SOLUCIÓN DEL TRIQUI

FUNDAMENTOS DE COMPUTACÓN

Profesora: Beatriz Ayala Hoyos

Estudiante: Gustavo Andrés Orozco

Para el problema planteado sobre crear un programa en el cual dos personas puedan jugar al TRIQUI, en donde gana la persona que llene de un mismo valor cualquier fila, columna, la diagonal principal o la diagonal invertida, para lograrlo, se utilizaron ciclos, procedimientos, parámetros y todo lo demás aprendido en clase que conforma un programa, iniciando las librerías necesarias para que funcione de la manera adecuada.

Específicamente para este juego lo primero que se debe hacer es crear una matriz de tamaño 3x3 con un ciclo para las filas y para las columnas que vaya desde 0 hasta 2 e inicializarla en un valor igual para todas las casillas, como la intención es jugar con los números cero (0) para el jugador 1 y uno (1) para el jugador 2, a través de condicionales nos aseguramos de que el valor con el que se inicializa la matriz sea diferente a dichos números, en este caso, la matriz se inicializó con el número ocho (8) utilizando un ciclo para las filas, otro para las columnas y asignando el número 8 a cada casilla.

Seguidamente se debe indicar con qué número juega cada jugador y hacer el procedimiento en el cual se les da instrucciones por turnos para que indiquen la fila y la columna en la que desean ubicar el número con el que juegan, esto se hizo a través de un ciclo de 0 a 8 (que son la cantidad de turnos totales en el juego) en el que el contador cuando se encuentra en un número par le da el turno al jugador 1 y cuando se encuentra en un número impar le cede el turno al jugador 2 y se debe verificar usando condicionales que la ubicación que da cada jugador en su turno, no esté ocupada.

Para comodidad de los jugadores, el programa imprime al final de cada turno la matriz con los cambios efectuados y finalmente para completar el objetivo del juego, el programa debe comprobar desde el quinto turno si hay una fila, columna, diagonal principal o diagonal invertida con el mismo valor, con un procedimiento que analiza cada una de ellas en búsqueda del TRIQUI, de ser así el programa especifica el jugador ganador, y si al final de los 9 turnos posibles en el juego no se logra el TRIQUI, el programa muestra un letrero diciendo que hubo un empate.

<p align="center">

<img src="https://s3.eu-west-2.amazonaws.com/dependabot-images/logo-with-name-horizontal.svg?v5" alt="Dependabot" width="336">

</p>

# Dependabot

Welcome to the public home of Dependabot. This repository serves 2 purposes:

1. It houses the source code for Dependabot Core, which is the heart of [Dependabot][dependabot]. Dependabot Core handles the logic for updating dependencies on GitHub (including GitHub Enterprise), GitLab, and Azure DevOps. If you want to host your own automated dependency update bot then this repo should give you the tools you need. A reference implementation is available [here][dependabot-script].

2. It is the public issue tracker for all things Dependabot, replacing the now-archived [feedback](https://github.com/dependabot/feedback/) repository.

## Got feedback?

Please file an issue. Bug reports, feature requests, and general feedback are all welcome.

## Contributing to Dependabot

Currently, the Dependabot team is not accepting support for new ecosystems. We are prioritising upgrades to already supported ecosystems at this time.

Please refer to the [CONTRIBUTING][contributing] guidelines for more information.

### Disclosing security issues

If you believe you have found a security vulnerability in Dependabot please submit the vulnerability to GitHub Security [Bug Bounty](https://bounty.github.com/) so that we can resolve the issue before it is disclosed publicly.

## What's in this repo?

Dependabot Core is a collection of packages for automating dependency updating

in Ruby, JavaScript, Python, PHP, Elixir, Elm, Go, Rust, Java and

.NET. It can also update git submodules, Docker files, and Terraform files.

Highlights include:

- Logic to check for the latest version of a dependency \*that's resolvable given

a project's other dependencies\*

- Logic to generate updated manifest and lockfiles for a new dependency version

- Logic to find changelogs, release notes, and commits for a dependency update

## Other Dependabot resources

In addition to this library, you may be interested in:

- The [dependabot-script][dependabot-script] repo, which provides a collection

of scripts that use this library to update dependencies on GitHub Enterprise,

GitLab or Azure DevOps

- The [API docs][api-docs] for Dependabot's hosted instance (dependabot.com)

## Setup

To run all of Dependabot Core, you'll need Ruby, Python, PHP, Elixir, Node, Go,

Elm, and Rust installed. However, if you just wish to run it for a single

language you can get away with just having that language and Ruby.

