# Git

## Git教程

代码托管平台：git.acwing.com

#### git基本概念

1. 工作区：仓库的目录。工作区是独立于各个分支的。
2. 暂存区：数据暂时存放的区域，类似于工作区写入版本库前的缓存区。暂存区是独立于各个分支的。
3. 版本库：存放所有已经提交到本地仓库的代码版本
4. 版本结构：树结构，树中每个节点代表一个代码版本。

#### git常用命令

1. git config --global user.name xxx：设置全局用户名，信息记录在~/.gitconfig文件中
2. git config --global user.email xxx@xxx.com：设置全局邮箱地址，信息记录在~/.gitconfig文件中
3. git init：将当前目录配置成git仓库，信息记录在隐藏的.git文件夹中
4. git add XX：将XX文件添加到暂存区

git add .：将所有待加入暂存区的文件加入暂存区

1. git rm --cached XX：将文件从仓库索引目录中删掉
2. git commit -m "给自己看的备注信息"：将暂存区的内容提交到当前分支
3. git status：查看仓库状态
4. git diff XX：查看XX文件相对于暂存区修改了哪些内容
5. git log：查看当前分支的所有版本
6. git reflog：查看HEAD指针的移动历史（包括被回滚的版本）
7. git reset --hard HEAD^ 或 git reset --hard HEAD~：将代码库回滚到上一个版本

git reset --hard HEAD^^：往上回滚两次，以此类推

git reset --hard HEAD~100：往上回滚100个版本

git reset --hard 版本号：回滚到某一特定版本

1. git checkout — XX或git restore XX：将XX文件尚未加入暂存区的修改全部撤销
2. git remote add origin git@git.acwing.com:xxx/XXX.git：将本地仓库关联到远程仓库
3. git push -u (第一次需要-u以后不需要)：将当前分支推送到远程仓库

git push origin branch\_name：将本地的某个分支推送到远程仓库

1. git clone git@git.acwing.com:xxx/XXX.git：将远程仓库XXX下载到当前目录下
2. git checkout -b branch\_name：创建并切换到branch\_name这个分支
3. git branch：查看所有分支和当前所处分支
4. git checkout branch\_name：切换到branch\_name这个分支
5. git merge branch\_name：将分支branch\_name合并到当前分支上
6. git branch -d branch\_name：删除本地仓库的branch\_name分支
7. git branch branch\_name：创建新分支
8. git push --set-upstream origin branch\_name：设置本地的branch\_name分支对应远程仓库的branch\_name分支
9. git push -d origin branch\_name：删除远程仓库的branch\_name分支
10. git pull：将远程仓库的当前分支与本地仓库的当前分支合并

git pull origin branch\_name：将远程仓库的branch\_name分支与本地仓库的当前分支合并

1. git branch --set-upstream-to=origin/branch\_name1 branch\_name2：将远程的branch\_name1分支与本地的branch\_name2分支对应
2. git checkout -t origin/branch\_name 将远程的branch\_name分支拉取到本地
3. git stash：将工作区和暂存区中尚未提交的修改存入栈中
4. git stash apply：将栈顶存储的修改恢复到当前分支，但不删除栈顶元素
5. git stash drop：删除栈顶存储的修改
6. git stash pop：将栈顶存储的修改恢复到当前分支，同时删除栈顶元素
7. git stash list：查看栈中所有元素

## 2. 创建作业 & 测试作业的正确性

homework 5 create # 可以重新创建所有lesson\_5的作业

homework 5 test # 可以评测lesson\_5的所有作业

## 3. 作业

注意： 本次作业的10个题目不是独立的，每个题目会依赖于前一个题目，因此评测时如果当前题目错误，则不再评测后续题目。

创建好作业后，先进入文件夹/home/acs/homework/lesson\_5/，然后：

(0) 在当前目录下创建文件夹homework，并将homework目录配置成git仓库。后续作业均在homework目录下操作；

(1) 创建文件readme.txt，内容包含一行：111；

将修改提交一个commit；

(2) 在readme.txt文件末尾新增一行：222；

将修改提交一个commit；

(3) 创建文件夹：problem1和problem2；

创建文件problem1/main.cpp。文件内容为下述链接中的代码：https://www.acwing.com/problem/content/submission/code\_detail/7834813/；

创建文件problem2/main.cpp。文件内容为下述链接中的代码：https://www.acwing.com/problem/content/submission/code\_detail/7834819/；

将修改提交一个commit；

(4) 删除文件夹problem2；

创建文件夹problem3；

创建文件problem3/main.cpp。文件内容为下述链接中的代码：https://www.acwing.com/problem/content/submission/code\_detail/7834841/；

