Белорусский государственный технологический университет

Кафедра Программной инженерии

**Курс “Математическое программирование”**

**Отчет по лабораторной работе №5**

**АЛГОРИТМЫ НА ГРАФАХ**

**Вариант 13**

Выполнил: Савин Владислав Александрович

ФИТ 2 курс, 8 группа

Минск 2021

**ЦЕЛЬ РАБОТЫ:** освоить сущность и программную реализацию: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов.

|  |  |
| --- | --- |
| 13 |  |

**Задание 1.**  Ориентированный граф **G** взять в соответствии с вариантом. Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на алгоритмах.

**Ниже приведена демонстрация работы поиска в глубину.**

Текущее состояние алгоритма хранится в следующих структурах памяти:

**C** – массив окраски вершин,

**D** – время окраски вершин в серый цвет,

**P** – массив предшествующих вершин,

**F** – время окраски в чёрный цвет.

Кроме того, используется переменная t, текущее значение которой – номер шага алгоритма.

По условию, граф имеет 5 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

Шаг 1. Начинаем с нулевой вершины и красим её. Находим смежные вершины вершине 0. Это вершина 1.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | W | W | W | W |
| D | 1 | I | I | I | I |
| P | N | N | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 2: затем переходим в 1-ю из 0-й и красим её. Находим смежные вершины вершине 1. Это вершина 2.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | W | W | W |
| D | 1 | 2 | I | I | I |
| P | N | 0 | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 3: потом переходим во 2-ю из 1-й и красим её. Находим смежные вершины вершине 2. Это вершина 3.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | W | W |
| D | 1 | 2 | 3 | I | I |
| P | N | 0 | 1 | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 4: потом переходим во 3-ю из 2-й и красим её. Ищем смежные вершины вершине 3. Это вершины 0 и 4.

0

1

2

4

3

|  |  |
| --- | --- |
| t | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | W |
| D | 1 | 2 | 3 | 4 | I |
| P | N | 0 | 1 | 2 | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 5: потом переходим во 4-ю из 3-й и красим её. Находим смежные вершины вершине 4. Это вершины 0 и 4.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | G |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 6. Белых вершин, смежный с 4 вершиной, нет, следовательно, мы погрузились в глубину. Закрашиваем 4 вершину в чёрный цвет.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 6 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 6 |

Шаг 7. На предыдущем и последующих шагах массивы D и P – не изменяются; заполняется массив F и изменяется массив C.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 7 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 7 | 6 |

Шаг 8. Окрашиваем вершину 2 в чёрный цвет.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 8 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 8 | 7 | 6 |

Шаг 9. Окрашиваем вершину 1 в чёрный цвет.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 9 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 9 | 8 | 7 | 6 |

Шаг 10. Последний шаг: окрашиваем 1 вершину в чёрный цвет и проверяем, что смежный ей вершин серого цвета нет. Значит алгоритм закончил свою работу.

1

0

2

3

4

|  |  |
| --- | --- |
| t | 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | B | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 10 | 9 | 8 | 7 | 6 |

**Ответ:** DFS-дерево имеет вид:

1

0

2

3

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 3 |

**Ниже представлен результат работы поиска в ширину.**

Текущее состояние алгоритма хранится в следующих структурах памяти:

**Q** – очередь вершин,

**C** – массив окраски вершин,

**D** – массив расстояний,

**P** – массив предшествующих вершин.

По условию, граф имеет 5 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

Шаг 1. Начинаем с нулевой вершины, помещаем ее в очередь и красим её.

1

0

2

3

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 0 |  |  |  |  |
| C | G | W | W | W | W |
| D | 0 | I | I | I | I |
| P | N | N | N | N | N |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 1 |  |  |  |  |
| C | B | G | W | W | W |
| D | 0 | 1 | I | I | I |
| P | N | 0 | N | N | N |

Шаг 2. Извлекаем из очереди 0-ую вершину и закрашиваем в black (B) цвет; помещаем 1-ую вершину в очередь.

1

0

2

3

4

Шаг 3. Извлекаем из очереди 1-ую вершину и закрашиваем в black (B) цвет; помещаем 2-ую вершину в очередь.

1

0

2

3

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 2 | 3 |  |  |  |
| C | B | B | G | G | W |
| D | 0 | 1 | 2 | 2 | I |
| P | N | 0 | 1 | 1 | N |

Шаг 4. Извлекаем из очереди 2-ую вершину и закрашиваем в black (B) цвет; 3-я вершина всё ещё в очереди.

