МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет

им. А.Н. Туполева – КАИ»

Институт компьютерных технологий и защиты информации

Отделение СПО ИКТЗИ (Колледж информационных технологий)

ЛАБОРАТОРНАЯ РАБОТА № 2

по дисциплине

[**Поддержка и тестирование программных модулей**](https://bb.kai.ru/webapps/blackboard/execute/courseMain?course_id=_16315_1)

Тема: «Внутренняя сортировка данных»

Работу выполнил

Студент гр.4335

Желваков А. С.

Принял

Преподаватель Шумилкин А. О.

Казань 2024

***ВАРИАНТ 9, 18***

1. **Цель работы**

Отработать навыки составления и тестирования программ как «белого ящика».

1. **Задание на лабораторную работу**

9. Даны натуральное число N и одномерный массив A1, A2, …, AN вещественных чисел. Получить все элементы, входящие в данный массив по одному разу.

18. Даны натуральное число N и одномерный массив A1, A2, …, AN целых чисел. Найти наименьший элемент в наиболее длинной непрерывной последовательности положительных значений.

1. **Результат выполнения работы**

Вариант 9:

**Алгоритм работы**

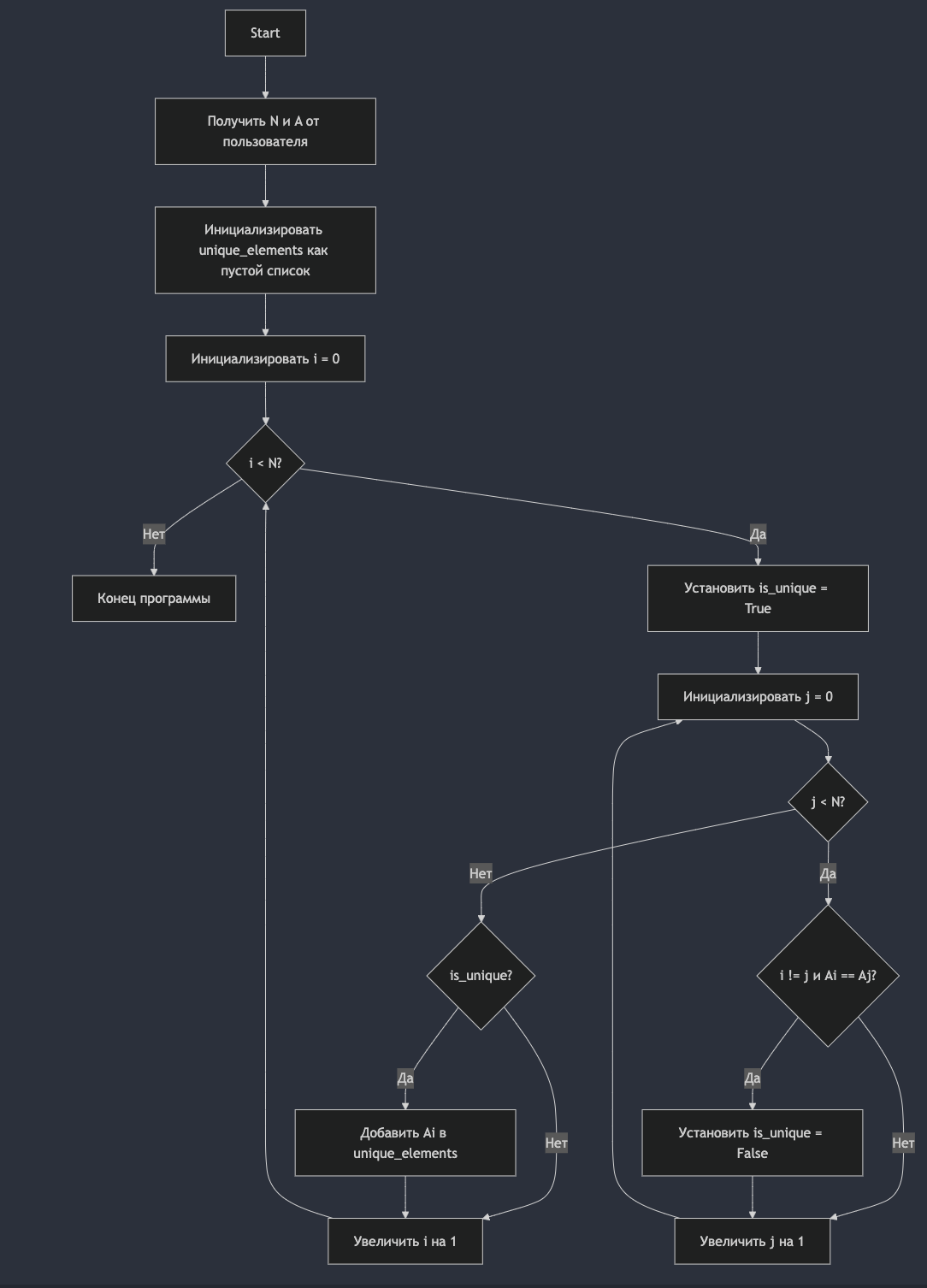


Рис. 1 - Алгоритм работы программы 9-го варианта

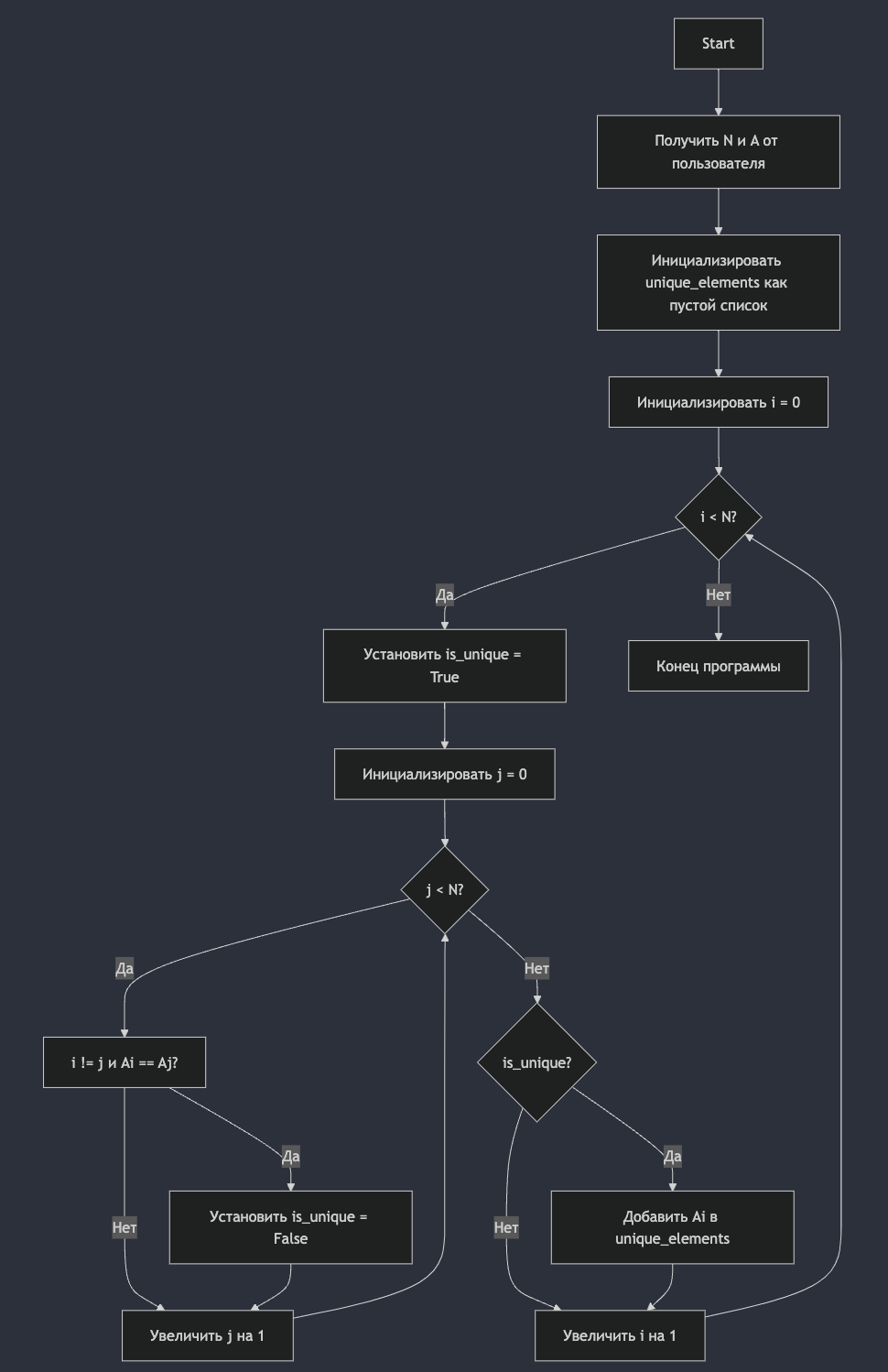


Рис. 2 - Потоковый граф работы алгоритма 9-го варианта

### **Определение цикломатической сложности**

Для вычисления цикломатической сложности воспользуемся формулами:

**Первая формула:** *V(G)=E−N+2V(G) = E - N + 2*V(G)=E−N+2

* + E — количество ребер в графе.
  + N — количество узлов в графе.

**Вторая формула:** *V(G)=P+1V(G) = P + 1*V(G)=P+1

* + P — количество независимых путей в графе.

Для данного алгоритма:

* Узлы (N) = 10
* Ребра (E) = 20

Цикломатическая сложность:

*V(G)=20−10+2=12V(G) = 20 - 10 + 2 = 12*V(G)=20−10+2=12

Либо по другой формуле:

*V(G)=11(количествоциклов+независимыхпутей)+1=12V(G) = 11 (количество циклов + независимых путей) + 1 = 12*V(G)=11(количествоциклов+независимыхпутей)+1=12

Таким образом, цикломатическая сложность программы равна 12, что отражает количество независимых путей через потоковый граф.

