Задание №8

Нахождение компонент сильной связности графа

Теоретические сведения. Описание алгоритма.

Дан ориентированный граф G, множество вершин которого V и множество рёбер E. Петли и кратные рёбра допускаются. Обозначим через n количество вершин графа, через m —- количество рёбер.

Две вершины (u и v) ориентированного графа называют сильно связными, если существует путь из u в v и существует путь из v в u.

Ориентированный граф называется сильно связным, если любые две его вершины сильно связны.

Отношение сильной связности – это отношение эквивалентности.

1) Рефлексивность: $\forall v\ v \to v$

2) Симметричность: $\forall v\ \forall u\ v \to u \Rightarrow u \to v$

3) Транзитивность: $\forall v\ \forall u\ \forall q\ v \to u \land u \to q \Rightarrow v \to q$

Компонентой сильной связности называется класс эквивалентности множества вершин ориентированного графа относительно отношения сильной связности. Другими словами компонента сильной связности является сильно связным подграфом. Так как сильная связность —- это отношение эквивалентности, то граф разбивается на сильно связные компоненты. Наша задача найти все такие классы эквивалентности.

Матрица сильной связности $\overline{S} = \|\overline{s}_{ij}\ \|$ орграфа – квадратная матрица порядка n с элементами

$$\overline{s}_{ij} = \begin{cases} 1, \text{если существует путь из } v_i \text{ в } v_j \text{ и из } v_j \text{ в } v_i \\ 0 \text{в противном случае} \end{cases}$$

Алгоритм нахождения компонент сильной связности орграфа по матрице сильной связности

1. В матрице $\overline{S}$ обнуляем столбцы (можно строки), у которых в первой строке стоят единицы. Получаем матрицу $\overline{S}_1$. Соответствующие единицам первой строки номера вершины принадлежат первой компоненте связности, k = 1.

2. Если $\overline{S}_1 \not\equiv (0)$, то k = k + 1. Ноходим не нулевую строку $\overline{S}_1$. Пусть ее номер i_1. Соответствующие единицам i_1 строки номера вершин принадлежат второй компоненте связности. В матрице $\overline{S}_1$ обнуляем столбцы (строки), у которых в строке i_1 стоят единицы. Получаем матрицу $\overline{S}_2$.

Процесс оканчивается, когда $\overline{S}_t = (0)$. В этом случае все вершины графа будут принадлежать какой-нибудь компоненте связности.

Этот алгоритм несколько уступает тому, что был использован в моём случае.

Алгоритм Косарайю

Инвертированием ориентированного графа назовем процедуру, в ходе которой поменяем направление каждого ребра на противоположное.

Пусть дан ориентированный граф $G = (V, E)$. Через $G' = (V, E')$ обозначим инвертирование G.

Идея этого алгоритма в том, что компоненты сильной связности есть циклы, то есть они совпадают и у исходного графа и у его инвертирования.

Алгоритм поиска в глубину

1. Двигаемся из начальной вершины.
2. Движемся в произвольную смежную вершину.
3. Из этой вершины обходим все возможные пути до смежных вершин.
4. Если таких путей нет или мы не достигли конечной вершины, то возвращаемся назад к вершине с несколькими исходящими ребрами и идем по другому пути.
5. Алгоритм повторяется, пока не будут исследованы все вершины и достигнута конечная вершина.

Алгоритм нахождения компонент связности

1. Выполняем серию обходов в глубину, посещающую весь граф. Для этого мы проходимся по всем вершинам графа и из каждой ещё не посещённой вершины вызываем обход в глубину. При этом для каждой вершины v запомним время выхода. Под временем понимаются логические часы: изначально время равно 0, при переходе в вершину или выходе из неё время увеличивается на 1.

2. Когда обход закончится, запоминаем все вершины в порядке увеличения времени выхода.

3. Теперь запустим обход в глубинку на инвертированном графе G'. Каждый раз для обхода будем выбирать ещё не посещенную вершину с максимальным индексом в списке, составленном на предыдущем шаге. Все вершины, посещенные в ходе одной итерации, образуют компоненту сильной связности.

