

Recursion

王慧妍

why@nju.edu.cn

南京大学



计算机科学与技术系



计算机软件研究所



特殊的函数：递归Recursion

- 一个调用本身的函数
 - A function that calls (调用) itself.
- 重点是如何能够递归的思考问题



It's a loooooooooong way to go to master recursion!

Mathematical induction

- 数学归纳法

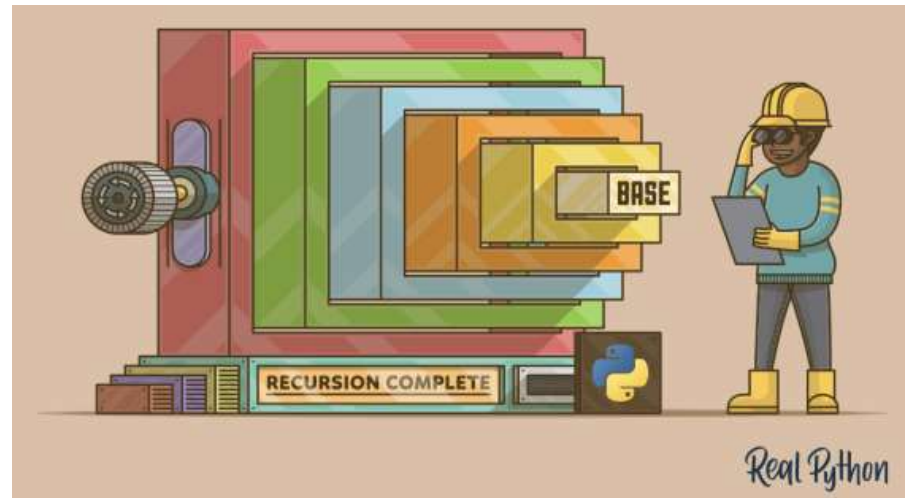
- Base Case (基础情况): $n = 0$
- Inductive Step (归纳步骤): $n = k \rightarrow n = k+1$

- 递归函数：自己调用自己的函数

- 不能变：函数名称，功能，返回类型
- 唯一能变的部分：参数

- 通过参数控制问题的解决规模

- 何时结束？



递归函数的堆栈管理

- [Visualization of Function Calls @ C Tutor](#)

