

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

В.С. ЗУБОВ, О.В.ШЕВЧЕНКО

**КУРСОВОЕ ПРОЕКТИРОВАНИЕ
ПО СТРУКТУРАМ ДАННЫХ
И МЕТОДАМ ПРОГРАММИРОВАНИЯ**

Методические указания

для студентов, обучающихся по направлению
«Прикладная математика и информатика»

Москва
Издательство МЭИ
2018

УДК 621.398
З-915

*Утверждено учебным управлением НИУ «МЭИ»
в качестве учебного издания*

Подготовлено на кафедре математического моделирования МЭИ
Рецензент: - А.В. Князев, канд. техн. наук, доц.

Зубов В.С.

З-915 Курсовое проектирование по структурам данных и методам
программирования: методические указания / В.С. Зубов,
О.В.Шевченко – М.: Издательство МЭИ, 2018. – 24 с.

Содержит указания и рабочие материалы, необходимые при выполнении
курсового проектирования по структурам данных и методам программирования с
использованием языков C++ и Object Pascal. Обеспечивает индивидуальную
работу студентов учебной группы. Использует как предметную основу графы.
Предназначено для студентов специальности «Прикладная математика».

**УДК 621.398
ББК 32.973-018.1**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. Требования к содержанию курсовой работы	4
2. Перечень вариантов искомого результата	5
3. Указания по выполнению курсовой работы	7
4. Требования к оформлению отчета по КР	10
Список литературы	16
Приложения	17

ВВЕДЕНИЕ

Завершающий этап подготовки студентов по методам программирования требует решения студентом сложных учебных задач. В дисциплине «Структуры данных и методы программирования», изучаемой в четвертом семестра обучения, предметной основой задач является сортировка данных и информационный поиск. В монографии [1] Д. Кнут указывает «... практически каждый важный аспект программирования возникает в связи с сортировкой и поиском». Они наглядны, просты для понимания.

В определенной степени это относится и к задачам на графах, которые реализуются студентами во второй половине семестра. Логика их решения сложнее, трудоемко и тестирование построенных программ. Поэтому задачи на графах являются предметом не лабораторных, а курсовых работ.

В курсовой работе используются понятия и термины, полученные студентом в курсе «Дискретная математика», а приобретает он навыки использования типовых систем исследования графов.

Некоторые термины, используемые в заданиях, требуют пояснений. *Длина пути* во взвешенном графе понимается как сумма весов ребер, образующих этот путь, а если веса ребер не заданы – как число ребер, образующих путь. *Кратчайший путь* характеризуется наименьшей длиной. Путь от вершины Y к вершине X обозначаем как $Y \rightarrow X$, а инцидентное вершинам Y, X ребро как $\{Y, X\}$. Число вершин в графе обозначим n .

Деревом поиска назовем множество всевозможных простых путей с общей начальной вершиной. На рис.1 слева дан пример графа, а справа – дерево поиска путей, начинающихся в вершине A . Общие начала путей в нем совмещены. Ветвь с вершинами D, C, E участвует в дереве поиска дважды: в составе путей, проходящих через узел B , и в составе путей, начинающихся с ребра $\{A, D\}$. Размер дерева поиска в плотных графах экспоненциально возрастает с увеличением n . Как следствие, увеличивается и вычислительная сложность поиска путей. Отбрасывая «неперспективные» ветви, можно уменьшать размер дерева поиска, производить его *усечение*. Критерий оценки ветви дерева поиска, если он существует, зависит от содержания задачи. Разновидность поиска с возвратами, в которой предусмотрено усечение дерева поиска, носит название *метода ветвей и границ*.



Рис. 1. Граф и дерево поиска.

1. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ КУРСОВОЙ РАБОТЫ

Приступая к курсовой работе (КР), студент должен овладеть тремя вариантами описания графа: а) в виде матрицы смежности ([2], § 8.2); б) в виде списков инцидентий ([2], § 8.4); в) в виде структуры с оглавлением ([2], § 8.4). Требуемый вариант описания графа указывает преподаватель.

