Глава 1

Элементы теории графов

Первая работа о теории графов была опубликована в 1736 году в Санкт-Петербурге Леонардом Эйлером. Эта была знаменитая задача о кенигсбергских мостах. С тех пор графы широко исследовались как теоретически, так и практически. В данном разделе будут рассмотрены только азы теории графов.

1.1 Основные понятия

В различных источниках встречаются разные определения графов.

Формально, можно сказать, что граф — это совокупность множества X, элементы которого называются *вершинами* и множества упорядоченных пар вершин, элементы которого называются *дугами*. Граф обычно обозначается (X, A).

В некоторых задачах имеет значение какая из вершин дуги является начальной, а какая конечной, а в некоторых нет. В первом случае граф называется *ориентированным* или *орграфом*. Во втором случае — *неориентированным*. Дуги в этом случае часто называются *ребрами*.

В качестве примера ориентированного и неориентированного графов можно привести поиск правильного маршрута от точки A до точки B в городе, например, в центре Саратова с его односторонним движением на улицах. Пешеходу не важно разрешенное направление движения — маршрут будет проложен по неориентированному графу, т. е., если существует путь от точки A до точки B, то существует путь от точки B до точки A. Для водителя необходимо учитывать разрешенное направление движения и из того, что существует путь от точки A до точки B, необязательно следует, что существует путь от точки B до точки A.

Довольно часто каждому ребру графа ставят в соответствие какую-нибудь метку, например расстояние между двумя точками. Такую метку называют *весом*, а граф — *взвешенным* или помеченным.

Графически ребра ориентированного графа представляются со стрелкой, показывающей направление от начальной точки к конечной. На рисунке слева представлен неориентированный граф, справа — ориентированный. Изображение графа помогает понять структуру, но только для графа с малым числом

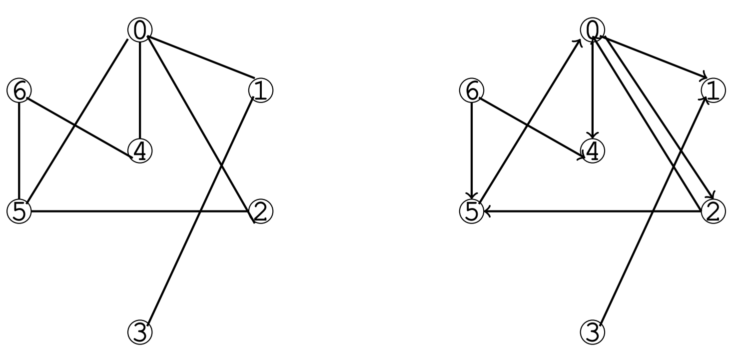


Рис. 1.1: Пример неориентированного и ориентированного графов

вершин и ребер. Однако не всегда по чертежу можно понять, что изображен один и тот же граф.

Например, на рисунке представлены два графа. На первый взгляд они разные, но, если выписать все ребра для обоих графов, становится понятно, что это один и тот же граф, просто по-разному представленный.

Действительно, список ребер: $0-1; 0-2; 0-5; 0-6; 3-4; 3-5; 4-5; 4-6; 7-8; 9-10; 9-11; 9-12; 11-12$. В примерах выше и дальше, будут рассматриваться *простые графы*. Простой граф — это граф, не

