

Guía para la resolución de los ejercicios

[1 - GIT cómo empezar](#)

[2- Manejo de lectura de datos](#)

1 - GIT cómo empezar

Cómo manejarnos en el proyecto de resolución de ejercicios y manejarse con GIT (a través de Bitbucket)


En esta prueba, la idea es que resuelvas algunos ejercicios. Nosotros iremos comentando tu código (code reviews), con observaciones o correcciones.

Vamos a manejarnos con Bitbucket, que es un servicio que nos permite tener código en internet y que varias personas a la vez puedan verlo y comentarlo.

Para utilizarlo:

- Si no tenés una cuenta, creá una en www.bitbucket.org. Si tenés una cuenta de gmail, usá el botón “loguearse via google”, sin crear usuario y clave.
- Creá el repositorio donde vas a subir el código
 - El nombre puede ser **msa_ejercicios**
 - No te olvides de configurarlo como privado!

Create a new repository [Import repository](#)

Workspace  ▼

Project* Select project ▼

Repository name* msa_ejercicios

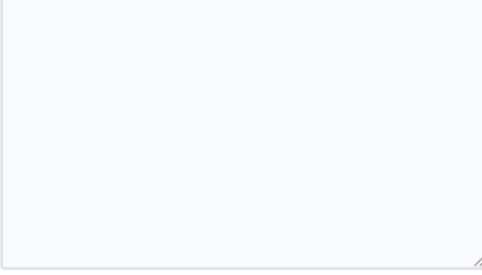
Access level ☒ Private repository
Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.

Include a README? No ▼

Default branch name e.g., 'main'

Include .gitignore? No ▼

▼ Advanced settings

Description 

Forking Allow only private forks ▼

Language Select language... ▼

[Create repository](#) [Cancel](#)

- Compartilo con el usuario **msa_reviews**. Mail: recruitingmsa.reviews@gmail.com. Este es el link con los pasos para hacerlo:

<https://support.atlassian.com/bitbucket-cloud/docs/grant-access-to-a-workspace/#Add-a-member-to-your-workspace>:

Lo importante es esta parte:

Add a member to your workspace


Once you've [created a workspace](#), you can start adding other users as members.

1. From your avatar, select the workspace or select **All workspaces** for a full list.
2. Click **Settings** in the left sidebar, and then select **User groups** under the Access Management heading.
3. Select the **Add members** button.
4. Enter an **email address** in the field provided.
Note: Entering the email address of a non-Bitbucket user sends an invitation to that person to create a Bitbucket account and join the workspace. You can add up to 10 people at a time.
5. Select the **User groups** field to display a list of existing groups in your workspace. Select the group that you want to add the member to, if necessary.
6. Click **Confirm**.
7. Repeat the last two steps for each user you want to add.

If you entered an email address and it has a corresponding Bitbucket account, the system resolves the account for you. If Bitbucket couldn't resolve the address, it sends the user an invitation to join the workspace by creating a Bitbucket account.

Una vez creado, el repositorio bitbucket explica cómo empezar. Seguí estos pasos para hacer el primer commit a la branch **master**.

[MSA Reviews](#) / [msa_ejercicios](#)



Repository setup

Your repository is empty — let's put some bits in your bucket.

Get code into Bitbucket fast using the command line

Command line

[I'm starting from scratch](#)

Set up your local directory

[Set up Git](#) on your machine if you haven't already.

```
$ mkdir /path/to/your/project
$ cd /path/to/your/project
$ git init
$ git remote add origin https://msa_reviews@bitbucket.org/msa_reviews/msa_ejercicios.git
```

Create your first file, commit, and push

```
$ echo "MSA Reviews" >> contributors.txt
$ git add contributors.txt
$ git commit -m 'Initial commit with contributors'
$ git push -u origin master
```

Great job, now you're all set up! Get started coding or [create a team](#) and invite people to work with you.

Para que se pueda hacer el review y comentar el código en bitbucket, es necesario crear una [branch](#) que salga de la branch principal (**master**).

Para cada ejercicio tiene que haber una branch distinta (Estas branches se pueden llamar ejercicio1, ejercicio2, etc.).

A continuación **todo el código** debe "comitearse y pushearse" ([Push code to Bitbucket](#)) en su branch correspondiente con estos archivos y carpetas:

- Una carpeta por problema, llamada **ejercicioN**, donde N es el número del problema a resolver, que contendrá un único archivo con la resolución del problema N, cuyo nombre será **EjercicioN.java**, **ejercicioN.py** ó el que corresponda según el lenguaje de programación usado.
- Dentro de la carpeta ejercicioN también deberá haber un archivo llamado **ejercicioN-solucion.txt** donde deberás explicar brevemente las ideas que surgieron durante la realización del ejercicio, así como comentarios sobre posibles mejoras, dudas, complicaciones que hayas tenido, y cualquier otra cosa que quieras agregar. La idea es que este archivo sea breve.
- Si el ejercicio necesita un input, en la misma carpeta deberá haber un archivo llamado **ejercicioN-input.txt** con algunos casos de ejemplo para usar de input. Este archivo debe contener casos básicos, no más de 4 o 5 inputs no muy largos. Si el ejercicio tenía inputs de ejemplo, con esos está bien (ver [2- Manejo de lectura de datos](#)).
- De ser necesario, se pueden agregar más archivos al directorio.

Todos los ejercicios deberán tener un **main** para ser ejecutado. Al correrse, el programa deberá resolver el problema planteado.

- Si el problema no recibe parámetros, el **main** no recibirá ningún parámetro, y simplemente deberá lograr el resultado pedido en el problema, realizando la salida via “standard output”
- Si el problema depende de una entrada, el **main** deberá recibir como parámetro la ubicación del archivo de input que deberá leer para obtener los datos necesarios, y generar la salida correspondiente.
- Ejemplo de ejecución para python: `python ejercicio1.py /path/al/archivo/input1.txt`
- Si está programado en un lenguaje precompilado no es necesario agregar el archivo compilado.

El objetivo de comitear el código en una branch es hacer un [pull request](#) (PR) hacia **master**, donde podemos hacer el code review.

Cuando consideres que un ejercicio está terminado y resuelto, crea un PR de la branch del ejercicio.

Al hacerlo tenés que [agregar](#) al usuario **msa_reviews** con el rol de reviewer.

En cada PR irás recibiendo comentarios de review de cada ejercicio.

Al terminar el review se irán incorporando los resultados a master ([git merge](#)).

2- Manejo de lectura de datos

Cómo generar la lectura de los datos necesarios para la resolución de un problema

Para obtener el input de los problemas que lo necesitan, habrá que leer un archivo de entrada, y éste deberá tener el formato que indiquen los problemas.

En cada archivo de entrada habrá uno o más casos de prueba, que se indicarán con una línea que dirá “# case x” donde “x” es el número del caso. Y luego vendrán los parámetros, y el formato está especificado

en cada problema. Por ejemplo, si un problema necesita como parámetros un número y una cadena (en ese orden según el problema), se darán dos líneas por caso, la primera con el número y la siguiente con la cadena.

Ejemplo:

```
# case 1
5
cadena de prueba del caso 1
# case 2
3
cadena de prueba del caso 2
```