Лабораторная работа 2

Рекурсия. Стратегия «разделяй и властвуй»

(20 баллов)

Выполните самостоятельно следующие задания и оформите отчет.

Требования по отчету:

Наличие титульного листа. Размер страницы должен соответствовать формату А4 (210х297), размеры полей: левое – 30 мм, правое – 10 мм, верхнее – 15 мм, нижнее – 20 мм. Шрифт Times new Roman, размер 14 pt полуторный междустрочный интервал. Выравнивание текста – по ширине, красная строка – 1,25 см, отступ слева и справа – 0 мм.

№ 1

(2 балла)

Реализуйте или найдите реализации двух сортировок: сортировка выбором и быстрая сортировка. Проведите вычислительные эксперименты и нарисуйте графики, показывающие зависимость времени выполнения двух алгоритмов от размера входных данных. Рассмотрите три варианта входных данных:

1. Список случайных чисел
2. Отсортированный список
3. Отсортированный в обратную сторону список

Для каждого из вариантов должен быть свой график зависимости.

№ 2

(2 балла)

Ряд Трибоначчи начинается с тройки 0, 0, 1, а каждое следующее число равно сумме трёх предыдущих. Числа нумеруются с 0.

Напишите функцию tribonacci(n), которая принимает в себя номер числа и возвращает n-ое число Трибоначчи. Функция должна быть рекурсивной.

При решении данной задачи не используйте циклы.

Укажите базовый и рекурсивный случаи функции.

№ 3

(2 балла)

Для задачи вашего варианта напишите подходящую рекурсивную функцию.

При решении задачи не используйте циклы.

Укажите базовый и рекурсивный случаи вашей функции.

1 вариант

Дано слово, состоящее только из строчных латинских букв. Проверьте, является ли это слово палиндромом. Напишите рекурсивную функцию recursive_poly(word). В качестве результата работы она должна возвращать строки «YES» или «NO». При решении нельзя использовать срезы с шагом, отличным от 1.

2 вариант

Напишите рекурсивную функцию recursive_count(some_list, t), которая возвращает количество вхождений элемента t в список some_list. Рекурсивный вызов должен быть устроен так же, как в функции find_max(A) из лекции.

3 вариант

Напишите рекурсивную функцию recursive_two(some_list), которая возвращает кортеж из двух значений – наибольший элемент списка some_list и второй наибольший его элемент.

4 вариант

Напишите рекурсивную функцию del_all_e(s, e), которая удаляет из строки s все вхождения символа e.

Например, del_all_e("мама мыла раму", "а") вернет строку "мм мыл рму".

5 вариант

Напишите функцию recursive_reverse(some_list), которая возвращает перевернутый список.

Например, recursive_reverse([1, 2, 3, 4, 5]) вернет список [5, 4, 3, 2, 1].

№ 4

(2 балла)

Числа Фибоначчи - это последовательность, которая вычисляется по рекуррентному соотношению $F_N = F_{N-1} + F_{N-2}$, где $F_0 = 0$, а $F_1 = 1$. Числа Люка задаются примерно так же: $L_N = L_{N-1} + L_{N-2}$, где $L_0 = 2$, а $L_1 = 1$. Пользуясь стандартным рекурсивным подходом, напишите две функции - fibonacci(n) и lucas(n), которые будут вычислять n-е число Фибоначчи и Люка соответственно. Измерьте время, которое требуется для вычисления F_N и L_N вплоть до N = 40. Если ваш компьютер достаточно быстрый, N стоит увеличить для повышения точности, а если слишком медленный - уменьшить, чтобы измерения заканчивались за приемлемое время.

Затем напишите функцию fib_with_lucas(n) и вспомогательную к ней, lucas_with_fib(n), пользуясь такими свойствами этих последовательностей:

• fib_with_lucas(n) – если поделить n нацело пополам и взять i = n // 2, а j = n - i, то $F_n = F_{i+j} = (F_i + L_j)\ (F_j + L_i) // 2$;

• lucas_with_fib(n) – в свою очередь, $L_N = F_{N-1} + F_{N+1}$.

Сравните быстродействие fibonacci() и fib_with_lucas(). Постройте график.

№ 5

(3 балла)

Хорошим примером для иллюстрации рекурсивных алгоритмов являются задачи рисования фракталов. Далее представлена программа, которая при помощи модуля turtle (turtle — Turtle graphics — Python 3.9.7 documentation) рисует фрактальное дерево.

```
import turtle


def tree(branchLen, t):
    if branchLen > 5:
        t.forward(branchLen)
        t.right(20)
        tree(branchLen - 15, t)
        t.left(40)
        tree(branchLen - 15, t)
        t.right(20)
        t.backward(branchLen)


def main():
    t = turtle.Turtle()
    myWin = turtle.Screen()
    t.left(90)
    t.up()
    t.backward(100)
    t.down()
    t.color("green")
    tree(75, t)
    myWin.exitonclick()


main()
```

Запустите программу и разберитесь в ее коде. Попробуйте поменять параметры при вызове функции tree и внутри нее.

Последовательно измените программу для рисования рекурсивного дерева по следующим пунктам:

1. Измените толщину ветвей, чтобы при уменьшении branchLen линии становились тоньше.
2. Измените цвет ветвей таким образом, чтобы самые короткие ветви окрашивались как листья.
3. Измените угол поворота черепахи, чтобы каждая ветвь поворачивалась произвольным образом в некотором диапазоне. Например, выбирайте угол между 15-ю и 45-ю градусами. Поэкспериментируйте в поисках лучшего вида.
4. Измените рекурсивную часть branchLen, чтобы каждый раз вычиталось произвольное значение из некоторого диапазона вместо некой постоянной величины.

В отчете приведите несколько примеров получившихся деревьев.

№ 6

(3 балла)

При помощи модуля turtle нарисуйте фрактал, указанный в вашем варианте (материалы Рекурсия: фракталы (mipt-cs.github.io)).

Вариант 1. Кривая Коха.

Вариант 2. Снежинка Коха.

Вариант 3. Кривая Минковского.

Вариант 4. Кривая дракона.

Вариант 5. Треугольник Серпинского.

№ 7

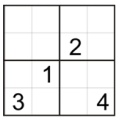
(3 балла)

Найдите или придумайте алгоритм для рисования фрактальных гор. Подсказка: одним из возможных методов будет использование треугольников.

№ 8

(3 балла)

Решите рекурсивно мини-судоку размером 4x4. Для этого напишите функцию solve_sudoku(matrix), где matrix - целочисленная матрица (список списков).



В мини-судоку числа от 1 до 4 встречаются ровно один раз в каждой вертикали и горизонтали, а также в квадратах 2x2.

Укажите базовый и рекурсивный случаи вашего алгоритма.

Судоку представлено в виде таблицы чисел, в которой нулями обозначены пустые места:

0000

0020

0100

3004

Правильный ответ:

2341

1423

4132

3214