Рассматриваются только связные графы. Разновидности графа определяются текстом задания. Если в нем упомянуты веса ребер или вершин, граф взвешенный. В таком графе длиной пути будем называть сумму весов ребер, образующих этот путь. Вес не может быть меньше 0. Ориентированный граф используется, только если это предписано в тексте задания. Осмысливая решение задачи КР, студент должен выбрать для реализации подходящие системы из четырех базовых вариантов систем исследования графа:

- модифицированный поиск в глубину ([2], § 8.9);
- ординарный поиск в ширину ([2], § 8.10);
- поиск кратчайших путей по методу Дейкстры ([2], § 8.4);
- поиск кратчайших путей на основе топологического упорядочения вершин ([2], § 9.4).

Вначале студент записывает и проверяет программу базового варианта. Внося корректные изменения в программу, студент приспособливает выбранную систему к своему варианту задания. Например, для выявления множества простых путей, начинающихся в заданной вершине А, следует применять модифицированный поиск в глубину (МППГ). Характерной ошибкой учащегося является замена МППГ *многократным* построением пути, начиная от вершины А. Этот неэффективный способ требует запоминания множества выявляемых путей, больших затрат времени на их сравнение. Применяя МППГ, *однажды* запускаем процесс поиска, в ходе которого автоматически создаются различные простые пути (или простые циклы, если этого требует задание). МППГ пригоден и для поиска единственного пути, например, пути экстремального в заданном смысле.

Для краткости в дальнейшем термины «простой путь», «простой цикл» заменяются терминами «путь» и «цикл», поскольку используются лишь простые пути и простые циклы.

Пример 1. Согласно заданию надо найти все циклы, проходящие через вершину A , не имеющие общих ребер ни с кратчайшим путем $B \rightarrow C$, ни с кратчайшим путем $D \rightarrow E$.

Для исчерпывающего перебора циклов применяем МПГ с той особенностью, что нужны замкнутые пути. Поиск такого пути начинаем с вершины A , в процессе поиска проверяем, возвращается ли путь в вершину A . В измененную программу надо внести проверку, не принадлежит ли очередное ребро перечню запрещенных ребер. Такие ребра игнорируются, в состав цикла не включаются. Проверку надо производить по ходу построения возможного цикла. Рассмотрено содержание подпрограммы, использующей метод ветвей и границ. До ее применения нужно создать перечень запрещенных ребер, выявить кратчайшие пути $B \rightarrow C$, $D \rightarrow E$.

Так как граф не взвешенный, для нахождения кратчайшего пути используем ординарный поиск в ширину. В базовую программу поиска вносим изменения: при помещении в очередь посещенной вершины запоминаем вершину-предшественника. При успешном завершении поиска в ширину (достижении заданной вершины X) берем ее предшественника Y , запоминаем ребро $\{Y, X\}$ и проходим всю цепь предшественников, запоминая ребра. Конец цепи – вершина без предшественника.

Рассмотренное задание требует уточнения. Может существовать несколько кратчайших путей $B \rightarrow C$ или $D \rightarrow E$. В курсовой работе исследовать, запоминать каждый вариант пути не следует (программа чрезмерно усложнится). Ординарный поиск в ширину дает нам один вариант кратчайшего пути, его и используем. Аналогичные уточнения могут потребоваться и для некоторых других вариантов задания. Нужно согласовать их с преподавателем *при получении индивидуального задания*.

Особенности индивидуального задания определяются указанным преподавателем вариантом ниже следующего перечня. Действия учащегося приведены ниже, он должен

- изучить указанный вариант перечня для определения типа графа, указанного явно или заданного по правилу умолчания.
- уточнить способ описания исходного графа и представление результата;
- обоснованно выбрать базовую систему исследования графа;
- внести в нее необходимые изменения и составить программу на языке C++, планируемый результат которой указан в пункте перечня;
- предусмотреть в программе изображение графа и результата, используя подходящую цветовую гамму;
- подготовить тесты для испытания программы. Среди исходных графов должны быть и такие, для которых искомый результат не реализуется. Программа должна сообщать об этом;
- испытать программу на множестве тестов, при необходимости выполняя отладочные действия;
- оформить отчет по выполненной курсовой работе (см. п. 4).

