Государственное бюджетное образовательное учреждение высшего образования
Московской области
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
имени дважды Героя Советского Союза, летчика-космонавта А.А. Леонова

Колледж космического машиностроения и технологий

ОТЧЕТ

по учебной практике УП.01.01

по профессиональному модулю

ПМ.01. Разработка программных модулей программного обеспечения для компьютерных систем

Специальность **09.02.03 «Программирование в компьютерных системах»**

Обучающегося 3 курса группы П1-20 формы обучения очной

Демьянова Артема Александровича

Место прохождения практики

Колледж космического машиностроения и технологий
«Технологического университета»

(Название организации)

Срок прохождения практики с «08» декабря 2022 г. по «21» декабря 2022 г.

Руководитель практики: преподаватель _____ С. Д. Стрельников

подпись

Итоговая оценка по практике _____

Оглавление

Введение

Я прошёл два курса на образовательной платформе «Stepik»: «Алгоритмы: теория и практика. Методы» и «Алгоритмы: теория и практика. Структуры данных». Всего было решена 18 задач на программирование, из которых 12 на курсе по методам, а 6 на курсе по структурам данных. Также было решено несколько теоретических задач, и были получены два сертификата за прохождение данных курсов. Задачи на программирование решались на языке Python версии 3.

Основная часть

Курс «Алгоритмы: теория и практика. Методы»

Тема №1. Занятие №1. Разбор задачи «Пример задачи на программирование»

Цель: Ознакомиться с проверяющей системой.

Задача: Требуется вычислить сумму двух входных целых чисел, лежащих в отрезке от нуля до десяти.

Реализация:

Листинг №1. Алгоритм 1.1.4

```
a, b = map(int, input().split())
res = a + b
print(res)
```

Ссылка на задачу: https://stepik.org/lesson/13140/step/4

Блок-схема:

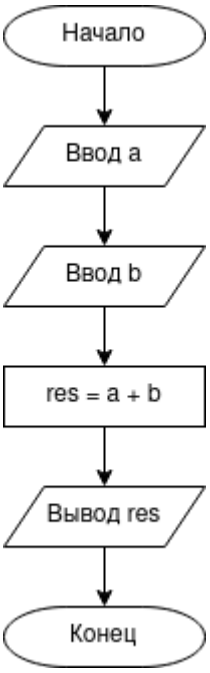

Рис. №1. Блок-схема задачи 1.1 «Пример задачи на программирование»

Тема №2. Занятие №2. Разбор задачи «Небольшое число Фибоначчи»

Цель: Ознакомиться с алгоритмом для поиска n-го числа Фибоначчи.

Задача: Дано целое число $1 \leq n \leq 40$, необходимо вычислить n-е число Фибоначчи.

Реализация:

Листинг №2. Алгоритм 2.2.6

```
n = int(input())

f = [0, 1]
for i in range(2, n + 1):
    t = f[i-1] + f[i-2]
    f.append(t)

print(f[-1])
```

Ссылка на задачу: https://stepik.org/lesson/13228/step/6

Блок-схема:

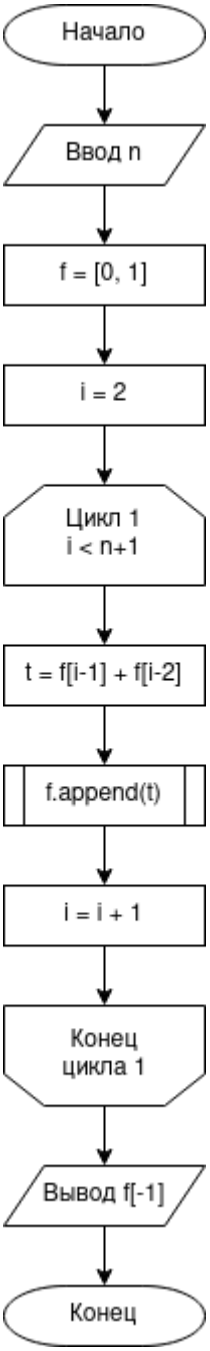

Рис. №2. Блок-схема задачи 2.2. «Небольшое число Фибоначчи»

Тема №2. Занятие №2. Разбор задачи «Последняя цифра большого числа Фибоначчи»

Цель: Ознакомиться с алгоритмом для поиска последней цифры большого числа Фибоначчи.

Задача: Дано число $1 \le n \le 10^7$, необходимо найти последнюю цифру n-го числа Фибоначчи.

Реализация:

Листинг №3. Алгоритм 2.2.7

```
n = int(input())

a, b = 0, 1
for i in range(2, n + 1):
    temp = b % 10
    b += a % 10
    a = temp
b %= 10

print(b)
```

Ссылка на задачу: https://stepik.org/lesson/13228/step/7

Блок-схема:

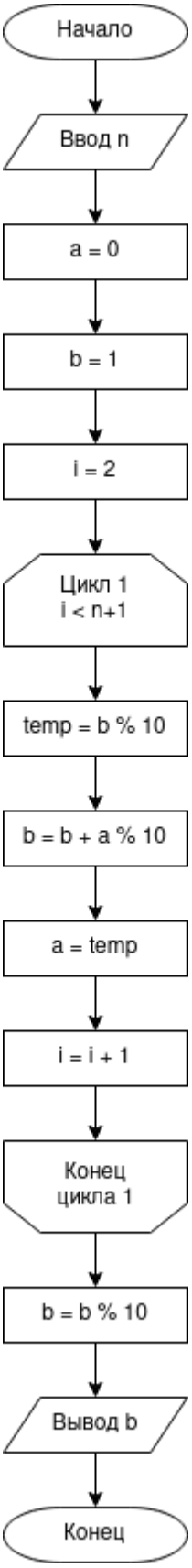

Рис. №3. Блок-схема задачи 2.2. «Последняя цифра большого числа Фибоначчи»

Тема №2. Занятие №2. Разбор задачи «Огромное число Фибоначчи по модулю»

Цель: Ознакомиться с алгоритмом поиска остатка от деления n-го числа Фибоначчи на m.

Задача: Даны целые числа $1 \le n \le 10^{18}$ и $2 \le m \le 10^{5}$, необходимо найти остаток от деления n-го числа Фибоначчи на m.

Реализация:

Листинг №4. Алгоритм 2.2.8

```
n, m = map(int, input().split())

f = [0, 1]
i = 1

while True:
    t = f[i-1] + f[i]
    t %= m
    f.append(t)
    i += 1
    if f[i-1] == 0 and f[i] == 1:
        f.pop()
        f.pop()
        break

l = len(f)
n %= l
print(f[n])
```

Ссылка на задачу: https://stepik.org/lesson/13228/step/8

Блок-схема:

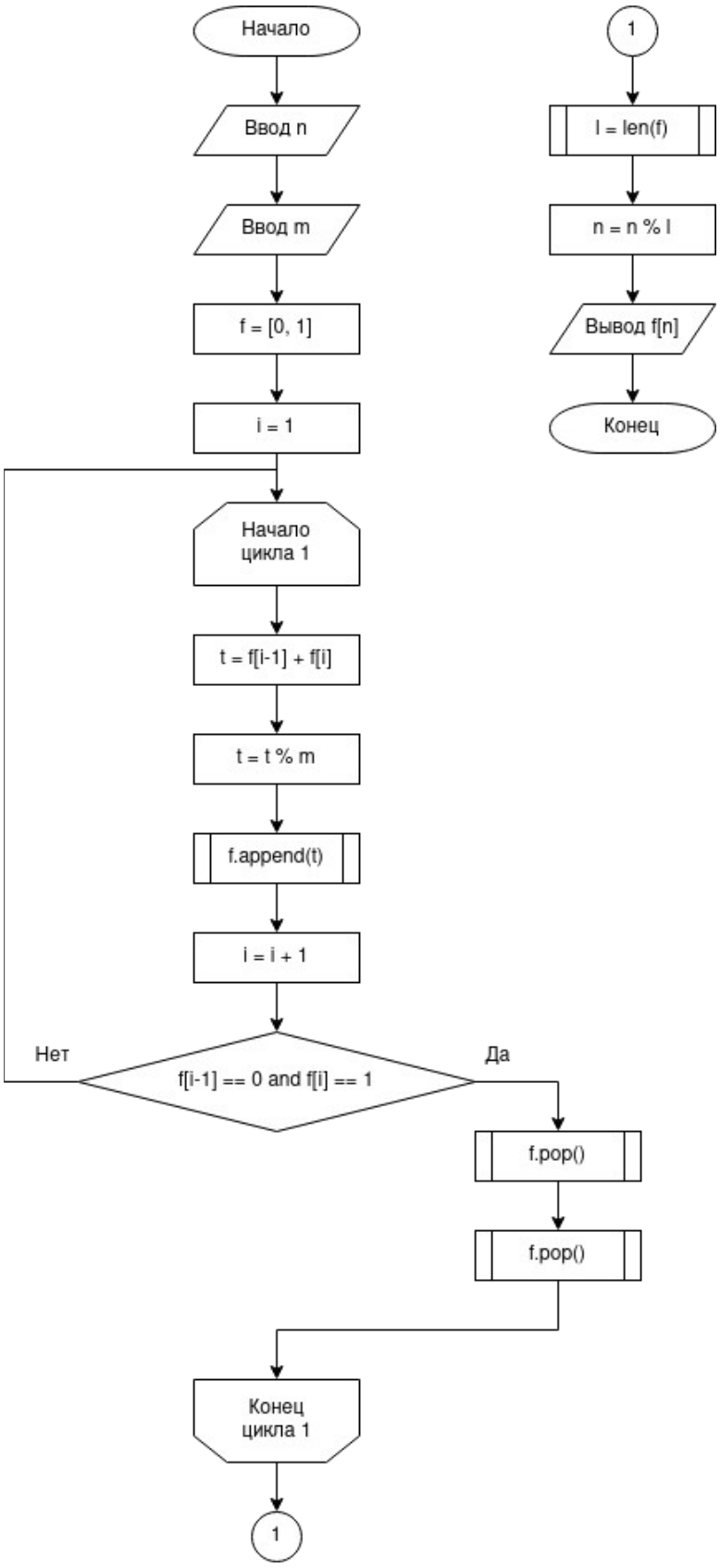

Рис. №4. Блок-схема алгоритма «Огромное число Фибоначчи по модулю»

Тема №2. Занятие №3. Разбор задачи «Наибольший общий делитель»

Цель: Ознакомиться с алгоритмом Евклида для поиска НОД двух чисел.