The main library is written in Ruby, while JavaScript, Python, PHP, Elm,

Elixir, Go, and Rust are required for dealing with updates for their respective

languages.

To install the helpers for each language:

1. `cd npm\_and\_yarn/helpers && yarn install --production && cd -`

2. `cd composer/helpers && composer install --no-dev && cd -`

3. `cd python/helpers && pyenv exec pip install -r requirements.txt && cd -`

4. `cd hex/helpers && mix deps.get && cd -`

5. `cd terraform && helpers/build "$(pwd)/helpers/install-dir/terraform" && cd -`

6. `cd go\_modules && helpers/build "$(pwd)/helpers/install-dir/go\_modules" && cd -`

## Local development

Run the tests by running `rspec spec` inside each of the packages. Style is

enforced by RuboCop. To check for style violations, simply run `rubocop` in

each of the packages.

### Running with Docker

While you can run Dependabot Core without Docker, we also provide a development

Dockerfile. In most cases, you'll be better off running Dependabot in the

development Docker container as it bakes in all required dependencies.

Start by building the initial Dependabot Core image, or pull it from the

Docker registry.

```shell

$ docker pull dependabot/dependabot-core # OR

$ docker build -f Dockerfile -t dependabot/dependabot-core . # This may take a while

```

Once you have the base Docker image, you can build and run the development

container using the `docker-dev-shell` script. The script will automatically

build the container if it's not present and can be forced to rebuild with the

`--rebuild` flag. The image includes all dependencies, and the script runs the

image, mounting the local copy of Dependabot Core so changes made locally will

be reflected inside the container. This means you can continue to use your

editor of choice while running the tests inside the container.

```shell

$ bin/docker-dev-shell

=> building image from Dockerfile.development

=> running docker development shell

[dependabot-core-dev] ~/dependabot-core $

```

### Dry run script

\*Note: you must have run `bundle install` in the `omnibus` directory before

running this script.\*

You can use the "dry-run" script to simulate a dependency update job, printing

the diff that would be generated to the terminal. It takes two positional

arguments: the package manager and the GitHub repo name (including the

account):

```bash

$ cd omnibus && bundle install && cd -

$ bin/dry-run.rb go\_modules rsc/quote

=> fetching dependency files

=> parsing dependency files

=> updating 2 dependencies

...

```

## Debugging with Visual Studio Code and Docker

There's built-in support for leveraging Visual Studio Code's [ability for

debugging](https://code.visualstudio.com/docs/remote/containers) inside a Docker container.

After installing the recommended [`Remote - Containers` extension](https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers),

simply press `Ctrl+Shift+P` (`⇧⌘P` on macOS) and select `Remote-Containers: Reopen in Container`.

You can also access the dropdown by clicking on the green button in the bottom-left corner of the editor.

If the development Docker image isn't present on your machine, it will be built automatically.

Once that's finished, start the `Debug Dry Run` configuration `(F5)` and you'll be prompted

to select a package manager and a repository to perform a dry run on.

Feel free to place breakpoints on the code.

## Releasing

Triggering the jobs that will push the new gems is done by following the steps below.

- Ensure you have the latest merged changes: `git checkout main` and `git pull`

- Generate an updated `CHANGELOG`, `version.rb`, and the rest of the needed commands: `bin/bump-version.rb patch`

- Edit the `CHANGELOG` file and remove any entries that aren't needed

- Run the commands that were output by running `bin/bump-version.rb patch`

## Architecture

Dependabot Core is a collection of Ruby packages (gems), which contain the

logic for updating dependencies in several languages.

### `dependabot-common`

The `common` package contains all general-purpose/shared functionality. For

instance, the code for creating pull requests via GitHub's API lives here, as

does most of the logic for handling Git dependencies (as most languages support

Git dependencies in one way or another). There are also base classes defined for

each of the major concerns required to implement support for a language or

package manager.