求阶乘

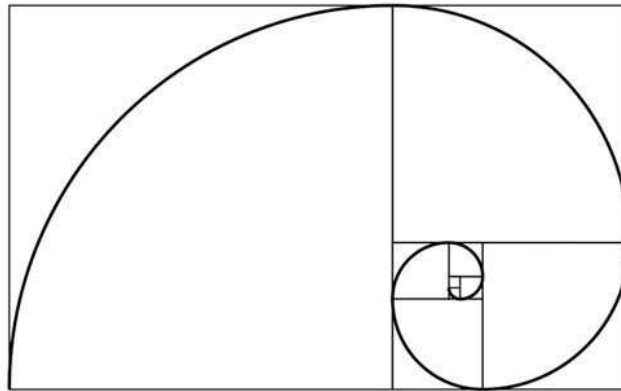
- $F(n) = n!$

$$n! = \begin{cases} 1 & (n = 0, 1) \\ n * (n - 1)! & (n > 1) \end{cases}$$

- 循环 v.s. 递归

斐布拉契数列

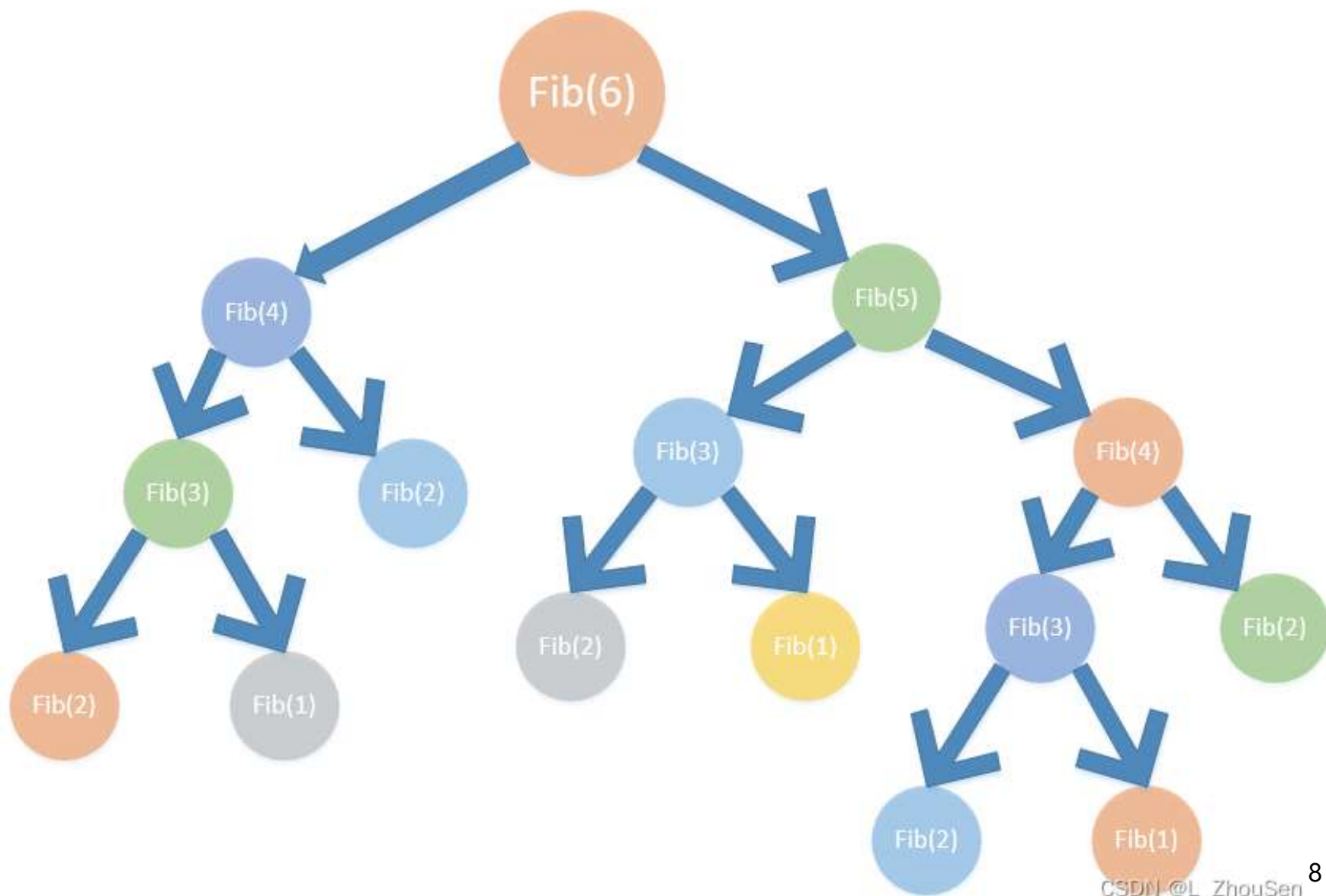
- 典型递归函数例子
 - [fib.c](#)
- 黄金分割数列：0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
 - $F(0) = 0, F(1) = 1, F(n) = F(n - 1) + F(n - 2)$

