PathFinder

Track C

November 27, 2019

ucode

Contents

Engage

DESCRIPTION

Hi, challenger.
Everyone has moments or even days when he needs to visit a large number of places.
Of course, you were considering the way to do this, covering the shortest path.
During this challenge, you'll improve your algorithmic skills and create an automating solution which will help to find the shortest path through your places of interest.
Besides, it's time to try your library out on the battlefield and show what it made of.

Remember, this is an individual challenge, so you have to develop all the code by yourself.
And, yes, it is preferable to consult with your colleagues, use P2P power to the fullest.
`P2P != Copy-Paste`
You will become a skilled software engineer only when you write 100500 lines of code yourself.

Your talent is in your hands, you know da wae.
Good luck!

BIG IDEA

Develop an algorithmic mindset.

ESSENTIAL QUESTION

What algorithms can help to use time and resources effectively?

CHALLENGE

Develop an algorithm that finds the shortest paths between the points.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. This will help you realize what knowledge you will get from this challenge and how to move forward.

Ask your neighbor on the right, left, or behind you, and discuss the following questions together. You can find the answers in the Internet and share it with student around you.

We encourage you to ask as many personal and contextual questions as possible. Note down your discussion.

- What is the path?
- What places do you want to visit during the weekend?
- How can you interpret the places and paths between them?
- What does the term shortest path mean?
- What is the graph theory?
- What is the application of graph theory in real life/products?
- How to find the shortest path between two places?
- What are the shortest path search algorithms?
- What is combinatorics?
- Can combinatorics math approaches be applied during the solution to this challenge?
- What is the meaning and application of the term FIFO?

GUIDING ACTIVITIES

These are only a set of example activities and resources. Do not forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

1. Read the story from cover to cover and study it carefully.
2. Read about software development methodologies. Try using one of them to develop a solution.
3. Make a simple task plan for this challenge.
4. Learn the basic algorithms for finding the shortest paths in graphs.
5. Read about and understand the basic rules of combinatorics.
6. Collaborate with other students to investigate challenge in-depth. Learn from others and share your knowledge.
7. Start developing your program.

ANALYSIS

You need to analyze all the collected information before you start.

1. Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
2. Perform only those tasks that are given in the story.
3. The challenge must be performed in `C`.
4. Your challenge must have the next structure:
   - `src` directory contains source files `.c`;
   - `obj` directory contains object files `.o`. It must be created only during compilation;
   - `inc` directory contains header files `.h`;
   - `libmx` directory contains source files of your library including its `Makefile`. You must use it.
   - `Makefile` that compiles the library `libmx` firstly and then compiles and builds `pathfinder`.
5. Makefile must be written in accordance to Auditor.
6. You should submit only files required to complete the task in the required directories and nothing else. Garbage shall not pass.
7. You should compile C files with clang compiler and use these flags: `-std=c11 -Wall -Wextra -Werror -Wpedantic`.
8. Your program must manage memory allocations correctly. Memory which is no longer needed must be released otherwise the task is considered as incomplete.
9. Usage of forbidden functions is considered as cheat and your challenge will be failed.
10. You must complete tasks according to the rules specified in the `Auditor`.
11. Your challenge will be checked and graded by students. The same as you. `Peer-to-Peer (P2P) learning`.
12. Also, your challenge will pass automatic evaluation which is called `Oracle`.
13. Got a question or you do not understand something? Ask the students or just Google that.
14. Use your brain and follow the white rabbit to prove that you are the Chosen one!!!

Act

SOLUTION DEVELOPMENT

Let's get started! And may the odds be ever in your Favor!
Repeat this cycle until your challenge is ready for evaluation.

1. Understand the information received, decide what is paramount and what is not for you.
2. Clone your git repository, what is issued on the challenge page in the LMS.
3. Read about header files and structures.
4. Open Auditor and read it. Seriously!
5. Brainstorm with other students and find more than one solution.
6. Try to realize your thoughts in code.
7. Check the work!

ALLOWED FUNCTIONS

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

BINARY

pathfinder

DESCRIPTION

You need to create a program that finds all the shortest paths between all the islands, using information about the bridges that connect them. So, the program:

1. reads data from the file which name is given as command-line argument;
2. finds all shortest paths between every combination of two islands.
   Any pair of islands always has a path between them;
3. prints path using the FIFO(first) rule for the islands on the standard output.
   More examples can be found in the CONSOLE OUTPUT.

File input data:

- the first line in the file is a number of islands;
- remaining lines are describing distance between two islands, one per line. Each line consists of two islands and bridge length between them in the format: `island1-island2,length_of_bridge` . Island names contain only alphabetic characters and cannot be empty. Length contains only digits.

