



# 实验 6 排序算法

计算机类3班 施家鑫 25020007105

## Content

- 实验 6 排序算法
  - Content
  - C语言实现
    - 生成100000个随机数
    - 实现冒泡排序功能
    - 实现快速排序功能
    - 分别记录两者执行时间，并进行比较
    - 完成主程序
  - Python语言实现
    - 生成100000个随机数到数组
    - 冒泡排序功能
    - 快速排序功能
    - 记录执行时间并进行比较
    - 主程序
  - 关于本文档的撰写

## C语言实现

随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。

## 生成100000个随机数

- 引用头文件

```
#include <stdlib.h>
// 使用其中的rand()函数
#include <time.h>
// 记录执行时间
// 使用其中的time()函数为随机数做种子，以保证“随机”
```

- 使用time()为随机数做种

```
srand(time(NULL));
```

- 生成100000个随机数

```
int a[100001] = {0};
for (int i = 0; i < 100000; i++) {
    a[i] = rand() % 1000000;
}
```

## 实现冒泡排序功能

```
void bubble_sort(int a[], int n) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (a[j] > a[j+1]) {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}
```

## 实现快速排序功能

```
void quick_sort(int a[], int low, int high) {
    if (low < high) {
        int pivot = a[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (a[j] < pivot) {
                i++;
                int temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
        int temp = a[i + 1];
        a[i + 1] = a[high];
        a[high] = temp;
        int pi = i + 1;
        quick_sort(a, low, pi - 1);
        quick_sort(a, pi + 1, high);
    }
}
```

## 分别记录两者执行时间，并进行比较

- 使用两个变量分别记录开始与结束时间

```
clock_t start, end;
```

- 在排序功能执行的代码前后分别加上

```
start = clock();
```

```
end = clock();
```

记录对应的时间

- 将时间转化为以秒为单位

```
double T = ((double)(end - start)) / CLOCKS_PER_SEC;
```

## 完成主程序

```
int main() {
    int a[100001] = {0};
    srand(time(NULL));
    for (int i = 0; i < 100000; i++) {
        a[i] = rand() % 1000000;
    }
    clock_t start, end;
    start = clock();
    bubble_sort(a, 100000);

    end = clock();
    double T = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Bubble Sort Time: %f seconds\n", T);

    for (int i = 0; i < 100001; i++) {
        a[i] = rand() % 1000000;
    }

    start = clock();
    quick_sort(a, 0, 99999);
    end = clock();
    T = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Quick Sort Time: %f seconds\n", T);

    return 0;
}
```

- 最终效果

```
output — zsh — 80x24

Last login: Fri Nov  7 14:11:44 on ttys013
jackthinshin@Jackthins-MacBook-Air output % ./"SortRandom"
Bubble Sort Time: 7.993485 seconds
Quick Sort Time: 0.007423 seconds
jackthinshin@Jackthins-MacBook-Air output % █
```

## Python语言实现

随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。

思路(与C语言类似):

- 生成随机数[功能](#)
- 冒泡排序[功能](#)
- 快速排序[功能](#)
- [记录](#)执行时间，[并](#)[进行](#)比较
- [主程序](#)

## 生成100000个随机数到数组

导入time包

```
import time
```

按照 Python 语言规范放入数组

```
a = [random.randint(0, 100000) for _ in range(100000)]
```

## 冒泡排序功能

```
def bubble_sort(a: list):
    n = len(a)
    for i in range(n - 1):
        for j in range(0, n - 1 - i):
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
```

## 快速排序功能

```
def quick_sort(a: list, low: int, high: int):
    if low < high:
        x = random.randint(low, high)
        a[x], a[high] = a[high], a[x]
        p = a[high]
        i = low - 1
        for j in range(low, high):
            if a[j] < p:
                i += 1
                a[i], a[j] = a[j], a[i]
        a[i + 1], a[high] = a[high], a[i + 1]
        p = i + 1
        quick_sort(a, low, p - 1)
        quick_sort(a, p + 1, high)
```

## 记录执行时间并进行比较

- 分别记录 start end 时间

```
start = time.perf_counter()
//排序操作
end = time.perf_counter()
```

## 主程序

```
a = [random.randint(0, 100000) for _ in range(100000)]
start = time.perf_counter()
bubble_sort(a)
end = time.perf_counter()
print(f"Bubble sort on 100000 elems: {end - start:.6f} s")

a = [random.randint(0, 100000) for _ in range(100000)]
start = time.perf_counter()
quick_sort(a, 0, len(a) - 1)
end = time.perf_counter()
print(f"Quick sort on 100000 elems: {end - start:.6f} s")
```