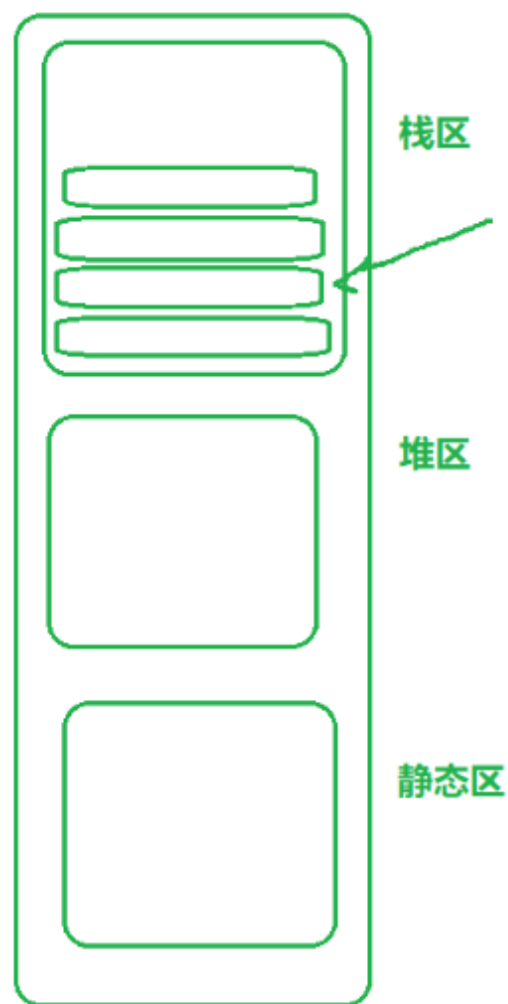


斐布拉契字符串

- $f(0) = b, f(1) = a,$
- $f(2) = f(1) + f(0) = ab,$
- $f(3) = f(2) + f(1) = aba,$
- $f(4) = f(3) + f(2) = abaab,$
- $f(n) = ?$

- [fib.c](#)
- [fib_string.c](#)
- [fib_long_iter.c](#)
- [fib_long_iter_space.c](#)