将readme.txt中最后一行222删掉，然后添加一行333；

将修改提交一个commit；

(5) 在https://git.acwing.com/上注册账号并创建仓库，仓库名称为homework；

将本地git仓库上传到AC Git云端仓库；

(6) 创建并切换至新分支dev；

在readme.txt文件中添加一行444；

将修改提交一个commit；

将dev分支推送至AC Git远程仓库；

(7) 切换回master分支；

在readme.txt文件中添加一行555；

将修改提交一个commit；

(8) 将dev分支合并到master分支；

手动处理冲突，使readme文件最终内容包含4行：111、333、555、444；

将修改提交一个commit；

(9) 将master分支的版本库push到AC Git云端仓库；

登录myserver服务器（4. ssh作业中配置的服务器）；

创建并清空文件夹：~/homework/lesson\_5/；

将AC Git云端仓库clone到~/homework/lesson\_5/中；

作业代码：

Homework\_0

mkdir homework

git init

mkdir homework

cd homework

git init

Homework\_1

vim readme.txt

git add .

git commit -m "111"

vim readme.txt

\*\*\*

111

\*\*\*

git add .

git commit -m "add readme.txt"

Homework\_2

vim readme.txt

git add .

git commit -m "222"

vim readme.txt

\*\*\*

111

222

\*\*\*

git add .

git commit -m "add 222"

Homework\_3

mkdir problem1 problem2

cd problem1

vim main.cpp

cd ../problem2

vim main.cpp

cd ..

git add .

git status

git commit -m "main.cpp"

mkdir problem1 problem2

cd problem1

vim main.cpp

\*\*\*

#include <iostream>

using namespace std;

int main()

{

int a, b;

cin >> a >> b;

cout << a + b << endl;

return 0;

}

\*\*\*

cd ../problem2

vim main.cpp

\*\*\*

#include <iostream>

using namespace std;

const int N = 1010;

int n, m;

int f[N];

int main()

{

cin >> n >> m;

while (n -- )

{

int v, w;

cin >> v >> w;

for (int j = m; j >= v; j -- )

f[j] = max(f[j], f[j - v] + w);

}

cout << f[m] << endl;

return 0;

}

\*\*\*

cd ..

git add .

git commit -m "add problem1 problem2"

Homework\_4

rm problem2 -r

mkdir problem3

cd problem3

vim main.cpp

cd ..

vim readme.txt

git add .

git commit -m "rm"

rm problem2 -r

mkdir problem3

cd problem3

vim main.cpp

\*\*\*

#include <iostream>

using namespace std;

const int N = 1010;

int n, m;

int f[N];

int main()

{

cin >> n >> m;

while (n -- )

{

int v, w;

cin >> v >> w;

for (int j = v; j <= m; j ++ )

f[j] = max(f[j], f[j - v] + w);

}

cout << f[m] << endl;

return 0;

}

\*\*\*

cd ..

vim readme.txt

\*\*\*

111

333

\*\*\*

git add .

git commit -m "many operations"

Homework\_5

git remote rm origin

git remote add origin [git@git.acwing.com:WilliamsWhites/homework.git](mailto:git@git.acwing.com:WilliamsWhites/homework.git)

git push -u origin master

本地操作

git remote add origin git@git.acwing.com:yxc/homework.git

git push -u origin master

Homework\_6

git branch dev

git checkout dev

vim readme.txt

git add .

git status

git push --set-upstream origin dev

git checkout -b dev

vim readme.txt

\*\*\*

111

333

444

\*\*\*

git add .

git commit -m "add 444"

git push origin dev

Homework\_7

git checkout master

vim readme.txt

git add .

git commit -m "555"

git checkout master # 切换回master分支

vim readme.txt

\*\*\*

111

333

555

\*\*\*

git add .

git commit -m "add 555"

Homework\_8

git merge dev

vim readme.txt

git status

git add .

git commit -m "fix conflict"

git branch

git log

git merge dev # 将dev分支合并到当前分支

vim readme.txt

\*\*\*

111

333

555

444

\*\*\*

git add .

git commit -m "fix conflicts"

Homework\_9

ssh myserver

cd homework

mkdir lesson\_5

cd lesson\_5

git clone git@git.acwing.com:yxc/homework.git

mkdir lesson\_5

cd lesson\_5

cd ~

cd .ssh/

ssh-keygen

cat id\_rsa.pub

cd ..

cd homework/

cd lesson\_5

git clone git@git.acwing.com:WilliamsWhites/homework.git