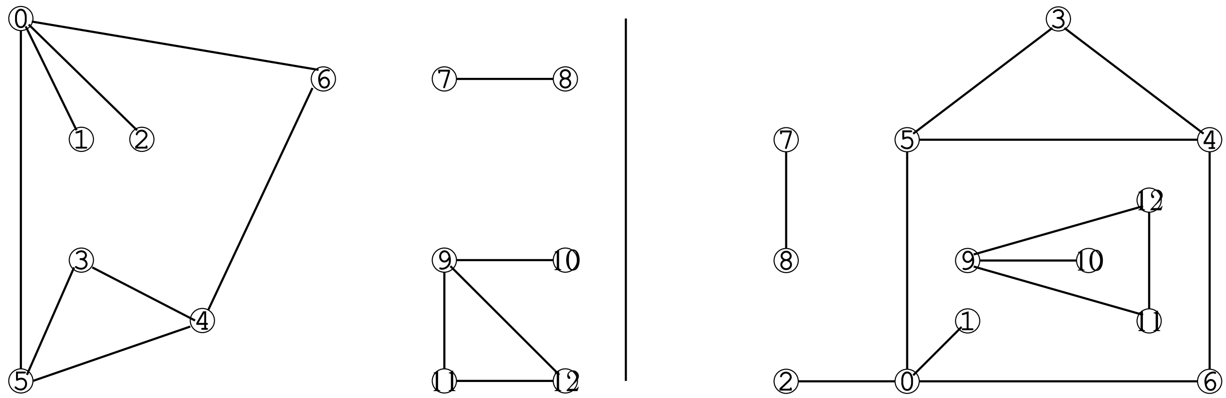


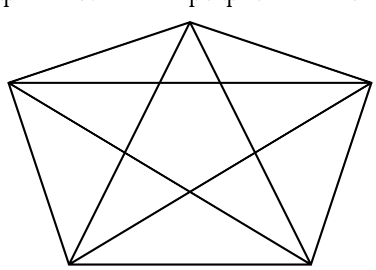
Рис. 1.2: Визуальное представление графа

имеющий кратных ребер (две вершины могут быть соединены только одним ребром) и петель (ребро, начинающееся и заканчивающиеся в одной вершине).

Граф является *полным*, если он содержит все возможные ребра. Таких ребер может быть не более $N(N-1)/2$, где N — количество вершин графа. Два графа называются *изоморфными*, если можно поменять метки вершин одного графа таким образом, чтобы набор ребер в итоге для обоих графов стал идентичным.

Если имеется ребро, соединяющее две вершины графа, то такие вершины называются *смежными*. А ребро — *инцидентным* этим вершинам. *Степень вершины* неориентированного графа — это количество инцидентных ей ребер. Для ориентированного графа можно говорить о *полустепени исхода* и *полустепени захода*. Это, соответственно, число ребер исходящих и заходящих в данную вершину.

Если все вершины имеют одинаковую степень, то говорят о *регулярных* графах. Например, K_5 граф — полный граф с пятью вершинами, имеющий степень вершин 5.



Подграф — множество ребер и вершин, которые сами представляют из себя граф.

Граф называется *планарным*, если на чертеже его ребра не пересекаются.

Путь в графе — это последовательность вершин $a_0, a_1, a_2, \ldots, a_n$, таких что для любых $i > 0$ вершина a_i смежна с вершиной a_{i-1}. *Простой* путь — путь, все вершины и ребра которого различны.

Если существует простой путь, начинающийся и заканчивающийся в одной вершине, такой путь называется *циклом*. Например, на рисунке 1.2 циклами являются: $0-6-4-5-0; 0-6-4-3-5-0; 3-4-5-3; 9-11-12-9$. Цикл должен содержать как минимум 3 различных вершины и три различных ребра. Длина пути — количество ребер, составляющих путь (или количество вершин минус единица) для невзвешенного графа и сумма весов соответствующих ребер для взвешенного графа.

Неориентированный граф называется *связным*, если из любой вершины графа существует путь в любую другую вершину.

Несвязный граф состоит из некоторого множества *связных компонент*, представляющих собой максимальные связные подграфы.

Например, представленный на рисунке 1.2 граф несвязный, содержит три связные компоненты. Первая состоит из вершин $0, 1, 2, 3, 4, 5, 6$; вторая — из вершин $7, 8$ и третья — из вершин $9, 10, 11, 12$.

В большинстве задач граф содержит малое количество из возможных ребер. Введем понятие *насыщенность* — среднее значение степени вершин, $\frac{2E}{N}$, где E — количество ребер, N — количество вершин.

Граф является *насыщенным* (плотным), если количество его ребер пропорционально N^2. и *разреженным* в противоположном случае.

В зависимости от насыщенности определяется какой из алгоритмов необходимо использовать. Пусть есть два алгоритма, решающих одну задачу. Один имеет сложность $O(N^2)$, другой — $O(E \log E)$.

В случае плотного графа с $N = 1000$ и $E = 10^6$ $O(N^2) = 10^6$, $O(E \log E) \approx 2 \times 10^7$. Очевидно, что первый алгоритм выгоднее.

В случае разреженного графа с $N = 10^6$ и $E = 10^6$ $O(N^2) = 10^{12}$, $O(E \log E) \approx 2 \times 10^7$. Очевидно, что выгоднее использовать второй алгоритм.

