git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/U-Mahajan/aa.git

git push -u origin main

……………….

Git bash used tool

git checkout branch-name

git push origin branch-name

To confirm that your branch is on the remote, list all branches:

bash

Copy code

git branch -r

**Use Cases for git rebase**

1. **Integrating Changes from Another Branch:**
   * Sync your branch with the latest changes from the base branch (e.g., main or master).
2. **Rewriting Commit History:**
   * Edit, squash, or reorder commits to clean up the history before sharing your changes.
3. **Avoiding Merge Commits:**
   * Keep a linear commit history by avoiding merge commits when integrating changes.

**Key Scenarios for git rebase**

**1. Rebase on Top of Another Branch**

To replay the commits from your branch onto the latest version of another branch (e.g., main):

bash

Copy code

# Switch to your feature branch

git checkout feature-branch

# Rebase onto the main branch

git rebase main

**Steps to Resolve Git Conflicts**

**1. Understand the Context**

* Conflicts typically arise during:
  + A **merge**: git merge branch-name.
  + A **rebase**: git rebase branch-name.
  + A **cherry-pick**: git cherry-pick commit-id.
* Git will pause the process and list the conflicting files.

**2. Identify Conflicted Files**

After a conflict occurs, Git marks the conflicting files as **unmerged**. Run the following command to see them:

bash

Copy code

git status

Example output:

plaintext

Copy code

On branch feature-branch

You have unmerged paths:

(use "git add <file>..." to mark resolution)

both modified: src/example.txt

**3. Open the Conflicted Files**

Git marks the conflicting sections in the file with **conflict markers**:

plaintext

Copy code

<<<<<<< HEAD

Code from your current branch (HEAD).

=======

Code from the branch you are merging or rebasing.

>>>>>>> branch-name

* **<<<<<<< HEAD**: Code from your current branch.
* **=======**: Separator between the two changes.
* **>>>>>>> branch-name**: Code from the branch you're merging or rebasing.

**4. Resolve the Conflict**

Manually edit the file to decide which changes to keep:

* Accept changes from one branch.
* Combine changes from both branches.
* Remove unnecessary lines.

For example: **Before Resolving:**

plaintext

Copy code

<<<<<<< HEAD

Current branch's code.

=======

Other branch's code.

>>>>>>> branch-name

**After Resolving:**

plaintext

Copy code

Resolved and combined code from both branches.

**5. Mark the Conflict as Resolved**

After editing and saving the file, mark it as resolved by adding it to the staging area:

bash

Copy code

git add <file>

For all conflicted files:

bash

Copy code

git add .

git clone https://github.com/madaan123/MyAlgorithms.git

git pull is a Git command used to fetch and integrate changes from a remote repository into your local branch. It's a combination of two commands:

1. **git fetch**: Downloads updates from the remote repository.
2. **git merge**: Merges those updates into your current branch.

**Usage**

bash

Copy code

git pull <remote> <branch>

* **<remote>**: The name of the remote repository (default is origin).
* **<branch>**: The name of the branch you want to pull changes from.