### **Базовое множество независимых линейных путей для программы**

**Путь 1:** Вход -> Получение N и A -> Начало внешнего цикла i (i < N) -> is\_unique = True -> Нет повторений -> Добавить A[i] в unique\_elements -> Увеличение i -> Конец программы.

* + Путь: Start -> A -> B -> C -> D -> E -> H (Да) -> I -> J -> D (Нет) -> End

**Путь 2:** Вход -> Получение N и A -> Начало внешнего цикла i (i < N) -> is\_unique = True -> Начало внутреннего цикла j (j < N) -> Нет совпадений для j -> Проверка is\_unique -> Добавить A[i] в unique\_elements -> Увеличение i -> Конец программы.

* + Путь: Start -> A -> B -> C -> D -> E -> F -> G (Нет) -> H (Да) -> I -> J -> D (Нет) -> End

**Путь 3:** Вход -> Получение N и A -> Начало внешнего цикла i (i < N) -> is\_unique = True -> Начало внутреннего цикла j (j < N) -> Найдено совпадение (A[i] == A[j]) -> is\_unique = False -> Проверка is\_unique -> Пропустить добавление в unique\_elements -> Увеличение i -> Конец программы.

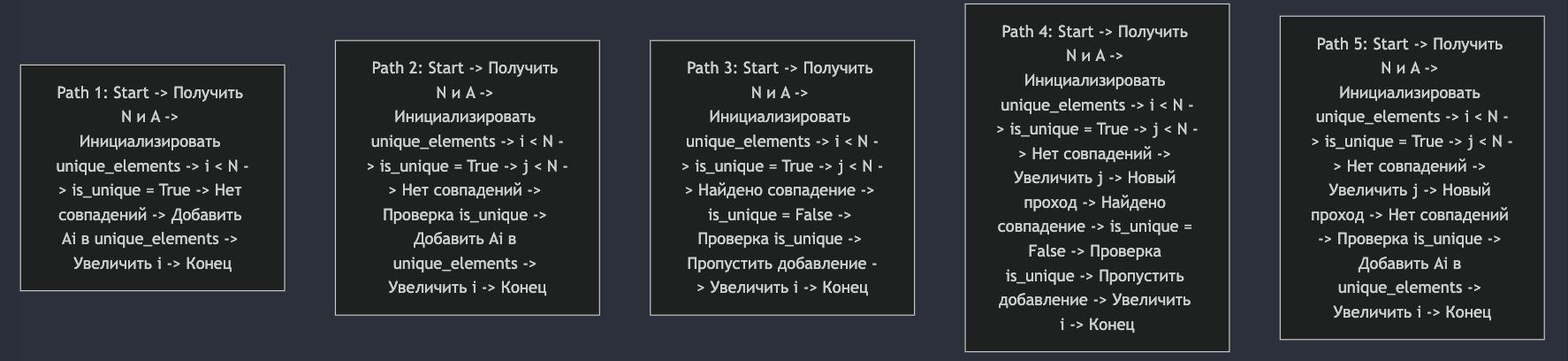
* + Путь: Start -> A -> B -> C -> D -> E -> F -> G (Да) -> K (Да) -> M -> H (Нет) -> J -> D (Нет) -> End