1

0

2

3

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 3 |  |  |  |  |
| C | B | B | B | G | W |
| D | 0 | 1 | 2 | 2 | I |
| P | N | 0 | 1 | 1 | N |

Шаг 5. Извлекаем из очереди 3-юю вершину и закрашиваем в black (B) цвет; помещаем 4-ую вершину в очередь.

1

0

2

3

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 4 |  |  |  |  |
| C | B | B | B | B | G |
| D | 0 | 1 | 2 | 2 | 3 |
| P | N | 0 | 1 | 1 | 3 |

Шаг 6: последний шаг алгоритма. Извлекаем последний элемент из очереди, очередь пуста, все вершины окрашиваются в В-цвет. Извлекаем последнюю 4-ую вершину, проверяем на смежность с другими вершинами.

1

0

2

3

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 5 |  |  |  |  |
| C | B | B | B | B | B |
| D | 0 | 1 | 2 | 2 | 3 |
| P | N | 0 | 1 | 1 | 3 |

Далее если мы попытаемся зайти в какую-либо вершину, то она уже будет закрашена и нас выкинет.

**Ответ:** Все вершины чёрного цвета, соответственно, алгоритм закончил свою работу. В результате получили вот такое BFS-дерево:

1

0

2

3

4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 3 |

**Алгоритм топологической сортировки**

Исходный граф:

1

0

2

3

4

По условию, граф имеет 5 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

Топологическая сортировка – это процедура упорядочивания вершин ориентированного графа, не имеющего циклов.

При реализации топологической сортировки с помощью алгоритма поиска в глубину используется массив меток вершин, с помощью которого моделируется удаление вершин из графа и сохраняются новые номера вершин.

Шаг 1. На первом шаге окрашиваем 0-ую вершину в серый цвет.

1/10

1/10

2/9

1

0

3

2

3

3/8

4

4/7

5/6

Формируем очередь по мере окрашивания вершин в чёрный цвет: 0 1 2 3 4

0

0

1

3

2

4

**Задание 2.** Осуществить программную реализацию алгоритмов на C++. Разработать структуры **AMatrix** и **АList**  для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию **BFS** обхода вершин графа, используя метод поиска в ширину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

|  |
| --- |
| public static void AMatrix()  {  int[,] matrix = new int[5, 5] { { 0, 1, 0, 0, 0 }, { 0, 0, 1, 1, 0 }, { 0, 0, 0, 1, 0 }, { 1, 0, 0, 0, 1 }, { 0, 0, 0, 0, 0 } };  for (int i = 0; i < 5; i++)  {  for (int j = 0; j < 5; j++)  {  Console.Write(matrix[i, j]);  }  Console.WriteLine();  }  }  public static void AList()  {  List<int> list = new List<int>();  int[] one = { 0, 1, 0, 0, 0 };  int[] two = { 0, 0, 1, 1, 0 };  int[] three = { 0, 0, 0, 1, 0 };  int[] four = { 1, 0, 0, 0, 1 };  int[] five = { 0, 0, 0, 0, 0 };  list.AddRange(one);  list.AddRange(two);  list.AddRange(three);  list.AddRange(four);  list.AddRange(five);  int k = 1;  foreach (int row in list)  {  Console.Write(row);  if (k % 5 == 0)  {  Console.WriteLine();  }  k++;  }  }  public static void BFS(int[,] matrix)  {  Queue<int> q = new Queue<int>(5);  int[] c = new int[5] { 0, 0, 0, 0, 0 };  int[] d = new int[5] { -1, -1, -1, -1, -1 };  int[] p = new int[5] { -1, -1, -1, -1, -1 };  for (int i = 0; i < 5; i++)  {  for (int j = 0; j < 5; j++)  {  if (matrix[i, j] == 1)  {  q.Enqueue(i);  Console.WriteLine(q.Dequeue());  }  }  }  } |

**Задание 3.**  Разработать функцию **DFS** обхода вершин графа, используя метод поиска глубину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

**Задание 4.** Доработайте функцию **DFS**, **для** выполнения топологической сортировки графа. Продемонстрировать работу функции. Копии экрана вставить в отчет.

Топологические сортировки для матрицы и списка смежностей реализованы как методы классов и их код был представлен выше. Ниже на рисунке 1 представлен результат работы:

