end with a newline characters `\n`, and contain only uppercase `WASD` characters:

- `W`: move the player up
- `A`: move the player left
- `S`: move the player down
- `D`: move the player right

It is not required that you output the solution with the least moves. Any sequence that solves the problem is valid.

Example

1. This is a valid input:

   ```
   #########
   #  xox..#
   #   #####
   #########
   ```

2. This input is invalid because there are tiles outside of wall-enclosed area:

   ```
   #########
   #  xox..#
   #   #####
   #####   #
   ```

3. This input is invalid because there are fewer target tiles than boxes:

```
#########
#  xox .#
#   #####
#########
```

4. This input is invalid because there are no solutions:

   ```
   #########
   # ox x..#
   #   #####
   #########
   ```

5. This is another valid input:

   ```
   #######
   #    .#
   #     #
   # xx  #
   #O    #
   #######
   ```

   One valid output is:

   ```
   DWWAWDDDSSAAWAS
   ```

   Another valid output is:

   ```
   WWDSDDSDWWSSAAA
   ```

Execution

We will compile your code with the following command:

```
g++ -std=c++17 -O3 -pthread -fopenmp hw1.cc -o hw1
```

We use ninja to build your code. The default ninja file for this homework is provided at `/home/ipc20/ta/hw1/build.ninja`. If you wish to change the compilation flags, include `build.ninja` in your submission.

To use ninja to build your code, make sure `build.ninja` and `hw1.cc` is in the working directory, then run `ninja` on the command line and it will build `hw1` for you. To remove the built files, run `ninja -t clean`.

Your code will be executed with a command equalviant to:

```
srun -n1 -c${threads} ./hw1 ${input}
```

where:

- `${threads}` is the number of threads
- `${input}` is the path to the input file

The time limit for each input test case is 30 seconds.

Report

Answer the following questions, in either English or Traditional Chinese.

1. Briefly (< 100 words) describe your implementation.
2. What are the difficulties encountered in this homework? How did you solve them? (You can discuss about hard-to-optimize hotspots, or synchronization problems)
3. What are the strengths and weaknesses of pthread and OpenMP?
4. Which one did you use to implement this homework? Why?
5. (Optional) Any suggestions or feedback for the homework are welcome.

Submission

Upload these files to iLMS:

- `hw1.cc` – the source code of your implementation.
- `build.ninja` – optional. Submit this file if you want to change the build command.
- `report.pdf` – your report.

Please follow the naming listed above carefully. Failing to adhere to the names above will result to points deduction. Here are a few bad examples: `hw1.c`, `HW1.cc`, `hw1.cpp`, `report.docx`, `report.pages`.

Grading

1. (40%) Correctness. Propotional to the number of test cases solved.
2. (30%) Performance. Based on the total time you solve all the test cases. For a failed test case, 75 seconds is added to your total time.
3. (30%) Report.

Appendix

Please note that this spec, the sample test cases and programs might contain bugs. If you spotted one and are unsure about it, please ask on iLMS 討論區!

Sample Testcases

The sample test cases are located at `/home/ipc20/ta/hw1/samples`.

Playing the game interactively

You can play sokoban in the terminal with `/home/ipc20/ta/hw1/play.py`.

For example, run `/home/ipc20/ta/hw1/play.py /home/ipc20/ta/hw1/samples/01.txt` to play the first sample input.

Reference implementation

There is a Python implementation at `/home/ipc20/ta/hw1/example_solver.py` for your reference. Please notice that the implementation is not very fast and there might be faster strategies.

Output validation

`/home/ipc20/ta/hw1/validate.py` can be used to validate your output.

For example, the following command validates your answer for `01.txt` when running 6 threads:

```
srun -c6 -o answer.txt ./hw1 01.txt
/home/ipc20/ta/hw1/validate.py 01.txt answer.txt
```

You can also use:

```
srun -c6 ./hw1 01.txt | /home/ipc20/ta/hw1/validate.py 01.txt -
```

Here, `-` instructs validate.py to read your output from stdin.

Automatic Judge

The `hw1-judge` command can be used to automatically judge your code against all sample test cases, it also submits your execution time to the scoreboard so you can compare your performance with others.

The scoreboard is here.

To use it, run `hw1-judge` in the directory that contains your code `hw1.cc`. It will automatically search for `build.ninja` and use it to compile your code, or fallback to the TA provided `/home/ipc20/ta/hw1/build.ninja` otherwise. If code compiliation is successful, it will then run all the sample test cases, show you the results as well as update the scoreboard.

> *Note:* `hw1-judge` *and the scoreboard has nothing to do with grading. Only the code submitted to iLMS is considered for grading purposes.*

The judge supports `-xpattern` and `-ipattern` filters to exclude or include test cases. These filters are applied in the order they are specified. These filters support glob(7) patterns. When multiple filters match a given test case, the later filter takes priority. For example, to skip `05.txt` and `06.txt`, you can run:

```
hw1-judge -x05.txt -x06.txt
```

To run only the case `02.txt`:

```
hw1-judge '-x*' -i02.txt
```

Judge Verdict Table

Verdict	Explaination
internal error	there is a bug in the judge
time limited exceeded+	execution time > time limit + 10 seconds
time limited exceeded	execution time > time limit
runtime error	your program didn’t return 0 or is terminated by a signal
no output	your program did not produce an output file
wrong answer	your output is incorrect
accepted	you passed the test case