**Путь 4:** Вход -> Получение N и A -> Начало внешнего цикла i (i < N) -> is\_unique = True -> Начало внутреннего цикла j (j < N) -> Нет совпадений -> Увеличение j -> Новый проход внутреннего цикла -> Найдено совпадение (A[i] == A[j]) -> is\_unique = False -> Проверка is\_unique -> Пропустить добавление в unique\_elements -> Увеличение i -> Конец программы.

* + Путь: Start -> A -> B -> C -> D -> E -> F -> G (Да) -> K (Нет) -> L -> F -> G (Да) -> K (Да) -> M -> H (Нет) -> J -> D (Нет) -> End

**Путь 5:** Вход -> Получение N и A -> Начало внешнего цикла i (i < N) -> is\_unique = True -> Начало внутреннего цикла j (j < N) -> Нет совпадений -> Увеличение j -> Новый проход внутреннего цикла -> Нет совпадений -> Проверка is\_unique -> Добавить A[i] в unique\_elements -> Увеличение i -> Конец программы.

* + Путь: Start -> A -> B -> C -> D -> E -> F -> G (Да) -> K (Нет) -> L -> F -> G (Да) -> K (Нет) -> L -> G (Нет) -> H (Да) -> I -> J -> D (Нет) -> End

Рис. 3 - Визуализация базового множества независимых линейных путей

**Результаты тестирования**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Данные** | **Ожидаемый результат** | **Фактический результат** | **Результаты тестирования** |
| 1 | N=5, A=[1.0, 2.0, 3.0, 4.0, 5.0] | [1.0, 2.0, 3.0, 4.0, 5.0] | [1.0, 2.0, 3.0, 4.0, 5.0] | Успешно |
| 2 | N=6, A=[1.0, 2.0, 2.0, 3.0, 4.0, 4.0] | [1.0, 3.0] | [1.0, 3.0] | Успешно |
| 3 | N=4, A=[2.0, 2.0, 2.0, 2.0] | [] | [] | Успешно |
| 4 | N=0, A=[] | [] | [] | Успешно |
| 5 | N=1, A=[10.5] | [10.5] | [10.5] | Успешно |
| 6 | N=5, A=[1.0, 1.0, 1.0, 1.0, 2.0] | [2.0] | [2.0] | Успешно |
| 7 | N=7, A=[-1.0, 2.0, -1.0, 3.0, 4.0, 2.0, 5.0] | [3.0, 4.0, 5.0] | [3.0, 4.0, 5.0] | Успешно |
| 8 | N=5, A=[1.1, 2.2, 3.3, 2.2, 4.4] | [1.1, 3.3, 4.4] | [1.1, 3.3, 4.4] | Успешно |

Вариант 18:

**Алгоритм работы**

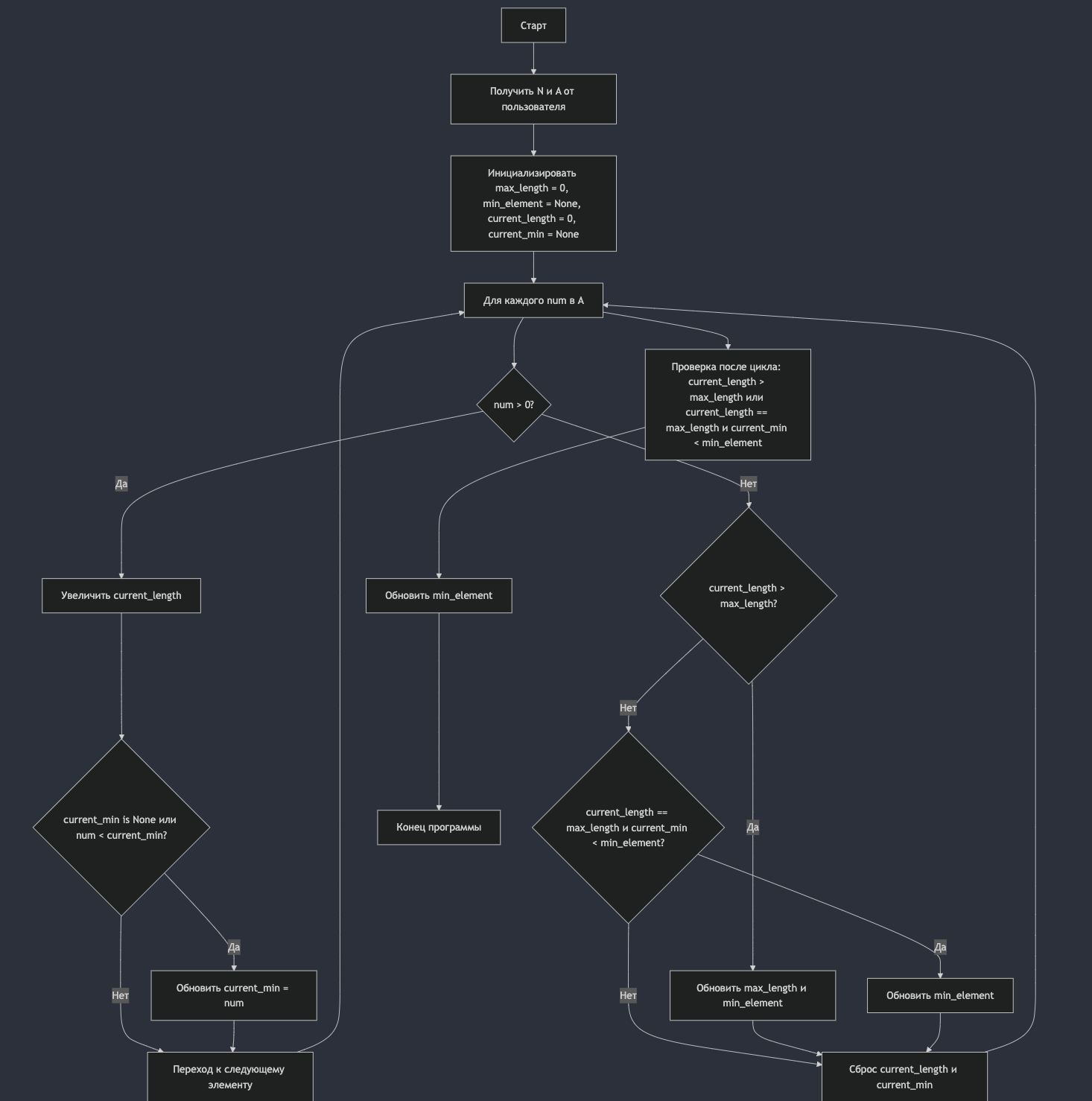


Рис. 4 - Алгоритм работы программы 18-го варианта

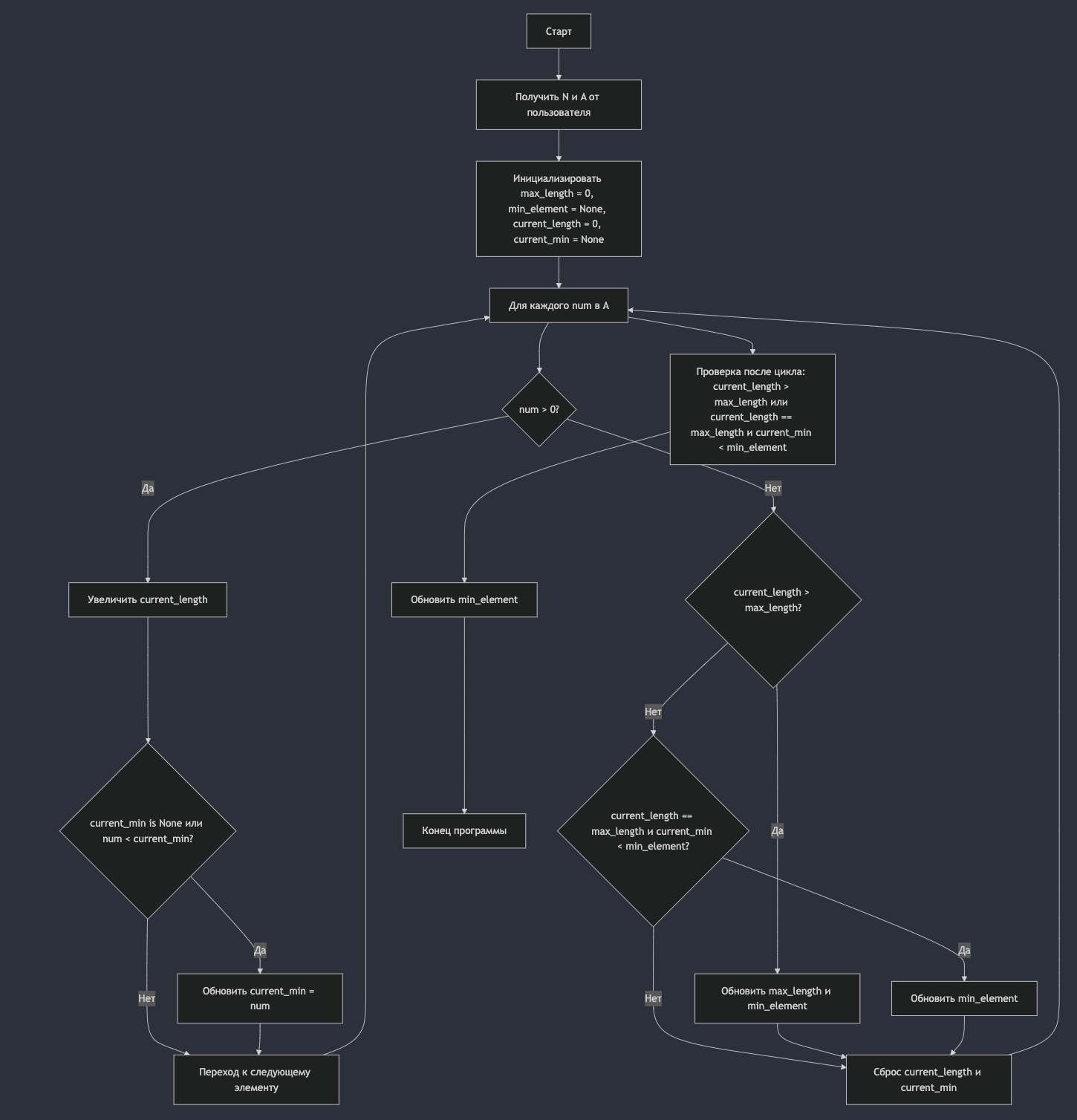


Рис. 5 - Потоковый граф работы алгоритма 18-го задания

### **Определение цикломатической сложности**

Для вычисления цикломатической сложности воспользуемся следующими формулами:

**Первая формула:** *V(G)=E−N+2V(G) = E - N + 2*V(G)=E−N+2

* + E — количество рёбер в графе.
