WEEK2

段雷

目 录

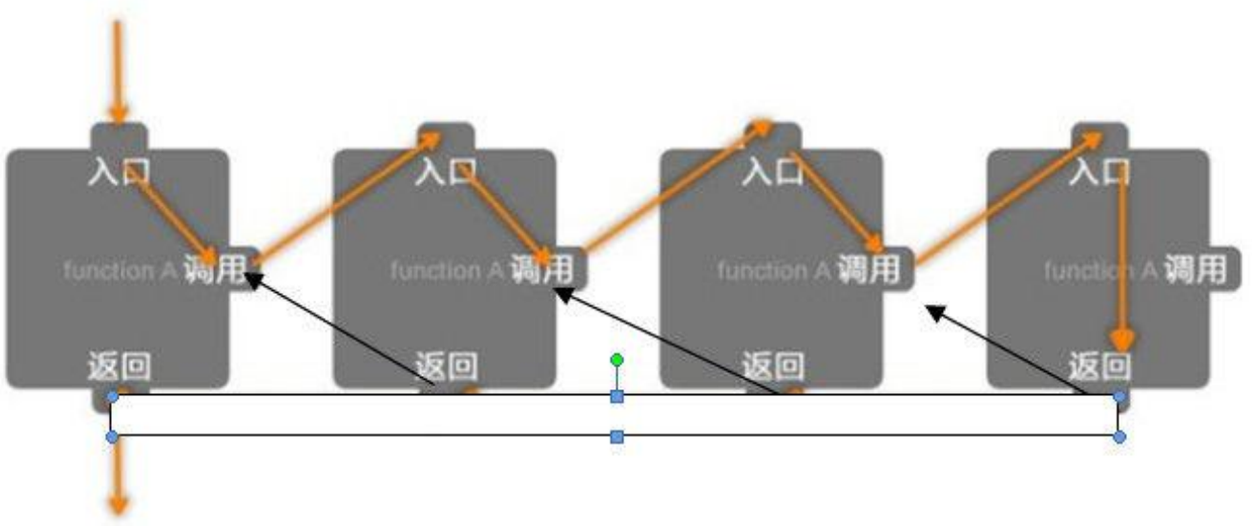
CONTENTS

01递归

递归

从前有座山，山里有座庙，庙里有个和尚，和尚在讲故事，从前有座山，山里有座庙，庙里有个和尚，和尚在讲故事，从前有座山...

要理解递归，就得先了解什么是递归



递归

```python
def recursion(depth):
    print("抱着")
    if depth == 0:
        print("我的小鲤鱼")
    else:
        recursion(depth - 1)
    print("的我")

print("吓得我抱起了")
recursion(2)
```

```
~/Desktop/a python3 a.py
吓得我抱起了
抱着
抱着
抱着
我的小鲤鱼
的我
的我
的我
```

递归

1.1 斐波那契数列 (def fib(n))

求斐波那契数列的第n项

```
1  def fib(n):
2      if n == 0 or n == 1:
3          return 1
4      return fib(n - 1) + fib(n - 2)
```

1.2 遍历文件 (def file_display(filepath))

遍历文件夹里的所有文件（包括子文件中的）

os.listdir(path)　　显示path路径下的所有文件和文件夹
os.path.join(path, name)　　将path和name连接成路径
os.path.isfile(filepath)　　判断filepath是否是一个文件

递归

1.2 遍历文件 (def file_display(filepath))

```python
import os

def file_display(filepath):
    for each in os.listdir(filepath):
        new_path = os.path.join(filepath, each)
        if os.path.isfile(new_path):
            print(each)
        else:
            file_display(new_path)
```

递归

1.3 Flip Game (poj 1753)

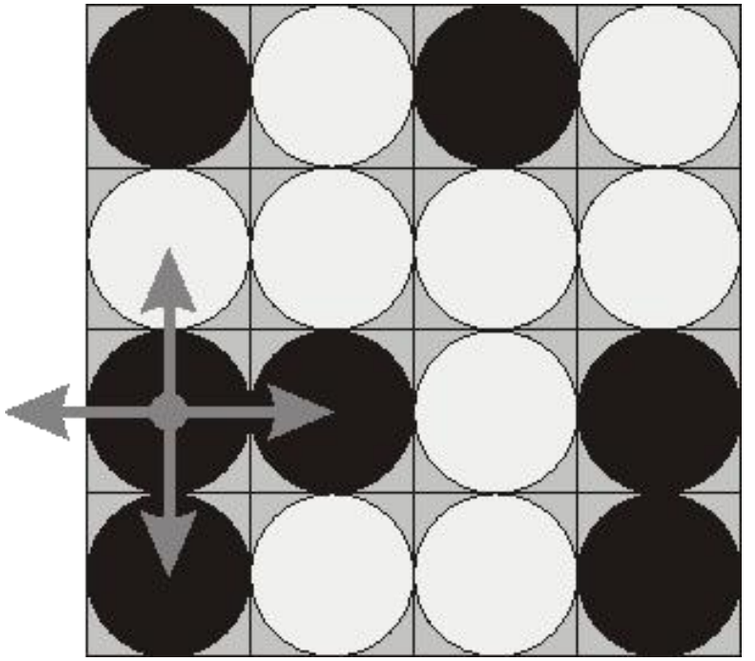
有4*4的正方形，每个格子要么是黑色，要么是白色，当把一个格子的颜色改变(黑->白或者白->黑)时，其周围上下左右(如果存在的话)的格子的颜色也被反转，问至少反转几个格子可以使4*4的正方形变为纯白或者纯黑?

