Описание программы

Основная идея

Конечная программа нацелена на реализацию алгоритма транзитивного замыкания Уоршелла.

То есть из начального бинарного отношения f, заданного в виде матрицы смежности (и визуализированного с помощью графа) не обладающего свойством транзитивности, будет достраиваться его замыкание по свойству транзитивности $\hat{f}$.

Причем, под бинарным отношением понимается функция $f : M^2 \to B$
сопоставляющая декартову квадрату $M^2 = \{(a, b) | a, b \in M\}$ из некоторого множества M значение из множества $B = \{0; 1\}$.
(т.е. если существует множество $M = (a, b, \ldots)$ то бинарное отношение говорит есть ли между элементами этого множества связь ($f(a, b) = 1$) или нет ($f(a, b) = 0$)). В таком случае такое бинарное отношение легко представимо на графе: если $f(a, b) = 1$, то в графе есть ребро $a \to b$, если$f(a, b) = 0$, то ребра $a \to b$ в графе нет.

Под транзитивностью бинарного отношения понимается наличие у f свойства, что для любых $a, b \in M : \begin{cases} f(a, b) = 1 \\ f(b, c) = 1 \end{cases}$, то $f(a, c) = 1$

Под транзитивным замыканием бинарного отношения $\hat{f}$ понимается такое бинарное отношение, что:

- f и $\hat{f}$ - согласованы (если $f(a, b) = 1$, то и $\hat{f}(a, b) = 1$)
- $\hat{f}$ обладает теми же свойствами, что и f, но еще и транзитивностью
- Свойство транзитивности было достигнуто путем добавление минимального кол-ва новых бинарных связей вида $\hat{f}(a, b) = 1$

Описание работы алгоритма

Рассмотрим как работает алгоритм построения транзитивного замыкания с помощью алгоритма Уоршелла:
Алгоритм работает с матрицей смежности. Идея алгоритма Уоршелла состоит в расширении множества промежуточных вершин по следующему правилу: на каждом шаге в рассмотрение добавляется одна новая вершина, после чего достижимости вершин пересчитываются "через нее".

То есть если и w — промежуточная вершина, то достижимость вершины v из вершины u через w пересчитывается по правилу:

$$d(u,v) := max(d(u,v), d(u,w) \cdot d(w,v))$$

т. е. не меняется, если v достижима из u, и меняется (с 0 на 1), если достижимости до введения промежуточной вершины w не было, а w достижима из u и v достижима из w. Таким образом, после k шагов будут соединены те вершины, которые достижимы по путям, проходящим только через первые k вершин (кроме первой и последней).

Также можно записать эту формулу в логических операциях (т.к. ребра принимают значения 0 или 1): $d(u,v) := d(u,v) \vee (d(u,w) \wedge d(w,v))$

Пусть бинарное отношение f задано на множестве M, тогда запишем псевдокод для алгоритма Уоршелла:

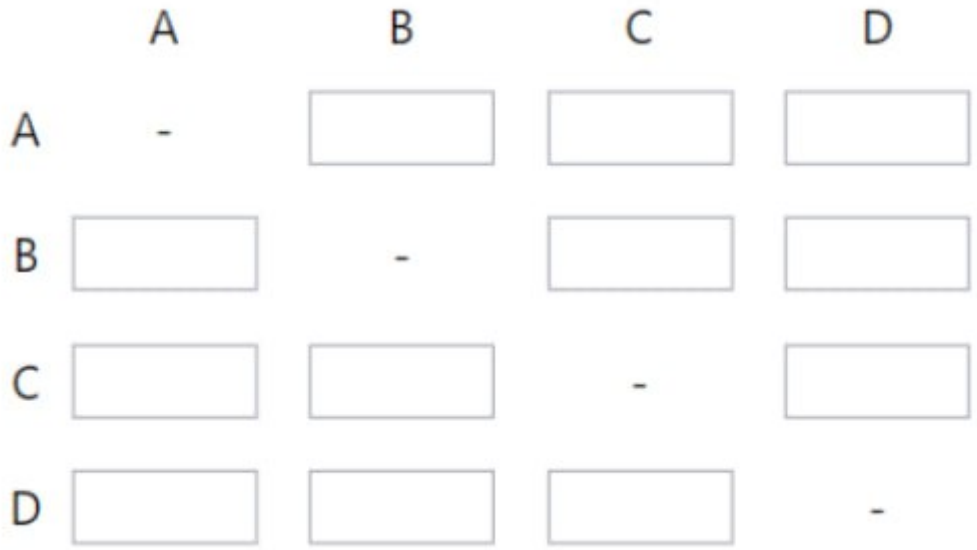
```
// Инициализация матрицы (если ее нет изначально)
for each u in M:
        for each v in M:
                d(u,v) = f(a,b) // Инициализация значения "ячейки" матрицы
смежности

//Основной алгоритм
for each w in M:
        for each u in M:
                for each v in M:
                        d(u,v) = d(u,v) or (d(u, w) and d(w, v))
```

Таким образом можем заметить что алгоритм исполняется за $O(N^3)$, где N - кол-во вершин в матрице.

