Race 04

Marathon C

April 30, 2020

ucode

Contents

ucode

Engage

DESCRIPTION

Welcome to this Race!
How to react when facing a difficult task that seems impossible to complete in the given time frame? It is certainly a common situation. And it's tempting to give up in such cases. However, having given up, we would fail at the challenge without even getting started. Isn't it better to at least do our best?
Why do people climb Mount Everest? Why do some run marathons? What drives them and why they are not afraid of lose?

We are all afraid, but we must not be afraid of defeat. A person is ready for almost anything if he has motivation.
How to motivate oneself? How to keep moving forward every day? The answer is inside ourselves. No matter the outcome, if you make an effort and believe in yourself, you're on the way to success. Break out from a fixed mindset. Motivate yourself every day and you'll conquer all the peaks.

This challenge invites you to prove the concept that beginners can implement an advanced algorithm to find the shortest/optimal route between two points.

BIG IDEA

Accept difficult challenges.

ESSENTIAL QUESTION

Is it possible to develop a pathfinding algorithm after only 3 weeks of programming?

CHALLENGE

Implement a pathfinding algorithm.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students in the Slack and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- How to organize work on the challenge productively?
- What workflow is best for a small developer team?
- What is the main skill needed for team challenges?
- What motivates you to grow in coding?
- What is the difference between pathfinding and navigation algorithms?
- What are pathfinding algorithms?
- What is a graph/graph theory?
- How to evaluate the complexity of a pathfinding algorithm?
- How to choose an efficient algorithm?
- What are the criteria of the shortest route?
- What is csv file format?
- How to open a file?
- How to read a file?
- What is a buffer?
- How to save read information?
- How to write data to a file?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Meet with your team in the Zoom. Discuss how you will organize teamwork. Create a channel for a team communication.
- Read about motivation. Watch a few videos on YouTube about this topic.
- Find your own personal reasons to learn and grow in programming.
- Involve all teammates in the research process.
- Read about branching and merging in `git`. Share your experience. These features improve your development efficiency.
- Learn about existing pathfinding algorithms. We advise you to focus on the Lee algorithm to deal with it.
- Read the story, taking everyone's ideas on board.
- Clone your git repository that is issued on the challenge page in the LMS.
- Distribute tasks between among team members.

- Start to develop the solution. Offer improvements. Test your code.
- Communicate with students and share information.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Challenge has to be carried out by the entire team.
- Each team member must understand the challenge and realization, and be able to reproduce it individually.
- It is your responsibility to assemble the whole team. Phone calls, SMS, messengers are good ways to stay in touch.
- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Analyze all information you have collected during the preparation stages.
- Perform only those tasks that are given in this document.
- Submit your files using the layout described in the story. Only useful files allowed, garbage shall not pass!
- Compile C-files with clang compiler and use these flags: `clang -std=c11 -Wall -Wextra -Werror -Wpedantic`.
- Your program must manage memory allocations correctly. Memory which is no longer needed must be released otherwise the task is considered as incomplete.
- Pay attention to what is allowed in a certain task. Use of forbidden stuff is considered a cheat and your tasks will be failed.
- Complete tasks according to the rules specified in the `Auditor`.
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- Also, the challenge will pass automatic evaluation which is called `Oracle`.
- If you have any questions or don't understand something, ask other students or just Google it.
- Use your brain and follow the white rabbit to prove that you are the Chosen one!

Act

NAME

Way home

DIRECTORY

`./`

SUBMIT

`Makefile, inc/*.[h], src/*.[c]`

ALLOWED FUNCTIONS

`read, write, malloc, free, exit, open, close`

BINARY

`race04`

LEGEND

This is my ship... the Nebuchadnezzar, it's a hovercraft. Help me to find the path to the Zion.

DESCRIPTION

Create a program that finds the shortest path in a maze between the entry and exit points using only orthogonal directions.
A maze is specified as a comma-separated sequence of obstacles and empty spaces, and represented by a map containing:

- `#` character is an obstacle
- `.` character is an empty space
- `,` character is a separation of obstacles and empty spaces

See an example in the EXAMPLE.