- 最终效果

```
SortRandom.py
import random
import time

def bubble_sort(a: list):
    n = len(a)
    for i in range(n - 1):
        for j in range(0, n - 1 - i):
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]

def quick_sort(a: list, low, high):
    if low < high:
        x = random.randint(low, high)
        a[x], a[high] = a[high], a[x]
        p = a[high]
        i = low - 1
        for j in range(low, high):
            if a[j] < p:
                i += 1
                a[i], a[j] = a[j], a[i]
        a[i + 1], a[high] = a[high], a[i + 1]
        p = i + 1
        quick_sort(a, low, p - 1)
        quick_sort(a, p + 1, high)

a = [random.randint(0, 100000) for _ in range(100000)]
start = time.perf_counter()
bubble_sort(a)
end = time.perf_counter()
print(f"Bubble sort on 100000 elems: {end - start:.6f} s")

a = [random.randint(0, 100000) for _ in range(100000)]
start = time.perf_counter()
quick_sort(a, 0, len(a) - 1)
end = time.perf_counter()
print(f"Quick sort on 100000 elems: {end - start:.6f} s")
```

5

5

Ln: 7 Col: 0

Ln: 21 Col: 45

# 关于本文档的撰写

第一次使用Markdown

在Visual Studio Code中添加扩展



扩展: 商店

markdown

Markdown Previe...  
Markdown Preview Enhanced por...  
Yiyi Wang

7.9M  
★ 4.5

安装

Markdown All in O...  
All you need to write Markdown (k...  
Yu Zhang

11.9M  
★ 4.5

安装

Markdown PDF  
Convert Markdown to PDF  
yzane

3.1M  
★ 4.5

安装

Markdown Previe...  
Adds Mermaid diagram and flow...  
Matt Bierner

3.4M  
★ 4.5

安装

Markdown Previe...  
Changes VS Code's built-in mark...  
Matt Bierner

2.4M  
★ 4.5

安装

Markdown Checkb...  
Adds checkbox support to the bui...  
Matt Bierner

1.1M  
★ 4.5

安装

Markdown Emoji  
Adds emoji syntax support to VS ...  
Matt Bierner

1.1M  
★ 4

安装

markdownlint  
Markdown linting and style checki...  
David Anson

9.7M  
★ 4.5

安装

Markdown Footnotes  
Adds [^footnote] syntax support t...  
Matt Bierner

703K  
★ 5

安装

Markdown yaml Pr...  
Renders yaml front matter as a ta...  
Matt Bierner

762K  
★ 2.5

安装

GitHub Markdown ...  
Changes VS Code's built-in mark...  
Mark

654K  
★ 5

安装

Markdown Preview Enhanced

Yiyi Wang | 7,975,100 | ★★★★★ (128)

Markdown Preview Enhanced ported to vscode

安装 自动更新

Markdown Preview Enhanced

test3.md

```
1 ---
2 title: "Hi there"
3 output: pdf_document
4 ---
5
6 # An exhibit of Markdown
7
8 H2* is cool.
9
10 This note demonstrates some of what (Markdown) [1] is
11 capable of doing.
12
13 $$\ln_{-}(-\lnfty)^{\lnfty} e^{(-x^2)} = \sqrt{\pi}!$$
14
15 *Note: Feel free to play with this page, unlike
16 regular notes, this doesn't automatically save itself.*
17
18 ## Basic formatting $x+12$
19
20 Paragraphs can be written like so. A paragraph is the
21 basic block of Markdown. A paragraph is what text will
22 turn into when there is no reason it should become
23 anything else.
24
25 Paragraphs must be separated by a blank line. Basic
26 formatting of italics and bold is supported.
27 This can be messed like so.
28
29 ## Lists
30
31 ### Ordered list
```

Markdown Preview Enhanced

Hi there pdf\_document

An exhibit of Markdown

H<sup>2</sup> is cool.

This note demonstrates some of what Markdown is capable of doing.

$$\int_{-\infty}^{\infty} e^{-x^2} = \sqrt{\pi}$$

*Note: Feel free to play with this page. Unlike regular notes, this doesn't automatically save itself.*

Basic formatting  $x + 12$

Paragraphs can be written like so. A paragraph is the basic block of Markdown. A paragraph is what text will turn into when there is no reason it should become anything else.