每一次函数调用都会在内存的栈区申请一块空间,直到栈区没有空间进行函数调用导致栈溢出

Greatest Common Divisor

- [gcd.c](#)

$$a > b \implies \gcd(a, b) = \gcd(a - b, b)$$

$$a < b \implies \gcd(a, b) = \gcd(a, b - a)$$



$$\gcd(a, b) = \gcd(b, a \% b)$$

数组与递归

- 数组求和

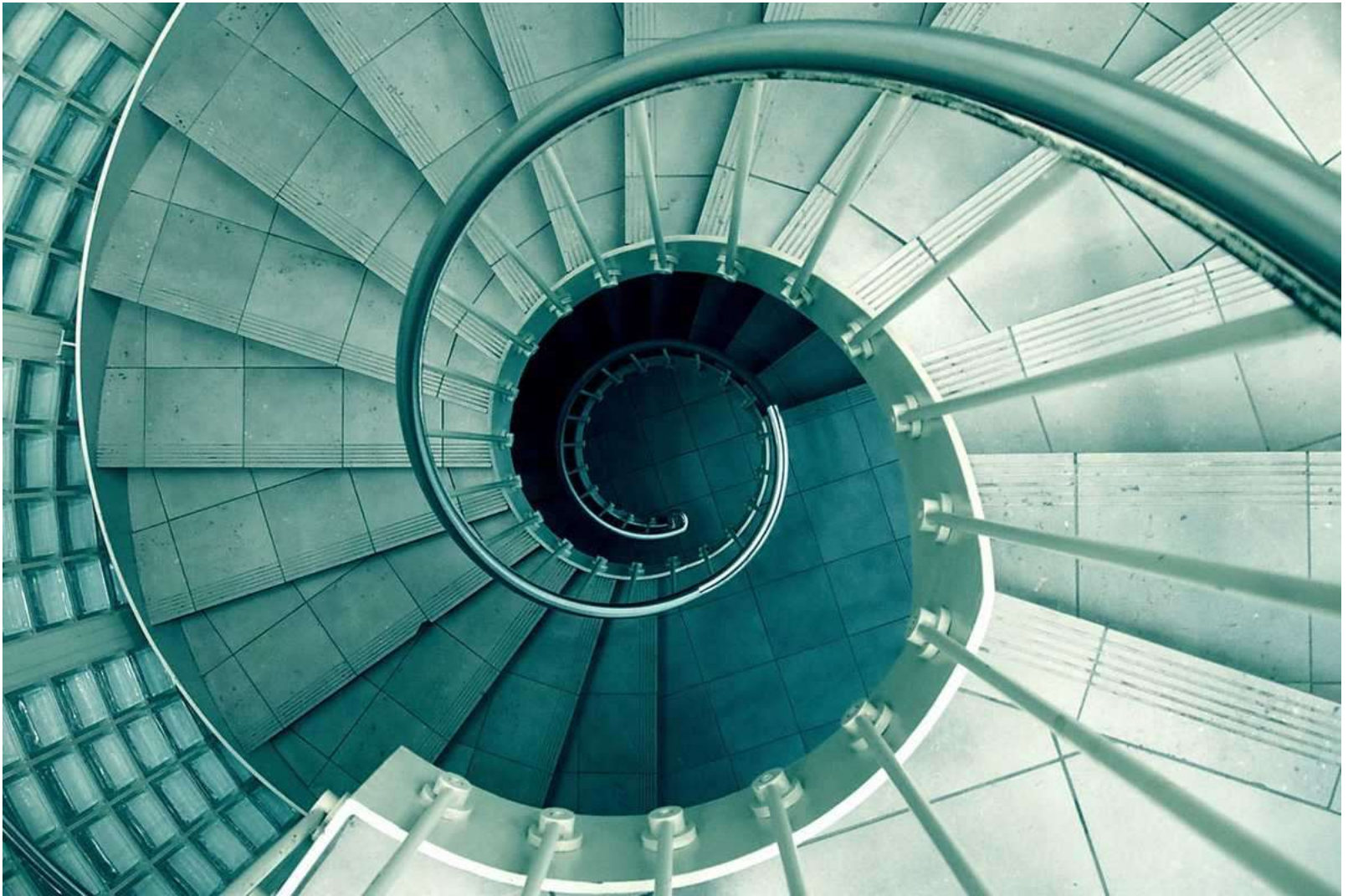
- [sum_array.c](#)