1.2 Представление графов

Существует несколько способов представления графов. Все зависит от постановки задачи.

Рассмотрим основные способы представления графов — матрицу смежности, список смежности.

Матрица смежности — это матрица $N \times N$, удовлетворяющая следующему свойству:

$$\begin{cases} a[i][j] = 1, \text{если вершины } i \text{ и } j \text{ смежные} \\ a[i][j] = 0, \text{если вершины } i \text{ и } j \text{ несмежные} \end{cases}$$

Для случая взвешенного графа в случае смежных вершин в ячейку матрицы ставится вес ребра.

Например, для графа, представленного на рисунке 1.2 матрица смежности имеет следующий вид:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	0	0	1	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	0	0	0	0	0	0	0
4	0	0	0	1	0	1	1	0	0	0	0	0	0
5	1	0	0	1	1	0	0	0	0	0	0	0	0
6	1	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1
10	0	0	0	0	0	0	0	0	0	1	0	0	0
11	0	0	0	0	0	0	0	0	0	1	0	0	1
12	0	0	0	0	0	0	0	0	0	1	0	1	0

Для простого неориентированного связного графа должны выполняться два условия:

1. Диагональ должна содержать нули (петель нет).

2. Матрица должна быть симметрична относительно главной диагонали.

Для ориентированного графа матрица не является симметричной.

В нашем случае граф несвязный, но можно увидеть, что для максимально связных подграфов (0 – 1 – 2 – 3 – 4 – 5 – 6) и (9 – 10 – 11 – 12) подматрицы симметричны.

Главный недостаток матрицы смежности — занимает $O(N^2)$ памяти и содержит слишком много нулей.

Преимущества использования матрицы смежности — при вставке, поиске и удалении ребра $i--j$. В этих случаях надо проверить чему равно $a[i][j]$. Для вставки ребра надо присвоить $a[i][j] = a[j][i] = 1$, для удаления — $a[i][j] = a[j][i] = 0$.

Для большинства остальных алгоритмов лучше воспользоваться списком смежности — каждой вершине ставится в соответствие набор смежных вершин.

Например, для графа, представленного на рисунке 1.2 матрица смежности имеет следующий вид:

$$\begin{array}{r|llllll}
0 & \to & 1\to & 2\to & 5\to & 6\to & \oslash \\
1 & \to & 0\to & \oslash & & & \\
2 & \to & 0\to & \oslash & & & \\
3 & \to & 4\to & 5\to & \oslash & & \\
4 & \to & 3\to & 5\to & 6\to & \oslash & \\
5 & \to & 0\to & 3\to & 4\to & \oslash & \\
6 & \to & 0\to & 4\to & \oslash & & \\
7 & \to & 8\to & \oslash & & & \\
8 & \to & 7\to & \oslash & & & \\
9 & \to & 10\to & 11\to & 12\to & \oslash & \\
10 & \to & 9\to & \oslash & & & \\
11 & \to & 9\to & 12\to & \oslash & & \\
12 & \to & 9\to & 11\to & \oslash & &
\end{array}$$

По памяти список смежности занимает $O(N + E)$.

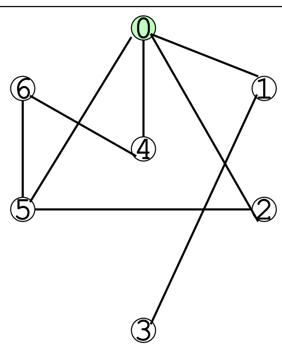
Так как размер каждого списка соответствует степени вершины, следовательно, поиск, вставка и удаление ребра занимает примерно $O(2E/V)$ (среднее значение степени вершины).