The program:

- takes a filename, entry point (x1, y1) and exit point (x2, y2) coordinates as command-line arguments
- uses your `mx_atoi` for getting coordinates without validation
- calculates the shortest path from the entry point to the most distant point (x3, y3) of the maze. Prints the number of movements `N` from (x1, y1) to (x3, y3) to the standard output in `dist=N` format
- calculates the shortest path from the entry point to the exit point. Prints the number of movements `N` from (x1, y1) to (x2, y2) to the standard output in `exit=N` format

- saves the maze and the path from the entry point to the exit point in the file `path.txt`. Note, that `path.txt` will be assessed only by peers. Obstacle - `#`, empty space - `.`, path - `*`. Most distant point - `X` if it is a part of the path or `D` in other cases if points do not match. `X` and `D` indicate where the farthest point is on the map.

Error handling. The program prints errors to the standard error stream `stderr`:

- if the number of command-line arguments are less or more than required, the program prints `usage: ./race04 [file_name] [x1] [y1] [x2] [y2]`
- in case of missing or empty file the program prints `map does not exist`
- if there are forbidden characters or the map is not rectangular, the program prints `map error`
- if points are outside of map coordinates, the program prints `points are out of map range`
- if the entry point is also an obstacle, the program prints `entry point cannot be an obstacle`
- if the exit point is also an obstacle, the program prints `exit point cannot be an obstacle`
- if there is no pass among points, the program prints `route not found`
- in case of any other errors the program prints `error`

If there is more than 1 error, the program prints only 1 error message according to priority listed above.

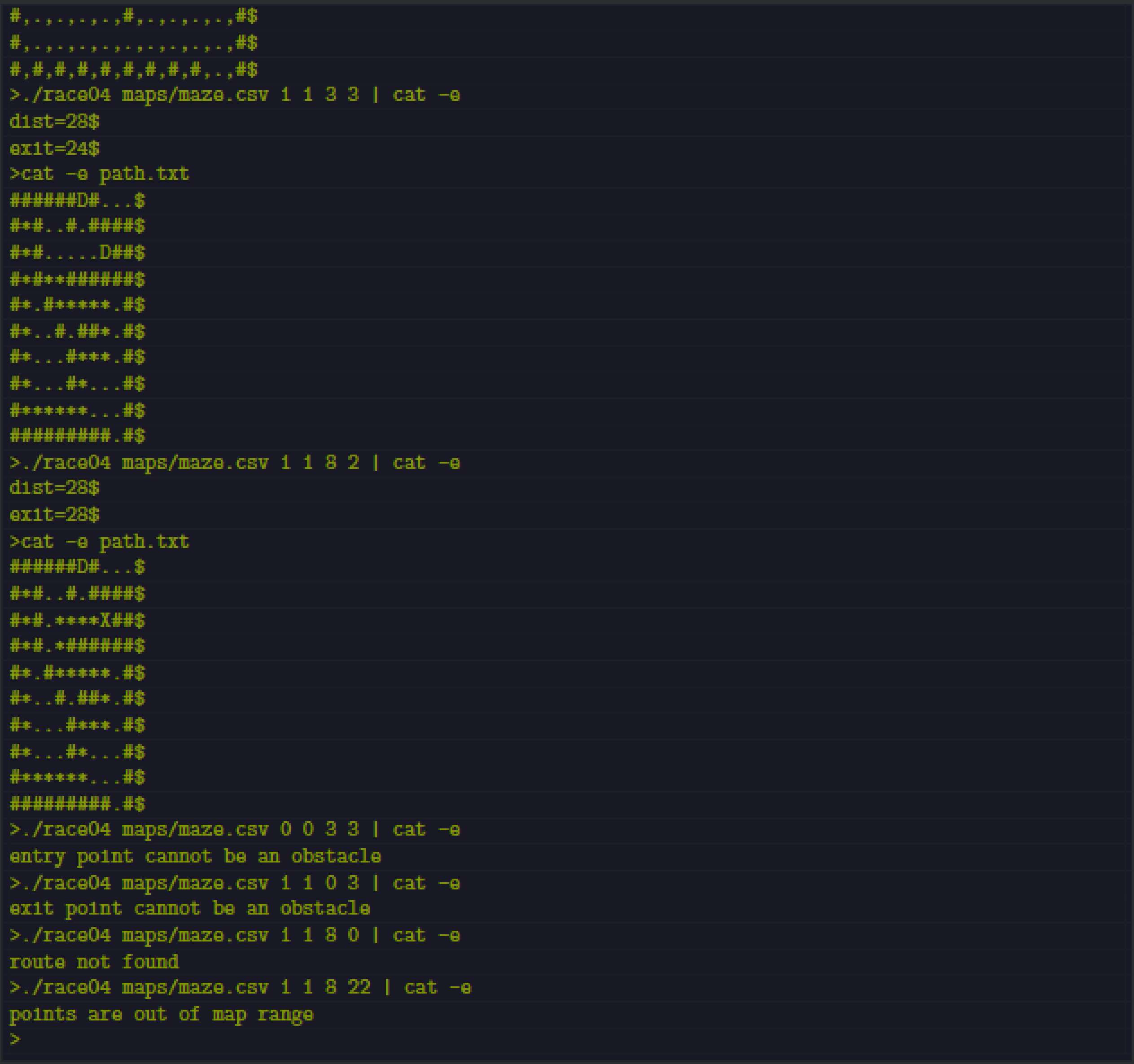
See a detailed example in the CONSOLE OUTPUT.