Задача: По данным двум числам $1 \le a, b \le 2 \cdot 10^9$ найдите их наибольший общий делитель.

Реализация:

Листинг №5. Алгоритм 2.3.5

```
a, b = map(int, input().split())

while a != 0 and b != 0:
    if a > b:
        a %= b
    else:
        b %= a

t = a + b
print(t)
```

Ссылка на задачу: https://stepik.org/lesson/13229/step/5

Блок-схема:

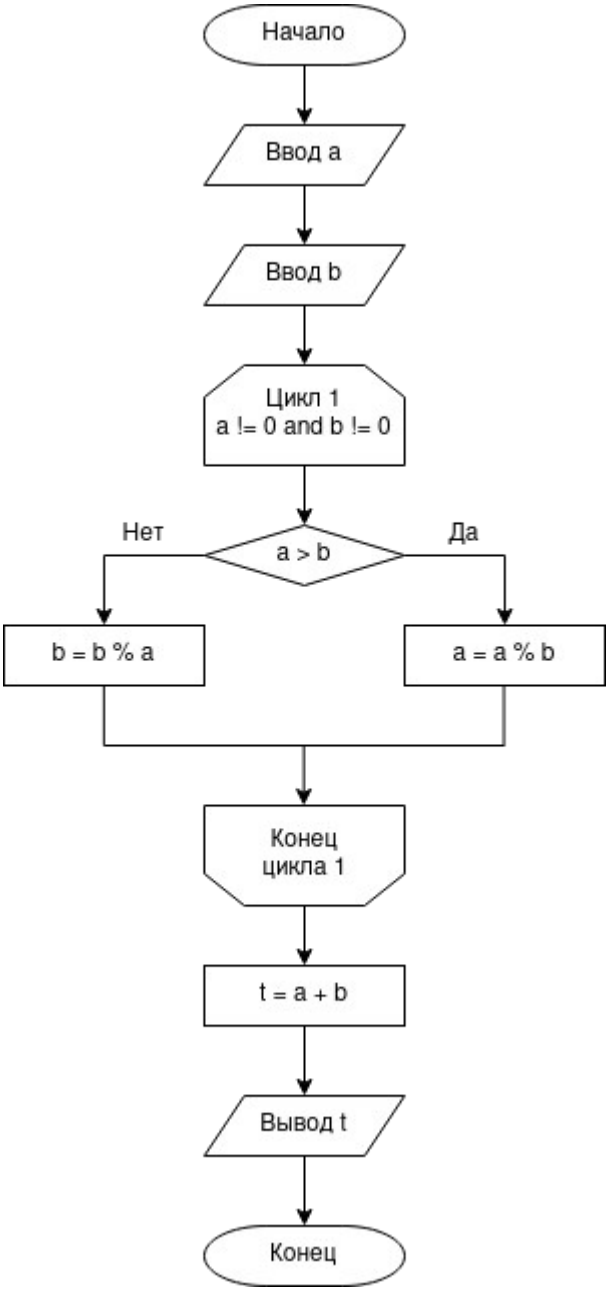

Рис. №5. Блок-схема алгоритма Евклида

Тема №4. Занятие №1. Разбор задачи «Покрыть отрезки точками»

Цель: Ознакомиться с алгоритмом для поиска множества точек минимального размера по данным n отрезкам, для которого каждый из отрезков содержит хотя бы одну из точек.

Задача: По данным n отрезкам необходимо найти множество точек минимального размера, для которого каждый из отрезков содержит хотя бы одну из точек. В первой строке дано число $1 \le n \le 100$ отрезков. Каждая из последующих n строк содержит по два числа $0 \le l \le r \le 10^9$, задающих начало и конец отрезка. Выведите оптимальное число m точек и сами m точек. Если таких множеств точек несколько, выведите любое из них.

Реализация:

Листинг №6. Алгоритм 4.1.9

```
lst = []
n = int(input())

for i in range(n):
    a, b = map(int, input().split())
    t = (b, a)
    lst.append(t)

lst.sort()

out = []
out.append(lst[0][0])

for i in range(n):
    if lst[i][1] > out[-1]:
        out.append(lst[i][0])

print(len(out))
print(*out)
```

Ссылка на задачу: https://stepik.org/lesson/13238/step/9

Блок-схема:

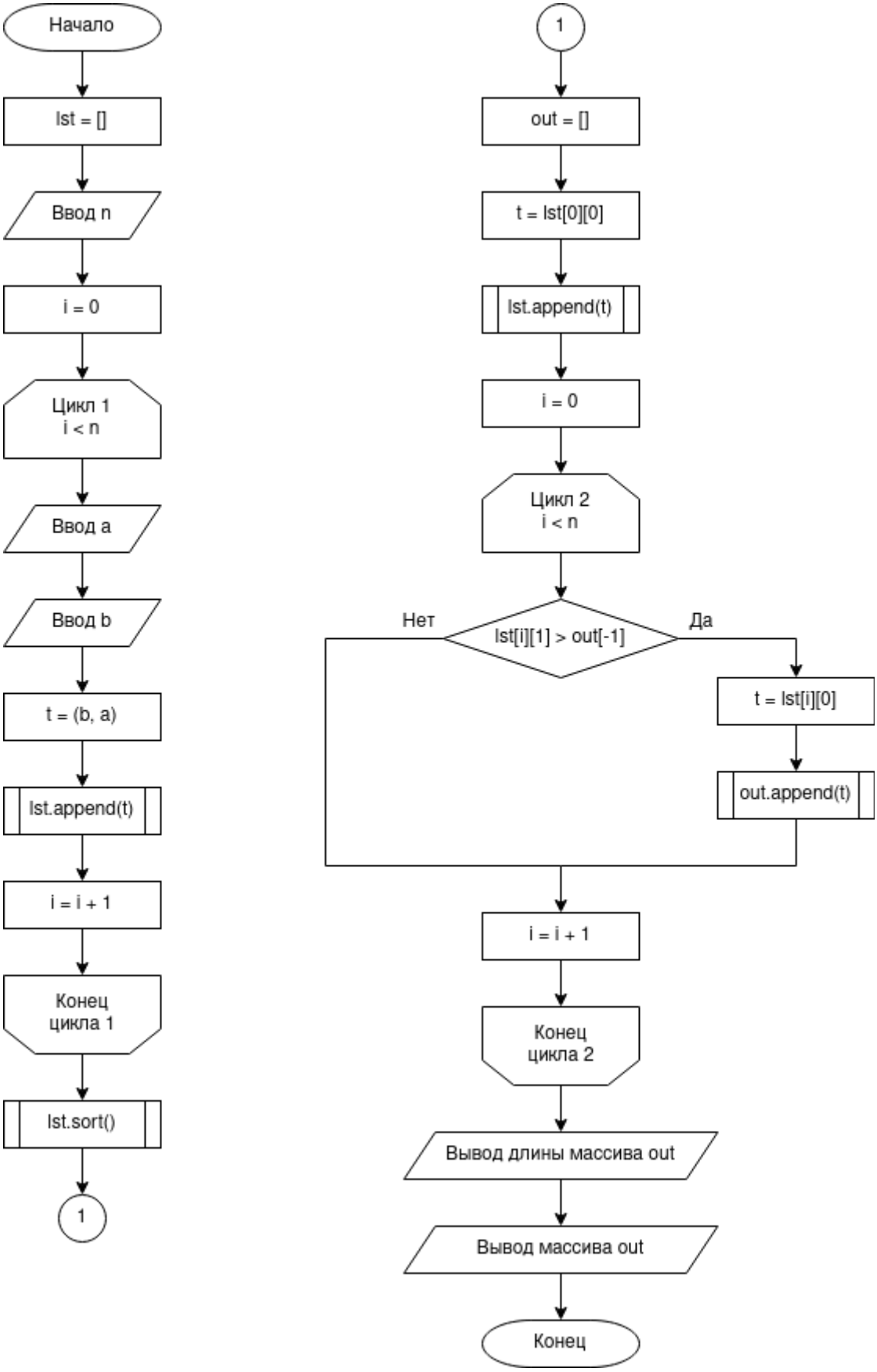

Рис. №6. Блок-схема алгоритма «Покрыть отрезки точками»

Тема №4. Занятие №1. Разбор задачи «Непрерывный рюкзак»

Цель: Ознакомиться с алгоритмом решения задачи о рюкзаке.

