КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кафедра інтелектуальних та інформаційних систем

Лабораторна робота № 3

з дисципліни

“Методи та системи парального програмування”

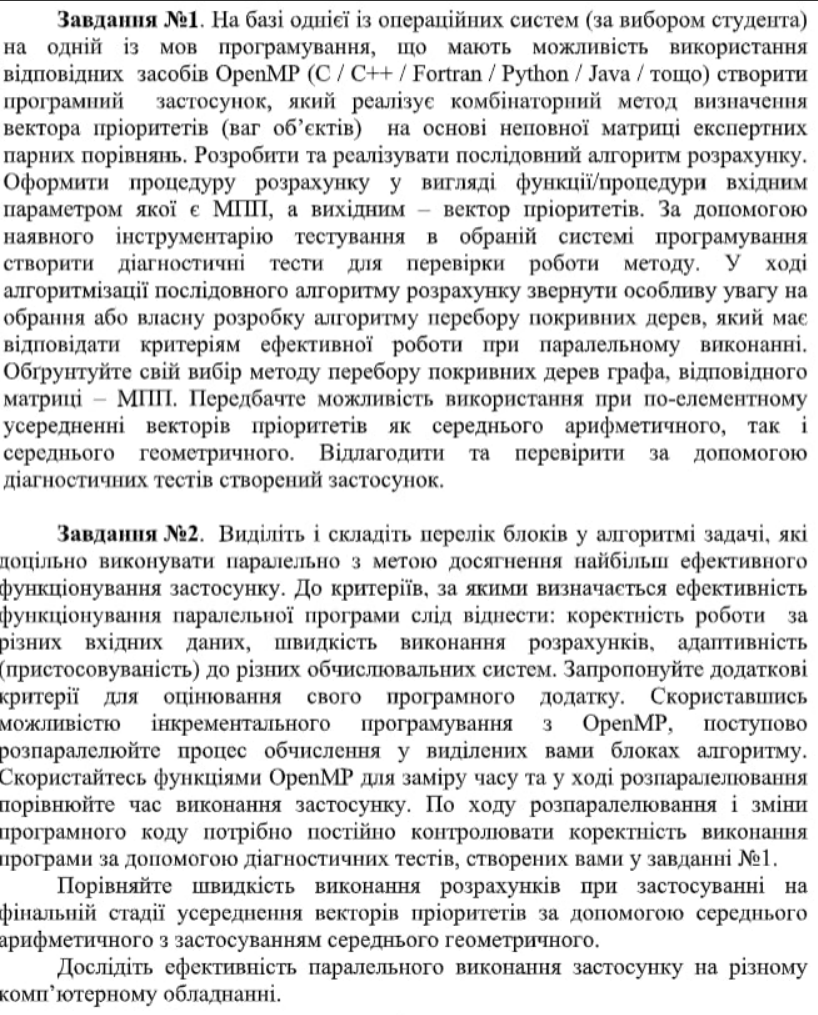
Виконав студент

групи КН- 31

Пашковський Павло Володимирович

Київ-2020

Завдання:



**1 Завдання**

import numpy  
from docutils.nodes import inline  
from numpy import \*  
from random import \*  
from time import time  
  
# для приятной печати времени  
MACRO = 100  
  
# C и openMP код  
codeOpenMP = \  
 """  
 int i = 0;  
  
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i)  
 {  
 #pragma omp for   
 for(i = 0 ; i < N\*M - 1; i++) {  
 // сравниваем два соседних элемента.  
 for(j = 0 ; j < n - i - 1 ; j++) {  
 if(matrix[j] > matrix[j+1]) {  
 // если они идут в неправильном порядке, то  
 // меняем их местами.  
 int tmp = matrix[j];  
 matrix[j] = matrix[j+1] ;  
 matrix[j+1] = tmp;  
 }  
 }  
 }  
  
 }  
 """  
  
# генерация случайной матрицы x на y  
# элементы матрицы - случайные числа от 0 до 99 включ.  
def randMat(x, y):  
 randRaw = lambda a: [randint(0, 100) for i in range(0, a)]  
 randConst = lambda x, y: [randRaw(x) for e in range(0, y)]  
 return array(randConst(x, y))  
  
  
def test():  
 results = []  
 n=5000  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
  
 print("\nTest on size: %dx%d" % (n, n))  
  
 """ python test """  
 matrix = array(sourceMat)  
 t1 = time()  
 matrix.sort()  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython)  
 results.append(matrix)  
  
 """ C и OpenMP test """  
 matrix = array(sourceMat)  
 t1 = time()  
 inline(codeOpenMP, ['matrix', 'c', 'randRow', 'N', 'M'],  
 extra\_compile\_args=['-O3 -fopenmp'],  
 compiler='gcc',  
 libraries=['gomp'],  
 headers=['<omp.h>'])  
 timeOpenMP = (time() - t1) \* MACRO  
 print("\tC plus OpenMP: %s" % (timeOpenMP))  
 results.append(matrix)  
  
 if array\_equal(results[0], results[1]) and \  
 array\_equal(results[1], results[2]):  
 print("\tTest - ok")  
 else:  
 print("\tTest - false")  
  
  
test()

Частина на Си:

"""  
int i = 0;  
  
omp\_set\_num\_threads(2);  
#pragma omp parallel shared(matrix, randRow, c) private(i)  
{  
#pragma omp for   
for(i = 0 ; i < N\*M - 1; i++) {  
 // сравниваем два соседних элемента.  
 for(j = 0 ; j < n - i - 1 ; j++) {  
 if(matrix[j] > matrix[j+1]) {  
 // если они идут в неправильном порядке, то  
 // меняем их местами.  
 int tmp = matrix[j];  
 matrix[j] = matrix[j+1] ;  
 matrix[j+1] = tmp;  
 }  
 }  
}  
  
}  
"""

**Частина на Python:**

matrix = array(sourceMat)  
t1 = time()  
matrix.sort()  
timePython = (time() - t1) \* MACRO  
print("\tPure python: ", timePython)  
results.append(matrix)

**2 Завдання**

from threading import Thread  
  
import numpy  
from docutils.nodes import inline  
from numpy import \*  
from random import \*  
from time import time  
# генерация случайной матрицы x на y  
# элементы матрицы - случайные числа от 0 до 99 включ.  
def randMat(x, y):  
 randRaw = lambda a: [randint(0, 100) for i in range(0, a)]  
 randConst = lambda x, y: [randRaw(x) for e in range(0, y)]  
 return array(randConst(x, y))  
def a():  
 # для приятной печати времени  
 MACRO = 50  
 # C и openMP код  
 codeOpenMP = \  
 """  
 int i = 0;  
   
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i)  
 {  
 #pragma omp for   
 for(i = 0 ; i < N\*M - 1; i++) {  
 // сравниваем два соседних элемента.  
 for(j = 0 ; j < n - i - 1 ; j++) {  
 if(matrix[j] > matrix[j+1]) {  
 // если они идут в неправильном порядке, то  
 // меняем их местами.  
 int tmp = matrix[j];  
 matrix[j] = matrix[j+1] ;  
 matrix[j+1] = tmp;  
 }  
 }  
 }  
   
 }  
 """  
 results = []  
 n = 5000  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
 """ C и OpenMP test """  
 matrix = array(sourceMat)  
 t1 = time()  
 inline(codeOpenMP, ['matrix', 'c', 'randRow', 'N', 'M'],  
 extra\_compile\_args=['-O3 -fopenmp'],  
 compiler='gcc',  
 libraries=['gomp'],  
 headers=['<omp.h>'])  
 timeOpenMP = (time() - t1) \* MACRO  
 print("\tC plus OpenMP: %s" % (timeOpenMP))  
 results.append(matrix)  
  
  
  
  
  
  
  
def b():  
 # для приятной печати времени  
 MACRO = 50  
 results = []  
 n = 50  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
  
 print("\nTest on size: %dx%d" % (n, n))  
  
 """ python test """  
 matrix = array(sourceMat)  
 t1 = time()  
 matrix.sort()  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython)  
 results.append(matrix)  
  
  
  
  
thread1 = Thread(target=a)# экземпляр потока Thread  
thread1.start()# Чтобы запустить отдельный поток, нужно создать экземпляр потока Thread и затем запустить его с помощью метода .start()  
thread2=Thread(target=b)# экземпляр потока Thread  
thread2.start()# Чтобы запустить отдельный поток, нужно создать экземпляр потока Thread и затем запустить его с помощью метода .start()

**потоки**

thread1 = Thread(target=a)# экземпляр потока Thread  
thread1.start()# Чтобы запустить отдельный поток, нужно создать экземпляр потока Thread и затем запустить его с помощью метода .start()  
thread2=Thread(target=b)# экземпляр потока Thread  
thread2.start()# Чтобы запустить отдельный поток, нужно создать экземпляр потока Thread и затем запустить его с помощью метода .start()

**Функція 1**

def a():  
 # для приятной печати времени  
 MACRO = 50  
 # C и openMP код  
 codeOpenMP = \  
 """  
 int i = 0;  
   
 omp\_set\_num\_threads(2);  
 #pragma omp parallel shared(matrix, randRow, c) private(i)  
 {  
 #pragma omp for   
 for(i = 0 ; i < N\*M - 1; i++) {  
 // сравниваем два соседних элемента.  
 for(j = 0 ; j < n - i - 1 ; j++) {  
 if(matrix[j] > matrix[j+1]) {  
 // если они идут в неправильном порядке, то  
 // меняем их местами.  
 int tmp = matrix[j];  
 matrix[j] = matrix[j+1] ;  
 matrix[j+1] = tmp;  
 }  
 }  
 }  
   
 }  
 """  
 results = []  
 n = 5000  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
 """ C и OpenMP test """  
 matrix = array(sourceMat)  
 t1 = time()  
 inline(codeOpenMP, ['matrix', 'c', 'randRow', 'N', 'M'],  
 extra\_compile\_args=['-O3 -fopenmp'],  
 compiler='gcc',  
 libraries=['gomp'],  
 headers=['<omp.h>'])  
 timeOpenMP = (time() - t1) \* MACRO  
 print("\tC plus OpenMP: %s" % (timeOpenMP))  
 results.append(matrix)

**Функція 2**

def b():  
 # для приятной печати времени  
 MACRO = 50  
 results = []  
 n = 50  
 sourceMat = randMat(n, n)  
 N, M = sourceMat.shape  
 randRow = sourceMat[randint(0, N)]  
 c = randint(0, n)  
  
 print("\nTest on size: %dx%d" % (n, n))  
  
 """ python test """  
 matrix = array(sourceMat)  
 t1 = time()  
 matrix.sort()  
 timePython = (time() - t1) \* MACRO  
 print("\tPure python: ", timePython)  
 results.append(matrix)