$$\begin{aligned}\text{Sum}(1, 3, 5, 7) &= 7 + \text{Sum}(1, 3, 5) \\ &= 7 + (5 + \text{Sum}(1, 3)) \\ &= 7 + (5 + (3 + \text{Sum}(1))) \\ &= 7 + (5 + (3 + 1)) \\ &= 7 + (5 + 4) \\ &= 7 + 9 \\ &= 16\end{aligned}$$

- 求最小值

- [min_re.c](#)

$$\begin{aligned}\text{Min}(3, 5, 2, 7) &= \text{min}(7, \text{Min}(3, 5, 2)) \\ &= \text{min}(7, \text{min}(2, \text{Min}(3, 5))) \\ &= \text{min}(7, \text{min}(2, \text{min}(5, \text{Min}(3)))) \\ &= \text{min}(7, \text{min}(2, \text{min}(5, 3))) \\ &= \text{min}(7, \text{min}(2, 3)) \\ &= \text{min}(7, 2) \\ &= 2\end{aligned}$$



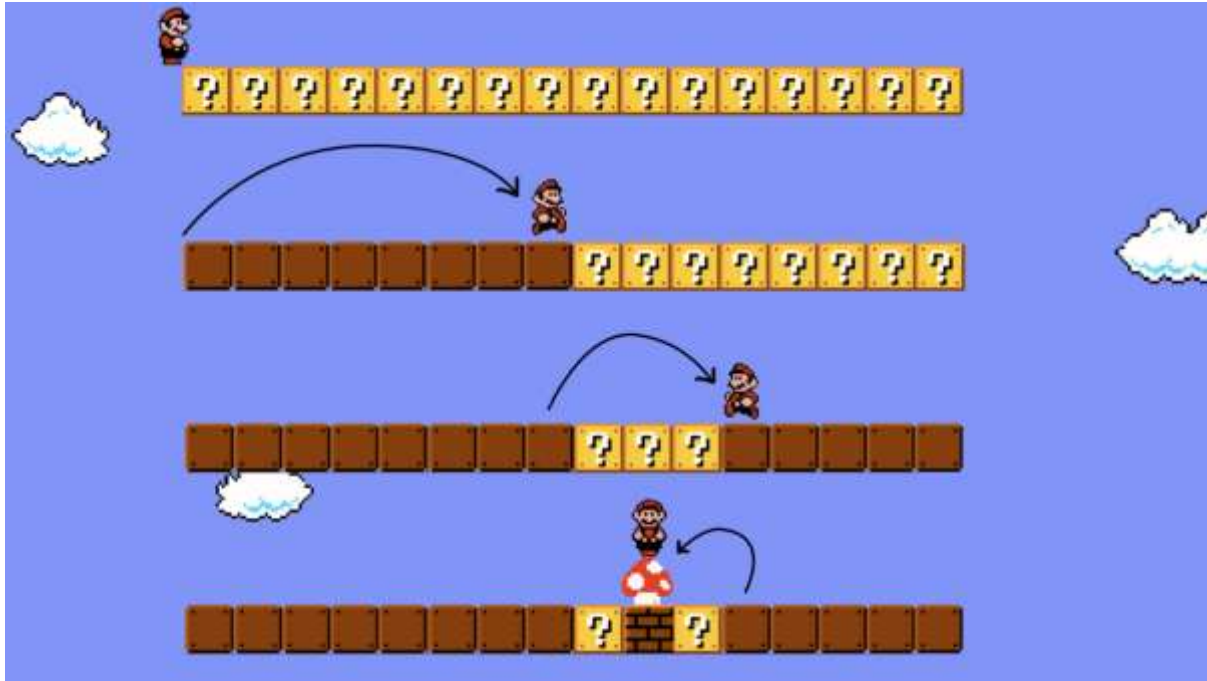
Binary Search

挠头...



- 典型二分查找算法

- 斐波那契数列: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
- binarysearch.c
- binarysearch_re.c



Divide and Conquer 分治法

- Divide the problem into a number of subproblems that are smaller instances of the same problem.
- Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
- Combine the solutions to the subproblems into the solution for the original problem.
- 重点是：问题分解和基线问题寻找

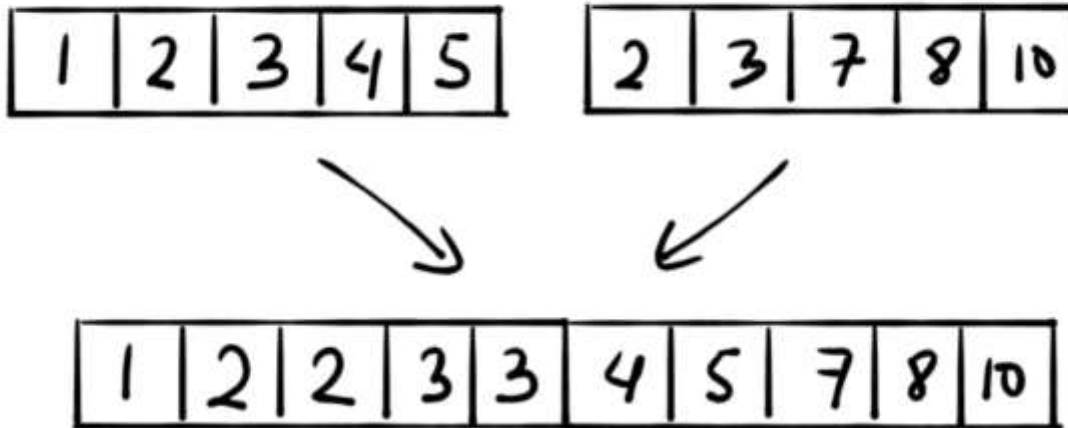
Merge Sort

挠头...