Задача: Первая строка содержит количество предметов $1 \le n \le 10^3$ и вместимость рюкзака $0 \le W \le 2 \cdot 10^6$. Каждая из следующих n строк задаёт стоимость $0 \le c_i \le 2 \cdot 10^6$ и объём $0 < w_i \le 2 \cdot 10^6$ предмета (n, W, c_i, w_i — целые числа). Выведите максимальную стоимость частей предметов (от каждого предмета можно отделить любую часть, стоимость и объём при этом пропорционально уменьшатся), помещающихся в данный рюкзак, с точностью не менее трёх знаков после запятой.

Реализация:

Листинг №7. Алгоритм 4.1.10

```
n, W = map(int, input().split())
l = []

for _ in range(n):
    a, b = map(float, input().split())
    t = (a, b, a / b)
    l.append(t)

l.sort(key=lambda x: -x[2])

out = 0

while True:
    if l[0][1] < W:
        W -= l[0][1]
        out += l[0][0]
        l.pop(0)
    else:
        out += l[0][0] * (1 - (l[0][1] - W) / l[0][1])
        break

    if not l:
        break

print(f"{out:.3f}")
```

Ссылка на задачу: https://stepik.org/lesson/13238/step/10

Блок-схема:

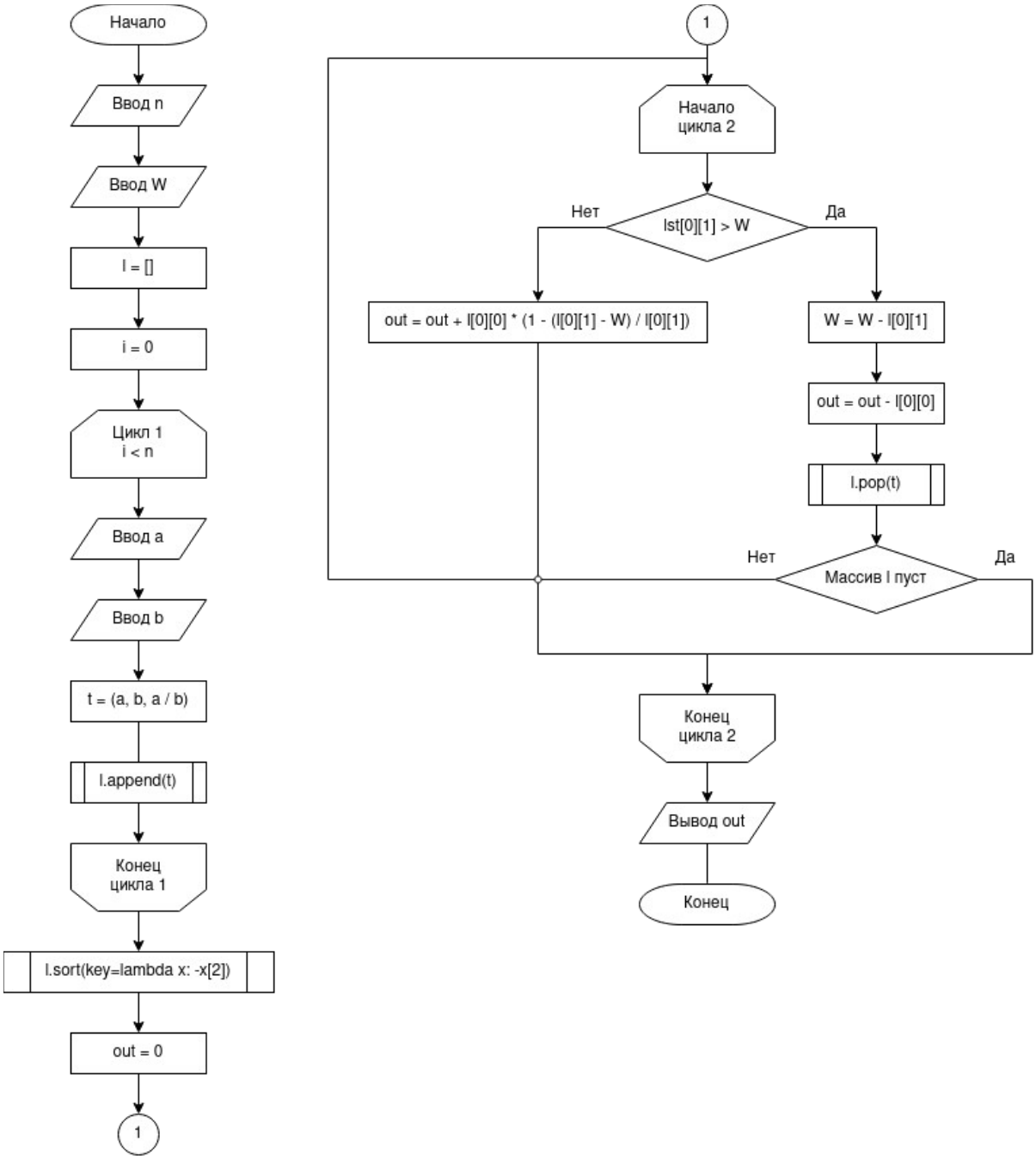

Рис. №7. Блок-схема алгоритма «Непрерывный рюкзак»

Тема №4. Занятие №1. Разбор задачи «Различные слагаемые»

Цель: Ознакомиться с алгоритмом для нахождения максимального числа k для которого по данному n можно представить как сумму k различных натуральных слагаемых.

Задача: По данному числу $1 \le n \le 10^9$ найдите максимальное число k, для которого n можно представить как сумму k различных натуральных слагаемых. Выведите в первой строке число k, во второй — k слагаемых.

Реализация:

Листинг №8. Алгоритм 4.1.11

```
n = int(input())
l = []

for i in range(1, n + 1):
    o = n
    n -= i
    if n >= i + 1:
        l.append(i)
    else:
        l.append(o)
        break

print(len(l))
print(*l)
```

Ссылка на задачу: https://stepik.org/lesson/13238/step/11

Блок-схема:

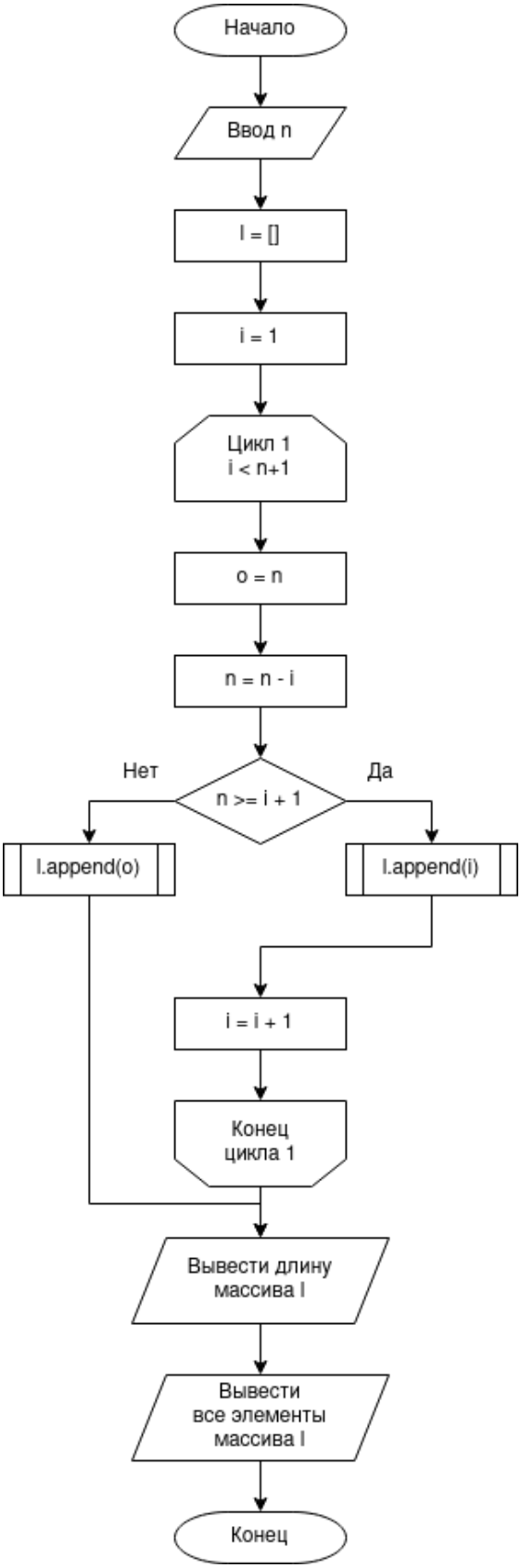

Рис. №8. Блок-схема алгоритма «Различные слагаемые»

Тема №4. Занятие №3. Разбор задачи «Очередь с приоритетами»

Цель: Ознакомиться со структорой данных «куча» и её свойствами и методами.

Задача: Первая строка входа содержит число операций $1 \le n \le 10^5$. Каждая из последующих n строк задают операцию одного из следующих двух типов:

- Insert x, где $0 \le x \le 10^9$ — целое число;
- ExtractMax.