2. ПЕРЕЧЕНЬ ВАРИАНТОВ ИСКОМОГО РЕЗУЛЬТАТА

1. Всевозможные пути $A \rightarrow B$, частью которых является кратчайший путь $C \rightarrow D$.
2. Цикл, проходящий через вершины A, B, C , не содержащий ребер, вес которых больше P .
3. Все пути $A \rightarrow B$, не имеющие общих ребер с самым длинным циклом, проходящим через вершину B (полагается, что он единственный).
4. Путь, начинающийся в вершине A и проходящий через максимальное число других вершин. Если таких простых путей несколько, найти все.
5. Все циклы, проходящие через вершину A , но не проходящие через центр графа; вершина A центральной не является.
6. Цикл, проходящий через вершины A, B , частью которого является кратчайший путь $C \rightarrow D$.
7. Путь $A \rightarrow B$ наименьшей стоимости в графе, где заданы не только веса ребер, но и веса вершин. Стоимость пути – это сумма весов пройденных ребер и вершин.
8. Путь $A \rightarrow B$, проходящий через вершину C , все ребра которого имеют вес в интервале от $P1$ до $P2$.
9. Все циклы, проходящие через вершину A , не имеющие общих ребер с кратчайшим путем $C \rightarrow D$ (полагается, что кратчайший путь $C \rightarrow D$ единственный).
10. Все пути $A \rightarrow B$, не имеющие общих ребер с кратчайшим путем $A \rightarrow B$, находимым по способу топологического упорядочения (граф – ориентированный, без циклов).
11. Путь $A \rightarrow B$, проходящий через вершины C, D и содержащий минимальное число ребер заданного веса P .
12. Все пути $A \rightarrow B$, проходящие хотя бы через одну центральную вершину графа; вершины A, B центральными не являются.
13. Кратчайший цикл, проходящий через вершину A , часть которого – путь $C \rightarrow D$ или путь $D \rightarrow C$.
14. Путь $A \rightarrow B$ наименьшей стоимости в графе, представляющем систему дорог; вес ребра – это стоимость топлива, расходуемого при проезде дороги, представленной ребром; особо заданы пошлины, взимаемые при въезде на дороги.
15. Все пути $A \rightarrow B$, проходящие через вершину C и не имеющие общих ребер с кратчайшим путем $A \rightarrow B$, находимым по методу Дейкстры (граф – взвешенный).
16. Путь $A \rightarrow B$, проходящий через вершины C, D и имеющий наибольший вес.
17. Все кратчайшие пути $A \rightarrow B$, не проходящие через вершины из заданного множества вершин.
18. Все пути $A \rightarrow B$, не имеющие общих ребер ни с одним циклом графа.

19. Цикл, проходящий через вершину A , но не проходящий через вершину B , и содержащий максимальное число ребер заданного веса P .
20. Цикл, проходящий через вершину A и максимальное число других вершин, не являющихся вершинами C, D .
21. Цикл, проходящий через вершины A, B , все ребра в котором имеют вес в интервале от P_1 до P_2 .
22. Каждый цикл, проходящий через вершины A, B , хотя бы одно ребро в котором принадлежит кратчайшему пути $C \rightarrow D$.
23. Все циклы, проходящие через узел A , вес которых не превышает P , не имеющие общих ребер с кратчайшим путем $C \rightarrow D$.
24. Все циклы, проходящие через узлы A, B, C и не содержащие ребер, вес которых равен нулю.
25. Путь $A \rightarrow B$, проходящий через вершины C, D , суммарный вес которого не превышает P .
26. Какой-либо путь, началом которого является кратчайший путь $A \rightarrow B$, а окончанием – кратчайший путь $C \rightarrow D$.
27. Кратчайший путь $A \rightarrow B$, проходящий через вершину C . Решить задачу двумя способами: а) на основе поиска в глубину; б) дважды используя поиск в ширину.
28. Во взвешенном графе наиболее удаленные вершины от вершины A , являющейся медианой графа (расстояние определяется как сумма весов ребер, соединяющих эту вершину с вершиной A).
29. Путь $A \rightarrow B$, начинающийся с ребра $\{A, E\}$ и содержащий минимальное число общих вершин с кратчайшим путем $C \rightarrow D$.
30. Кратчайший путь $A \rightarrow B$, проходящий через ребра $\{C, D\}, \{D, E\}$, но не проходящий через центр графа. Вершины C, D, E не центральные.

3. УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

Для исчерпывающего перебора простых путей или простых циклов применяем модифицированный поиск в глубину (МППГ). Во избежание ошибок следует придерживаться именно этой системы исследования графа, тщательно обдумывая вносимые изменения. Если характер задания определяет возможность усечения дерева поиска, обязательно используйте метод ветвей и границ.