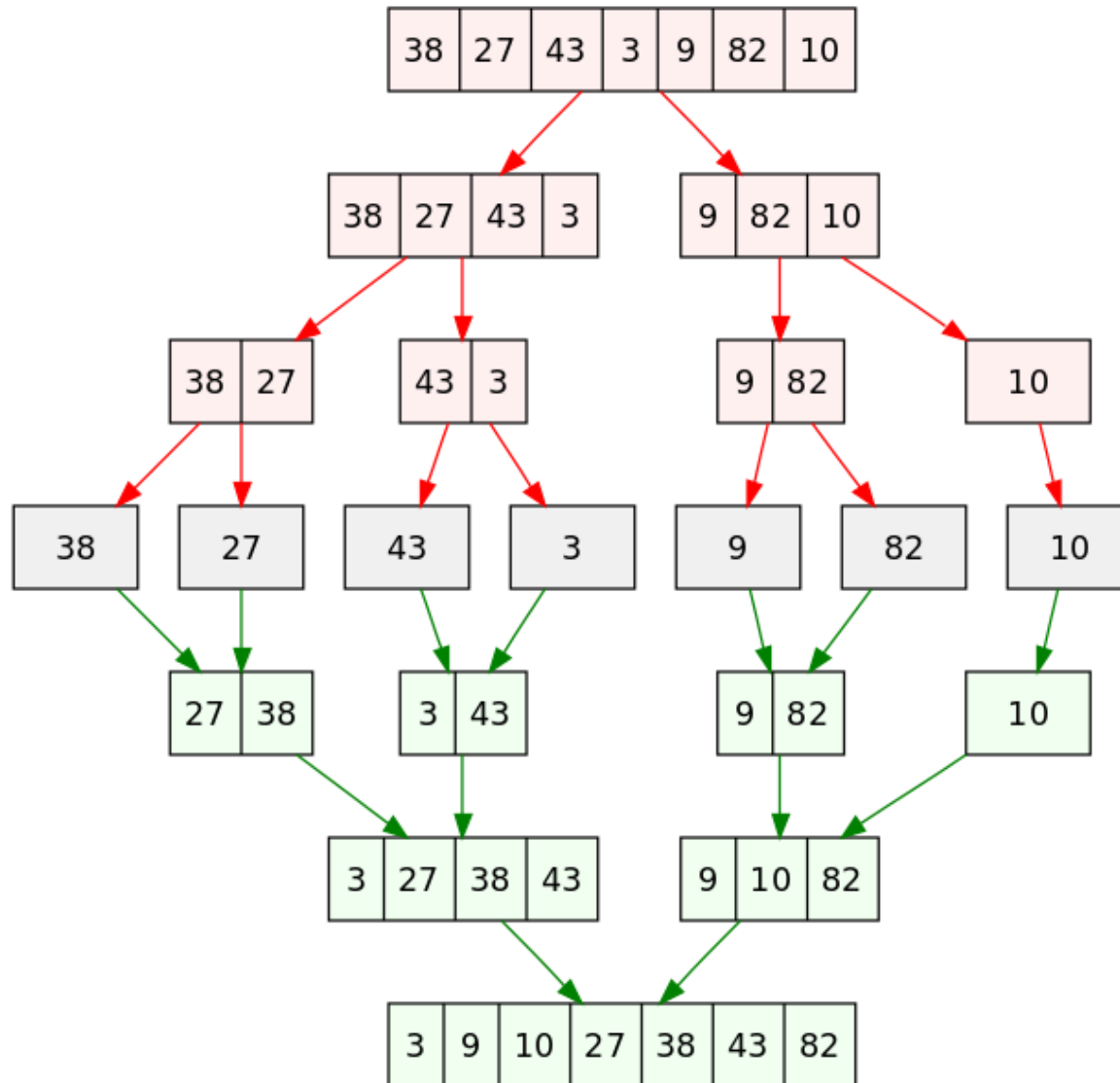


- [merge.c](#)
- [mergesort.c](#)

Merge Two Sorted Arrays



Merge Sort (dance!)



