D12组数学实践答辩

MATHEMATICAL PRACTICE

汇报人：D12组

答辩时间：2023-07-13

小组成员及分工

组长：梁家鸣

组员：陈文浩　　杨顺爽　　付宇晨　　姚竣译

时怡康　　唐永琦　　李昊瑞　　余炳焱

方案制作：李昊瑞　　付宇晨　　梁家鸣

代码编写：杨顺爽　　陈文浩　　时怡康　　唐永琦

PPT制作：姚竣译　　余炳焱

目 录

CONTENTS

01

项目方案

项目简介

- 对总量为120G（字节）的数据文件集进行排序，比拼排序速度。
- 文件内为长度15的随机英文字母字符串，’/n’换行（参见文件样例），输出按字符顺序对字符串排序的每隔100000+组号*100的字符串，同样’/n’换行，到一个文件。按正确率/完成时间的值大小排名得分。
- 每组随机抽1人代表小组答辩，要对小组项目实施过程、方案及代码进行说明，并回答老师提问，由老师进行评分。

项目方案

拿到包含大量待排序数据的文件，进行外部排序，一共五台电脑，每个电脑将文件分成24个子文件，利用内部排序的方法对每个子文件进行排序，存储在外存中，在将这些已经排好序的子文件进行归并，再把数据全部写入一个输出文件，得到有序的结果。

02

项目实施过程

实施过程

1.拿到项目任务后，组长带领组员开会商讨解决方案

2.根据对项目要求的分析和对组员编程水平的评估，组内经过讨论决定用更加简单高效的Python语言编写程序并进行合理的分工

3.组长进行组内分工，对应工作的组员主要负责各自项目，但都须快速学习Python语言,对编写程序有一定的理解，能够看懂代码

实施过程

4.经由组内商议分析决定，由于文件内存相当大，故采用外部排序和归并排序的方法，以达到快速准确排序的目的

5.任务明确后，组员各司其职、集思广益，不断查找资料、完善代码。在经过反复修改后终于得到了能够完成测试数据排序的代码

03

代码说明

步骤

01

将五个U盘中的数据分开来排序

02

再将五份排好的数据合并到一起排序，最后打印

以下为步骤1的代码说明

定义一个外部排序函数，用于将大文件分成若干个小文件进行排序

步骤1

```
5   def external_sort(input_file , output_files, chunk_size):
6       with open(input_file, 'r') as f_in:    # 以只读模式打开输入文件
7           i = 0
8           while True:
9               chunk = f_in.read(chunk_size)    # 读取指定大小的数据块
10              if not chunk:    # 如果数据块为空，跳出循环
11                  break
12              chunk = chunk.splitlines()    # 将数据块分割为行
13              chunk.sort()    # 进行排序
14              with open(output_files[i], 'w') as f_out:    # 将排序后的数据块写入对应的小文件中
15                  for line in chunk:
16                      f_out.write(line + '\n')    # 遍历排序后的数据块，逐行写入文件
17              i += 1
```

定义一个归并排序函数，合并若干个有序的小文件并输出到一个文件中

步骤1

```
25  def merge_sort(output_files, output_file):
26      # 打开指定的输入文件，返回文件对象，并将文件对象放入列表中
27      file_objects = [open(file, 'r') for file in output_files]
28      # 创建一个输出文件对象，用于写入排序结果
29      f_out = open(output_file, 'w')
30      # 读取每个文件的第一行并去除首尾的空格，存入current_lines列表
31      current_lines = [file.readline().strip() for file in file_objects]
32
33      # 归并排序
34      while current_lines:...
46
47      for file in file_objects:    # 关闭所有尚未关闭的输入文件
48          file.close()
49      f_out.close()    # 关闭输出文件
```

归并排序代码见下页

归并排序代码

步骤1

```
while current_lines:
    smallest = min(current_lines)    # 找出current_lines列表中的最小元素
    smallest_idx = current_lines.index(smallest)    # 找出current_lines列表中最小元素的索引值
    f_out.write(smallest + '\n')    # 写入最小值到输出文件
    next_line = file_objects[smallest_idx].readline().strip()    # 从对应的输入文件读取下一行，并去除首尾的空格

    if next_line:
        current_lines[smallest_idx] = next_line    # 如果下一行不为空，则更新当前行列表中的对应行
    else:
        file_objects[smallest_idx].close()    # 关闭已经读取完毕的输入文件
        del file_objects[smallest_idx]    # 从文件对象列表中删除该文件对象
        del current_lines[smallest_idx]    # 从当前行列表中删除对应的行
```

定义main函数并执行

步骤1

```
46  def main():
47      input_file = 'data.txt'
48      output_files = ['1.txt', '2.txt', '3.txt']
49      output_file = '4.txt'
50      chunk_size = 3 * 1024 * 1024 # 每个小文件的大小
51      # 调用外部排序函数和归并排序函数
52      external_sort(input_file, output_files, chunk_size)
53      merge_sort(output_files, output_file)
54
55  # 执行main函数
56  if __name__=='__main__':
57      main()
```

//main( ): 程序的主函数，负责调用以上函数完成整个流程。

注释：最终，主函数`main( )`调用了`external_sort( )`和`merge_sort( )`来实现外部排序的整个过程，将输入文件按照指定的块大小分割并排序，然后将排序后的结果输出到一个文件中

以下为步骤2的代码说明

定义一个归并排序的函数，合并步骤1的五个有序小文件并输出到一个文件中

步骤2

```
4   def merge_sort(output_files, output_file):
5       # 打开指定的输入文件，返回文件对象，并将文件对象放入列表中
6       file_objects = [open(file, 'r') for file in output_files]
7       # 创建一个输出文件对象，用于写入排序结果
8       f_out = open(output_file, 'w')
9       # 读取每个文件的第一行并去除首尾的空格，存入current_lines列表
10      current_lines = [file.readline().strip() for file in file_objects]
11
12      # 归并排序
13      while current_lines:...
25
26      for file in file_objects:    # 关闭所有尚未关闭的输入文件
27          file.close()
28      f_out.close()    # 关闭输出文件
```

归并排序代码见下页

归并排序代码

步骤2

```
while current_lines:
    smallest = min(current_lines)    # 找出current_lines列表中的最小元素
    smallest_idx = current_lines.index(smallest)    # 找出current_lines列表中最小元素的索引值
    f_out.write(smallest + '\n')    # 写入最小值到输出文件
    next_line = file_objects[smallest_idx].readline().strip()    # 从对应的输入文件读取下一行，并去除首尾的空格

    if next_line:
        current_lines[smallest_idx] = next_line    # 如果下一行不为空，则更新当前行列表中的对应行
    else:
        file_objects[smallest_idx].close()    # 关闭已经读取完毕的输入文件
        del file_objects[smallest_idx]    # 从文件对象列表中删除该文件对象
        del current_lines[smallest_idx]    # 从当前行列表中删除对应的行
```

读取

步骤2

```
30  def read_lines(filename, starting_index):
31      with open(filename, 'r') as f:
32          for _ in range(starting_index-1):
33              next(f)    # 跳过指定的起始行前的行
34          for line in f:
35              yield line.strip()   # 生成被strip后的每一行内容
```

//read_lines( ): 用于按行读取一个文件，并返回指定开始行之后的内容。

注释：因为原本文件太大，无法直接读取，便使用yield进行迭代读取，每次读一行，然后再读下一行，进行迭代

打印

步骤2

```
37  def print_results(output_file, starting_index, interval):
38      # 使用read_lines函数读取指定文件的指定行，并将读取的行存入lines
39      lines = read_lines(output_file, starting_index)
40      with open("E:/1234/10", "w") as file:
41          # 使用enumerate将lines中的每一行与其对应的行号i一起返回
42          for i, line in enumerate(lines, start=starting_index):
43              if i % interval == starting_index % interval:    # 判断当前行号是否满足给定的间隔条件
44                  file.write(line+"\n")    # 如果满足条件，则将当前行的内容写入输出文件
```

//print_results( ): 用于写入指定位置的输出文件内容。

定义main函数并执行

步骤2

```
45  def main():
46      output_files = ['1.txt', '2.txt', '3.txt','4.txt','5.txt']
47      output_file = '6.txt'
48      #调用归并排序函数和打印函数
49      merge_sort(output_files, output_file)
50      print_results(output_file, 1200, 100000)
51
52  #执行main函数
53  if __name__=='__main__':
54      main()
```

//main( ): 程序的主函数，负责调用以上函数完成整个流程。

恳请老师批评指正

THANKS FOR LISTENING

汇报人：D12组

答辩时间：2023-07-13