Визуализация

Визуализация матрицы

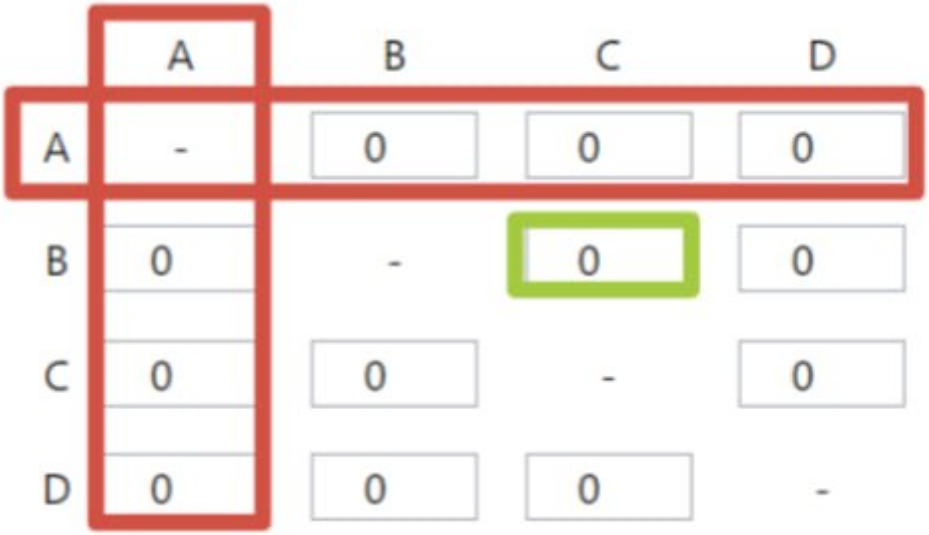
Планируется что на экране после запуска программы будет изображение вида:



где можно будет "вручную" добавить значение каждой ячейки матрицы смежности из диапазона (0,1) (есть ребро между вершинами или нет), а также возможность сгенерировать эту матрицу случайно или импортировать из файла.
Также будет возможность добавить вершину в матрицу или удалить ее из матрицы для получения аналогичной таблицы большего или меньшего вида.

В ходе работы алгоритма планируется возможность выбора режима:

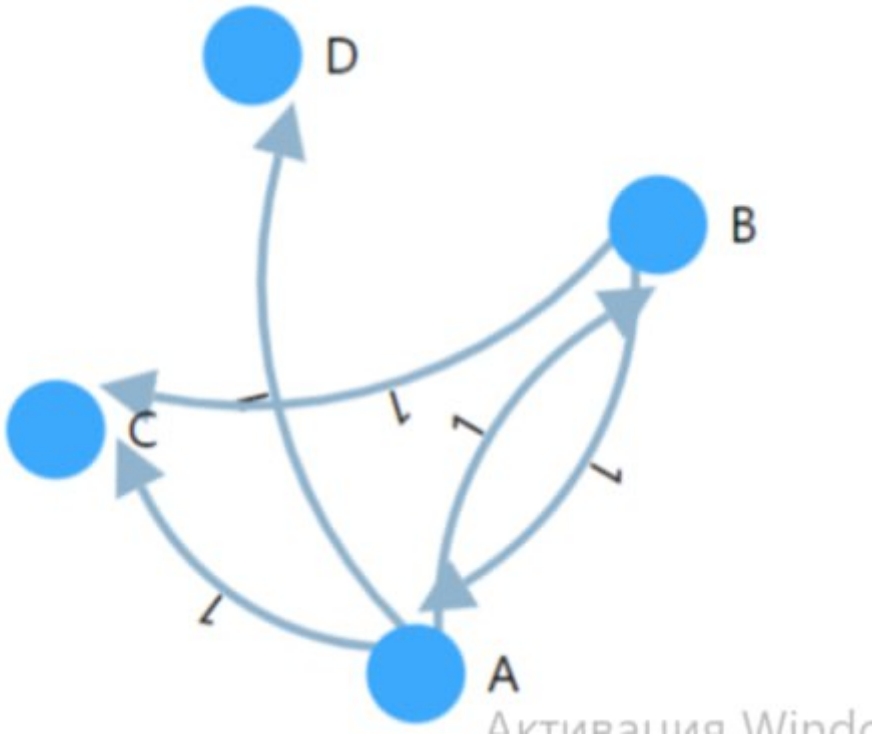
- Режим отладки (с подробной визуализацией изменения матрицы в ходе работы алгоритма)
- Обычный режим (просто появляется итоговая матрица смежности с транзитивным отношением)
  Для режима откладки планируется визуальное отображение главной для этого шага вершины и обрабатываемой ячейки примерно следующего вида:



(где красным обведена главная для этого шага вершина (ту, через которую смотрим можно ли пройти), а зеленым обрабатываемая на данном шаге ячейка))

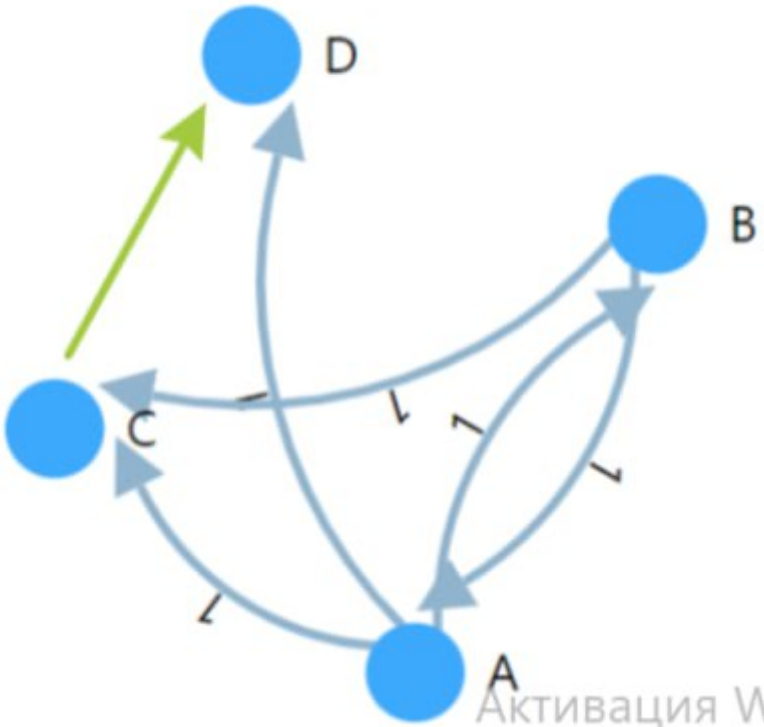
Визуализация графа

Для полученной матрицы смежности планируется визуализация графа для этой матрицы примерно следующего вида:



Тогда для обычного режима будет визуализирована в виде графа сначала начальная матрица, а затем (после отработки алгоритма) конечная.

Для режима отладки планируется что вновь добавленное ребро (если оно добавлено) будет отмечаться цветом на данном шаге примерно следующим образом:



(если ребро $C \to D$ было добавлено на данном шаге) и всплывать текстовое сообщение о добавленном ребре вида "добавлено ребро $V \to U$"

То есть граф будет постепенно обновляться в соответствии с обновленными значениями в матрице смежности на каждом шаге

Дополнительные взаимодействия пользователя с интерфейсом

Помимо изложенной информации о визуализации планируется, что пользователю на главном экране будут доступны кнопки вида:

- Добавления/удаления вершин в матрице
- Выбора режима (обычный/отладки)
- Кнопка запуска работы алгоритма
- Для режима отладки будет выбор режима показа пошаговой работы алгоритма (ручной/ автоматический)
  - "Ручной" предполагает, что переход к следующему шагу будет осуществлен только при нажатии пользователем кнопки "следующий шаг"
  - "Автоматический" предполагает, что переход к следующему шагу будет осуществляться каждую секунду (возможно можно будет указать время самому, например в диапазоне от 0.1 секунды до 10 секунд с помощью ползунка)

План разработки

Приблизительный план разработки будет разделен на следующие этапы:

- **Создание прототипа** (предполагает создание готового, но не рабочего (или частично рабочего) интерфейса приложения: визуализация матрицы с рабочими для ввода ячейками, строящийся для данной матрицы граф, наличие перечисленных кнопок, но без их работоспособности)
- **Создание первой версии приложения** (предполагает доработку прототипа, чтобы работал алгоритм, но только в обычном режиме (без режима отладки))
- **Создание второй версии приложения** (предполагает доработку первой версии до состояния, чтобы алгоритм работал в двух режимах (простом режиме и режиме отладки))
- **Создание финальной версии приложения и написание отчета** (предполагает исправление возможных ошибок и написания отчета для разработанного приложения)