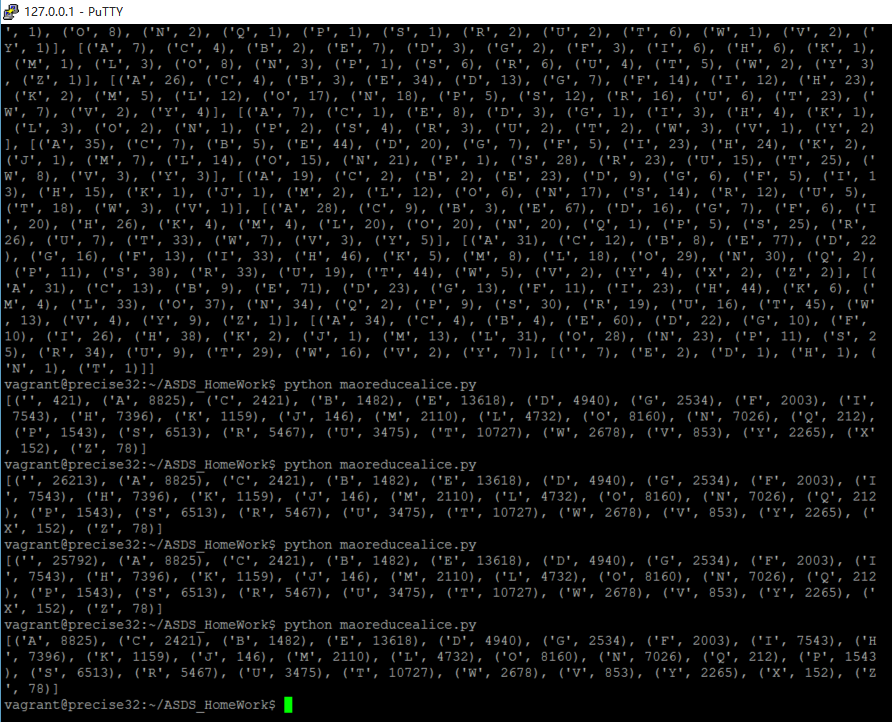
ADSD HW10 MapReduce



MapReduce Code:

import multiprocessing.dummy

p = multiprocessing.dummy.Pool(4)

f=open("alice.txt","r")

x=f.read()

x=x.strip().split("\n")

#print x

definchr = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

def mapper(data):

pairs=[]

for i in data:

i=i.upper().strip()

if i in definchr:

pairs.append((i,1))

return pairs

def combiner(pairs):

index={}

for (key,value) in pairs:

if not index.has\_key(key):

index[key]=value

else:

index[key]+=value

pairs=[]

for key in index:

pairs.append((key,index[key]))

return pairs

def reducer(con):

index={}

for y in con:

for (key,value) in y:

if not index.has\_key(key):

index[key]=value

else:

index[key]+=value

pairs=[]

for key in index:

pairs.append((key,index[key]))

return pairs

#pairs=mapper(x)

#pairs=combiner(pairs)

#pairs=mapper(x[0])

#print"x[0] and x[1]:",

#print pairs

re=p.map(mapper,x)

#print re

re=p.map(combiner,re)

#print re

con=reducer(re)

print con[1:]

def findletter(w):

for i in w:

if i=='z':

print i

break

return i

#print'find z:',

#print findletter(con)

#con.sort(key=lambda x:x[0])

#print con

#con.sort(key=lambda x:x[1])

#print con