Лабораторная работа №1
по дисциплине МССАПРКЭС

Анализ структурных характеристик проектируемой системы электронных средств
(методами теории графов).

В качестве проектируемой системы электронных средств рассматривается некоторая коммутируемая пакетная сеть связи, состоящая из проектируемых устройств коммутации и линий связи между ними. Структура сети связи в виде неориентированного графа, полагается такой же, как структура сети интервалов (ребер графа) и станций (вершин графа) на схеме метрополитена на сайте https://metro.yandex.ru для трех городов: Харькова, Санкт-Петербурга и Москвы.

В качестве веса каждого ребра сети выступает время (в микросекундах) передачи пакетов в линии связи между смежными вершинами (устройствами коммутации), численные значения которых соответствуют времени (в минутах) проезда (перемещения) между соответствующими (по структуре) станциями метро.

Исходные данные о структуре сети и времени проезда (перемещения) между соседними станциями, а также о кратчайших маршрутах между любыми двумя станциями следует определять с помощью интерактивной программы на указанном выше сайте.

Задание 1.

С использованием интерактивной программы на сайте https://metro.yandex.ru для расчета кратчайших маршрутов на графах сети метрополитена городов Харькова, СПб и Москвы определить (методом направленного перебора только крайних станций и станций пересадок) диаметр, радиус и центр (центры) графа сети (см. Приложение 1)

Изобразить графически на структуре каждой сети самый длинный (по всем возможным парам соединяемых вершин) из всех кратчайших маршрутов, длина которого соответствует диаметру графа сети, и самые короткие (по всем возможным центральным вершинам) из всех самых длинных (по всем возможным остальным вершинам в паре с центральной) кратчайших маршрутов, длина которых соответствует радиусу графа (с выделением соответствующих центров графа).

Задание 2.

Выделить из структур трех сетей подграфы, включающие только станции пересадок и промежуточные станции между ними. Для Харькова такой подграф включает 9 станций с раздельными подстанциями пересадок и 6 с совмещенными, для СПб, соответственно, 16 и 8 станций, а для Москвы (в пределах внутреннего кольца) – 57 и 28 станций (см. Приложение 2)

Для каждого подграфа $\boldsymbol{G}$ определить его цикломатическое число $\nu(\boldsymbol{G})$ и хроматическое число $X(\boldsymbol{G})$ (другое обозначение$K(\boldsymbol{G})$) (см. Приложение 3).

Отметить также, является ли каждый из подграфов планарным или нет (см. Приложение 4).

Задание 3.

Проверить, являются ли указанные в задании 2 подграфы эйлеровыми (полуэйлеровыми), т.е. содержит ли они эйлеров цикл (путь), представляющий собой замкнутый (незамкнутый) маршрут, проходящий **по всем ребрам только по одному разу** (см. пояснения, например, в https://ru.wikipedia.org/wiki/Эйлеров_цикл или в http://studopedia.ru/3_80686_eylerovi-grafi.html).

Проверить также, являются ли указанные подграфы гамильтоновыми (полугамильтоновыми), т.е. содержит ли они гамильтонов цикл (путь), представляющий собой

замкнутый (незамкнутый) маршрут, проходящий **по всем вершинам только по одному разу** (см. пояснения, например, в https://ru.wikipedia.org/wiki/Гамильтонов_граф или в http://www.studfiles.ru/preview/1722050/)

Проанализировать два варианта подграфов: 1) с совмещением станций пересадок в виде одной вершины, не различая переходов; 2) с раздельным представлением станций пересадок в виде нескольких подстанций-вершин с дополнительными ребрами, соответствующими переходам между ними.

Изобразить указанные варианты подграфов и показать на них эйлеровы и гамильтоновы циклы и/или пути (если они есть).

Задание 4.

Используя алгоритм Дейкстры (https://ru.wikipedia.org/wiki/Алгоритм_Дейкстры) для **нечетных** номеров N по списку группы или Беллмана-Форда (https://ru.wikipedia.org/wiki/Алгоритм_Беллмана_—_Форда) для **четных** номеров N по списку определить в описанном выше подграфе для СПб (с учетом подстанций пересадок и промежуточных станций между ними) кратчайшие маршруты (и их длину) между заданной вершиной (см. далее – по вариантам) и всеми остальными вершинами (станциями).

С алгоритмами Дейкстры и Беллмана-Форда можно разобраться на примере алгоритмов маршрутизации в сетях связи, где они широко используются (см. Приложение 5).