Оценка сложности алгоритма

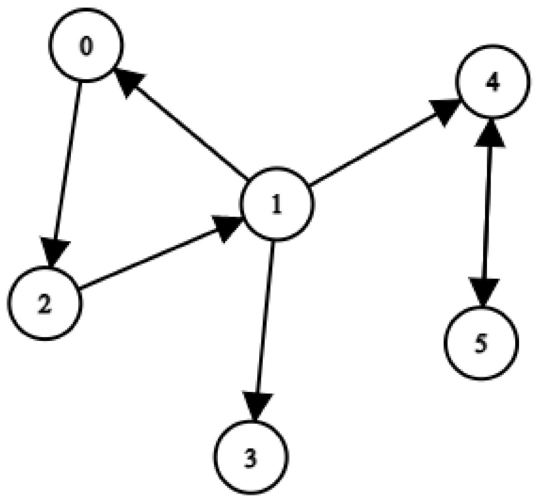
Если оценивать сложность алгоритма нахождения компонент сильной связности через матрицу сильной связности $\overline{S}$, то наибольшую сложность представляет нахождение матрицы односторонней связности графа. Так как при перемножении матриц необходимо использоавть три вложенных цикла, то сложность составляет $O(n^3)$. По этой причине мною был выбран алгоритм Косарайю, сложность которого составляет всего $O(n^2)$ в случае насыщенных графов и $O(n+m)$ в случае разреженных графов, где m – количество дуг. Этот метод состоит из двух процедур поиска в глубину, подвергнутых незначительным изменениям, в результате время его выполнения гораздо меньше.

Логическая блок-схема алгоритма

Программа написана на языке программирования C++ с применением библиотеки wxWindgets. Программа проходит несколько этапов компиляции, среди которых препроцессинг, ассемблирование, компилирование и линковка. Этот язык является кроссплатформенным, поэтому программа будет работать как на оперционыых системах семейства Windows, так и на ОС UNIX.

Тестовые примеры

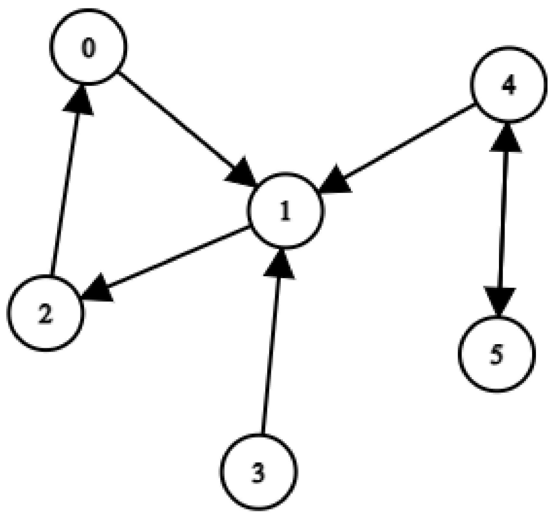
Пример 1.



Тогда при первом обходе массив с временами выхода будет иметь следующий вид: out = [2, 8, 9, 7, 6, 5].

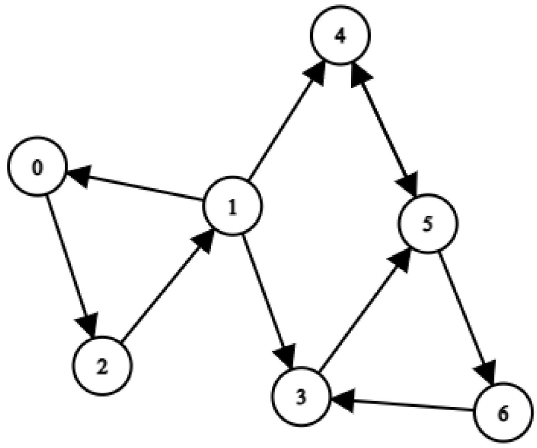
После обхода отсортированный массив с вершинами имеет вид order = [0, 5, 4, 3, 1, 2].