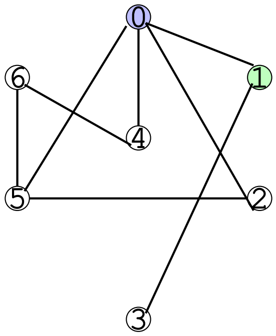
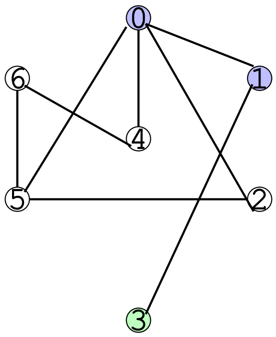
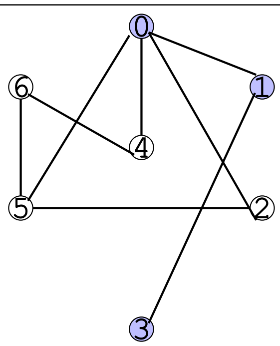
1.3 Обход графа в глубину

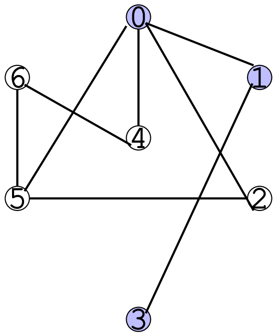
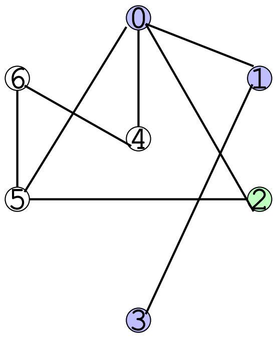
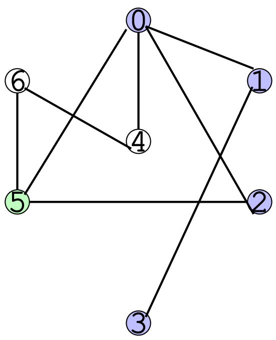
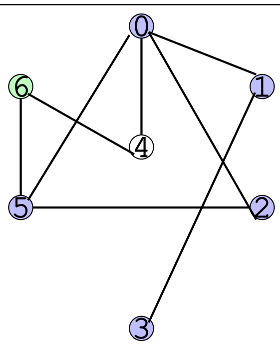
Алгоритм обхода в глубину: посетив вершину, помечаем ее меткой, что она посещена. Ищем смежную непосещенную вершину и рекурсивно вызываем обход в глубину для этой вершины. Повторяем процесс до тех пор, пока все вершины не будут посещены.

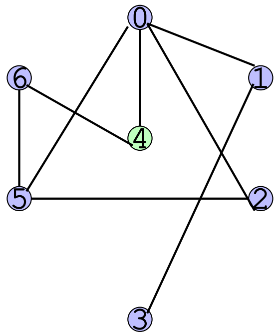
Название в глубину происходит из-за того, что идем «вглубь» графа до тех пор, пока это возможно.

Например, для заданного графа начинаем с нулевой вершины. Зеленым цветом будет отмечена текущая вершина, синим — уже посещенные.

Граф	Результат	Действие с текущей вершиной	Массив посещенных вершин	Поиск смежной вершины
	0	Помечаем 0 как посещенную	$[1, 0, 0, 0, 0, 0, 0]$	Смежная вершина — 1

Граф	Результат	Действие с текущей вершиной	Массив посещенных вершин	Поиск смежной вершины
	0, 1	Помечаем 1 как посещенную	$[1, 1, 0, 0, 0, 0, 0]$	Смежная вершина — 3
	0, 1, 3	Помечаем 3 как посещенную	$[1, 1, 0, 1, 0, 0, 0]$	Смежных непосещенных вершин нет. Возвращаемся на шаг назад. Ищем смежную вершину для 1.
	0, 1, 3		$[1, 1, 0, 1, 0, 0, 0]$	Смежных непосещенных вершин нет. Возвращаемся на шаг назад. Ищем смежную вершину для 0.

Граф	Результат	Действие с текущей вершиной	Массив посещенных вершин	Поиск смежной вершины
	0, 1, 3		$[1, 1, 0, 1, 0, 0, 0]$	Смежная вершина — 2
	0, 1, 3, 2	Помечаем 2 как посещенную	$[1, 1, 1, 1, 0, 0, 0]$	Смежная вершина — 5
	0, 1, 3, 2, 5	Помечаем 5 как посещенную	$[1, 1, 1, 1, 0, 1, 0]$	Смежная вершина — 6
	0, 1, 3, 2, 5, 6	Помечаем 6 как посещенную	$[1, 1, 1, 1, 0, 1, 1]$	Смежная вершина — 4

Граф	Результат	Действие с текущей вершиной	Массив посещенных вершин	Поиск смежной вершины
	0, 1, 3, 2, 5, 6, 4	Помечаем 4 как посещенную	$[1, 1, 1, 1, 1, 1, 1]$	Все вершины посещены. Завершаем алгоритм

Алгоритм 1: Обход в глубину. Рекурсивный случай.