Вес каждого ребра задать в соответствии с временем перемещения между соседними станциями (включая станции пересадок, связанные переходами), предварительно рассчитанном с помощью интерактивной программы на сайте https://metro.yandex.ru

При выполнении задания 4 необходимо использовать отдельную автономную нумерацию вершин подграфа от 1 до M, где M=16 (см.рис.2 в Приложении 1)

В качестве заданной вершины (по вариантам), относительно которой необходимо определять маршруты к другим (пятнадцати) вершинам, следует использовать вершину с номером W (в интервале от 1 до M), вычисляемым по формуле: W=1+mod(N,M), где N – номер исполнителя лабораторной работы по списку студентов в группе, mod(N,M) – остаток от деления числа N на M, где M=16.

Представить промежуточные этапы расчетов (в виде таблицы или на графе) согласно заданным алгоритмам Дейкстры или Беллмана-Форда, выполненные вручную или с помощью программы расчетов в системе MathCad (v.14 или v.15), составленной по соответствующему алгоритму.

Сравнить найденные маршруты с маршрутами, которые отображаются после расчетов для аналогичных пар вершин (станций) на сайте https://metro.yandex.ru

Отметить в выводах, совпали маршруты (и их длина) или нет (если нет, проанализировать – почему).

Приложение 1

Рис.1. Граф (схема) метрополитена г. Харькова

Рис.2. Граф (схема) метрополитена г. Санкт-Петербурга

Рис.4. Граф (схема) метрополитена г. Москвы с нумерацией станций пересадок в центре без выделения подстанций

Рис.5. Граф (схема) центральной части метрополитена г. Москвы с нумерацией подстанций пересадок

Приложение 2

Расстояния в графе, диаметр, центр, радиус графа

Утверждение. Если для двух вершин существует маршрут, связывающий их, то обязательно найдется минимальный маршрут, соединяющий эти вершины. Обозначим длину этого маршрута через d(v, w).

Определение. Величину d(v, w) (конечную или бесконечную) будем называть **расстоянием между вершинами v, w**. Это расстояние удовлетворяет аксиомам метрики:

d(v, w) ≥ 0, причем d(v, w) = 0 тогда и только тогда, когда v=w;

d(v, w) = d(w, v);

d(v, w) ≤ d(v, u) + d(u, w).

Определение. **Диаметром** связного графа называется максимально возможное расстояние между двумя его вершинами.

Определение. **Центром** графа называется такая вершина, что максимальное расстояние между ней и любой другой вершиной является наименьшим из всех возможных; это расстояние называется **радиусом** графа.

Пример

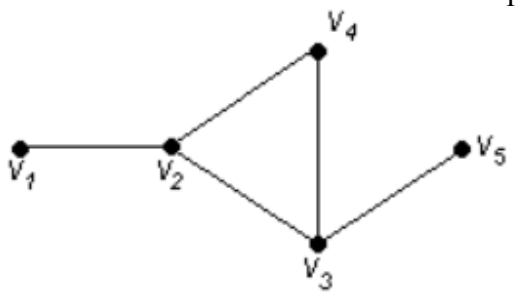


Рис. .

Для графа G, изображенного на рисунке , найти радиус, диаметр и центры.

Решение.

Чтобы определить центры, радиус, диаметр графа *G*, найдем матрицу *D(G)* расстояний между вершинами графа, элементами *dij* которой будут расстояния между вершинами *vi* и *vj*. Для этого воспользуемся графическим представлением графа. Заметим, что матрица *D(G)* симметрична относительно главной диагонали.

$$D(G) = \begin{bmatrix} 0 & 1 & 2 & 2 & 3 \\ 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 1 \\ 2 & 1 & 1 & 0 & 2 \\ 3 & 2 & 1 & 2 & 0 \end{bmatrix}.$$

С помощью полученной матрицы для каждой вершины графа *G* определим

$$r(v_i) = \max_j d(v_i, v_j)$$

наибольшее удаление из выражения: для *i, j = 1, 2, ..., 5*.

В результате получаем: *r(v1) = 3, r(v2) = 2, r(v3) = 2, r(v4) = 2, r(v5) = 3*. Минимальное из полученных чисел является радиусом графа *G*, максимальное - диаметром графа *G*. Значит, *R(G) = 2* и *D(G) = 3*, центрами являются вершины *v2, v3, v4*.