Инвертированный граф



Запускаем второй обход. Полученные компоненты связности: {2, 0, 1}, {3}, {4, 5}.

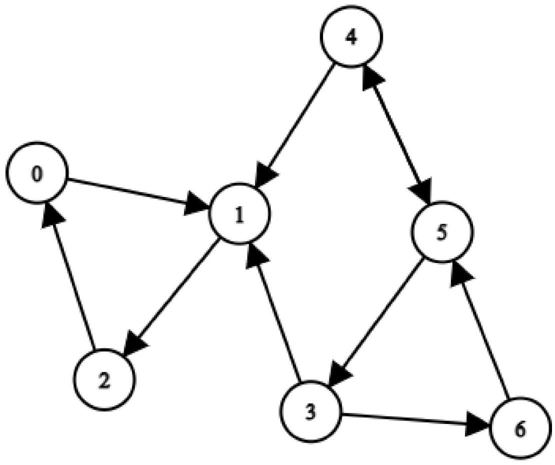
Пример 2. Добавился узел и некоторые дуги.



Тогда при первом обходе массив с временами выхода будет иметь следующий вид: out = [2, 10, 11, 7, 9, 8, 6].

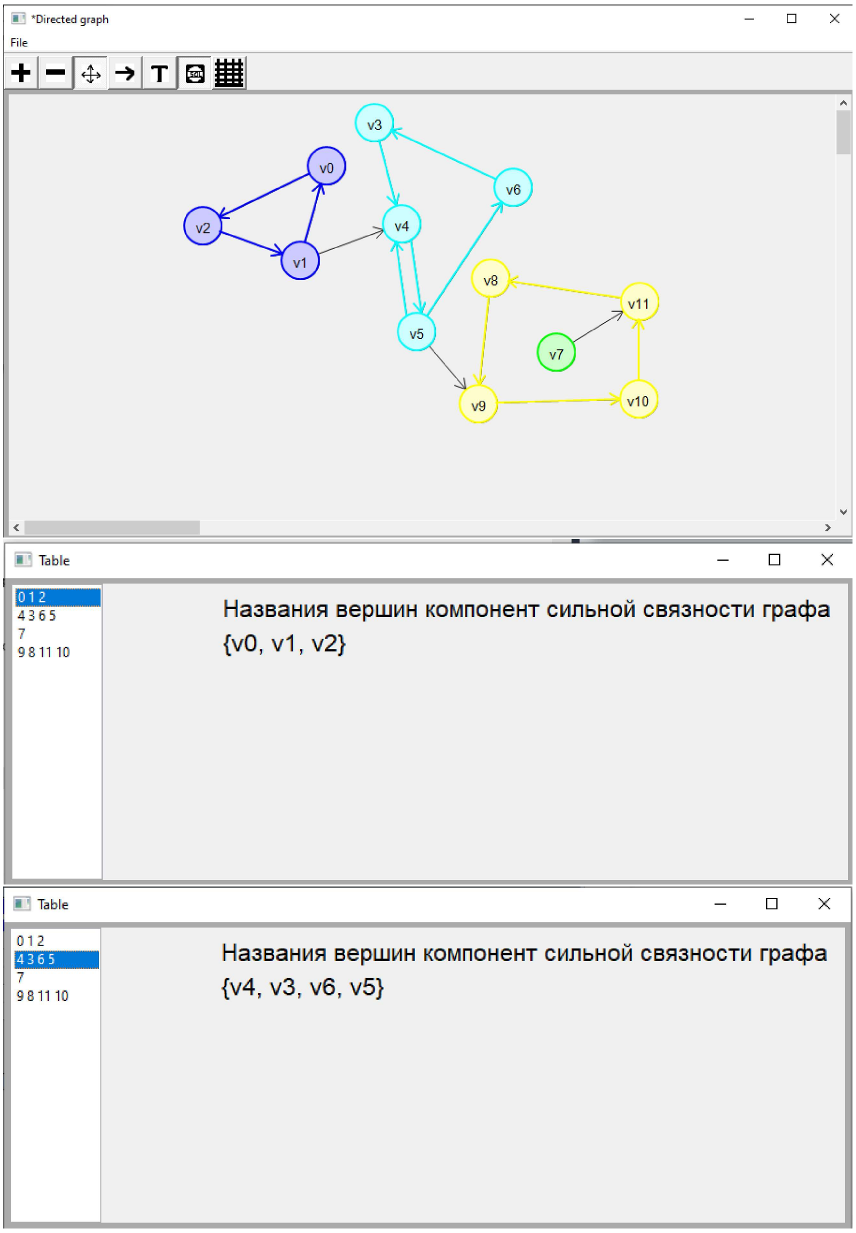
После обхода отсортированный массив с вершинами имеет вид order = [0, 6, 3, 5, 4, 1, 2].