Пример 2. Во втором варианте перечня предполагаются запрещенные ребра. Допустим, таким является ребро $\{Y, X\}$. При достижении вершины Y выбирается одно из инцидентных ребер. Если выбрано ребро $\{Y, X\}$, безусловное выполнение шага назад от вершины Y будет ошибкой. Нужно проверять другие ребра, инцидентные Y , для движения вперед, и только если таких ребер нет, выполнять шаг назад. Возможным следствием ошибки явится исключение из рассмотрения некоторых циклов.

Пример 3. Целый ряд вариантов задания предполагают усечение области поиска. Обратимся к 6 варианту. Начальный этап решения – нахождение множества ребер, образующих кратчайший путь $C \rightarrow D$. Второй этап – МППГ

с изменениями. Трудоемкий перебор простых путей или циклов главный недостаток МПГ. Сократить его в 6 варианте задания можно, отказавшись от рассмотрения *многочисленных* циклов, делая проверки по ходу построения цикла. Пусть $C = 5$, $D = 3$, а найденный кратчайший путь $C \rightarrow D$ образован ребрами $\{5, 2\}$, $\{2, 4\}$, $\{4, 3\}$. При достижении вершины C выполняем перебор ребер, инцидентных C . Соответствующий цикл предусмотрен в МПГ, изменение состоит в том, что шаг вперед не выполняем, пока не выявится ребро $\{5, 2\}$. Затем в указанном цикле выполняем перебор ребер, инцидентных вершине 2, делая шаг вперед только при выявлении ребра $\{2, 4\}$ и т.д. Таким образом, исключаем из процесса поиска ветви, отходящие от вершин кратчайшего пути $C \rightarrow D$.

Пример 4. Рассмотрим 8 вариант. Когда выявляется ребро запрещенного веса, шаг вперед по этому ребру не выполняется и отвергается ветвь дерева поиска, возможно большая.

Пример 5. Обратимся к 25 варианту, очевидно требующему применения метода ветвей и границ. Суммирование веса ребер по ходу движения вперед позволяет сокращать дерево поиска. Всякий раз, когда сумма превышает P , делается шаг назад (возможно не один). При шаге назад от вершины X к вершине Y из накопленной суммы вычитается вес ребра $\{Y, X\}$. Такая коррекция нужна и в случае посещения висячих вершин, висячих ветвей.

Пример 6. Для использования метода ветвей и границ в 11 варианте нужен счетчик K ребер заданного веса P . При шаге вперед по такому ребру счетчик увеличивается на 1, а при шаге назад по такому ребру – уменьшается на 1. Если при достижении вершины B подтверждается факт прохождения пути через вершины C и D , запоминаются значение K_0 счетчика K и путь $A \rightarrow B$, делается шаг назад и процесс МПГ продолжается с целью выявления следующих вариантов пути. Теперь на каждом шаге вперед проверяется истинность условия $K < K_0$. Если оно оказывается ложным, происходит усечение дерева поиска. Процесс МПГ вновь приводит к вершине B , только если на пути к ней значение K осталось меньше K_0 . Значение K_0 заменяется, путь $A \rightarrow B$ запоминается и процесс МПГ повторяется, пока всевозможные допустимые пути не будут рассмотрены.

Для упрощения алгоритма условие $K < K_0$ проверяйте с самого начала поиска пути, выполнив предварительно присваивание $K_0 = n$.

Запрещенные вершины. Если в варианте задания упомянуты вершины, прохождение которых на этапе МПГ запрещено, то чтобы сократить дерево поиска, вершины следует проверять по ходу построения пути. Простой перебор запрещенных вершин на каждом шаге МПГ не рекомендуется, так как существенно замедляет построение пути. Имеются альтернативы: а) дихотомический поиск в *упорядоченном* списке запрещенных вершин; б) использование вектора Q признаков вершин; $Q[w] = \text{false}$ означает, что вершина w запрещена. Это самый быстрый способ.

Обязательные вершины. Если задание предписывает прохождение пути через заданные вершины, сократить дерево поиска не удастся.

Проверку нужно выполнять после построения очередного варианта пути. Двукратный перебор вершин не потребуется. Если метка $R[v]$ вершины v больше 0, путь проходит через v . Выявление вариантов пути продолжается, пока не найдется путь, проходящий через все заданные вершины.

