MapReduce 是利用多进程并行处理文件数据的典型场景。作为一种编程模型，其甚至被称为 Google 的”三驾马车“之一(尽管目前由于内存计算等的普及已经被逐渐淘汰)。在编程模型中，Map 进行任务处理, Reduce 进行结果归约。本周作业要求利用 Python 多进程实现 MapReduce 模型下的文档库(***搜狐新闻数据(SogouCS)，下载地址：https://www.sogou.com/labs/resource/cs.php***)，注意仅使用页面内容，即新闻正文）词频统计功能。具体地：

```
from multiprocessing import Process
from multiprocessing import Queue
import os
import jieba
import re
import math
import pickle
from tqdm import tqdm
import collections
import time

filepath = r"C:\Users\86186\Desktop\现代程序设计\11\news_sohusite_xml.dat"
respath=r"C:\Users\86186\Desktop\现代程序设计\11"
txtlist = []
```

1. Map 进程读取文档并进行词频统计，返回该文本的词频统计结果。

利用正则表达式对于新闻信息进行处理，只保留正文部分；然后进行分词统计词频，将结果保存到文件中，在队列中放入文件路径

```
def map_task(n,count,q,txtlist):
    res = {}
    start = n*count
    end = n*(count+1)
    pattern = '[(<content>)(</content>)]'
    jieba.setLogLevel(jieba.logging.INFO)
    for i in range(start,end):
        if i < len(txtlist):
            line =re.sub(pattern,'',txtlist[i])
            data = jieba.lcut(line)
            for j in data:
                if '\u4e00'<= j <='\u9fff':
                    if j not in res.keys():
                        res[j] = 1
                    else:
                        res[j] += 1
    with open(os.path.join(respath,"map_result{}.txt".format(count)),'wb') as f:
        pickle.dump(res,f)
    q.put(os.path.join(respath,"map_result{}.txt".format(count)))
```

```python
class Map(Process):
    def __init__(self,n,count,q,txtlist) -> None:
        super().__init__()
        self.n = n
        self.count = count
        self.q = q
        self.txtlist = txtlist

    def run(self):
        print("Map Process {} starts.".format(self.count))
        map_task(self.n,self.count,self.q,self.txtlist)
        print("Map Process {} ends.".format(self.count))
```

2. Reduce 进程收集所有 Map 进程提供的文档词频统计，更新总的文档库词频，并在所有 map 完成后保存总的词频到文件。

利用队列中的路径更新总的字典，并将结果输出到文件中

```python
class Reduce(Process):
    def __init__(self,q,process_num) -> None:
        super().__init__()
        self.q = q
        self.process_num = process_num
        self.result={}

    def run(self):
        print("Reduce process is runing...")
        pbar = tqdm(total=self.process_num,desc="Reduce")
        while not self.q.empty():
            path = self.q.get()
            with open(path,'rb') as f:
                res = pickle.load(f)
            for i in res.keys():
                if i in self.result.keys():
                    self.result[i] += res[i]
                else:
                    self.result[i] = res[i]
            pbar.update(1)
        pbar.close()
        self.result=dict(collections.OrderedDict(sorted(self.result.items(),key=lambda dc:dc[1],reverse=True)))
        with open(os.path.join(respath,"result.txt"),'w+',encoding='ansi') as f:
            for m,n in self.result.items():
                f.write(m+':'+str(n)+'\n')
        print("Reduce process finished.")
```

3. 主进程可提前读入所有的文档的路径列表，供多个 Map 进程竞争获取文档路径；或由主进程根据 Map 进程的数目进行分发；或者单独实现一个分发进程，与多个 MAP 进程通信。

主进程提前对信息进行处理，只保留<content>行；并根据进程数目得到每个进程要处理的新闻数量

```
def give_out(filepath,process_num):
    global txtlist
    i=1
    with open(filepath,'r',encoding='ansi') as f:
        for line in f.readlines():
            if i % 6 == 5:
                txtlist.append(line.strip())
            i += 1
    #txtlist=txtlist[:100]
    length = len(txtlist)
    n=int(math.ceil(length/process_num))
    return n
```

Main 函数：

```
if __name__ == "__main__":
    a=time.time()
    p_list=[]
    process_num = 8
    q=Queue(process_num)
    n=give_out(filepath,process_num)
    for i in range(process_num):
        p=Map(n,i,q,txtlist)
        p_list.append(p)
    for i in p_list:
        i.start()
    for i in p_list:
        i.join()
    b=time.time()
    r=Reduce(q,process_num)
    r.start()
    r.join()
    print("time:{:.0f}min{:.0f}s".format((b-a)//60,(b-a)%60))
```

运行结果：

```
PS C:\Users\86186\Desktop\现代程序设计> c:; cd 'c:\Users\86186\Desktop\现代程序设计'; & 'C:\Users\86186\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\86186\.vscode\extensions\ms-python.python-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher' '1930' '--' 'c:\Users\86186\Desktop\现代程序设计\11\p.py'
Map Process 0 starts.
Map Process 1 starts.
Map Process 2 starts.
Map Process 3 starts.
Map Process 4 starts.
Map Process 5 starts.
Map Process 6 starts.
Map Process 7 starts.
Map Process 4 ends.
Map Process 3 ends.
Map Process 2 ends.
Map Process 1 ends.
Map Process 0 ends.
Map Process 7 ends.
Reduce process is runing...
Reduce: 100%|                                                                                | 8/8 [00:05<00:00,  1.58it/s]
Reduce process finished.
time:32min39s
```

result.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

的:13150179
在:3294681
了:2828306
是:2037399
和:1949345
月:1282203
也:1106906
有:1084942
为:1071075
将:1005800
日:1002622
年:929559
等:873420
对:830688
中:812335
与:798422
上:778239
都:718932
他:714291
不:696994
到:673660
就:672174
记者:643072
后:619954
多:604950
中国:600520
我:587094
更:565668
人:555547
但:548691
说:546568
系列:528887
被:518963
而:510624
元:508072
从:505777
公司:503115
并:492846
这:490927

4. 记录程序运行时间，比较不同 Map 进程数量对运行时间的影响，可以做出运行时间-进程数目的曲线并进行简要分析。

随着进程数目增加运行时间显著减少，担当进程数目多于 8 时，时间并未减少反而有所增加。这是因为电脑的 cpu 数量为 8，最多只能同时运行 8 个进程，而设定的进程数目多于 8 个时，部分进程并不能同时运行，而是在等待前面的完成才能运行。

```
from matplotlib import pyplot as plt
import numpy as np

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False

x=np.array([2,4,8,15])

y=np.array([4620,3076,1959,2020])

plt.plot(x,y)
plt.title("运行时间与进程数")
plt.xlabel("进程数")
plt.ylabel("运行时间（s）")
plt.show()
```