Paragraphs must be separated by a blank line. Basic formatting of *italics* and **bold** is supported. This can be ~~messed~~ like so.

English 简体中文 繁體中文 日本語

市场

标识符

shd101wy: markdown-preview-enhanced

版本

0.8.20

已发布

2017-06-13, 08:43:02

上次发布时间

2025-11-01, 20:22:12

类别

Other

资源

市场

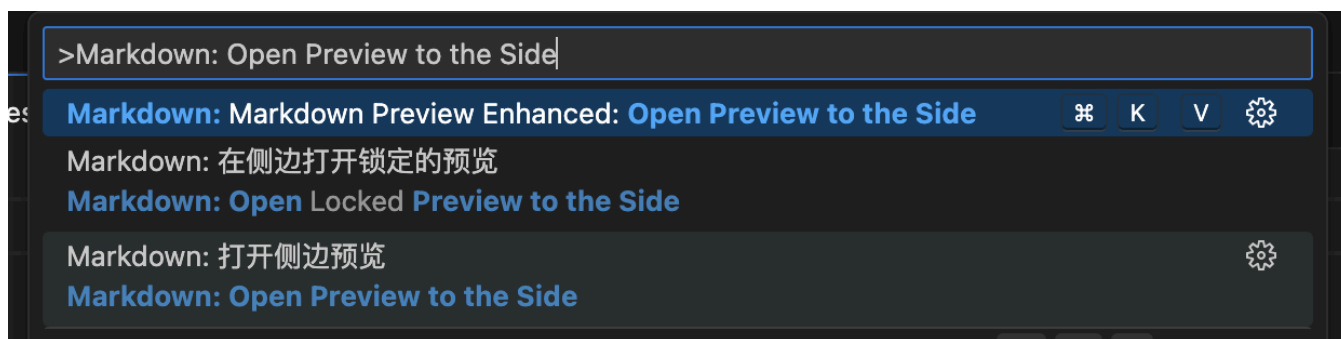
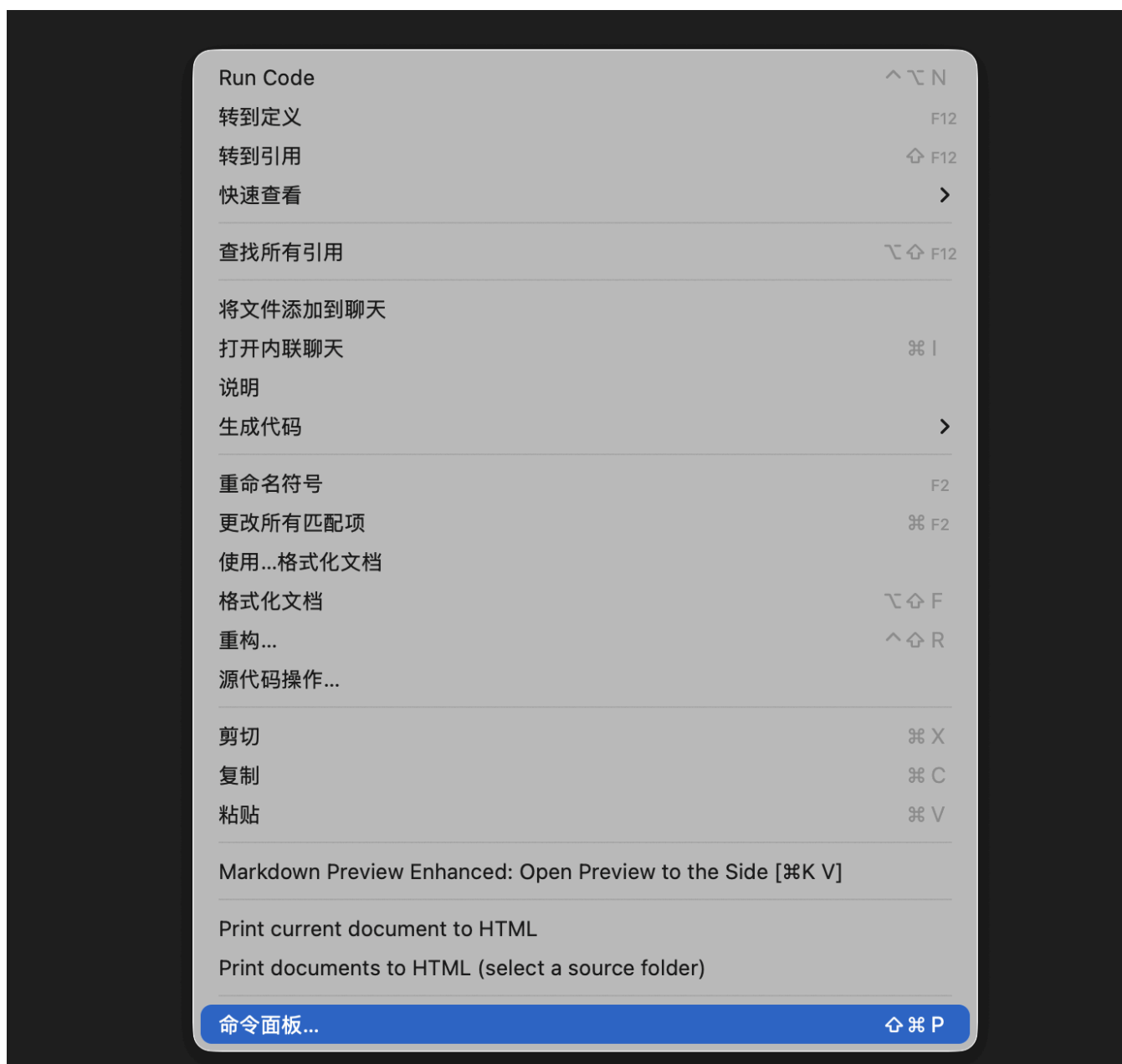
问题