Sample Input:

```
bwwb
bbwb
bwwb
bwww
```

Sample Output:

```
4
```



```
bwwb
bbwb
bwwb
bwww

wbwb
wbwb
bwwb
bwww

bwbb
wwwb
bwwb
bwww

bbbb
bbbb
bbwb
bwww

bbbb
bbbb
bbbb
bbbb
```

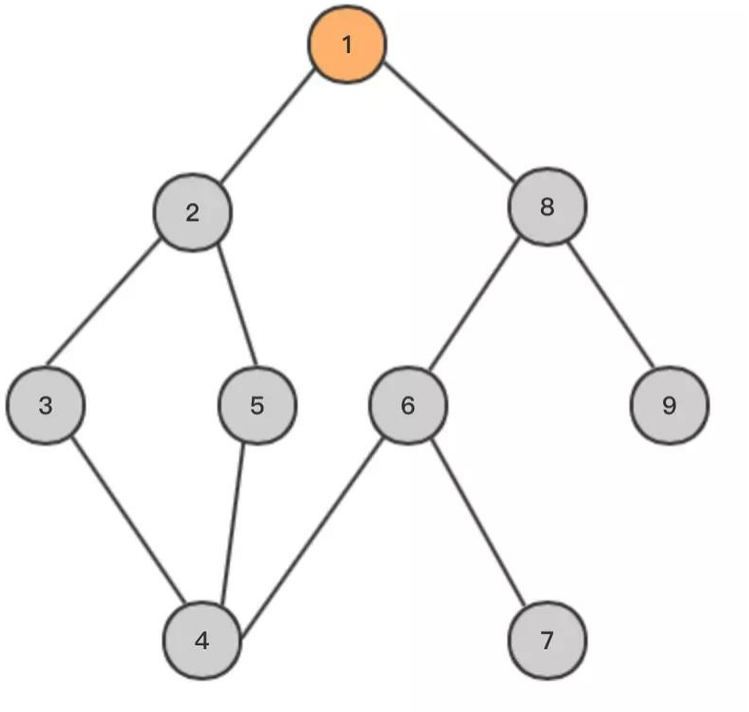
递归

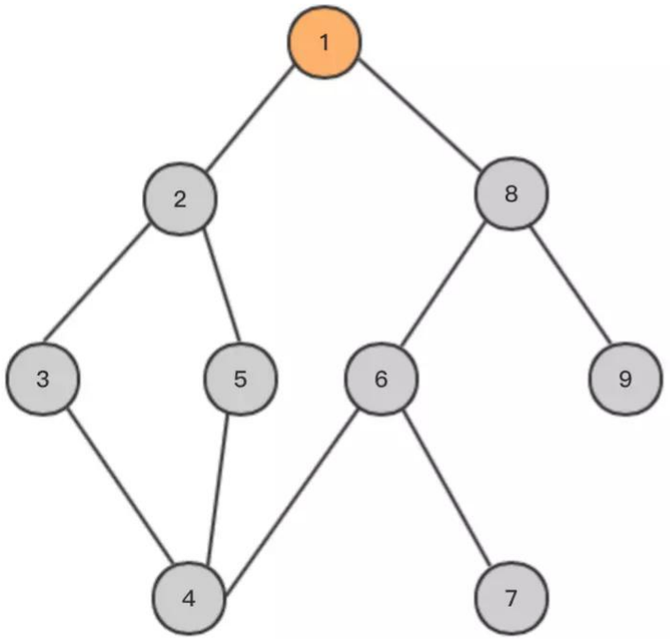
1.3 Flip Game (poj 1753)

```
36  bool Dfs(int x, int y, int d)
37  {
38      if(x == 4)
39          return check();
40      b[x][y] = 0;
41      if(Dfs(get_x(x, y), get_y(x, y), d))
42          return true;
43      b[x][y] = 1;
44      a[x][y] ^= 1;
45      if(x > 0) a[x-1][y] ^= 1;
46      if(x < 3) a[x+1][y] ^= 1;
47      if(y > 0) a[x][y-1] ^= 1;
48      if(y < 3) a[x][y+1] ^= 1;
49      if(Dfs(get_x(x, y), get_y(x, y), d + 1))
50          return true;
51      b[x][y] = 0;
52      a[x][y] ^= 1;
53      if(x > 0) a[x-1][y] ^= 1;
54      if(x < 3) a[x+1][y] ^= 1;
55      if(y > 0) a[x][y-1] ^= 1;
56      if(y < 3) a[x][y+1] ^= 1;
57      return false;
58  }
```

