Лабораторная работа №1

Выполнил студент группы БВТ2003 Степанов Михаил Николаевич

Задание №1

Вывести "Hello, World!"

```
print("Hello, World!")
```

```
Hello, World!
```

Задание №2

Написать генератор случайных матриц(многомерных), который принимает опциональные параметры m, n, min_limit, max_limit, где m и n указывают размер матрицы, а min_lim и max_lim - минимальное и максимальное значение для генерируемого числа . По умолчанию при отсутствии параметров принимать следующие значения: m=50, n=50, min_limit=-250, max_limit=1000+(номер своего варианта)

```
from random import randint

def randomMatrixGenerate(m=50, n=50, min_limit=-250, max_limit=1000):
    return [[randint(min_limit,max_limit) for i in range(m)] for i in
range(n)]
```

Задание №3

Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.

Методы: выбором, вставкой, обменом, Шелла, турнирная, быстрая сортировка, пирамидальная

```
a=randomMatrixGenerate(4,3,0,10)

# Сортировка выбором

def selectionSort(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            min_j = j
            for k in range (j+1, len(matrix[0])):
                if matrix[i][k]<matrix[i][min_j]:
                    min_j=k
```

```
            matrix[i][j], matrix[i][min_j] = matrix[i][min_j],
matrix[i][j]


# Сортировка вставкой

def insertionSort(matrix):
    for i in range(len(matrix)):
        for j in range(1, len(matrix[0])):
            k = j-1
            while k >= 0 and matrix[i][k+1] < matrix[i][k]:
                matrix[i][k+1], matrix[i][k] = matrix[i][k], matrix[i]
[k+1]
                k-=1

# Сортировка обменом

def bubbleSort(matrix):
    for i in range(len(matrix)):
        for j in range(0, len(matrix[0])):
            for k in range(0, len(matrix[0])-j-1):
                if matrix[i][k+1] < matrix[i][k] :
                    matrix[i][k+1], matrix[i][k] = matrix[i][k],
matrix[i][k+1]

# Сортировка Шелла

def shellSort(matrix):
    for i in range(len(matrix)):
        gap = len(matrix[i])//2
        while gap > 0:
            for j in range(gap, len(matrix[i])):
                temp = matrix[i][j]
                k=j
                while k >= gap and matrix[i][k-gap] > temp:
                    matrix[i][k] = matrix[i][k-gap]
                    k-=gap
                matrix[i][k] = temp
            gap //= 2

# Турнирная сортировка

import math

def build(A, B, x, left, right):
    if left == right:
        B[x] = (A[left], x)
    else:
        middle = (left + right) // 2
        build(A, B, 2 * x, left, middle)
```

```
        build(A, B, (2 * x) + 1, middle + 1, right)
        if B[(2 * x) + 1][0] < B[2 * x][0]:
            B[x] = B[(2 * x) + 1]
        else:
            B[x] = B[2 * x]

def pop(B):
    result = B[1][0]
    index = B[1][1]
    B[index] = None
    while index > 1:
        index = index // 2
        if B[index * 2] is None:
            minimum = B[(index * 2) + 1]
        elif B[(index * 2) + 1] is None:
            minimum = B[index * 2]
        else:
            minimum = min(B[index * 2], B[(index * 2) + 1])
        if minimum == B[index]:
            break
        B[index] = minimum
    return result

def tournamentSort(array):
    temp = [None] * (2 ** (math.ceil(math.log2(len(array))) + 1))
    build(array, temp, 1, 0, len(array) - 1)
    sorted_array = [pop(temp) for i in range(len(array))]
    return sorted_array

def tournamentMatrixSort(matrix):
    for i in range(len(matrix)):
        matrix[i] = tournamentSort(matrix[i])

# Быстрая сортировка

def partition(array, start, end):
    pivot = array[start]
    low = start+1
    high = end

    while True:
        while low <= high and array[high] >= pivot:
            high-=1
        while low <= high and array[low] <= pivot:
            low+=1
        if low <= high:
            array[low], array[high] = array[high], array[low]
        else:
            break
```

```
    array[start], array[high] = array[high], array[start]

    return high

def quickSort(array, start, end):
    if (start < end):
        p = partition(array, start, end)
        quickSort(array, start, p-1)
        quickSort(array, p+1, end)


def quickMatrixSort(matrix):
    for i in range(len(matrix)):
        quickSort(matrix[i], 0, len(matrix[i])-1)

# Пирамидальная сортировка

def heapify(array, n, i):
    winner = i
    left = i*2+1
    right = i*2+2

    if left < n and array[i] < array[left]:
        winner = left

    if right < n and array[winner] < array[right]:
        winner = right

    if winner != i:
        array[i], array[winner] = array[winner], array[i]
        heapify(array, n, winner)


def heapSort(array):
    for i in range(len(array)//2, -1, -1):
        heapify(array, len(array), i)

    for i in range(len(array)-1, 0, -1):
        array[i], array[0] = array[0], array[i]

        heapify(array,i,0)

def heapMatrixSort(matrix):
    for i in range(len(matrix)):
        heapSort(matrix[i])

# Замеряем время, требующееся на сортировку

from timeit import default_timer as timer
```

```
# Сортировка выбором

a = randomMatrixGenerate(5,5,-4,10)
start = timer()
selectionSort(a)
end = timer()
print(end-start)
```

```
0.000210000000151922
```

```
# Сортировка вставкой

a = randomMatrixGenerate(5,5,-4,10)
start = timer()
insertionSort(a)
end = timer()
print(end-start)
```

```
5.020000025979243e-05
```

```
# Сортировка обменом

a = randomMatrixGenerate(5,5,-4,10)
start = timer()
bubbleSort(a)
end = timer()
print(end-start)
```

```
5.86999999541149e-05
```

```
# Сортировка Шелла

a = randomMatrixGenerate(5,5,-4,10)
start = timer()
shellSort(a)
end = timer()
print(end-start)
```

```
7.370000002993038e-05
```

```
# Турнирная сортировка

a = randomMatrixGenerate(5,5,-4,10)
start = timer()
tournamentMatrixSort(a)
end = timer()
print(end-start)
```

```
0.0001046000002133951
```

```
# Быстрая сортировка
```

```
a = randomMatrixGenerate(5,5,-4,10)
start = timer()
quickMatrixSort(a)
end = timer()
print(end-start)
```

```
0.0001378000001750479
```

```
# Пирамидальная сортировка

a = randomMatrixGenerate(5,5,-4,10)
start = timer()
heapMatrixSort(a)
end = timer()
print(end-start)
```

```
9.399999999004649e-05
```

```
# Стандартная сортировка питона

a = randomMatrixGenerate(5,5,-4,10)
start = timer()
for i in range(len(a)):
    a[i] = sorted(a[i])
end = timer()
print(end-start)
```

```
7.049999976516119e-05
```

Задание №4

Создать публичный репозиторий на github, и запушить выполненное задание в .ipynb формате.

https://github.com/Exidelius/IiITLabs.git