Приложение 3

Цикломатическое число

Граф, любая пара вершин которого связана, называют *связным графом*. В связном графе, перемещаясь по ребрам из вершины в вершину, можно попасть в каждую вершину. Граф, состоящий из отдельных фрагментов, называют *несвязным*, состоящим из отдельных компонент связности.

Если граф не связный, то множество его вершин можно единственным образом разделить на непересекающиеся подмножества, каждое из которых содержит все связанные между собой вершины и вместе с инцидентными им ребрами образует связный подграф.

Таким образом, несвязный граф представляет собой совокупность отдельных частей (подграфов), называемых *компонентами связности*.

При рассмотрении произвольного неориентированного графа G(X, U) без петель с $|X| = n$ и $|U| = r$, состоящего из «p» компонент связности, величину

$$v(G) = r - n + p$$

называют *цикломатическим числом графа.*

Цикломатическое число графа указывает то наименьшее число ребер, которое нужно удалить из данного графа, чтобы получить дерево (для связного графа) или лес (для несвязного графа), т. е. добиться отсутствия у графа циклов.

Цикломатическое число всегда неотрицательно.

Основное свойство цикломатического числа формулируется в виде теоремы:

Цикломатическое число мультиграфа равно максимальному числу независимых циклов.

Знание цикломатического числа оказывается полезным при анализе топологии электронных схем, а также для решения целого класса задач конструкторского проектирования РЭС.

Пример. Если рассматривать конструкцию печатной платы, то схему электрических соединений можно интерпретировать графом G (X, U), где X – множество областей контактных площадок, внутри которых проведение проводников запрещено, а U — множество трасс.

При этом все коммутационное пространство разбивается проведенными соединениями на отдельные, локально замкнутые области

$$Q = \{q_1, q_2, ..., q_g\}.$$

Любые две вершины $x_i, x_j \in X$, находящиеся в различных областях $q_s, q_i \in Q$, не могут быть соединены ребром $u_k \in U$ без пересечения ребер (соединений), ограничивающих области q_s и q_t.

В связи с этим возникает необходимость контроля непопадания смежных вершин в различные изолированные области.

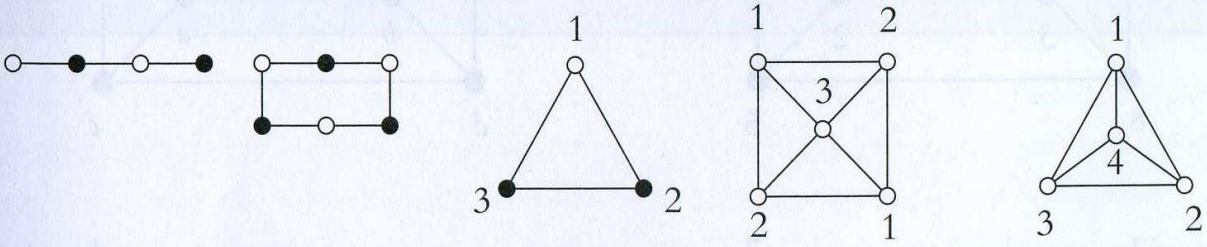
Цикломатическое число v (G) позволяет определить число таких локально замкнутых областей и перейти к решению задачи рационального перераспределения ребер графа G (X, U).

Хроматическое число

Раскраска вершин $V(G)$ графа G множеством цветов C состоит в присвоении каждой вершине графа цвета из множества C таким образом, что смежные вершины будут окрашены в разные цвета. Хроматическое число $X(G)$ графа G определяется так: это минимальное количество цветов, в которое можно раскрасить вершины графа G так, чтобы любые смежные вершины имели разные цвета.

Если G имеет как минимум одно ребро, то $X(G)$ будет больше либо равно 2. Очевидно, что $X(G)$ не может быть больше числа вершин V (граничным случаем будет раскраска каждой вершины в свой цвет). Разумеется, хроматическое число является инвариантом, так как полностью эквивалентные (изоморфные) графы имеют одинаковое хроматическое число.

Рассмотрим следующие графы:



Если n вершин графа расположены в одну линию, его хроматическое число равно 2, так как для его раскраски будет достаточно чередовать цвета. Это же справедливо и для любого дерева. Если же все вершины графа образуют цикл и число вершин четно, то хроматическое число такого графа равно 2. Если же число вершин нечетно, то хроматическое число равно 3. И наконец, если граф является колесом (граф с n вершинами, $(n-1)$-я вершина которого принадлежит простому циклу, а одна вершина вне этого цикла смежна со всеми остальными), то его хроматическое число равно 3, если внешний цикл состоит из четного числа вершин. Если же это число нечетное, хроматическое число будет равно 4.