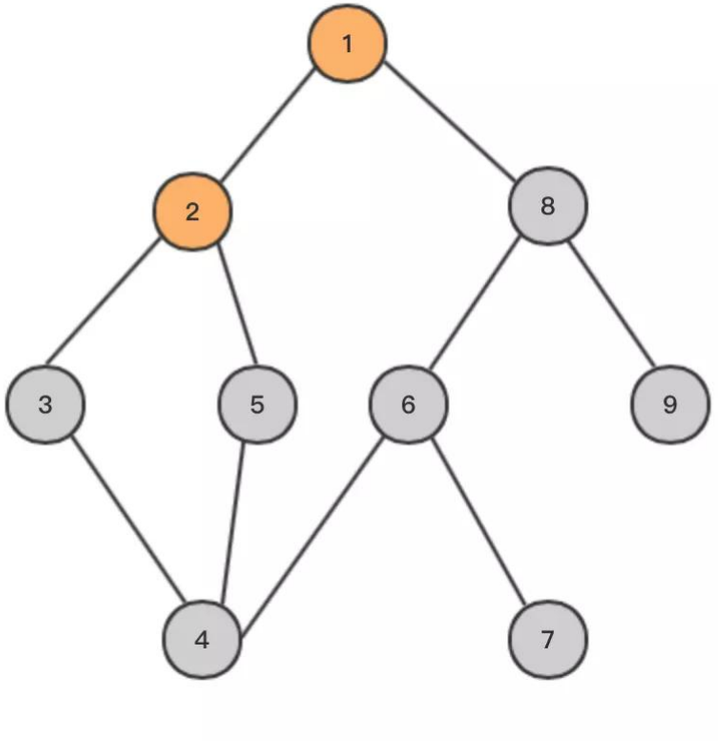
02深度优先搜索

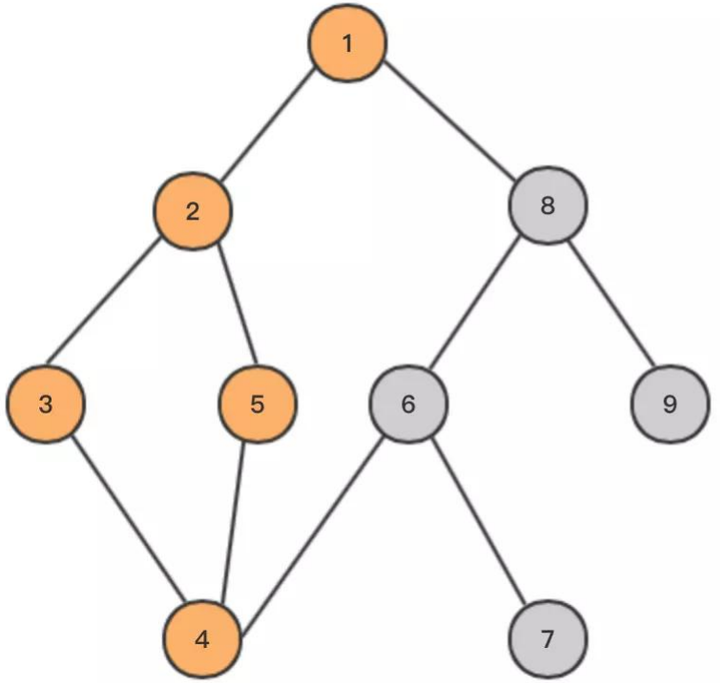
dfs

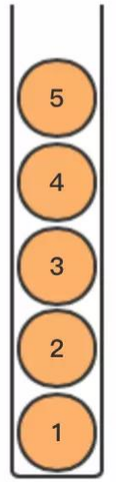




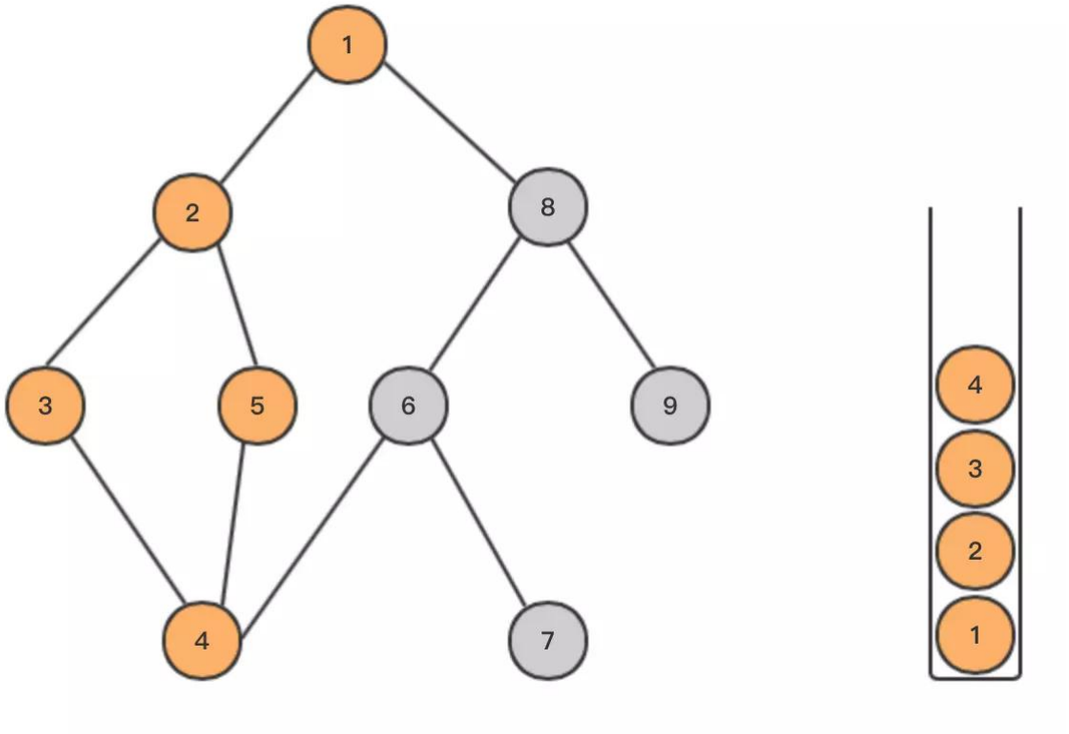
dfs

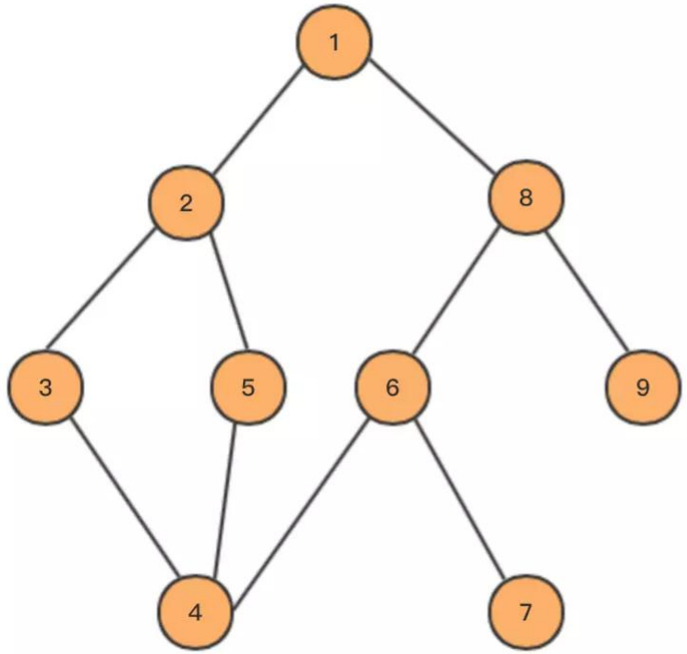


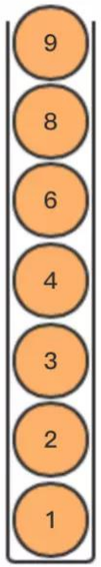




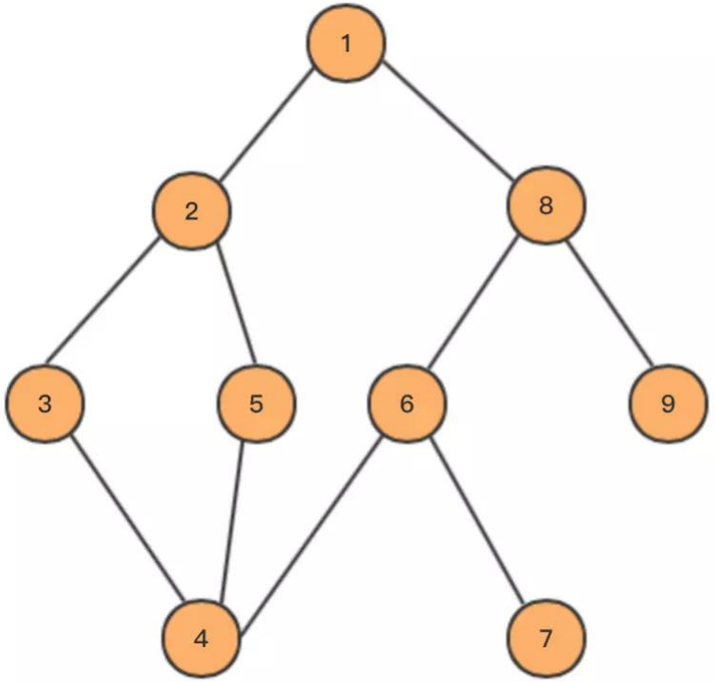