Output consist of information blocks about certain shortest path. Every block contains:

1. `========================================` upper boundary that consist of 40 `=` characters;
2. `Path: <island1> -> <island2>` - shows initial point and final destination;
3. `Route: <island1> -> <all_islands_between> -> <island2>` - shows the full route between two islands;
4. `Distance: <length1> + <length2> = <sum>` - shows the distance between every island in the route as well as their sum that indicates distance to the final destination.
5. `========================================` bottom boundary that consist of 40 `=` characters;

Error handling. Print on the standard error stream:

- `usage: ./pathfinder [filename]`, if there are an invalid number of command-line arguments;
- `error: file [filename] does not exist`, if the file does not exist;
- `error: file [filename] is empty`, if the file is empty;
- `error: line 1 is not valid`, if the first line contains something other than digits;
- `error: line [line_value] is not valid`, if one of the remaining lines does not match the format above;
- `error: invalid number of islands`, if the number received on the first line does not coincide with the number of islands.

CONSOLE OUTPUT

```
>./pathfinder | cat -e
usage: ./pathfinder [filename]
>cat -e islands
cat: islands: No such file or directory
>./pathfinder islands | cat -e
error: file islands does not exist
>cat -e empty
>./pathfinder empty | cat -e
error: file empty is empty
>cat -e invalid1
4f$
Greenland-Bananal,8$
Fraser-Greenland,10$
Bananal-Fraser,3$
Java-Fraser,5$
>./pathfinder invalid1 | cat -e
error: line 1 is not valid
>cat -e invalid2
4$
Greenland-Bananal,8$
Fraser--Greenland,10$
Bananal-Fraser,3$
Java-Fraser,5$
>./pathfinder invalid2 | cat -e
```

```
error: line 3 is not valid
>cat -e invalid3
3$
Greenland-Bananal,8$
Fraser-Greenland,10$
Bananal-Fraser,3$
Java-Fraser,5$
>./pathfinder invalid3 | cat -e
error: invalid number of islands
>cat -e example1
4$
Greenland-Bananal,8$
Fraser-Greenland,10$
Bananal-Fraser,3$
Java-Fraser,5$
>./pathfinder example1 | cat -e
========================================$
Path: Greenland -> Bananal$
Route: Greenland -> Bananal$
Distance: 8$
========================================$
========================================$
Path: Greenland -> Fraser$
Route: Greenland -> Fraser$
Distance: 10$
========================================$
========================================$
Path: Greenland -> Java$
Route: Greenland -> Fraser -> Java$
Distance: 10 + 5 = 15$
========================================$
========================================$
Path: Bananal -> Fraser$
Route: Bananal -> Fraser$
Distance: 3$
========================================$
========================================$
Path: Bananal -> Java$
Route: Bananal -> Fraser -> Java$
Distance: 3 + 5 = 8$
========================================$
========================================$
Path: Fraser -> Java$
Route: Fraser -> Java$
Distance: 5$
========================================$
>cat -e example2
5$
A-B,11$
A-C,10$
B-D,5$
C-D,6$
C-E,15$
D-E,4$
>./pathfinder example2 | cat -e
```

```
========================================$
Path: A -> B$
Route: A -> B$
Distance: 11$
========================================$
========================================$
Path: A -> C$
Route: A -> C$
Distance: 10$
========================================$
========================================$
Path: A -> D$
Route: A -> B -> D$
Distance: 11 + 5 = 16$
========================================$
========================================$
Path: A -> D$
Route: A -> C -> D$
Distance: 10 + 6 = 16$
========================================$
========================================$
Path: A -> E$
Route: A -> B -> D -> E$
Distance: 11 + 5 + 4 = 20$
========================================$
========================================$
Path: A -> E$
Route: A -> C -> D -> E$
Distance: 10 + 6 + 4 = 20$
========================================$
========================================$
Path: B -> C$
Route: B -> D -> C$
Distance: 5 + 6 = 11$
========================================$
========================================$
Path: B -> D$
Route: B -> D$
Distance: 5$
========================================$
========================================$
Path: B -> E$
Route: B -> D -> E$
Distance: 5 + 4 = 9$
========================================$
========================================$
Path: C -> D$
Route: C -> D$
Distance: 6$
========================================$
========================================$
Path: C -> E$
Route: C -> D -> E$
Distance: 6 + 4 = 10$
========================================$
```

```
========================================$
Path: D -> E$
Route: D -> E$
Distance: 4$
========================================$
>
```

Share

PUBLISHING

The final important and integral stage of your work is its publishing. This allows you to share your challenges, solutions, and reflections with a local and global audience.

During this stage, you will find how to get a global assessment. You will get representative feedback. As a result, you get the maximum experience from the work you have done.

What you can create to disseminate information

- Text post, summary from reflection.
- Charts, infographics or any other ways to visualize your information.
- Video of your work, reflection video.
- Audio podcast. You can record a story with your experience.
- Photos from ucode with small post.

Example techniques

- Canva - a good way to visualize your data.
- QuickTime - easy way to record your screen, capture video, or record audio.

Example ways to share your experience

- Facebook - create a post that will inspire your friends.
- YouTube - upload a video.
- GitHub - share your solution.
- Telegraph - create a post. This is a good way to share information in a Telegram.
- Instagram - share a photos and stories from ucode. Don't forget to tag us :)

Share what you learned with your local community and the world. Use #ucode and #CBLWorld on social media.