Первая операция добавляет число x в очередь с приоритетами, вторая — извлекает максимальное число и выводит его.

Реализация:

Листинг №9. Алгоритм 4.3.8

```
from heapq import *

n = int(input())

l = []

for i in range(n):
    j = input().split()
    if j[0] == 'Insert':
        j[1] = -int(j[1])
        heappush(l, j[1])
    else:
        t = heappop(l)
        t = -t
        print(t)
```

Ссылка на задачу: https://stepik.org/lesson/13240/step/8

Блок-схема:

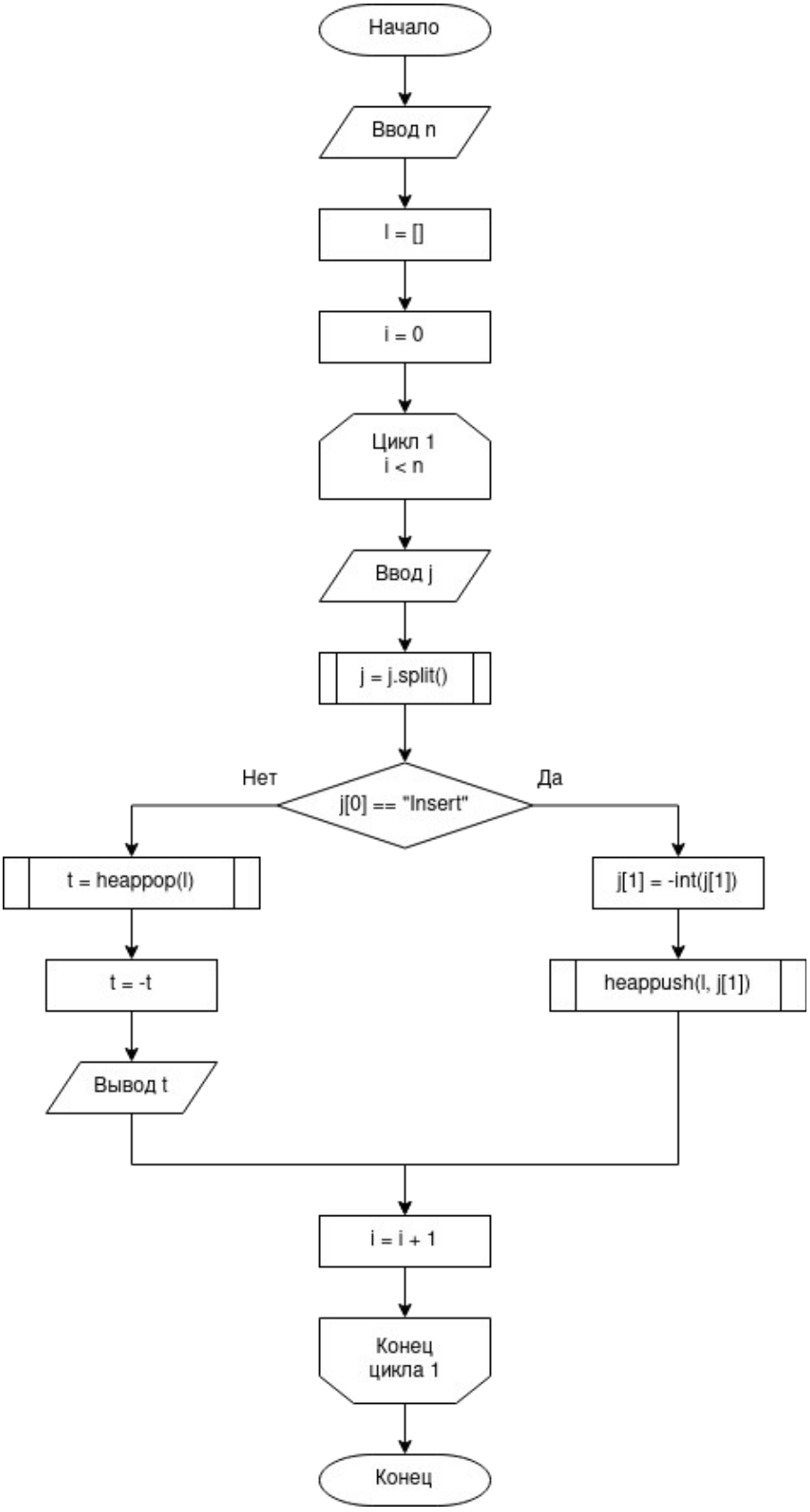

Рис. №9. Блок-схема алгоритма «Очередь с приоритетами»

Тема №6. Занятие №5. Разбор задачи «Точки и отрезки»

Цель: Ознакомиться с алгоритмом для нахождения количества отрезков к котором принадлежат точки.

Задача: В первой строке задано два целых числа $1 \le n \le 50000$ и $1 \le m \le 50000$ — количество отрезков и точек на прямой, соответственно. Следующие n строк содержат по два целых числа a_i и b_i ($a_i \le b_i$) — координаты концов отрезков. Последняя строка содержит m целых чисел — координаты точек. Все координаты не превышают 10^8 по модулю. Точка считается принадлежащей отрезку, если она находится внутри него или на границе. Для каждой точки в порядке появления во вводе выведите, скольким отрезкам она принадлежит.

Реализация:

Листинг №10. Алгоритм 6.5.6

```
n, m = [int(x) for x in input().split()]

a = []
for i in range(0, n):
    l, r = [int(x) for x in input().split()]
    a.append([l, -1])
    a.append([r, 1])

p = [int(x) for x in input().split()]
for i in range(m):
    a.append([p[i], 0, i])


a.sort()
res = [0] * m
curr = 0
for i in range(len(a)):
    if a[i][1] == 1:
        curr += -1
    if a[i][1] == -1:
        curr -= -1
    if a[i][1] == 0:
        res[a[i][2]] = curr

print(*res)
```

Ссылка на задачу: https://stepik.org/lesson/13249/step/6

Блок-схема:

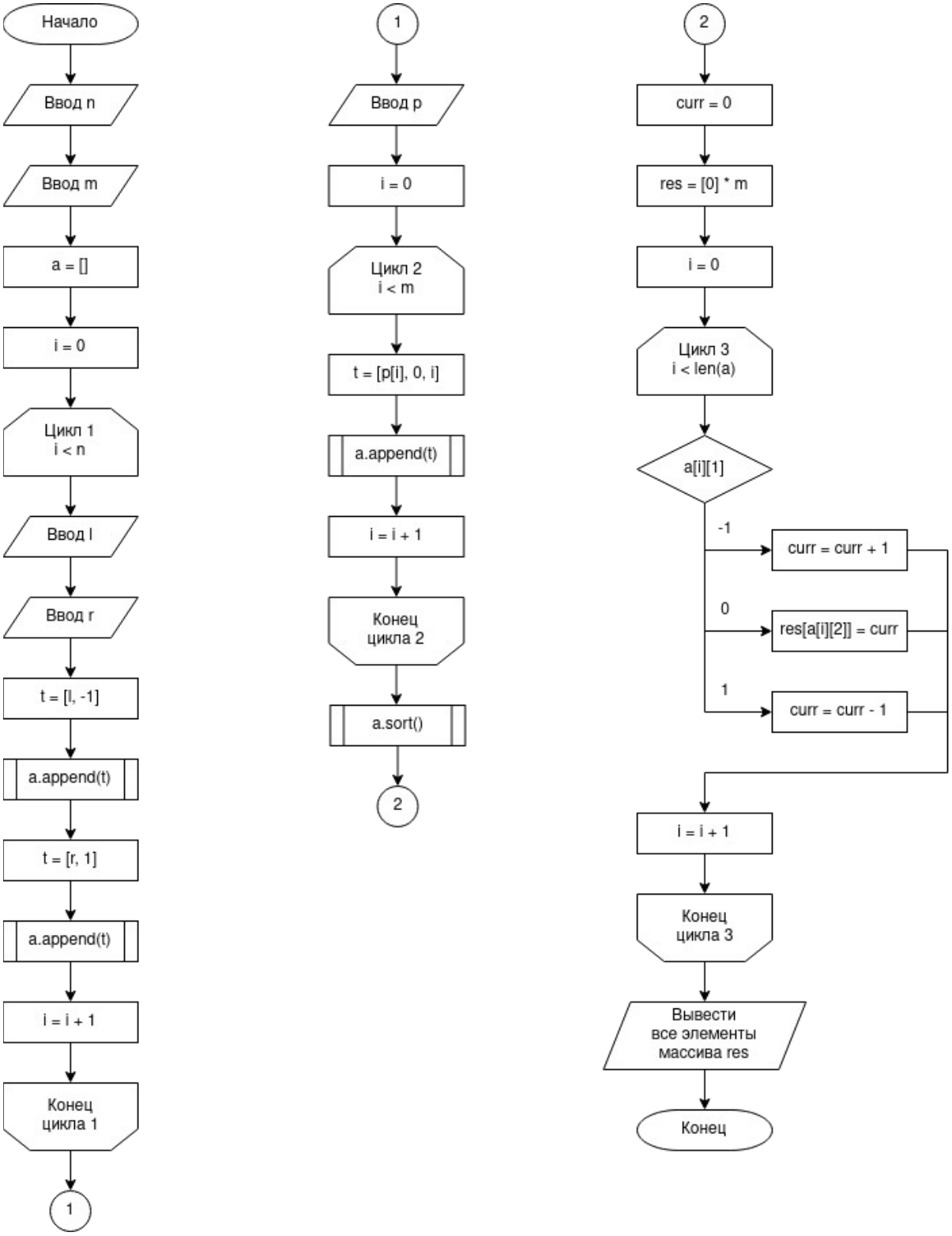

Рис. №10. Блок-схема алгоритма «Точки и отрезки»

Тема №6. Занятие №8. Разбор задачи «Сортировка подсчётом»

Цель: Ознакомиться с алгоритмом сортировки подсчётом.

Задача: Первая строка содержит число $1 \le n \le 10^4$, вторая — n натуральных чисел, не превышающих 10. Выведите упорядоченную по неубыванию последовательность этих чисел.

Реализация:

Листинг №11. Алгоритм 6.8.3

```
n = int(input())
s = list(map(int, input().split()))

mn = min(s)
mx = max(s)

k = mx - mn + 1
count = [0] * k

for i in s:
    count[i - mn] += 1

i = 0
for j in range(k):
    for _ in range(count[j]):
        s[i] = j + mn
        i += 1

print(*s)
```

Ссылка на задачу: https://stepik.org/lesson/13252/step/3

Блок-схема:

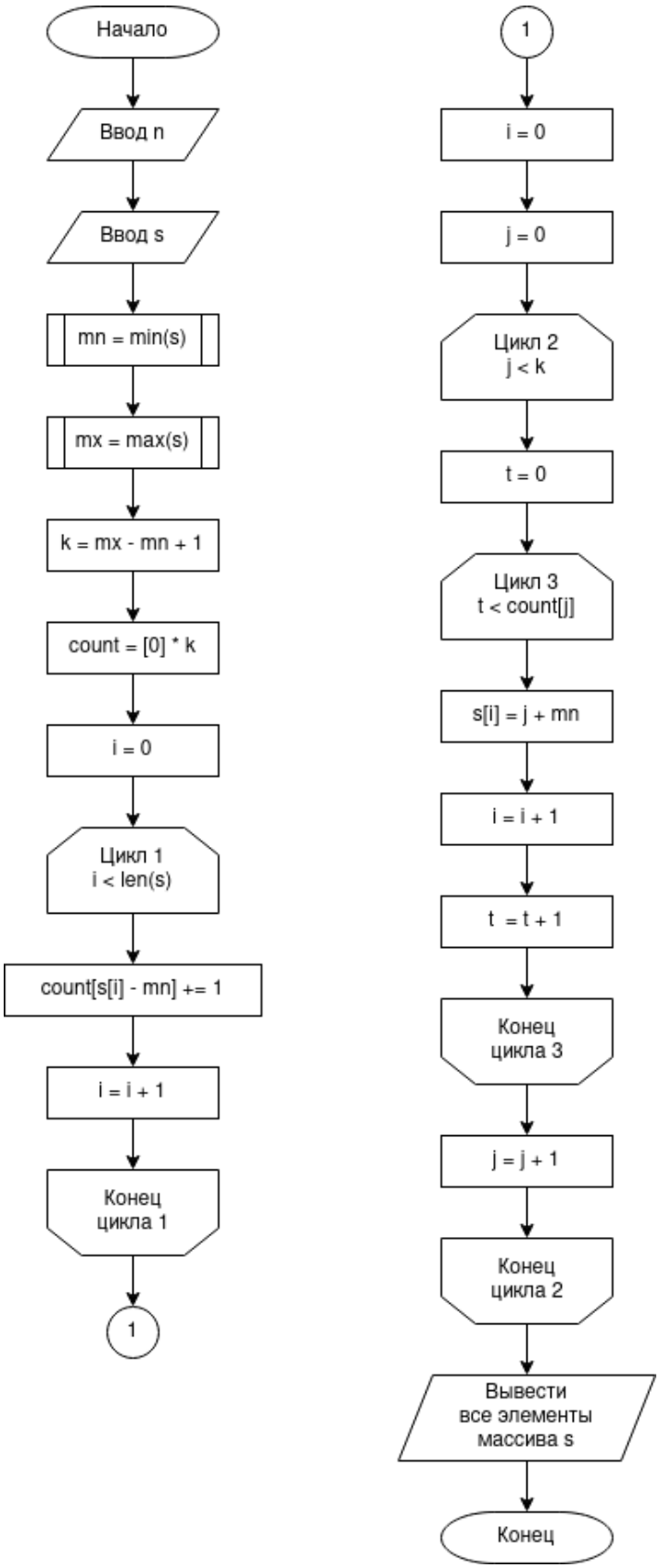

Рис. №11. Блок-схема алгоритма сортировки подсчётом

Тема №8. Занятие №3. Разбор задачи «Расстояние редактирования»

Цель: Ознакомиться с алгоритмом расстояния редактирования Левенштейна.

Задача: Вычислите расстояние редактирования двух данных непустых строк длины не более 10^2, содержащих строчные буквы латинского алфавита.

Реализация:

Листинг №12. Алгоритм 8.3.8

```
a, b = input(), input()
n, m = len(a), len(b)

if n > m:
    a, b = b, a
    n, m = m, n

curr_row = range(n + 1)

for i in range(1, m + 1):
    prev_row, curr_row = curr_row, [i] + [0] * n
    for j in range(1, n + 1):
        add = prev_row[j] + 1
        delete = curr_row[j - 1] + 1
        change = prev_row[j - 1]
        if a[j - 1] != b[i - 1]:
            change += 1
        curr_row[j] = min(add, delete, change)

print(curr_row[n])
```

Ссылка на задачу: https://stepik.org/lesson/13258/step/8

Блок-схема:

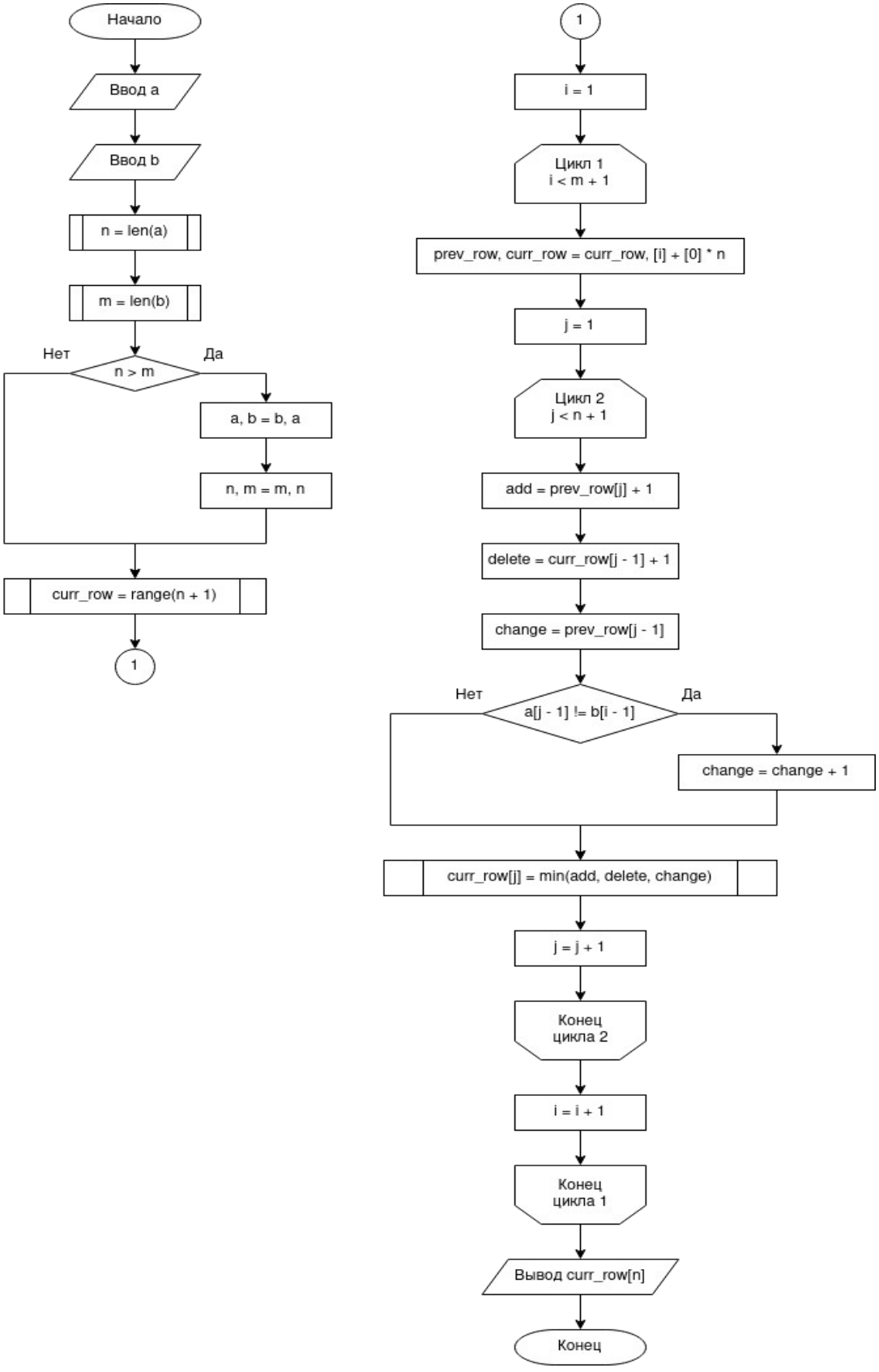

Рис. №12. Блок-схема алгоритма Левенштейна

Курс «Алгоритмы: теория и практика. Структуры данных»

Тема №1. Занятие №2. Разбор задачи «Расстановка скобок в коде»

Цель: Ознакомиться с алгоритмом корректности расстановки скобок.