dfs



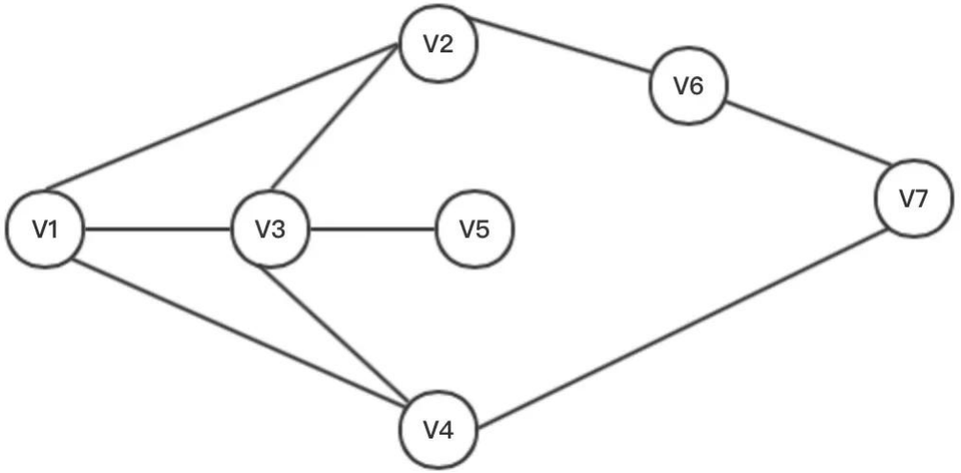


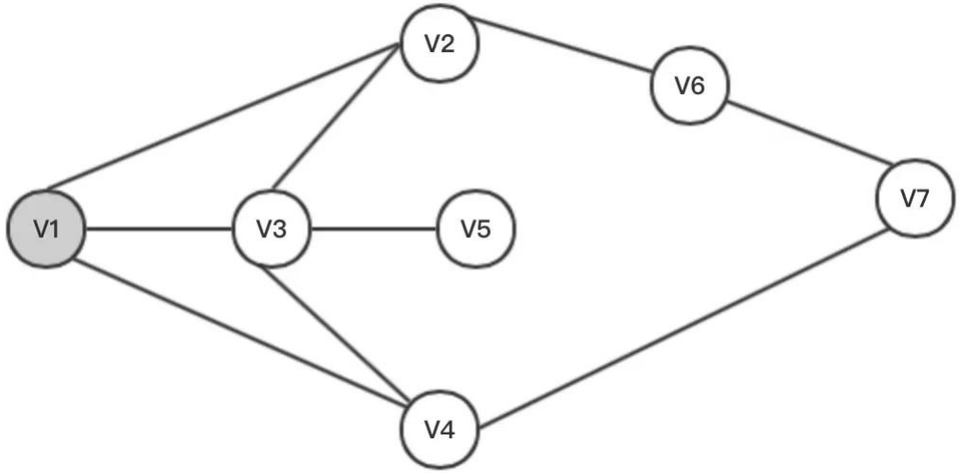


dfs

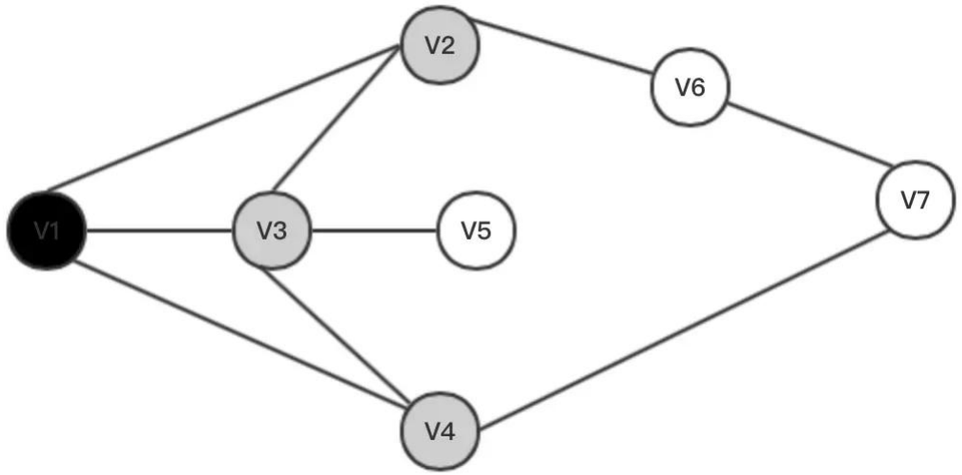


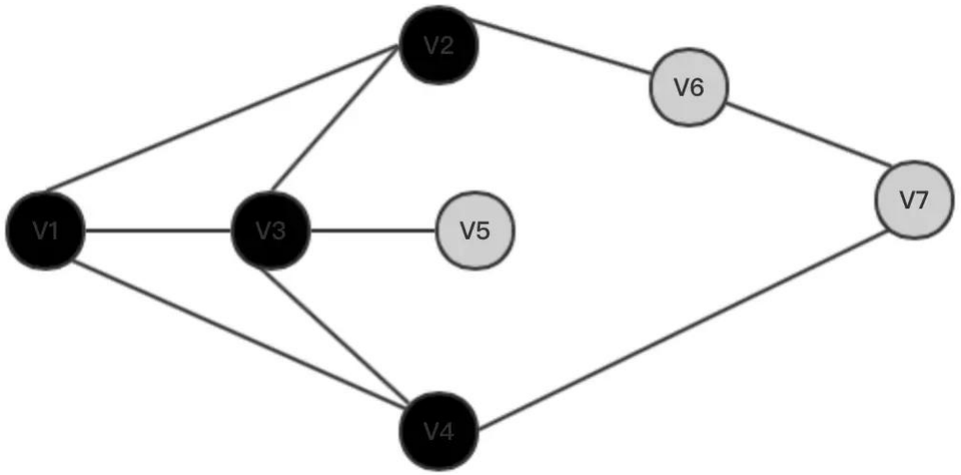
bfs



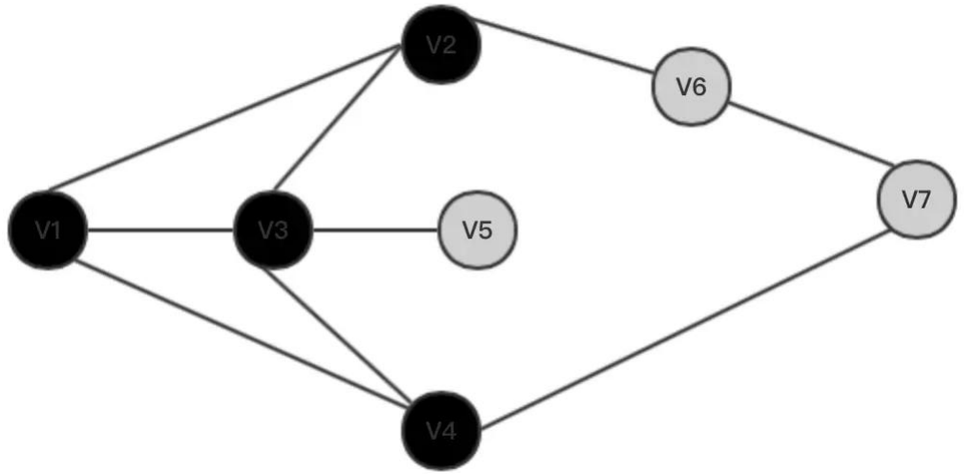


bfs





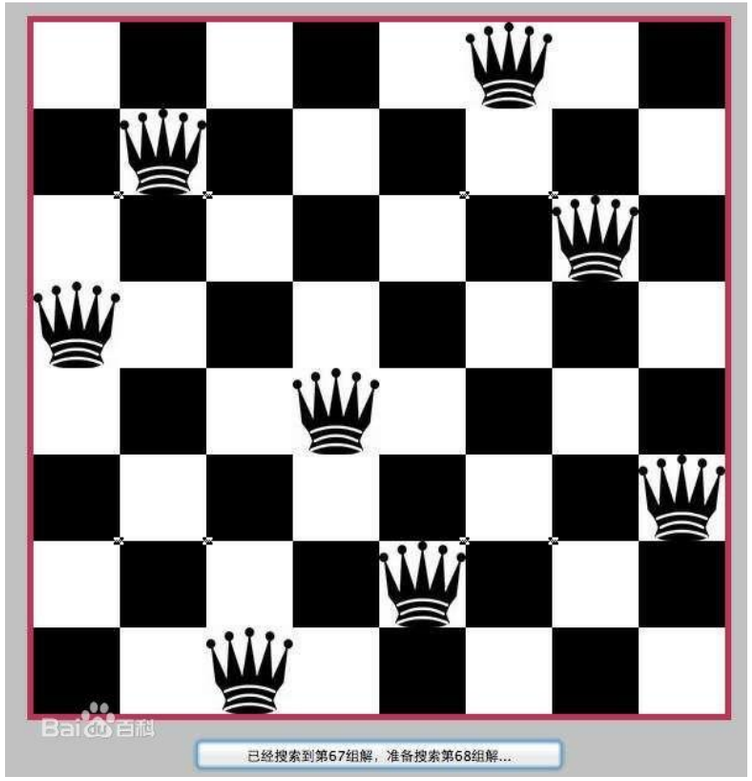
bfs



dfs

2.1 八皇后问题

会下国际象棋的人都很清楚：皇后可以在横、竖、斜线上不限步数地吃掉其他棋子。如何将8个皇后放在棋盘上（有8 * 8个方格），使它们谁也不能被吃掉，这就是著名的八皇后问题。



bfs

2.2 迷宫问题

Description

定义一个二维数组：

```
int maze[5][5] = {
        0, 1, 0, 0, 0,
        0, 1, 0, 1, 0,
        0, 0, 0, 0, 0,
        0, 1, 1, 1, 0,
        0, 0, 0, 1, 0,
};
```