### `dependabot-{package-manager}`

There is a gem for each package manager or language that Dependabot

supports. At a minimum, each of these gems will implement the following

classes:

| Service | Description |

|------------------|-----------------------------------------------------------------------------------------------|

| `FileFetcher` | Fetches the relevant dependency files for a project (e.g., the `Gemfile` and `Gemfile.lock`). See the [README](https://github.com/dependabot/dependabot-core/blob/main/common/lib/dependabot/file\_fetchers/README.md) for more details. |

| `FileParser` | Parses a dependency file and extracts a list of dependencies for a project. See the [README](https://github.com/dependabot/dependabot-core/blob/main/common/lib/dependabot/file\_parsers/README.md) for more details. |

| `UpdateChecker` | Checks whether a given dependency is up-to-date. See the [README](https://github.com/dependabot/dependabot-core/tree/main/common/lib/dependabot/update\_checkers/README.md) for more details. |

| `FileUpdater` | Updates a dependency file to use the latest version of a given dependency. See the [README](https://github.com/dependabot/dependabot-core/tree/main/common/lib/dependabot/file\_updaters/README.md) for more details. |

| `MetadataFinder` | Looks up metadata about a dependency, such as its GitHub URL. See the [README](https://github.com/dependabot/dependabot-core/tree/main/common/lib/dependabot/metadata\_finders/README.md) for more details. |

| `Version` | Describes the logic for comparing dependency versions. See the [hex Version class](https://github.com/dependabot/dependabot-core/blob/main/hex/lib/dependabot/hex/version.rb) for an example. |

| `Requirement` | Describes the format of a dependency requirement (e.g. `>= 1.2.3`). See the [hex Requirement class](https://github.com/dependabot/dependabot-core/blob/main/hex/lib/dependabot/hex/requirement.rb) for an example. |

The high-level flow looks like this:

<p align="center">

<img src="https://s3.eu-west-2.amazonaws.com/dependabot-images/package-manager-architecture.svg" alt="Dependabot architecture">

</p>

### `dependabot-omnibus`

This is a "meta" gem, that simply depends on all the others. If you want to

automatically include support for all languages, you can just include this gem

and you'll get all you need.

## Why is this public?

As the name suggests, Dependabot Core is the core of Dependabot (the rest of the

app is pretty much just a UI and database). If we were paranoid about someone

stealing our business then we'd be keeping it under lock and key.

Dependabot Core is public because we're more interested in it having an

impact than we are in making a buck from it. We'd love you to use

[Dependabot][dependabot] so that we can continue to develop it, but if you want

to build and host your own version then this library should make doing so a

\*lot\* easier.

If you use Dependabot Core then we'd love to hear what you build!

## License

We use the License Zero Prosperity Public License, which essentially enshrines

the following:

- If you would like to use Dependabot Core in a non-commercial capacity, such as

to host a bot at your workplace, then we give you full permission to do so. In

fact, we'd love you to and will help and support you however we can.

- If you would like to add Dependabot's functionality to your for-profit

company's offering then we DO NOT give you permission to use Dependabot Core

to do so. Please contact us directly to discuss a partnership or licensing

arrangement.

If you make a significant contribution to Dependabot Core then you will be asked

to transfer the IP of that contribution to Dependabot Ltd so that it can be

licensed in the same way as the above.

## History

Dependabot and Dependabot Core started life as [Bump][bump] and

[Bump Core][bump-core], back when Harry and Grey were working at

[GoCardless][gocardless]. We remain grateful for the help and support of

GoCardless in helping make Dependabot possible - if you need to collect

recurring payments from Europe, check them out.

[dependabot]: https://dependabot.com

[dependabot-status]: https://api.dependabot.com/badges/status?host=github&identifier=93163073

[dependabot-script]: https://github.com/dependabot/dependabot-script

[contributing]: https://github.com/dependabot/dependabot-core/blob/main/CONTRIBUTING.md

[api-docs]: https://github.com/dependabot/api-docs

[bump]: https://github.com/gocardless/bump

[bump-core]: https://github.com/gocardless/bump-core

[gocardless]: https://gocardless.com