Вход: Граф, представленный списком смежности Gr, N— число вершин графа, A— массив посещенных вершин, x— вершина, с которой начинаем обход

Выход: Список последовательно посещенных вершин

начало алгоритма

· присваиваем $A[x] = 1$ (помечаем вершину x как посещенную);

· выводим x на экран;

цикл для $i = 0$ *до* $Gr[x].size()$ **выполнять**

если *существует непосещенная вершина, смежная вершине стека* ($A[Gr[x][i]] == 0$) **то**

· вызываем эту функцию для $x = Gr[x][i]$;

конец алгоритма

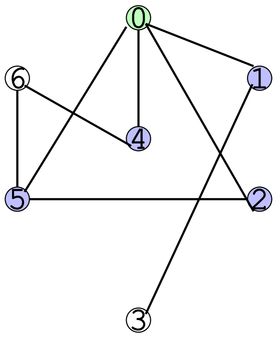
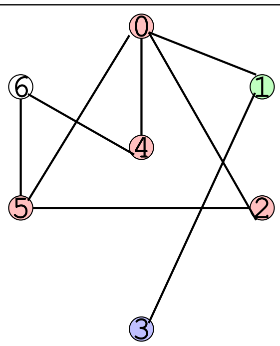
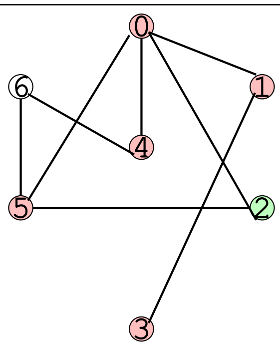
Текущая вершина	$Gr[x]$	Смежная вершина	Массив A	fl	Стек	Результат
0			[1 0 0 0 0 0 0]	`true`	0	0
0	[1, 2, 4, 5]	1	[1 1 0 0 0 0 0]	`true`	[1, 0]	0, 1
1	[0, 3]	3	[1 1 0 1 0 0 0]	`true`	[3, 1, 0]	0, 1, 3
3	[1]		[1 1 0 1 0 0 0]	`false`	[1, 0]	0, 1, 3
1	[0, 3]		[1 1 0 1 0 0 0]	`false`	[0]	0, 1, 3
0	[1, 2, 4, 5]	2	[1 1 1 1 0 0 0]	`true`	[2, 0]	0, 1, 3, 2
2	[0, 5]	5	[1 1 1 1 0 1 0]	`true`	[5, 2, 0]	0, 1, 3, 2, 5
5	[0, 2, 6]	6	[1 1 1 1 0 1 1]	`true`	[6, 5, 2, 0]	0, 1, 3, 2, 5, 6
6	[4, 5]	4	[1 1 1 1 1 1 1]	`true`	[4, 6, 5, 2, 0]	0, 1, 3, 2, 5, 6, 4
4	[0, 6]		[1 1 1 1 1 1 1]	`false`	[6, 5, 2, 0]	0, 1, 3, 2, 5, 6, 4
6	[4, 5]		[1 1 1 1 1 1 1]	`false`	[5, 2, 0]	0, 1, 3, 2, 5, 6, 4
5	[0, 2, 6]		[1 1 1 1 1 1 1]	`false`	[2, 0]	0, 1, 3, 2, 5, 6, 4
2	[0, 5]		[1 1 1 1 1 1 1]	`false`	[0]	0, 1, 3, 2, 5, 6, 4
0	[1, 2, 4, 5]		[1 1 1 1 1 1 1]	`false`	$\oslash$	0, 1, 3, 2, 5, 6, 4

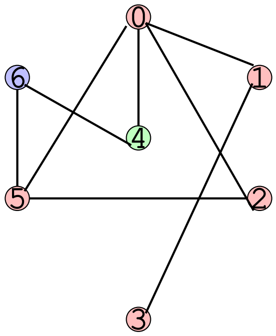
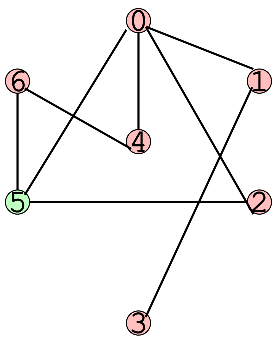
В таблице под стеком понимается стек вызовов рекурсивных функций.

