Министерство образования и науки Российской федерации

Санкт-Петербургский государственный электротехнический университет

“ЛЭТИ”

кафедра математического обеспечения и применения ЭВМ

**Спецификация**

**Преподаватель: Фирсов М.А.**

Факультет: КТИ

Выполнили:

Марущак Ю.В.

Левицкий Д.В.

Карпенко Д.Р.

Санкт-Петербург

2016

**Список терминов:**

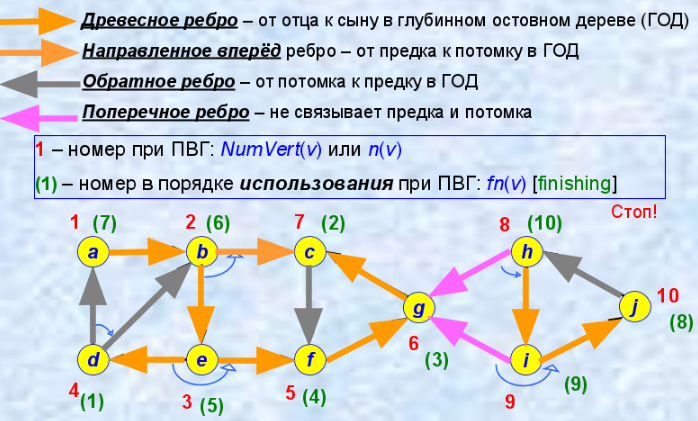
**Граф** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *graph*) — основной объект изучения математической теории графов, совокупность непустого множества вершин и наборов пар вершин (связей между вершинами).

**Ориентированный граф** (кратко**орграф**) — (мульти) граф, рёбрам которого присвоено направление.

**Направленный ациклический граф**(*ориентированный ациклический граф*, *DAG* от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *directed acyclic graph*) — орграф, в котором отсутствуют направленные циклы, то есть пути, начинающиеся и кончающиеся в одной и той же вершине. Направленный ациклический граф является обобщением дерева.

**Топологическая сортировка** — упорядочивание вершин направленного ациклического графа согласно частичному порядку, заданному ребрами орграфа на множестве его вершин.

**Виды ребер в орг. Графе**:



**Цель:**

Реализовать алгоритм Топологической сортировки на базе языка Java и выполнить топологическую сортировку для заданного направленного ациклического графа и визуализировать его работу и интерфейс, для удобства использования программы.

**Алгоритм Тарьяна:**

Изначально все вершины «белые». Производим серию обходов в глубину, пока каждая вершина не будет посещена.

* При входе в вершину делаем её «серой», при выходе — «чёрной» и одновременно заносим в окончательный список.
* Если вдруг вошли в серую вершину — найден цикл, топологическая сортировка невозможна.

**Цвет.** Во время обхода в глубину используется 3 цвета. Изначально все вершины белые. Когда вершина обнаружена, красим ее в серый цвет. Когда просмотрен список всех смежных с ней вершин, красим ее в черный цвет.

**Сложность.** O(n)

**Наш псевдокод:**

void main()

{

Graph G;

вврод графа G;

Node \* final\_list;

для каждой вершины графа G вызываем : algorithm(curr\_top, final\_list)

}

void algorithm(Node curr\_top, Node \* final\_list)

{

if (curr.color == Gray)

{

Невозможно отсортирвоать, т.к.найден цикл.

}

curr\_top.color = Gray;