  + N — количество узлов в графе.

**Вторая формула:** *V(G)=P+1V(G) = P + 1*V(G)=P+1

* + P — количество независимых путей в графе.

Для данного алгоритма:

* **Узлы (N)** = 17
* **Рёбра (E)** = 26

Цикломатическая сложность:

*V(G)=26−17+2=11V(G) = 26 - 17 + 2 = 11*V(G)=26−17+2=11

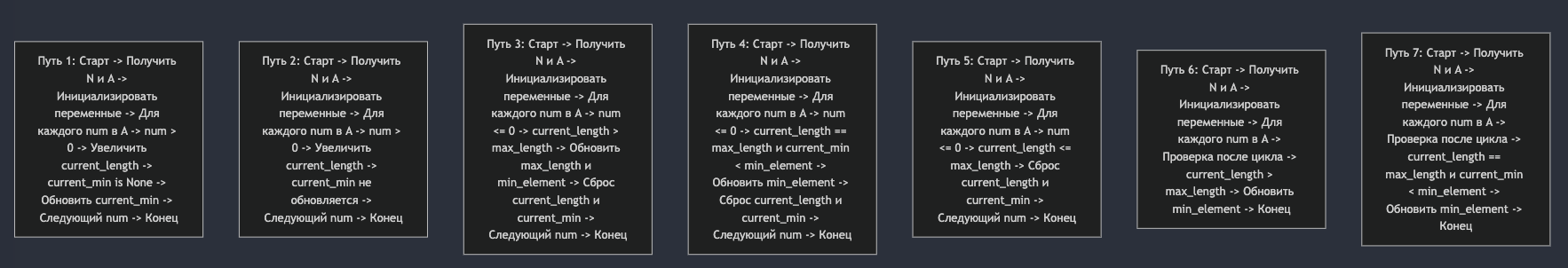
Либо по другой формуле:

*V(G)=10(количествонезависимыхпутей)+1=11V(G) = 10 (количество независимых путей) + 1 = 11*V(G)=10(количествонезависимыхпутей)+1=11

Таким образом, цикломатическая сложность программы равна 11, что отражает количество независимых путей через потоковый граф.

### **Базовое множество независимых линейных путей для программы**

* **Путь 1:** Start -> Получить N и A -> Инициализировать переменные -> Для каждого num в A -> num > 0 -> Увеличить current\_length -> current\_min is None -> Обновить current\_min -> Следующий num -> Конец.
* **Путь 2:** Start -> Получить N и A -> Инициализировать переменные -> Для каждого num в A -> num > 0 -> Увеличить current\_length -> current\_min не обновляется -> Следующий num -> Конец.
* **Путь 3:** Start -> Получить N и A -> Инициализировать переменные -> Для каждого num в A -> num <= 0 -> current\_length > max\_length -> Обновить max\_length и min\_element -> Сброс current\_length и current\_min -> Следующий num -> Конец.
* **Путь 4:** Start -> Получить N и A -> Инициализировать переменные -> Для каждого num в A -> num <= 0 -> current\_length == max\_length и current\_min < min\_element -> Обновить min\_element -> Сброс current\_length и current\_min -> Следующий num -> Конец.
* **Путь 5:** Start -> Получить N и A -> Инициализировать переменные -> Для каждого num в A -> num <= 0 -> current\_length <= max\_length -> Сброс current\_length и current\_min -> Следующий num -> Конец.
* **Путь 6:** Start -> Получить N и A -> Инициализировать переменные -> Для каждого num в A -> Проверка после цикла -> current\_length > max\_length -> Обновить min\_element -> Конец.
* **Путь 7:** Start -> Получить N и A -> Инициализировать переменные -> Для каждого num в A -> Проверка после цикла -> current\_length == max\_length и current\_min < min\_element -> Обновить min\_element -> Конец.

Рис. 6 - Визуализация базового множества независимых линейных путей

**Результаты тестирования**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Данные** | **Ожидаемый результат** | **Фактический результат** | **Результаты тестирования** |
| 1 | N=5, A=[3, 2, 5, 4, 1] | 1 | 1 | Успешно |
| 2 | N=4, A=[-1, 0, -2, -3] | None | None | Успешно |
| 3 | N=6, A=[-1, 2, 3, 4, -2, -3] | 2 | 2 | Успешно |
| 4 | N=10, A=[1, 2, -1, 3, 4, 5, -2, 6, 7, -3] | 3 | 3 | Успешно |
| 5 | N=8, A=[1, 2, -1, 3, 4, -2, 5, 6] | 1 | 1 | Успешно |
| 6 | N=9, A=[3, 4, -1, 2, 3, -2, 4, 5, 6] | 4 | 4 | Успешно |
| 7 | N=5, A=[-1, -2, 5, -3, -4] | 5 | 5 | Успешно |
| 8 | N=7, A=[0, 1, 2, 0, 3, 4, 0] | 1 | 1 | Успешно |
| 9 | N=0, A=[] | None | None | Успешно |
| 10 | N=4, A=[0, 0, 0, 0] | None | None | Успешно |
| 11 | N=7, A=[1, 3, -1, 1, 3, -2, 1] | 1 | 1 | Успешно |
| 12 | N=10, A=[2, 4, -1, 1, 3, 5, -2, 0, 1, 2] | 1 | 1 | Успешно |

1. **Листинг программы**

<https://github.com/ArseniyZh/CIT/tree/main/3rd_year/%D0%9F%D0%B8%D0%A2%D0%9F%D0%9C/%D0%BB%D0%B0%D0%B1%D0%B0%201>

**Вывод:**

Изучены принципы тестирования методом «белого ящика», используя анализ граничных значений.