1.4 Обход графа в ширину

Другой способ обхода графа — это обход в ширину. Для работы алгоритма используем очередь. Начинаем рассматривать с некоторой вершины, записываем в очередь все смежные с ней непосещенные вершины, отмечаем их как посещенные. Извлекаем из очереди голову и повторяем процесс поиска смежных вершин до тех пор, пока очередь не пуста.

Рассмотрим тот же пример, что и в предыдущем разделе. Начинаем с вершины 0. Также создаем массив посещенных вершин. Зеленым цветом показана текущая вершина, синим — смежные, красным — уже посещенные (кроме текущей).

Граф	Результат	Действие с текущей вершиной	Массив посещенных вершин	Поиск смежных вершин
	0	Помечаем 0 как посещенную	$[1,0,0,0,0,0,0]$	Смежные вершины — 1, 2, 4, 5 Помечаем их как посещенные и записываем в очередь: $1,2,4,5$.
	0, 1, 2, 4, 5	Текущая вершина — 1. Извлекаем ее из очереди.	$[1,1,1,0,1,1,0]$	Смежная вершина — 3. Помечаем ее как посещенную и записываем в очередь: $2,4,5,3$.
	0, 1, 2, 4, 5, 3	Текущая вершина — 2. Извлекаем ее из очереди.	$[1,1,1,1,1,1,0]$	Смежных непосещенных вершин нет. Очередь: $4,5,3$.

Граф	Результат	Действие с текущей вершиной	Массив посещенных вершин	Поиск смежных вершин
	0, 1, 2, 4, 5, 3	Текущая вершина — 4. Извлекаем ее из очереди.	$[1,1,1,1,1,1,1]$	Смежная вершина — 6. Помечаем ее как посещенную и записываем в очередь: $5,3,6$.
	0, 1, 2, 4, 5, 3, 6	Текущая вершина — 5. Извлекаем ее из очереди.	$[1,1,1,1,1,1,1]$	Все вершины обошли, дальше последовательно извлекаем вершины из очереди $3,6$.

Текущая вершина	$Gr[x]$	Массив A	Очередь	Результат
0	[1, 2, 4, 5]	[1 1 1 0 1 1 0]	$[1,2,4,5]$	$0,\ 1,2,4,5$
1	[0, 3]	[1 1 1 1 1 1 0]	$[2,4,5,2]$	$0,1,2,4,5,3$
2	$[0,5]$	[1 1 1 1 1 1 0]	$[4,5,3]$	$0,1,2,4,5,3$
4	[0, 6]	[1 1 1 1 1 1 1]	$[5,3,6]$	$0,1,2,4,5,3,6$
5	$[0,2,6]$	[1 1 1 1 1 1 1]	$[3,6]$	$0,1,2,4,5,3,6$
3	[1]	[1 1 1 1 1 1 1]	[6]	$0,1,2,4,5,3,6$
6	$[0,4,5]$	[1 1 1 1 1 1 1]	$[\oslash]$	0, 1, 3, 2, 5, 6

Алгоритм 2: Обход в ширину.

Вход: Граф, представленный списком смежности Gr, N— число вершин графа, x— вершина, с которой начинаем обход

Выход: Список последовательно посещенных вершин

начало алгоритма

- · создаем очередь и инициализируем ее;
- · создаем массив A размерности N и заполняем его нулями;
- · присваиваем $A[x] = 1$ (помечаем вершину x как посещенную);
- · помещаем вершину x в очередь;
- · выводим x на экран;
- **цикл пока** *очередь не пуста* **выполнять**
  - · извлекаем голову очереди (x);
  - **цикл для** $i = 0$ *до* $Gr[x].size()$ **выполнять**
    - **если** *существует непосещенная вершина, смежная* x *(*$A[Gr[x][i]] == 0$*)* **то**
      - · $y = Gr[x][i]$;
      - · помечаем y как посещенную вершину;
      - · помещаем ее в очередь;
      - · выводим на экран;
- **если** *остались непосещенные вершины* **то**
  - · вызываем рассмотренный алгоритм для непосещенной вершины;

конец алгоритма