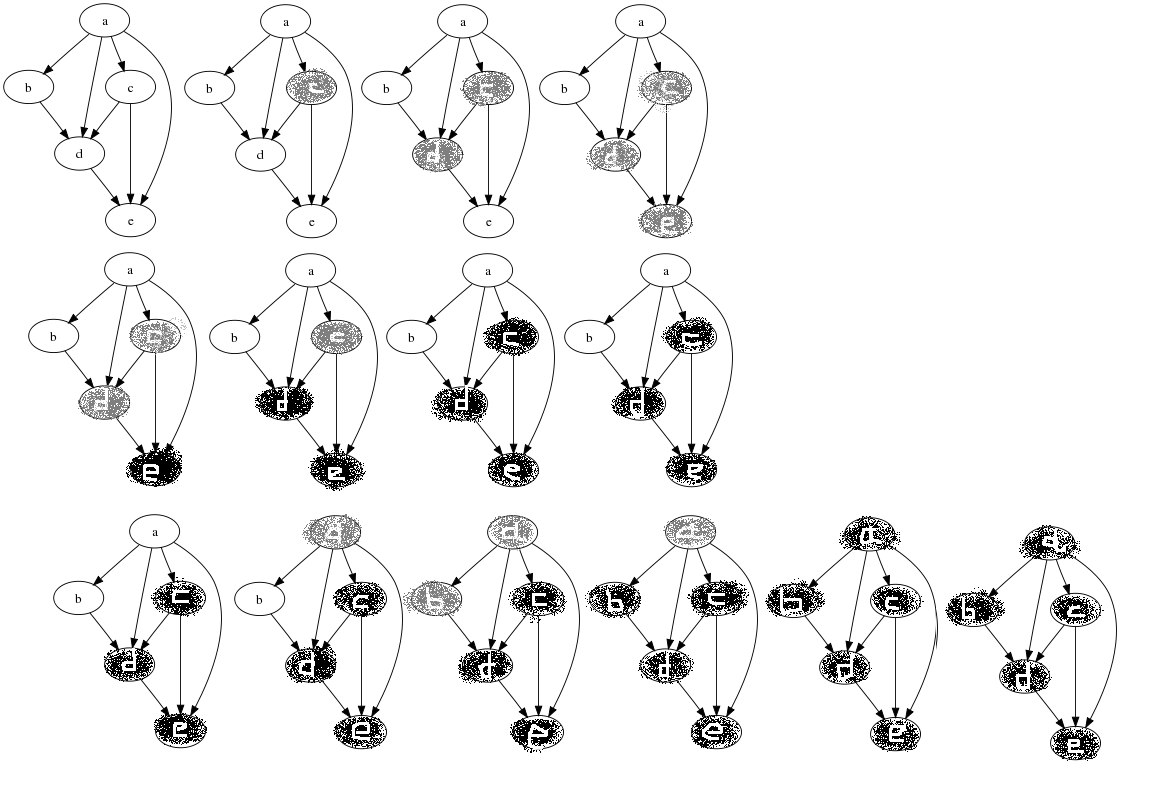
для каждого сына curr\_top вызываем функцию algorithm(текущий сын curr\_top, final\_list)

curr\_top.color = Black;

добавляем curr\_top в final\_list;

}

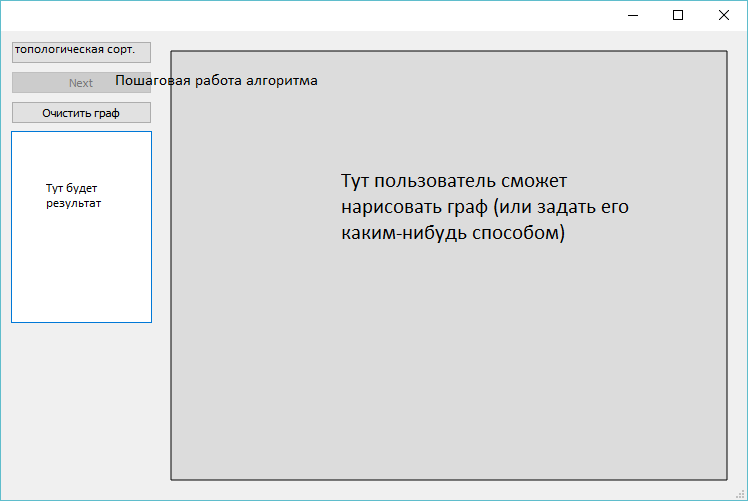
**Наш пример:**



A,b,c,d,e

**Примерный вид интерфейса:**

Поле для вывода работы алгоритма



Пользователь будет иметь возможность вводить граф графически(например: Левая кнопка – создание вершины, правая кнопка – провести ребро), и будет иметь возможность удалять и перемещать вершины. Так же вводить граф с клавиатуры вручную.

В программе будет возможность выделить вершину кликнув по ней левой кнопкой, таким образом чтобы провести ребро, нужно выделить 2 вершины и затем кликнуть правой кнопкой мыши по пустому месту( в том порядке, в которым были выбраны вершины и будет проведено ребро)

Удаление будет сделано аналогично, выделяются вершины и какой-нибудь горячей клавишей они будут удалены.