. Рассмотрим особенности реализации *поиска в ширину*. Такой этап предполагается в вариантах 1, 5, 6, 9, 12, 17, 22, 23, 26, 27, 28, 29, 30 приведенного выше перечня. При поиске в ширину посещаемые вершины снабжаются метками, препятствующими повторному занесению в очередь. Помечая каждую вершину (кроме начальной) номером вершины-предшественника, можем получить цепь предшественников, в которой каждая пара соседних элементов определяет ребро – элемент кратчайшего пути от *конечной* вершины искомого пути до отправной вершины поиска в ширину.

Этап поиска в ширину наиболее сложен в реализации вариантов 5, 12, 30, в которой сначала надо для каждой вершины найти эксцентриситет, а затем выявить множество *центральных вершин*, для которых эксцентриситет минимален.

Рекурсия в реализации МПГ. Проверку корректности изменений МПГ удобно осуществлять на простой модели. Используйте рекурсию. Ниже очередность ее шагов показана на псевдокоде.

Подпрограмма МПГ (v) // v – текущая вершина МПГ
 // (вначале это исходная вершина z)

1. Для v выполняем поиск не посещенной смежной вершины w
2. Если вершина w не найдена, переходим к пункту 5
3. Вершину w снабжаем меткой $R[w] = v$, запрещающей повторное ее посещение
4. Рекурсивный переход МПГ (w)
5. Снимаем запрет на повторное посещение вершины v (действием $R[v] = -1$)
6. Конец шага рекурсии

Данная подпрограмма проходит всевозможные пути, начинающиеся в вершине z , без их обработки. Если нас интересует путь $z \rightarrow u$ до некоторой вершины u , поставим в конец пункта 3 проверку условия $w = u$. Когда оно истинно, обрабатываем найденный путь с переходом к пункту 1. Текущей вершиной остается v , процесс МПГ продолжается и могут быть найдены другие пути $z \rightarrow u$. Если все множество путей нас не интересует, выполнение пункта 4 ставим в зависимость от логической переменной b , задавая ей значение false при обнаружении первого пути $z \rightarrow u$. Отмена рекурсивного вызова МПГ вызовет завершение всех копий подпрограммы.

Для обработки найденного пути надо иметь последовательность образующих его вершин. Метка $R[w]$ вершины w содержит номер предшественника вершины w . Поэтому, если начиная с вершины $j = u$ выполнять действие $j = R[j]$, можно получить цепь предшественников, заканчивающуюся начальной вершиной z пути.

Если согласно заданию запрещается проходить ребро $\{X, Y\}$, вставляем пункт 1а между пунктами 1 и 2:

- 1а. Если $(v = X) \text{ and } (w = Y) \text{ or } (v = Y) \text{ and } (w = X)$ перейти к пункту 1.

В приложении А подпрограмма МПГ, в которой учтены изменения, записана на языке Паскаль. Ее изучение способствует разработке этапа МПГ в индивидуальном задании.

4. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА ПО КР

Отчет по курсовой работе должен содержать следующие разделы:

а) Титульный лист. Вверху страницы -- вертикально расположенные поля: <университет>, <институт>, <кафедра>; в центре страницы -- поле "Отчет по курсовой работе по курсу: <наименование курса> на тему: <тема варианта курсовой работы>"; справа от центра, выровненные по правому краю, -- вертикально поля "Студент группы <наименование группы><ФИО студента>", "Принял: <ФИО преподавателя>"; внизу страницы -- вертикально поля "г.Москва", <текущий год>; центрирование текста титульного листа (кроме особо указанных полей); размер шрифта 14pt, гарнитура Times New Roman.

б) Постановка задачи (в общем виде с указанием данных конкретного варианта задания, выданного преподавателем).

в) Ход решения задачи (построение алгоритма решения задачи в виде *псевдокода*; последовательное перечисление всех используемых для решения задачи графовых алгоритмов с обоснованием их применимости в случае конкретного варианта задания, выданного преподавателем).

г) Структура программы на С++ (описание файлов, составляющих проект, с указанием прототипа главной (вызывающей) функции и перечислением прототипов вспомогательных (вызываемых) функций)

д) Скриншоты (снимки экрана) программы (главная форма приложения с ее основными элементами управления; визуализация исходного графа и результата работы программы -- искомого пути, цикла, ребер и т.п. -- на произвольном простом примере)

Главная форма приложения должна, как минимум, содержать:

- поля ввода исходных данных (размер графа -- поле редактируемого ввода -- например, типа TEdit в IDE Borland C++ Builder; граф -- прямоугольная таблица -- например, типа TStringGrid, в случае задания графа матрицей смежности, или граф -- список -- например, типа TStringGrid или другого подходящего графического компонента среды разработки (IDE), если граф задан списком смежных вершин (инцидентий), или иные графические компоненты в соответствии с заданием конкретного варианта; плюс поля для задания условий поиска в графе);
- кнопки управления (Старт, Выход, и т.д.);
- желательно - главное меню формы (с пунктами "Файл" - подпункты "Создать", "Сохранить", "Сохранить как...", "Открыть", "Удалить"; пункт

"Визуализация" -- подпункты "Сохранить", "Сохранить как...", пункт "О программе", пункт "Помощь").