它表示一个迷宫，其中的1表示墙壁，0表示可以走的路，只能横着走或竖着走，不能斜着走，要求编程序找出从左上角到右下角的最短路线。

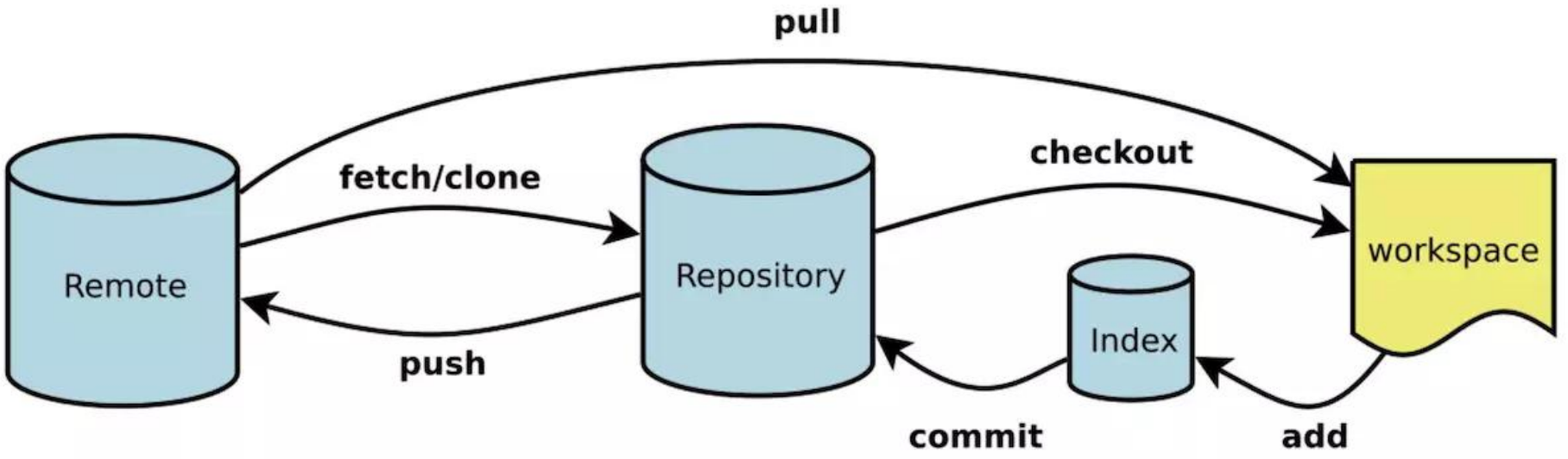
03git讲解

git

- 免费
- 管理历史代码版本
- 代码网盘
- 多人协作

- 非纯文本不上传
- 封号警告

git



Remote:远端仓库
Repository:本地仓库
Index:缓存区
workspace:工作区

github - 1 创建github账号

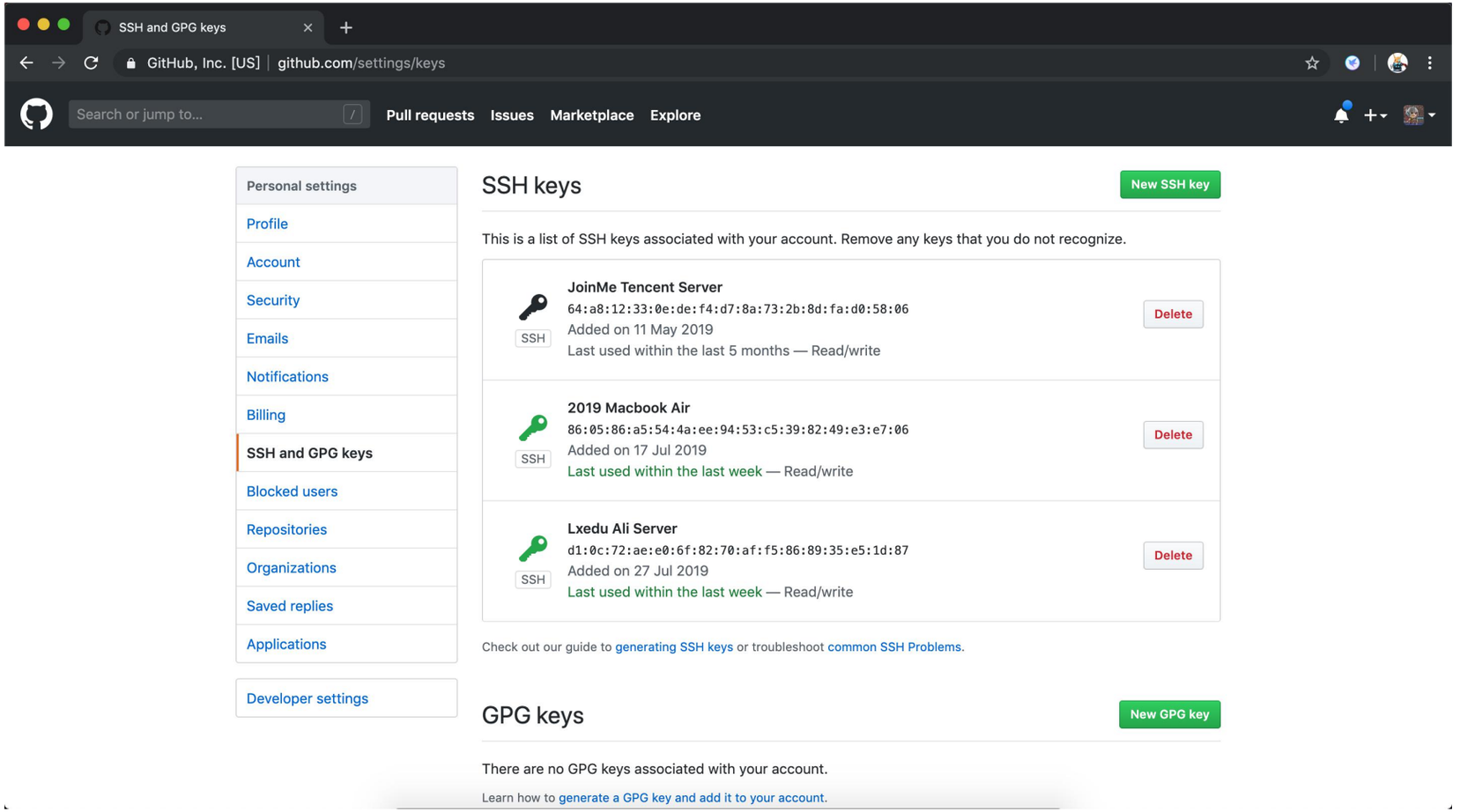
github - 2 创建ssh key

$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