Задача: Если скобки расставлены правильно, вывести строку «Success». В противном случае вывести индекс (используя индексацию с единицы) первой закрывающей скобки, для которой нет соответствующей открывающей. Если такой нет, вывести индекс первой открывающей скобки, для которой нет соответствующей закрывающей.

Реализация:

Листинг №13. Алгоритм 1.2.1

```
def check(string):

    stack = []
    open_brackets = ['(','[','{']
    closed_brackets = [')',']','}']

    t = len(string)
    for i in range(t):

        if string[i] in open_brackets:
            stack.append(string[i])
            stack.append(i+1)

        if string[i] in closed_brackets:
            if len(stack) == 0:
                return i+1
            if open_brackets.index(stack[-2]) ==
closed_brackets.index(string[i]):
                stack.pop()
                stack.pop()
            else:
                return i+1

    if not stack:
        return "Success"

    return stack[-1]

t = input()
t = check(t)
print(t)
```

Ссылка на задачу: https://stepik.org/lesson/41234/step/1

Блок-схема:

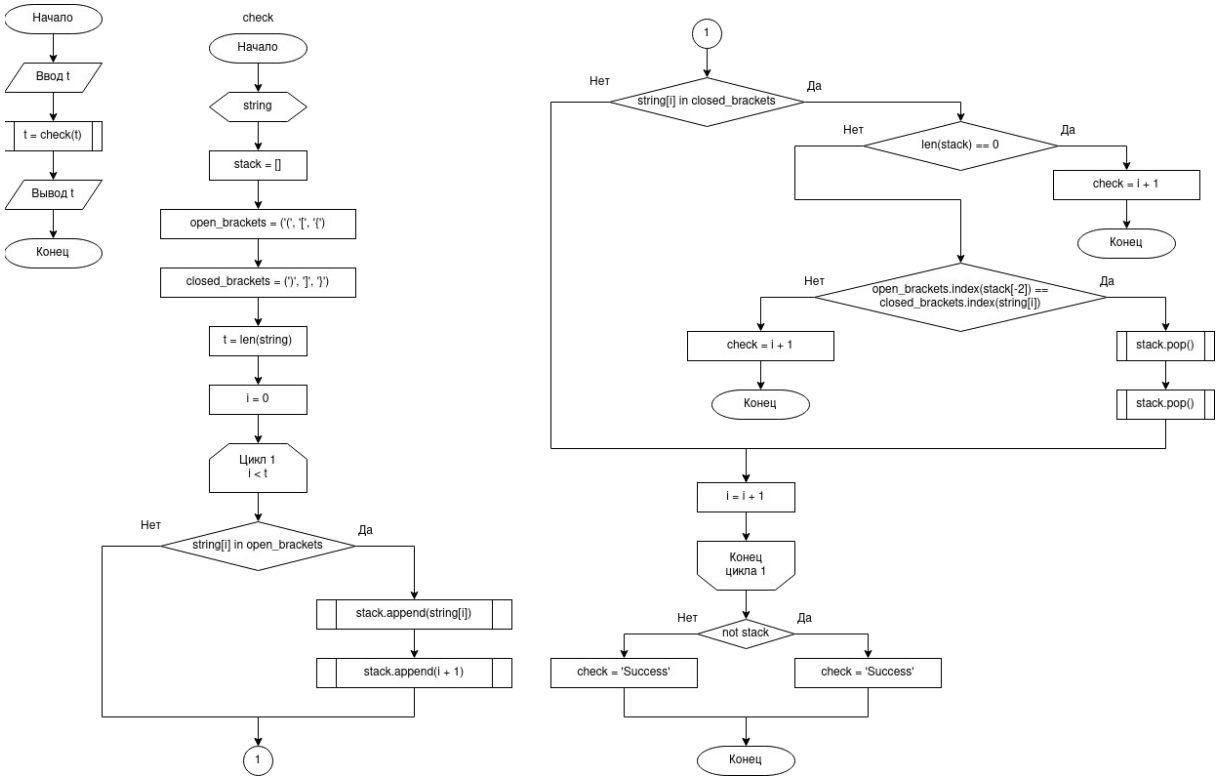

Рис. №13. Блок-схема задачи 1.2 «Расстановка скобок в коде»

Тема №1. Занятие №2. Разбор задачи «Высота дерева»

Цель: Научиться хранить и эффективно обрабатывать деревья, даже если в них сотни тысяч вершин.

Задача: Вычислить высоту данного дерева.

Реализация:

Листинг №14. Алгоритм 1.2.2

```
n = int(input())
nodes = list(map(int, input().split()))

tree = {}
for i in range(n):
    parent = nodes[i]
    if parent in tree:
        tree[parent].append(i)
    else:
        tree[parent] = [i]
depth = 0
children = tree[-1]
while len(children):
    depth += 1
    new_children = []
    for child in children:
        if child in tree:
            new_children.extend(tree[child])
    children = new_children

print(depth)
```

Ссылка на задачу: https://stepik.org/lesson/41234/step/2

Блок-схема:

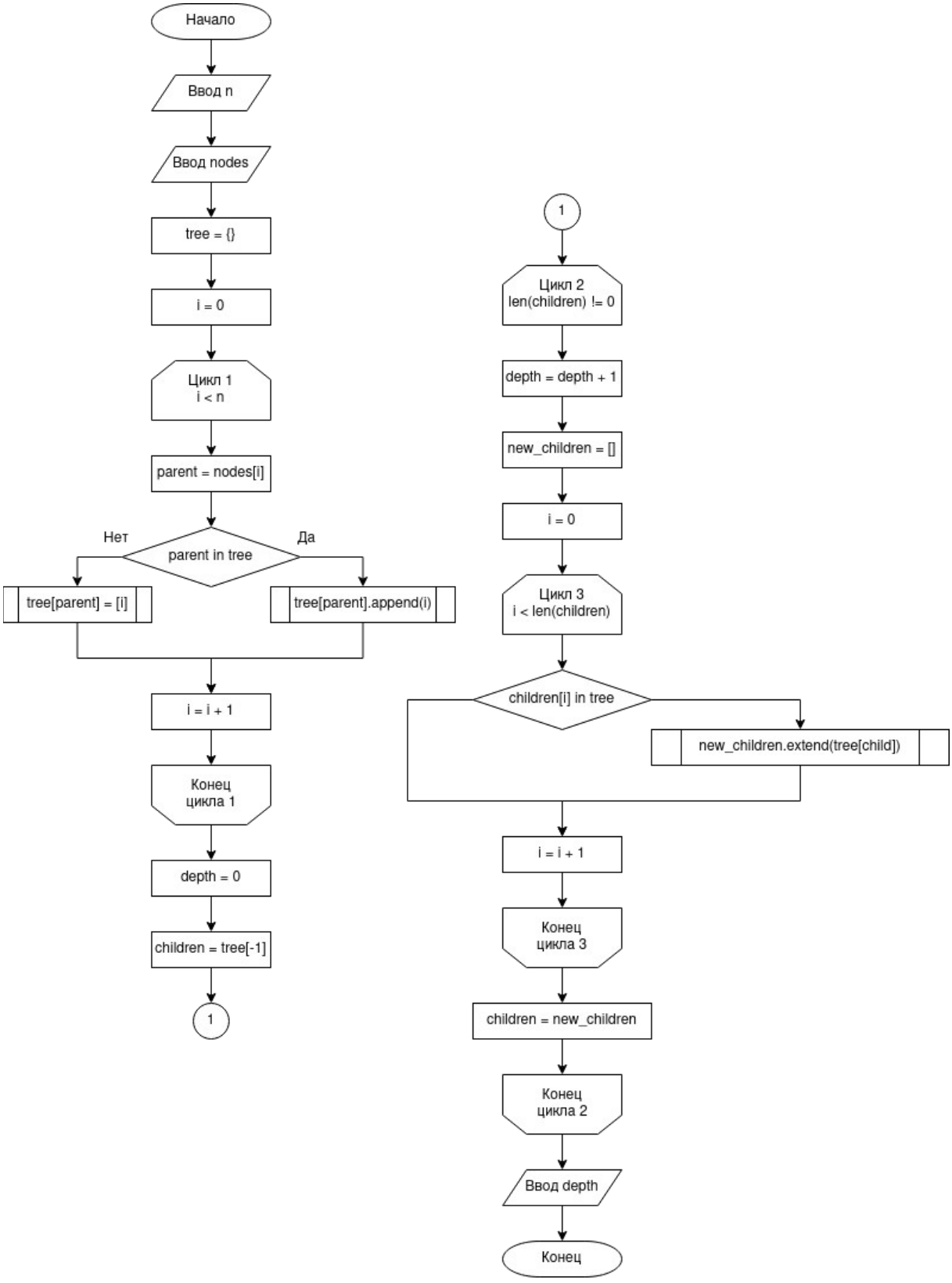

Рис. №14. Блок-схема задачи 1.2 «Высота дерева»

Тема №1. Занятие №2. Разбор задачи «Стек с поддержкой максимума»

Цель: Расширить интерфейс стека так, чтобы он дополнительно поддерживал операцию max и при этом чтобы время работы всех операций по-прежнему было константным.

