# Contributing to this project

Please take a moment to review this document in order to make the contribution

process easy and effective for everyone involved.

Following these guidelines helps to communicate that you respect the time of

the developers managing and developing this open source project. In return,

they should reciprocate that respect in addressing your issue or assessing

patches and features.

## Using the issue tracker

The issue tracker is the preferred channel for [bug reports](#bugs),

[features requests](#features) and [submitting pull

requests](#pull-requests), but please respect the following restrictions:

\* Please \*\*do not\*\* use the issue tracker for personal support requests (use

our [igraph support forum](https://igraph.discourse.group)).

\* Please \*\*do not\*\* derail or troll issues. Keep the discussion on topic and

respect the opinions of others.

Please also take a look at our [tips on writing igraph code](#tips) before

getting your hands dirty.

<a name="bugs"></a>

## Bug reports

A bug is a \_demonstrable problem\_ that is caused by the code in the repository.

Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. \*\*Make sure that the bug is in the C code of igraph and not in one of the

higher level interfaces\*\* &mdash; if you are using igraph from R, Python

or Mathematica, consider submitting your issue in

[igraph/rigraph](https://github.com/igraph/rigraph/issues/new),

[igraph/python-igraph](https://github.com/igraph/python-igraph/issues/new)

or [szhorvat/IGraphM](https://github.com/szhorvat/IGraphM/issues/new)

instead. If you are unsure whether your issue is in the C layer, submit

a bug report in the repository of the higher level interface &mdash;

we will transfer the issue here if it indeed affects the C layer.

2. \*\*Use the GitHub issue search\*\* &mdash; check if the issue has already been

reported.

3. \*\*Check if the issue has been fixed\*\* &mdash; try to reproduce it using the

latest `master` or development branch in the repository.

4. \*\*Isolate the problem\*\* &mdash; create a [short, self-contained, correct

example](http://sscce.org/).

A good bug report shouldn't leave others needing to chase you up for more

information. Please try to be as detailed as possible in your report. What is

your environment? What steps will reproduce the issue? What would you expect to

be the outcome? All these details will help people to fix any potential bugs.

Example:

> Short and descriptive example bug report title

>

> A summary of the issue and the compiler/OS environment in which it occurs. If

> suitable, include the steps required to reproduce the bug.

>

> 1. This is the first step

> 2. This is the second step

> 3. Further steps, etc.

>

> `<url>` - a link to the reduced test case

>

> Any other information you want to share that is relevant to the issue being

> reported. This might include the lines of code that you have identified as

> causing the bug, and potential solutions (and your opinions on their

> merits).

<a name="features"></a>

## Feature requests

Feature requests are welcome. But take a moment to find out whether your idea

fits with the scope and aims of the project. It's up to \*you\* to make a strong

case to convince the project's developers of the merits of this feature. Please

provide as much detail and context as possible.

<a name="pull-requests"></a>

## Pull requests

Good pull requests - patches, improvements, new features - are a fantastic

help. They should remain focused in scope and avoid containing unrelated

commits.

\*\*Please ask first\*\* before embarking on any significant pull request (e.g.

implementing features, refactoring code, porting to a different language),

otherwise you risk spending a lot of time working on something that the

project's developers might not want to merge into the project.

Please adhere to the coding conventions used throughout a project (indentation,

accurate comments, etc.) and any other requirements (such as test coverage).

Follow this process if you'd like your work considered for inclusion in the

project:

1. [Fork](http://help.github.com/fork-a-repo/) the project, clone your fork,

and configure the remotes:

```bash

# Clone your fork of the repo into the current directory

git clone https://github.com/<your-username>/<repo-name>

# Navigate to the newly cloned directory

cd <repo-name>

# Assign the original repo to a remote called "upstream"

git remote add upstream https://github.com/<upstream-owner>/<repo-name>

```

2. Please checkout the section on [branching](#branching) to see whether you

need to branch off from the `master` branch or the `develop` branch.