Ручной ввод с клавиатуры будет в виде:

a,b,c,d (список всех вершин)

a b (все связи в графе)

c d

Аналогичным образом можно вводить граф с файла

Результат: отсортированный список вершин

Генерация(начальная версия): Будет происходить случайно(но будет проверка на цикличность) если граф будет с циклом генерация будет происходить заново(входные параметры кол-во вершин и % ребер).

**План работы:**

**25.06.2016** Встреча группы для обсуждения алгоритмов, создания репозитория и четкого разделения труда.

**26.06.2016** Создание Прототипа(отдельно работающий графический интерфейс и алгоритм).

**28.06.2016** Попытка соединения между собой Алгоритма и граф. интерфейса. Необходимо, чтобы присутствовал хотя бы примитивный способ задания графа (с граф. выводом на экран) и вывод результата в интерфейс(без демонстрации работы алгоритма).

**30.06.2016** Графическая демонстрация работы алгоритма, с реализованным алгоритмом выделения различных видов ребер, и несколькими способами задания графа пользователем(в том числе с случайной генерацией).

**1.07.2016** Написание отчета по финальной версии проекта.

Дополнение : к 30.06.2016 доделать проверку на циклы и выдачу соответствующих сообщений

**Разделение труда:**

**Группа :** Обсуждение выбора алгоритма и тонкостей его реализации

Юлия: Разработка алгоритма на Java ( топологическая сортировка и разделение различных видов ребер)

Project manage (слежение за планом, создание репозитория)

Даниил: Создание интерфейса и ручных/неручных способов задания ребер и вершин.

Денис: Графическая визуализация различных видов ребер и алгоритма.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Работа алгоритма | Визуализация алгоритма  Гр Рез Алг | Визуализация интерфейса | Работа кнопок | Работа алгоритма различных видов ребер | Визуализация  различных видов ребер | Способ задания графов  К Г Ф |
| **Тест 1** | **+** | + + + | **+** |  | + | + | + + + |
| **Тест 2** | + | + + + | **+** |  | + | + | + + + |
| **Тест 3** | + | + + + | **+** |  | + | + | + + + |
| **Тест 4** |  |  | **+** | + |  |  |  |
| **Тест 5** |  |  | **+** | + |  |  |  |
| **Тест 6** | + | + + + | **+** |  | + | + | + + + |

**Тест кейсы:**

Конечная версия:

Гр- граф ,Рез- результат, Алг- алгоритм, К – клавиатура, Ф – файл, Г – графически

Тест1: 1->2 , 1->3 , 2->4 , 5->1 ( обычный тест)

4->1 , 4->2 , 4->3 , 5->3

2->1 , 2->3 ,4->2 , 5->2

Тест2: Тест из спецификации

1->2 , 2->3 , 2-.4 , 3->4 , 5->4

1->8 , 8->2 , 7->2 , 7->3 , 6->3 , 6->7 , 4->3 , 5->4 ( усложненные)

Тест3: 1->2 , 2->3 , 3->1 ( циклический тест)

1->2 , 3->2 , 2->4 , 4->3

1->2 , 1->4 , 1->3 , 3->3 , 3->4

Тест4: Проверка работы всех кнопок на основе теста 1.1

Тест5: Проверка всех кнопок на основе теста 3.1

Тест6: 2->1 , 3->2 , 4->2 , 4->5 , 2->5 , 5->6 , 6->3 (усложненный циклический тест)

1->5 , 2->5 , 6->2 , 3->2 , 2->7 , 6->7 , 7->3 , 7->8 , 8->4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Работа алгоритма | Визуализация алгоритма  Гр Рез Алг | Визуализация интерфейса | Работа кнопок | Способ задания графов  К Г Ф |
| **Тест 1** | **+** | + + | **+** |  | + |
| **Тест 2** | + | + + | **+** |  | + |
| **Тест 4** |  |  | **+** | + |  |

**Тест кейсы:**

Первая версия:

Тест1: 1->2 , 1->3 , 2->4 , 5->1 ( обычный тест) ( Ожидаемый результат: 5,1,3,2,4)

4->1 , 4->2 , 4->3 , 5->3 ( Ожидаемый результат: 4,3,5,2,1)