Задача: Реализовать стек с поддержкой операций push, pop и max.

Реализация:

Листинг №15. Алгоритм 1.2.4

```
n = int(input())
stack = []

for i in range(n):
    operation = input().split()
    if operation[0] == "push":
        v = int(operation[1])
        v_max = max(v, stack[-1][1]) if stack else v
        stack.append((v, v_max))
    elif operation[0] == "pop":
        if stack:
            stack.pop()
    elif operation[0] == "max":
        if stack:
            print(stack[-1][1])
```

Ссылка на задачу: https://stepik.org/lesson/41234/step/4

Блок-схема:

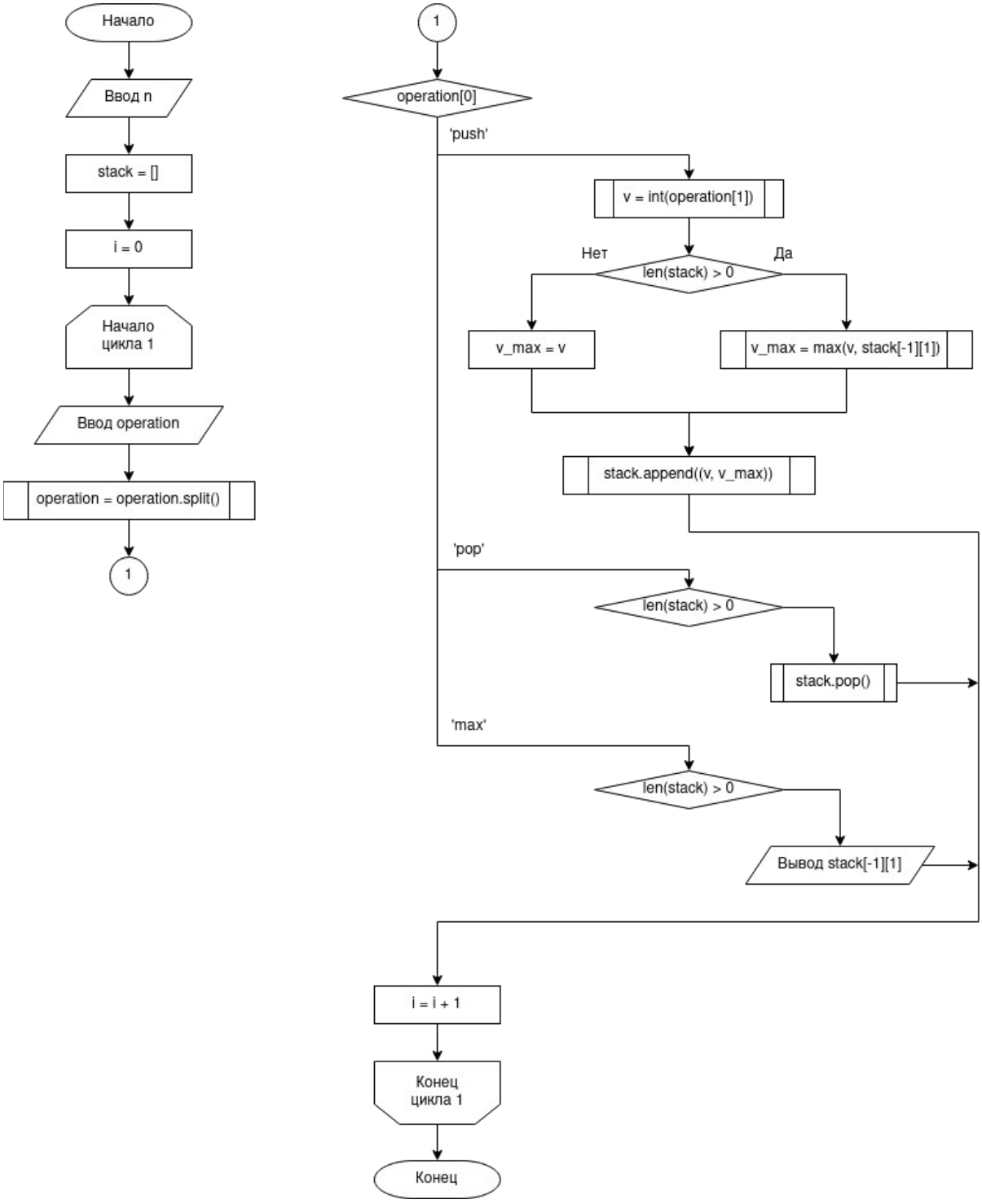

Рис. №15. Блок-схема задачи 1.2 «Стек с поддержкой максимума»

Тема №1. Занятие №2. Разбор задачи «Максимум в скользящем окне»

Цель: Реализовать алгоритм задачи со временем работы O(n)

Задача: Найти максимум в каждом окне размера m данного массива чисел.

Реализация:

Листинг №16. Алгоритм 1.2.5

```
from collections import deque

n = int(input())
arr = list(map(int, input().split()))
k = int(input())

Qi = deque()

for i in range(k):
    while Qi and arr[i] >= arr[Qi[-1]]:
        Qi.pop()
    Qi.append(i)

for i in range(k, n):
    print(str(arr[Qi[0]]) + " ", end="")
    while Qi and Qi[0] <= i-k:
        Qi.popleft()
    while Qi and arr[i] >= arr[Qi[-1]]:
        Qi.pop()
    Qi.append(i)

print(arr[Qi[0]])
```

Ссылка на задачу: https://stepik.org/lesson/41234/step/5

Блок-схема:

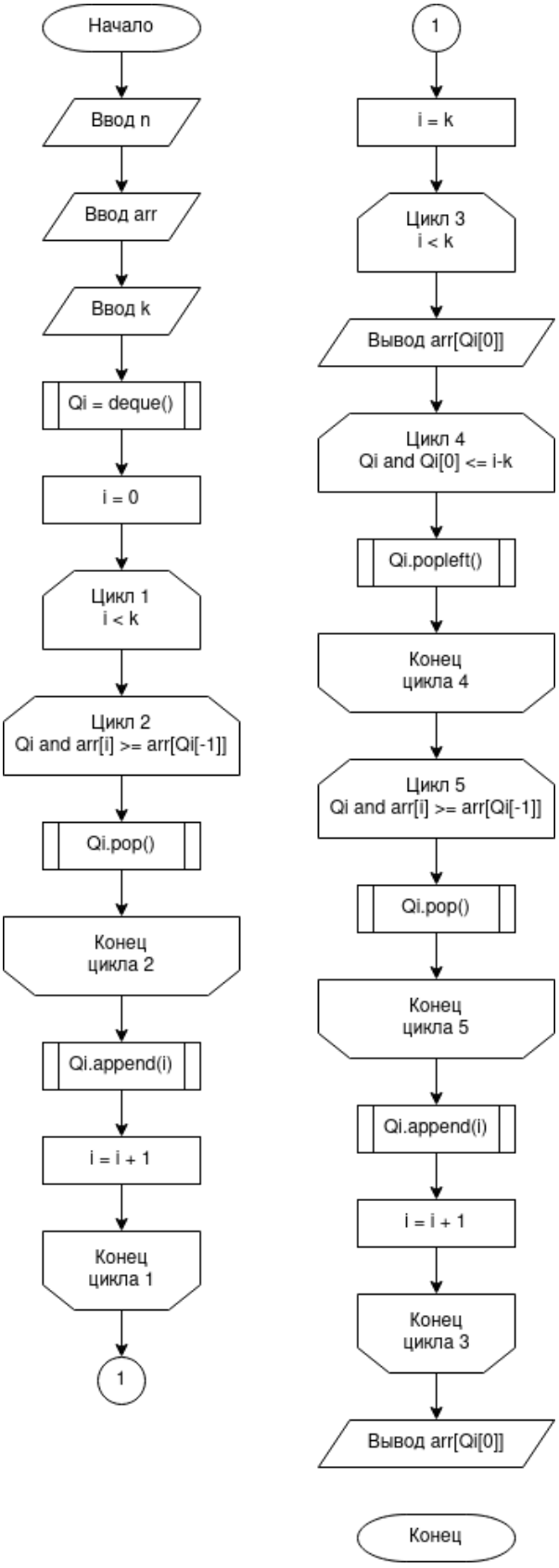

Рис. №16. Блок-схема задачи 1.2 «Максимум в скользящем окне»

Тема №2. Занятие №3. Разбор задачи «Построение кучи»

Цель: Преобразовать заданный массив в кучу за линейное количество обменов.

Задача: Переставить элементы заданного массива чисел так, чтобы он удовлетворял свойству мин-кучи.