一路回车（包括输入密码）

~/.ssh/id_rsa.pub 中复制公钥

粘贴到
github的SSH and GPG keys



github - 3 设置本地git信息

$ ssh -T git@github.com

第一次会提示是否continue，输入yes后显示：
You've successfully authenticated, but GitHub does not provide shell access
即成功

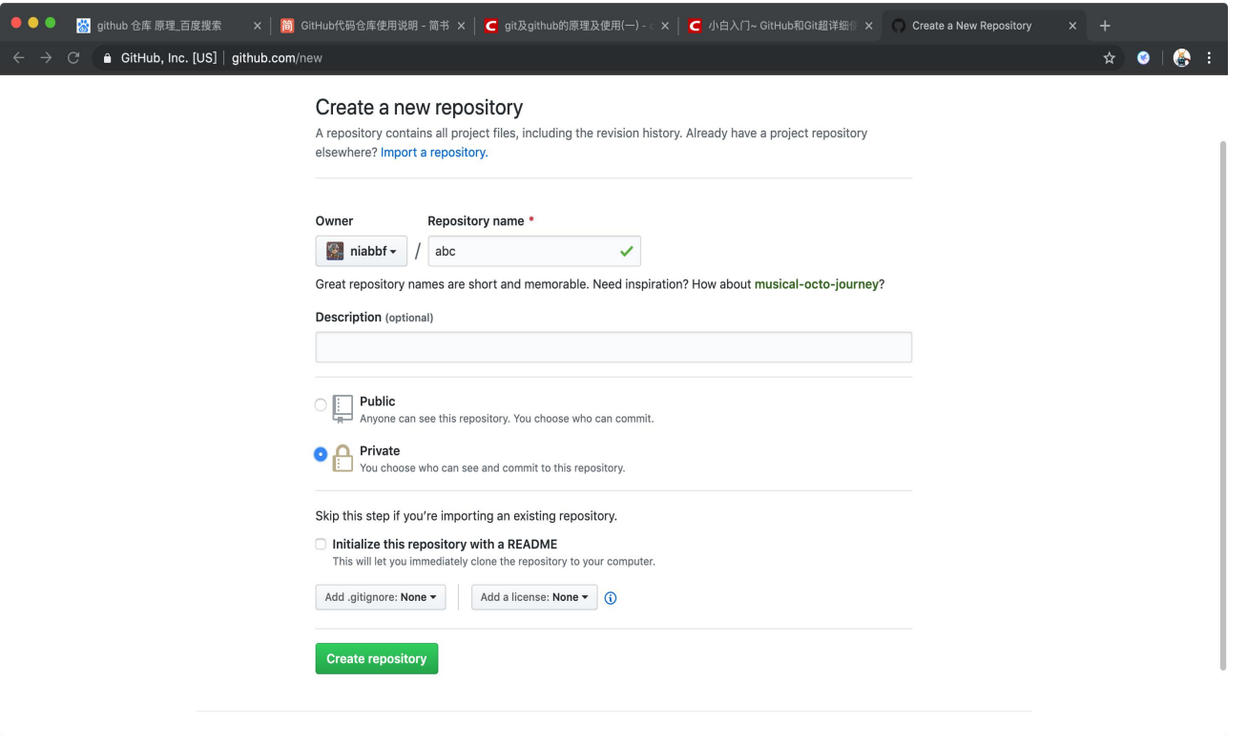
$ git config --global user.name "niabbf"
$ git config --global user.email "duanlei601@163.com"