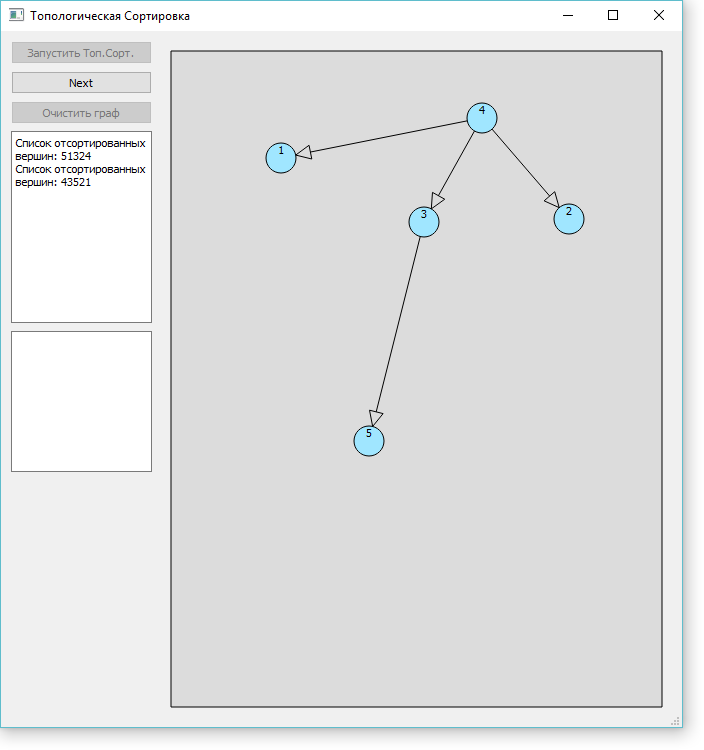
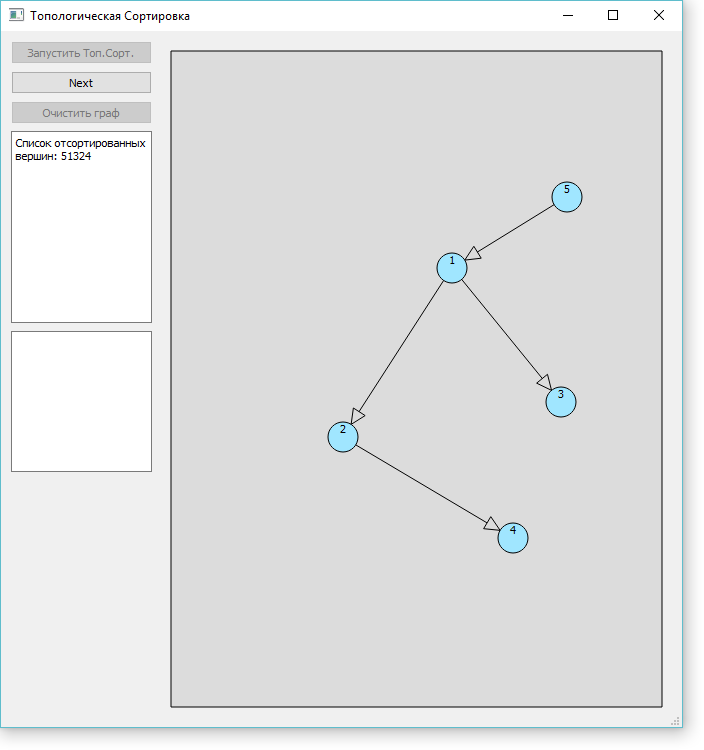
Тест2: Тест из спецификации (Ожидаемый результат 1,3,2,4,5 | 1,2,3,4,5)

1->8 , 8->2 , 7->2 , 7->3 , 6->3 , 6->7 , 4->3 , 5->4 ( усложненные) ( Ожидаемый результат: 6,7,5,4,3,1,8,2)