На форме отрисовки результата располагается графическое изображение решения задачи КР. Граф и структуры, соответствующие условиям варианта задания, следует отображать информативно, эргономично, в подходящей цветовой гамме, с соответствующим графическим разрешением экрана.

е) Результаты тестирования программы (аналогично пункту д) для максимально полной системы тестов, покрывающей все возможные логические ветви вариантов исходных данных графа)

ж) Листинг программы (описание на языке C++ главной и вспомогательной функций программы)

з) Вывод (резюме возможностей разработанной программы, с указанием языка и среды разработки)

Примечание. Альтернативный описанному выше в пп. д), е) вариант организации интерфейсной части КР -- использование готового приложения Graphviz для визуализации графов -- предложен в приложении Б. Этот вариант может использоваться студентом вместо формы отрисовки при выполнении заданий пп. д), е).

СПИСОК ЛИТЕРАТУРЫ

1. **Кнут Д.Э.** Искусство программирования, том 3. Сортировка и поиск, 2-е изд. М.: Изд. дом «Вильямс», 2000. – 822 с.: ил.
2. **Зубов В.С., Шевченко И.В.** Структуры и методы обработки данных: Практикум в среде Delphi. – М.: Информационно-издательский дом «Филинь», 2004. – 304 с.
3. **Князев А.В.** Основы языка C++. – М.: Издательство МЭИ, 2013. – 80 с.
4. **Князев А.В.** Разработка программ в среде C++ Builder: учебное пособие / А.В. Князев. – М.: Издательство МЭИ, 2012. – 80 с.
5. **Седжвик Р.** Алгоритмы на C++. СПб: Вильямс, 2016.

А. Рекурсивная реализация метода МПГ

Переменные n , b , u , X , Y являются глобальными. Именем *Obrab* обозначена пользовательская подпрограмма обработки найденного пути. Она не приведена. Описанием графа является матрица смежности. Перед запуском процедуры MPG меткам всех вершин графа должно быть присвоено значение -1, а метке отправной вершины графа – значение n . Подчеркнутый оператор реализует запрет прохождения пути через ребро $\{X, Y\}$ в любом направлении. Он упрощается, если запрещен проход ребра лишь в одном направлении, например, от Y к вершине X . Если путь *должен проходить* через ребро $\{X, Y\}$ в направлении от X к Y , заменяем подчеркнутый оператор оператором

$$\text{if } (v = X) \text{ and } (w \neq Y) \text{ then continue;}$$

Procedure MPG (v : integer);

Var j, w : integer;

begin

for $w := 0$ to $n-1$ do

begin if $(A[v, w] > 0)$ and $(R[w] = -1)$ then

begin if $(v = X) \text{ and } (w = Y)$ or $(v = Y) \text{ and } (w = X)$ then continue;

if $w = u$ then begin $b := \text{false}$; *Obrab* (v, w) end

else begin $R[w] := v$; if b then MPG (w) end // Шаг вперед, если b – истина

end

end;

$R[v] := -1$ // Действие, выполняемое при шаге назад

end;

Б. Вариант реализации интерфейсной части КР

Наиболее простой способ визуализировать результаты решения задач на графах — использовать свободно доступную систему визуализации Graphviz.

Данную систему можно скачать в сети интернет по адресу <http://www.graphviz.org> (есть версии для ОС Window/Linux/MacOS).

Система Graphviz представляет собой набор программ, принимающих на вход описание графа в виде текстового файла на специальном языке описания графов DOT и выдающих изображение графа в виде файлов в популярных графических форматах (bmp/gif/jpeg/png).

Язык DOT достаточно прост для генерации описания графа из программы.