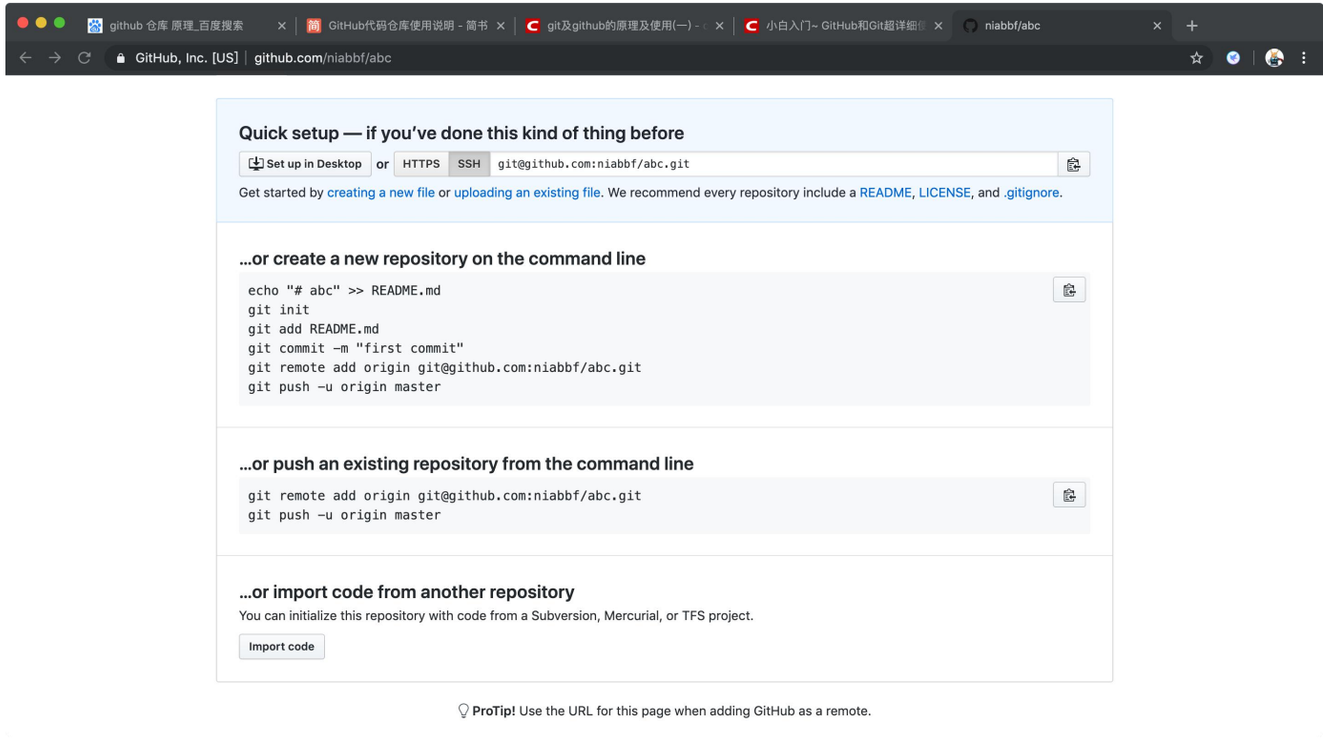
github - 4 常用指令

创建项目：

```
git init
git remote add origin git@xxxx
git add .
git commit –m “comment”
git push –u origin master
```

初始化项目
将本地仓库与远程仓库建立连接
将当前所有变化添加到暂存区
将暂存区添加到本地仓库
将本地仓库“推”到远程仓库





github - 4 常用指令

日常使用：

git clone	拷贝项目
git add .	将修改添加到暂存区
git commit -m “comment”	将暂存区内容添加到本地仓库
git push	将本地仓库“推”到远程仓库
git pull	将远程仓库修改“拉”到本地仓库
git status	查看文件修改情况
git log	查看commit记录 (:q退出，由于默认有一个:，所以按q即退出)
git log --pretty=oneline	单行显示commit记录

github - 5 “少”用指令

很少使用：

git branch	创建分支
git checkout	切换分支
git reset	版本回退
git merge	版本合并

04题目讲解

GS 20190904 OT P1

有一个 n*m 的数字棋盘，两个人轮流从里面挑选数字，每个人的得分是挑选过的数字之和。两个人的目的都是让自己的得分尽可能高 。如果有一个人挑选了(i,j)这个位置的数字，那么第j列其他数字都不能再被挑选。问两人都采取最优策略下，第一个人的得分减第二个的得分为多少?

Sample Input:

3 6
3 7 5 3 4 5
4 5 2 6 5 4
7 4 9 7 8 3

Sample Output:

3

GS 20190904 OT P2

0~9每个数字都会被映射到0~9中的数字，且构成双射。现在给出映射后的数字列表，要求将这个数字列表按原数字大小从小到大排序后输出，如果原数字大小相同，那么按照映射后数字出现次序排列。

Sample Input:

```
10
3 5 4 6 2 7 9 8 0 1
3
990
332
32
```

(-> 669)
(-> 004)
(-> 04)

Sample Output:

```
332
32
990
```

HW 1
leetcode191

191. Number of 1 Bits

Easy 499 420 Favorite Share

Write a function that takes an unsigned integer and return the number of '1' bits it has (also known as the Hamming weight).

Example 1:

```
Input: 00000000000000000000000000001011
Output: 3
Explanation: The input binary string 00000000000000000000000000001011 has a total of three '1' bits.
```

Example 2:

```
Input: 00000000000000000000000010000000
Output: 1
Explanation: The input binary string 00000000000000000000000010000000 has a total of one '1' bit.
```

Example 3:

```
Input: 11111111111111111111111111111101
Output: 31
Explanation: The input binary string 11111111111111111111111111111101 has a
```

HW 2
leetcode201

201. Bitwise AND of Numbers Range

Medium 488 66 Favorite Share

Given a range [m, n] where 0 <= m <= n <= 2147483647, return the bitwise AND of all numbers in this range, inclusive.

Example 1:

```
Input: [5,7]
Output: 4
```

Example 2:

```
Input: [0,1]
Output: 0
```

HW 3
leetcode338

338. Counting Bits

Medium 1615 116 Favorite Share

Given a non negative integer number **num**. For every numbers **i** in the range **0 ≤ i ≤ num** calculate the number of 1's in their binary representation and return them as an array.

Example 1:

```
Input: 2
Output: [0,1,1]
```

Example 2:

```
Input: 5
Output: [0,1,1,2,1,2]
```

Follow up:

- It is very easy to come up with a solution with run time **O(n*sizeof(integer))**. But can you do it in linear time **O(n)** /possibly in a single pass?
- Space complexity should be **O(n)**.
- Can you do it like a boss? Do it without using any builtin function like **__builtin_popcount** in c++ or in any other language.

HW 4
leetcode162

162. Find Peak Element

Medium 1000 1504 Favorite Share

A peak element is an element that is greater than its neighbors.

Given an input array `nums`, where `nums[i] ≠ nums[i+1]`, find a peak element and return its index.

The array may contain multiple peaks, in that case return the index to any one of the peaks is fine.

You may imagine that `nums[-1] = nums[n] = -∞`.

Example 1:

```
Input: nums = [1,2,3,1]
Output: 2
Explanation: 3 is a peak element and your function should return the index
number 2.
```

Example 2:

```
Input: nums = [1,2,1,3,5,6,4]
Output: 1 or 5
Explanation: Your function can return either index number 1 where the peak
element is 2,
             or index number 5 where the peak element is 6.
```