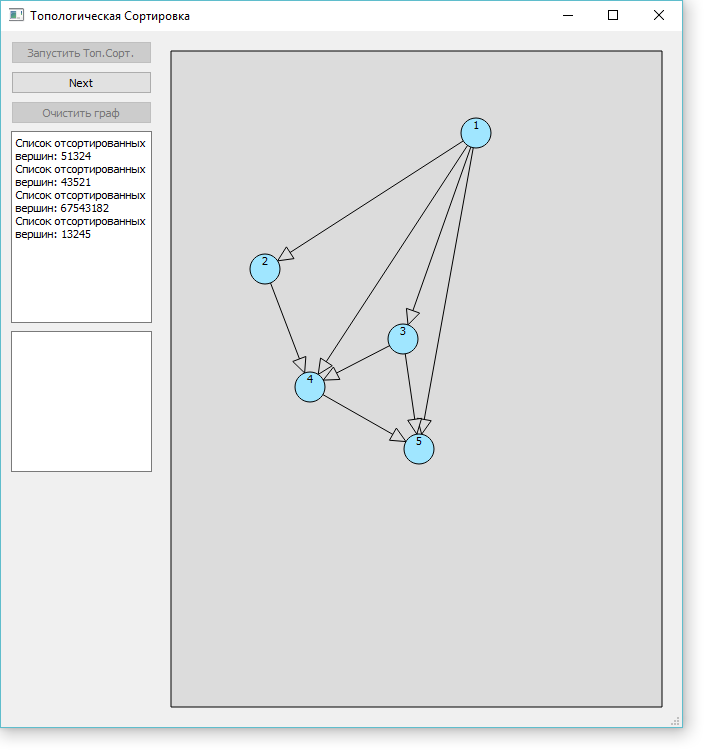
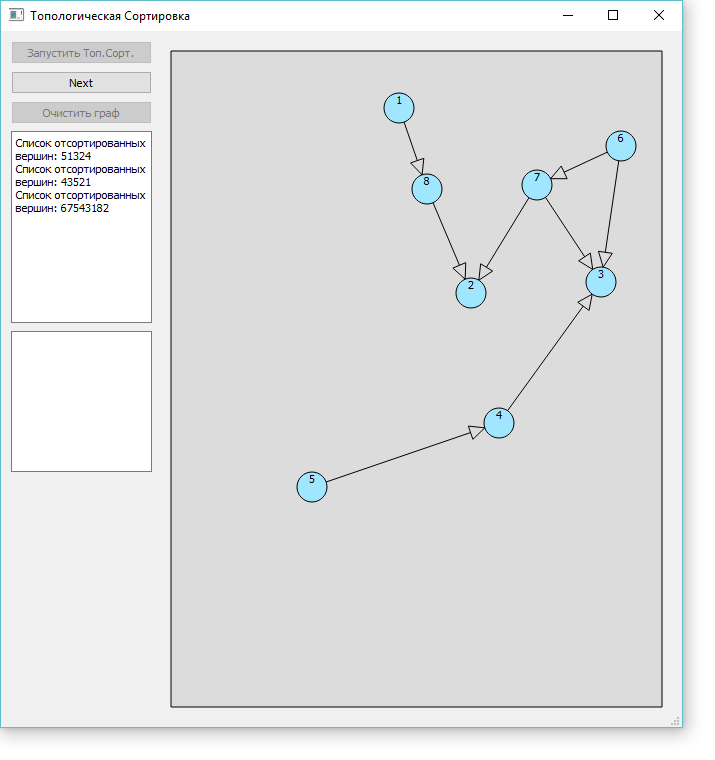
Тест4: Проверка работы всех кнопок на основе теста 1.1

Тестирование программы 1:

**Тест1:**

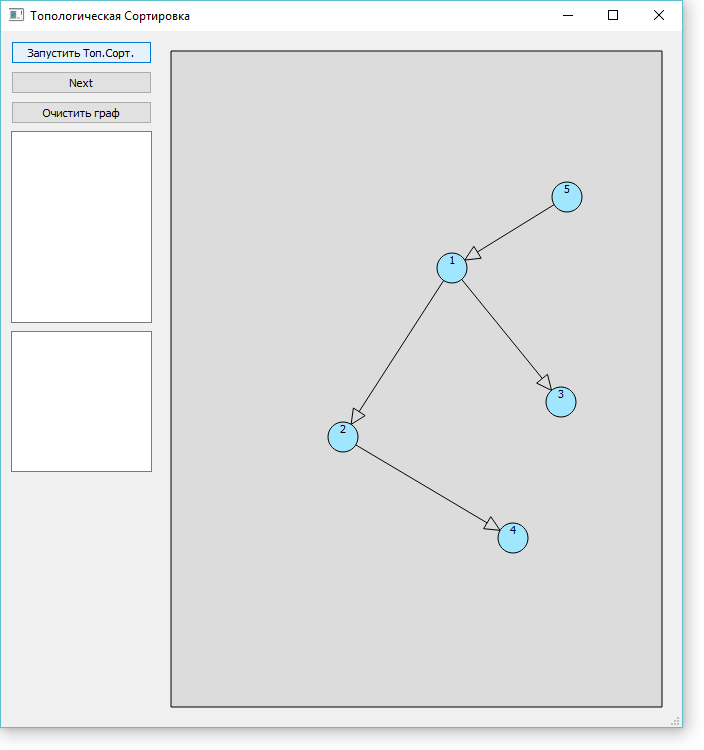


**Тест 2:**

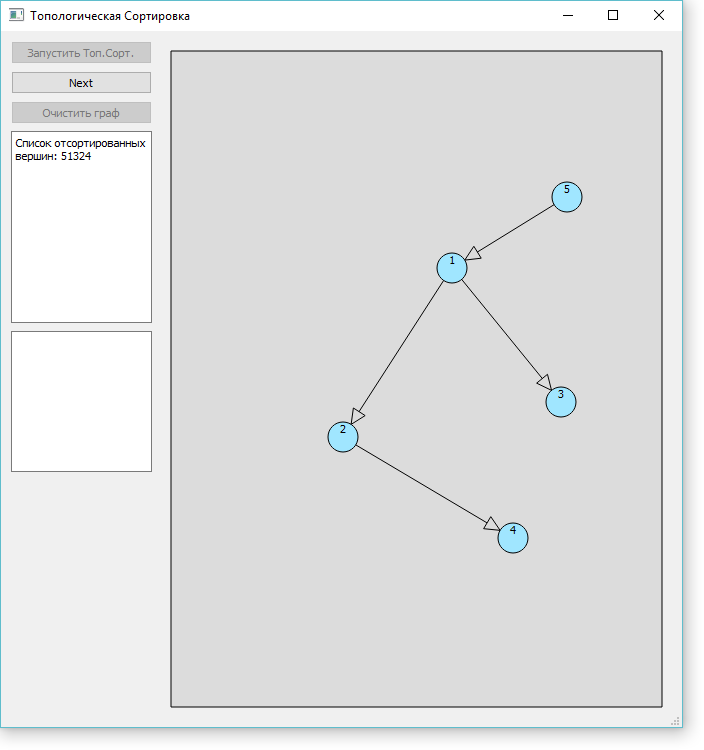


**Тест 3 :**

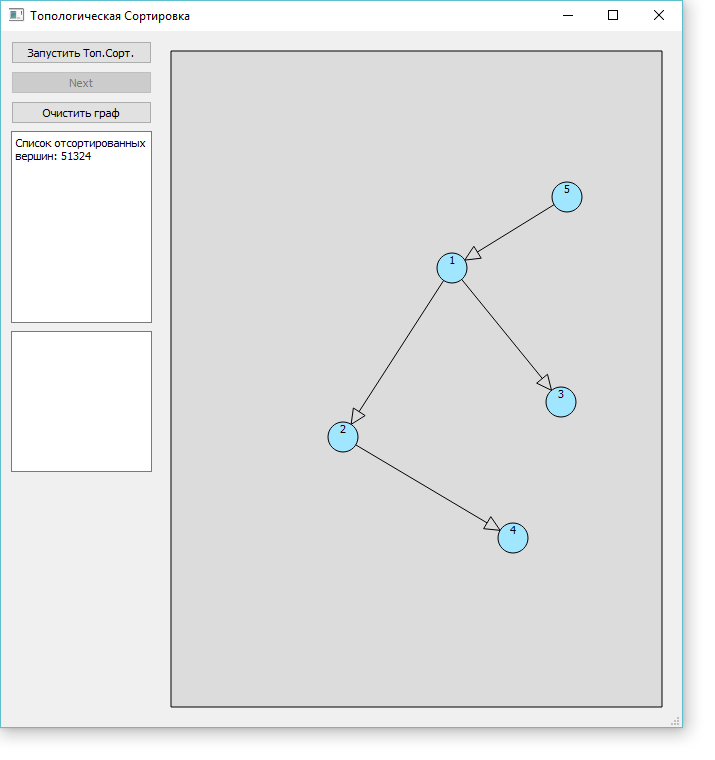
Ввели граф:



Нажатие кнопки Запустить Топ.Сорт.



Нажатие кнопки Next(завершает работу алгоритма)



Нажатие кнопки Очистить Граф