Описание простейшего неориентированного и ориентированного графа приведено ниже:

Неорграф

graph G {

"1"

"2"

"3"

Орграф

digraph G {

"1"

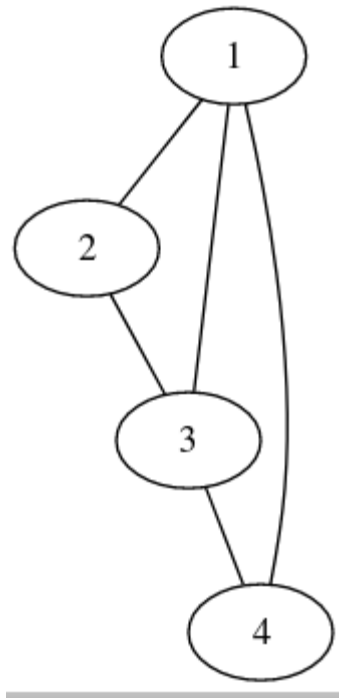
"2"

"3"

```

"4"
"1" -- "2"
"2" -- "3"
"3" -- "4"
"4" -- "1"
"1" -- "3"
}

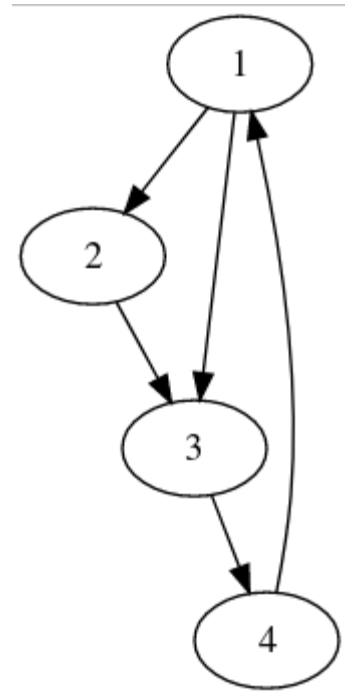
```



```

"4"
"1" -> "2"
"2" -> "3"
"3" -> "4"
"4" -> "1"
"1" -> "3"
}

```

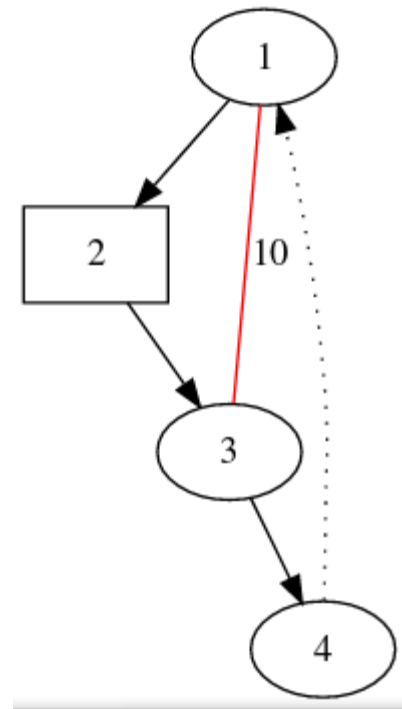


Для создания графов содержащих ориентированные и неориентированные ребра, а также выделения цветом и задания пометок ребер необходимо указывать дополнительные атрибуты.

```

digraph G {
    "1"
    "2" [shape=box]
    "3"
    "4"
    "1" -> "2"
    "2" -> "3"
    "3" -> "4"
    "4" -> "1" [style=dotted]
    "1" -> "3" [dir=none, color=red, label="10"]
}

```



Для построения изображения графа по готовому текстовому описанию можно использовать:

1. Сервисы <http://www.webgraphviz.com/> или <https://stamm-wilbrandt.de/GraphvizFiddle/> в сети Интернет;
2. Утилиты `dot` или `circo` из пакета Graphviz, запускаемые из командной строки на своем компьютере.

Для этого необходимо запустить командную оболочку (`cmd.exe` в Windows) и выполнить команду `dot -Tpng input.dot > output.png` где `-Tpng` задает формат выходного изображения(png), `input.dot` — текстовый файл с описанием графа на языке DOT, `output.png` — имя файла выходного изображения.