Реализация:

Листинг №17. Алгоритм 2.3.1

```
from sys import stdin

n, *arr = [int(x) for x in stdin.read().split()]
answer = []

for i in range((n - 2) // 2, -1, -1):
    node = i

    while True:
        min_index = node
        left = 2 * node + 1
        right = 2 * node + 2

        if left < n and arr[min_index] > arr[left]:
            min_index = left

        if right < n and arr[min_index] > arr[right]:
            min_index = right

        if arr[node] == arr[min_index]:
            break
        answer += [(node, min_index)]
        arr[node], arr[min_index] = arr[min_index], arr[node]
        node = min_index

print(len(answer))
for item in answer:
    print(*item)
```

Ссылка на задачу: https://stepik.org/lesson/41560/step/1

Блок-схема:

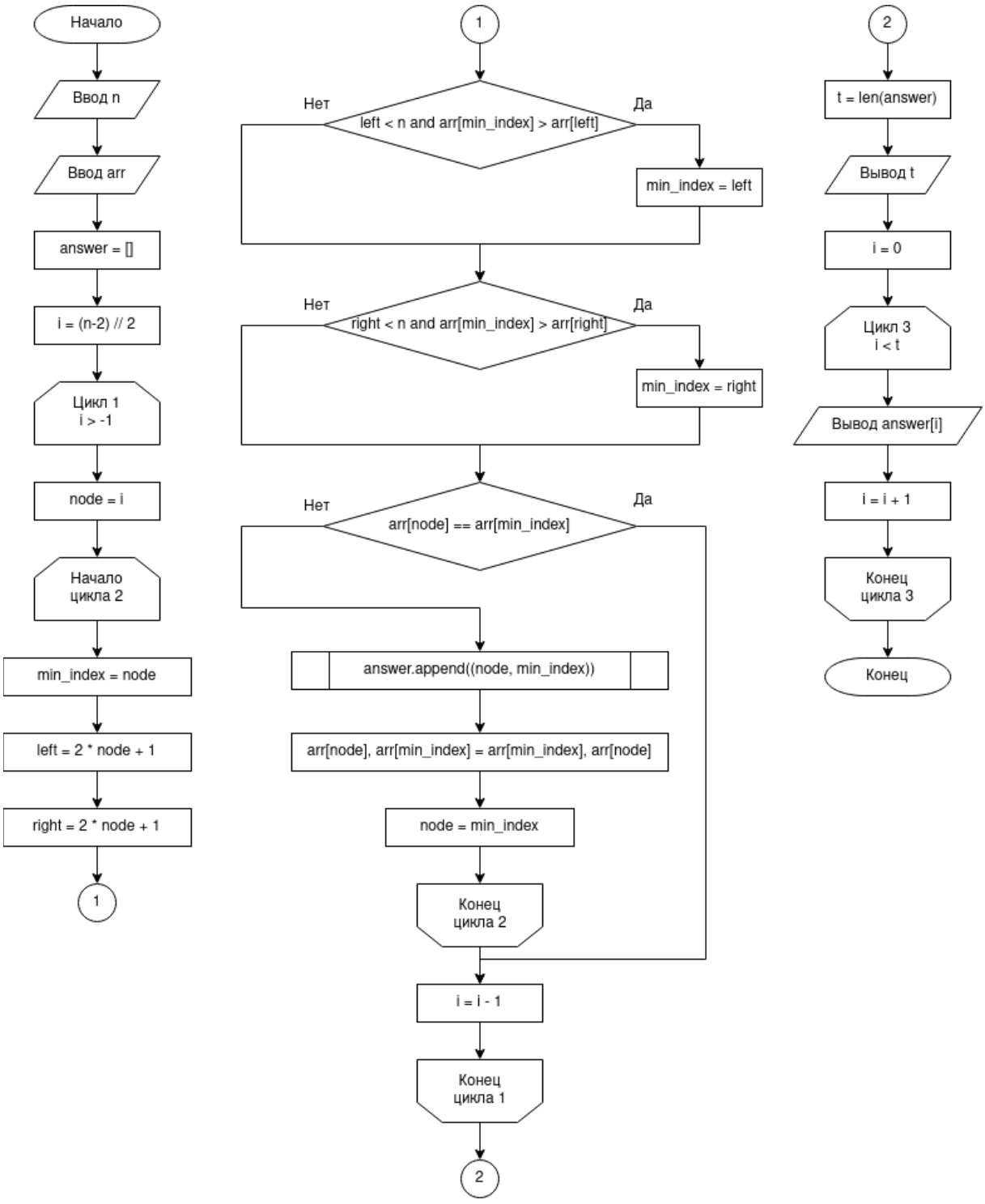

Рис. №17. Блок-схема задачи 2.3 «Построение кучи»

Тема №3. Занятие №2. Разбор задачи «Телефонная книга»

Цель: Реализовать простую телефонную книгу, поддерживающую следующие типы запросов:

- add number name: добавить запись с именем name и телефонным номером number. Если запись с таким телефонным номером уже есть, нужно заменить в ней имя на name;
- del number: удалить запись с соответствующим телефонным номером. Если такой записи нет, ничего не делать;
- find number: найти имя записи с телефонным номером number. Если запись с таким номером есть, вывести имя. В противном случае вывести «not found» (без кавычек).

Задача: Реализовать структуру данных, эффективно обрабатывающую запросы вида add number name, del number и find number.

Реализация:

Листинг №18. Алгоритм 3.2.1

```
n = int(input())

s = {}

for i in (input().split() for _ in range(n)):
    if i[0] == 'add':
        s[i[1]] = i[2]
    elif i[0] == 'del':
        s.pop(i[1], None)
    elif i[0] == 'find':
        print(s.get(i[1], 'not found'))
```

Ссылка на задачу: https://stepik.org/lesson/41562/step/1

Блок-схема:

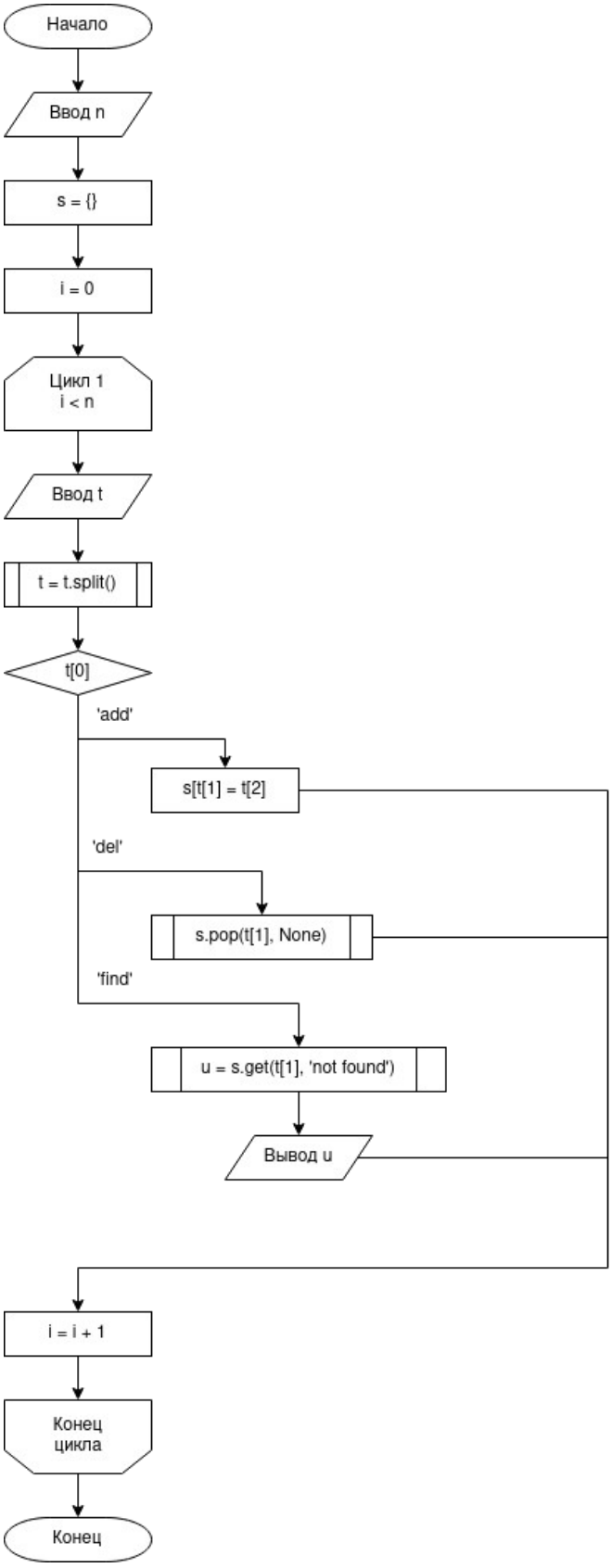

Рис. №18. Блок-схема задачи 3.2 «Телефонная книга»

Заключение

В процессе прохождения учебной практики на образовательной платформе «Stepik» на курсах по методам и структурам данных по предмету Системное программирование были получены следующие навыки в области базовых алгоритмических методов и структур данных, наиболее часто использующихся на практике:

- «Жадные» алгоритмы;
- Метод «Разделяй и властвуй»;
- Динамическое программирование;
- Массивы;
- Спики;
- Очереди и стеки;
- Динамические массивы;
- Очереди с приоритетами;
- Системы непересекающихся множеств;
- Хеш-таблицы;
- Сбалансированные деревья.