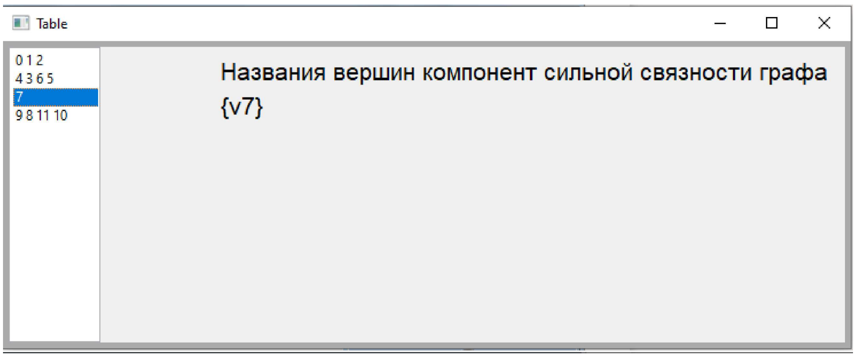
Инвертированный граф

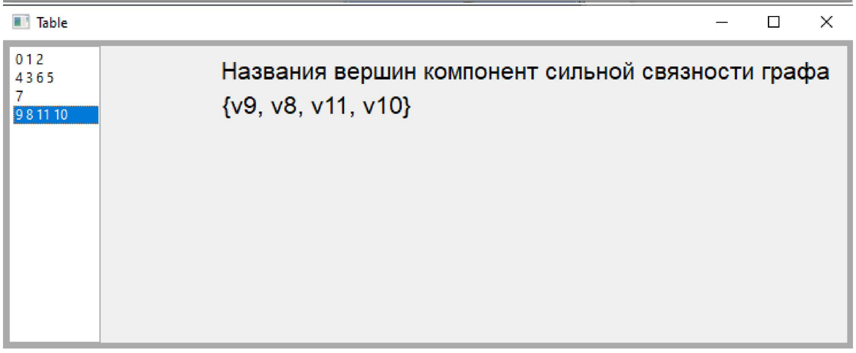


Запускаем второй обход. Полученные компоненты связности: {2, 0, 1}, {4, 5, 3, 6}.

Скриншоты программы







Пример прикладной задачи

Некоторая компания, предоставляющая услуги по курьерским доставкам, решила разрабоать программу для составления маршрутов в пределах города. Чтобы не возникало ситуаций, когда курьер не смог добраться от одного пункта до другого, разработчики приложения извлекли данные об улицах и интерпретировали улицы в виде графа. Оказалось, что этот граф не обязательно сильно связный. В некоторых местах оказалось, что может быть дорога, которая входит и выходит из области, но не связана с какой-либо другой дорогой внутри области. Затем было решено предварительно обрабатывать подграф, перечислив все сильно связанные компоненты и отбросив все, кроме самого большого компонента. Проблема с запутавшимися курьерами была решена.