Ниже приведен пример программы, формирующей описание графа на языке DOT по матрице смежности (предварительно в нее вставляется заданный в массиве путь).

```

#include <iostream>

#define NODES      5
#define PATH_SIZE  4

//матрица смежности
int my_graph[NODES][NODES]={ {0,1,1,1,0},
                             {1,0,0,1,1},
                             {1,1,0,0,1},
                             {0,1,1,0,0},
                             {0,1,0,1,0}};

//путь
int my_path[PATH_SIZE]={1,3,2,0};

//вставка пути в матрицу смежности

```

```

void insert_path_in_graph(int graph[NODES][NODES],int path[PATH_SIZE]){
    for(int i=0;i<PATH_SIZE-1;i++){
        if(graph[path[i]][path[i+1]]>0){ //mark edge as path part
            graph[path[i]][path[i+1]]=2;
            if(graph[path[i+1]][path[i]]>0) graph[path[i+1]][path[i]]=2; //if undirected mart too
        }
        else graph[path[i]][path[i+1]]=-1; //mark edge as error
    }
}

//печать DOT описания графа
void print_graph_dot(int graph[NODES][NODES]){

    std::cout<<"digraph G{\n";

    for(int i=0;i<NODES;i++) //output nodes
        std::cout<<"\"<i<<"\n";

    for(int i=0;i<NODES;i++) //output edges
        for(int j=0;j<NODES;j++){
            if(j<i){
                if((graph[i][j]==1)&&(graph[j][i]==1)) //undirected edge not from path
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" [dir=none]\n";
                else if((graph[i][j]==2)&&(graph[j][i]==2)) //undirected edge from path
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" [dir=none,color=red,style=bold]\n";
                else if((graph[i][j]==1)) //directed edge not from path
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" \n";
                else if((graph[i][j]==2)) //directed edge from path
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" [color=red,style=bold]\n";
                else if((graph[i][j]==-1)) //error edge
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" [color=red,style=dotted]\n";
            }
            else{
                if((graph[i][j]==1)&&(graph[j][i]==0)) //directed edge not from path
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" \n";
                else if((graph[i][j]==2)&&(graph[j][i]==0)) //directed edge from path
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" [color=red,style=bold]\n";
                else if((graph[i][j]==-1)&&(graph[j][i]==0)) //error edge
                    std::cout<<"\"<i<<" ->"<<"\"<j<<" [color=red,style=dotted]\n";
            }
        }
    std::cout<<"}\n";
}

int main(void){
    insert_path_in_graph(my_graph,my_path); //mark path for visualization
    print_graph_dot(my_graph); //create DOT description
}

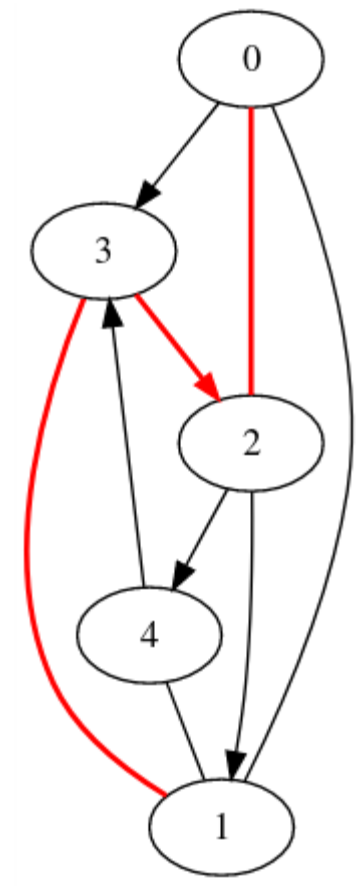
```

Получившееся описание графа и изображение:


```

digraph G{
"0"
"1"
"2"
"3"
"4"
"0" ->"3"
"1" ->"0" [dir=none]
"2" ->"0" [dir=none,color=red,style=bold]
"2" ->"1"
"2" ->"4"
"3" ->"1" [dir=none,color=red,style=bold]
"3" ->"2" [color=red,style=bold]
"4" ->"1" [dir=none]
"4" ->"3"
}

```



Учебное издание

Зубов Валерий Сергеевич
Шевченко Ольга Васильевна

**КУРСОВОЕ ПРОЕКТИРОВАНИЕ
ПО СТРУКТУРАМ ДАННЫХ
И МЕТОДАМ ПРОГРАММИРОВАНИЯ**

Методические указания

для студентов, обучающихся по направлению
«Прикладная математика и информатика»

Редактор Л.И. Веселовский
Компьютерная верстка Л.В. Валдаевой

Подписано в печать 21.04.2018	Печать офсетная	Формат 60x84/16
Печ.л. 1,5	Тираж 100 экз. Изд.№ 17-125	Заказ №142