If you cloned a while ago, get the latest changes from upstream:

```bash

git checkout <dev-branch>

git pull --rebase upstream <dev-branch>

```

3. Create a new topic branch (off the targeted branch, see

[branching](#branching) section) to contain your feature, change, or fix:

```bash

git checkout -b <topic-branch-name>

```

4. Commit your changes in logical chunks. Please adhere to these [git commit

message guidelines](http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html)

or your code is unlikely be merged into the main project. Use Git's

[interactive rebase](https://help.github.com/articles/interactive-rebase)

feature to tidy up your commits before making them public.

5. We have a handy [checklist for new igraph

functions](https://github.com/igraph/igraph/wiki/Checklist-for-new-(and-old)-functions).

If you have added any new functions to igraph, please go through the

checklist to ensure that your functions play nicely with the rest of the

library.

6. Locally merge (or rebase) the upstream development branch into your topic branch:

```bash

git pull [--rebase] upstream <dev-branch>

```

7. Push your topic branch up to your fork:

```bash

git push origin <topic-branch-name>

```

8. [Open a pull request](https://help.github.com/articles/using-pull-requests/)

with a clear title and description.

\*\*IMPORTANT\*\*: By submitting a pull request, you agree to allow the project owner to

license your work under the same license as that used by the project.

<a name="branching"></a>

### Branching

`igraph` is committed to [semantic versioning](https://semver.org/). We are currently still in the development release (0.x), which in principle is a mark that the public API is not yet stable. Regardless, we try to maintain semantic versioning also for the development releases. We do so as follows. Any released minor version (0.x.z) will be API backwards-compatible with any previous release of the \*same\* minor version (0.x.y, with y < z). This means that \*if\* there is an API incompatible change, we will increase the minor version. For example, release 0.8.1 is API backwards-compatible with release 0.8.0. A new release 0.9.0 will be API incompatible with version 0.8.1. Note that this only concerns the \*public\* API, internal functions may change also within a minor version.

There will always be two versions of `igraph`: the most recent released version, and the next upcoming minor release, which is by definition not yet released. The most recent release version is in the `master` branch, while the next upcoming minor release is in the `develop` branch. If you make a change that is API incompatible with the most recent release, it \*\*must\*\* be merged to the `develop` branch. If the change is API backwards-compatible, it \*\*can\*\* be merged to the `master` branch. It is possible that you build on recent improvements in the `develop` branch, in which case your change should of course target the `develop` branch. If you only add new functionality, but do not change anything of the existing API, this should be backwards-compatible, and can be merged in the `master` branch.

When you make a new pull request, please specify the correct target branch. The maintainers of `igraph` may decide to retarget your pull request to the correct branch. Retargeting you pull request may result in merge conflicts, so it is always good to decide \*\*before\*\* starting to work on something whether you should start from the `master` branch or from the `develop` branch. In most cases, changes in the `master` branch will also be merged to the `develop` branch by the maintainers.

<a name="tips"></a>

## Writing igraph Code

[Some tips on writing igraph code](https://github.com/igraph/igraph/wiki/Tips-on-writing-igraph-code).

## Ask Us!

In general, if you are not sure about something, please ask! You can

open an issue on GitHub, open a thread in our

[igraph support forum](https://igraph.discourse.group), or write to

[@ntamas](https://github.com/ntamas), [@vtraag](https://github.com/vtraag),

[@szhorvat](https://github.com/szhorvat) or

[@gaborcsardi](https://github.com/gaborcsardi).

We prefer the igraph support forum, because then others can learn from it

too.

## Legal Stuff

This is a pain to deal with, but we can't avoid it, unfortunately.

So, igraph is licensed under the "General Public License (GPL) version 2, or

later". The igraph manual is licensed under the "GNU Free Documentation

License". By submitting a patch or pull request, you agree to allow the project

owner to license your work under the same license as that used by the project.