仓库

许可证

Yiyi Wang

打开Side Preview以实现一边撰写一边预览效果的功能



最终工作区域实际效果如下：

SortRandom.cSortRandom.pyTest6\_25020007105.md 9+

Test6\_25020007105.md > 实验 6 排序算法 > ## Content

1# <center> 实验 6 排序算法 </center>

2

3\*\*\*<center>计算机类3班 施家鑫 25020007105</center>\*\*\*

4

5## Content

6Table of Contents (out of date)

7- [C语言实现] (#c语言实现)

8- [生成100000个随机数] (#生成100000个随机数)

9- [实现\*\*冒泡排序\*\*功能] (#实现冒泡排序功能)

10- [实现\*\*快速排序\*\*功能] (#实现快速排序功能)

11- [分别记录两者\*\*执行时间\*\*，并进行\*\*比较\*\*] (#分别记录两者执行时间并

12间并比较)

13- [完成主程序] (#完成主程序)

14- [Python语言实现] (#python语言实现)

15- [生成100000个随机数到数组] (#生成100000个随机数到数组)

16- [冒泡排序功能] (#冒泡排序功能)

17- [快速排序功能] (#快速排序功能)

18- [记录执行时间并比较] (#记录执行时间并比较)

19- [主程序] (#主程序)

20- [关于本文档的撰写] (#关于本文档的撰写)

21

22## C语言实现

23> 随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。

24

25思路：

26Table of Contents (out of date)

27- [实现\*\*生成随机数\*\*功能] (#生成100000个随机数)

28- [实现\*\*冒泡排序\*\*功能] (#实现冒泡排序功能)

29- [实现\*\*快速排序\*\*功能] (#实现快速排序功能)

30- [分别记录两者\*\*执行时间\*\*，并进行\*\*比较\*\*] (#分别记录两者执行时间并

31间并比较)

32- [完成主程序] (#完成主程序)

33

34### 生成100000个随机数

35

36- 引用头文件

37

实验 6 排序算法

计算机类3班 施家鑫 25020007105

Content

- C语言实现
  - 生成100000个随机数
  - 实现冒泡排序功能
  - 实现快速排序功能
  - 分别记录两者执行时间，并进行比较
  - 完成主程序
- Python语言实现
  - 生成100000个随机数到数组
  - 冒泡排序功能
  - 快速排序功能
  - 记录执行时间并比较
  - 主程序
- 关于本文档的撰写

C语言实现

随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。

思路：

- 实现生成随机数功能
- 实现冒泡排序功能
- 实现快速排序功能
- 分别记录两者执行时间，并进行比较
- 完成主程序

生成100000个随机数

- 引用头文件

```
#include <stdlib.h>
// 使用其中的rand() 函数
#include <time.h>
// 记录执行时间
// 使用其中的time() 函数为随机数做种子，以保证“随机”
```

行 18, 列 26 空格: 4 UTF-8 LF Markdown