快速排序

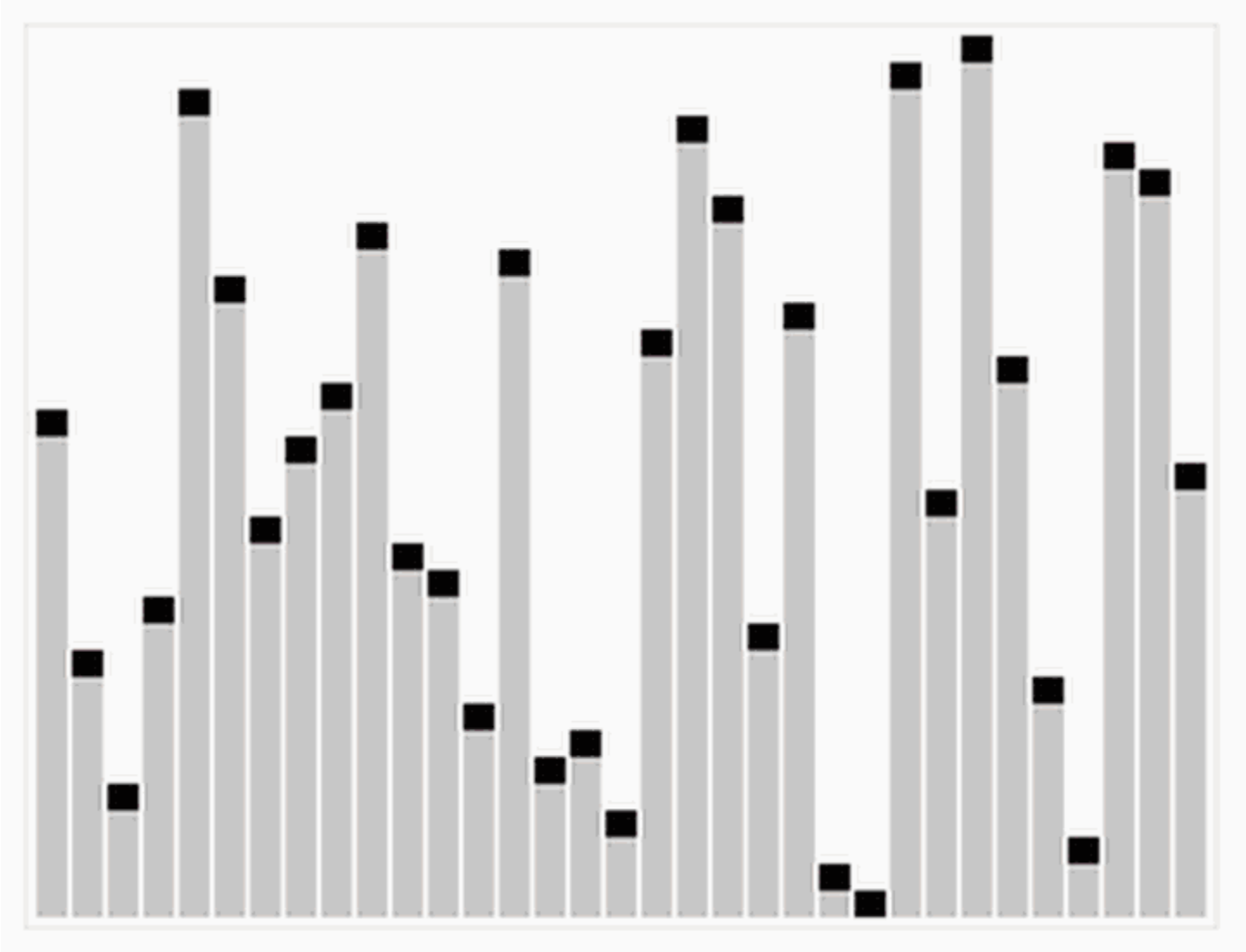
- 基本思想是

- 通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序
- 整个排序过程可以递归进行，以此达到整个数据变成有序序列。

- [quicksort.c](#)

挠头...





End

- 为努力（煎熬）的自己鼓掌！
- 继续加油！