EXAMPLE

```
#,#,#,#,#,#,.,#,.,.,.$
#,.,#,.,.,#,.,#,#,#,#$
#,.,#,.,.,.,.,.,.,#,#$
#,.,#,.,.,#,#,#,#,#,#$
#,.,.,#,.,.,.,.,.,.,#$
#,.,.,.,#,.,#,#,.,.,#$
#,.,.,.,.,#,.,.,.,.,#$
#,.,.,.,.,#,.,.,.,.,#$
#,.,.,.,.,.,.,.,.,.,#$
#,#,#,#,#,#,#,#,#,.,#$
```

CONSOLE OUTPUT

```
>./race04 | cat -e
usage: ./race04 [file_name] [x1] [y1] [x2] [y2]
>cat -e maps/maze.csv
#,#,#,#,#,#,.,#,.,.,.$
#,.,#,.,.,#,.,#,#,#,#$
#,.,#,.,.,.,.,.,.,#,#$
#,.,#,.,.,#,#,#,#,#,#$
#,.,.,#,.,.,.,.,.,.,#$
#,.,.,.,#,.,#,#,.,.,#$
#,.,.,.,.,#,.,.,.,.,#$
```



```
#,.,.,.,.,#,.,.,.,.,#$
#,.,.,.,.,.,.,.,.,.,#$
#,#,#,#,#,#,#,#,#,.,#$
>./race04 maps/maze.csv 1 1 3 3 | cat -e
dist=28$
exit=24$
>cat -e path.txt
######D#...$
#*#..#.####$
#*#.....D##$
#*#**######$
#*.#*****.#$
#*..#.##*.#$
#*...#***.#$
#*...#*...#$
#******...#$
#########.#$
>./race04 maps/maze.csv 1 1 8 2 | cat -e
dist=28$
exit=28$
>cat -e path.txt
######D#...$
#*#..#.####$
#*#.****X##$
#*#.*######$
#*.#*****.#$
#*..#.##*.#$
#*...#***.#$
#*...#*...#$
#******...#$
#########.#$
>./race04 maps/maze.csv 0 0 3 3 | cat -e
entry point cannot be an obstacle
>./race04 maps/maze.csv 1 1 0 3 | cat -e
exit point cannot be an obstacle
>./race04 maps/maze.csv 1 1 8 0 | cat -e
route not found
>./race04 maps/maze.csv 1 1 8 22 | cat -e
points are out of map range
>
```