В отличие от цикломатического числа определение хроматического числа осуществляется с помощью сравнительно сложных алгоритмов, в основу большинства которых положены методы целочисленного линейного программирования.

Оценка хроматического числа через число вершин графа G (X, U) имеет вид:

$$1 \leqslant k(G) \leqslant |X| = n.$$

В этом выражении нижняя граница соответствует пустым, а верхняя – полным графам.

На практике для простых связных графов оценить величину k (G) можно следующим образом.

Сначала выбираем вершину с минимальной локальной степенью и пометим (раскрасим) ее, затем произведем хроматическую раскраску вершин, смежных с данной, и так далее.

Знание хроматического числа графа позволяет в ряде случаев упростить алгоритмы, используемые на этапе конструкторского проектирования РЭС.

Пример. Рассмотрим задачу оценки числа слоев многослойной печатной платы. Пусть, граф G (X, U) интерпретирует фрагмент коммутационного поля платы, где x – множество областей контактных площадок конструктивных элементов, внутри которых проведение проводников запрещено, а U – множество трасс платы.

Построим граф пересечений Q (Y, V), в котором вершинами являются отдельные компоненты связности (электрические цепи) графа G (X, U), причем y_i, $y_j \in Y$ смежны, если соответствующие им компоненты связности $G_i (X_i , U_i)$ и $G_j (X_j , U_j)$ перекрывают друг друга.

При этом хроматическое число k (Q) графа Q (Y, V) определит верхнюю границу числа коммутационных слоев рассматриваемого фрагмента платы (в нашем примере k (Q) = 2). В этом случае цепи, соответствующие вершинам одного цвета графа Q (Y, V), можно располагать в одном слое печатной платы.

Приложение 4

Плоские (планарные) графы

При проектировании электрических соединений печатных плат к печатному монтажу предъявляются требования полного отсутствия или минимального числа пересечений проводников при однослойном (многослойном) печатном монтаже. Будем интерпретировать рисунок электрических соединений графом, вершины которого соответствуют контактным площадкам и переходам из слоя в слой, а ребра — печатным проводникам платы. Для выполнения указанных требований необходимо, чтобы граф (частичные графы, представляющие отдельные слои коммутаций многослойных печатных плат) полностью размещался на плоскости, а его ребра пересекались только в вершинах при минимизации числа вершин, соответствующих контактным переходам, т. е. граф (частичные графы) должен быть плоским. В связи с этим задача определения планарности графа и построения его плоского изображения приобретает особое значение.

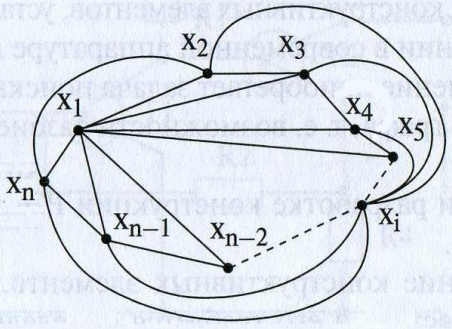
Граф G (X, U) называют *плоским (планарным)* тогда и только тогда, когда он имеет геометрическую реализацию в двухмерном евклидовом пространстве, т. е. может быть расположен на плоскости так, что все его ребра пересекаются только в вершинах X графа.

Геометрическая реализация графа в евклидовой плоскости, если она вообще возможна, возможна лишь при определенном расположении его вершин и ребер на плоскости.

Критерии планарности графов

Прежде всего, определим максимально возможное число ребер плоского графа.

Пусть задан граф G (X, U), имеющий гамильтонов цикл. С помощью изоморфных преобразований перейдем к графу G' (X, U), в котором ребра гамильтонова цикла не пересекаются. Тогда во внутренней и внешней областях выпуклой фигуры, образованной ребрами гамильтонова цикла G' (X, U), можно пронести без пересечений не более (n – 3) ребер.



Следовательно, максимальное число некратных ребер у плоского графа

$$r_{max} = n + (n - 3) + (n - 3) = 3\,(n - 2)\,.$$

Кроме того, если известно, что число некратных ребер графа

$$r \leqslant n + 2,$$

то такой граф заведомо плоский.

Таким образом, можно записать следующие условия для предварительного исследования планарности графа:

$r > 3\,(n - 2)$ – граф заведомо неплоский;

$r \leqslant n + 2$ – граф заведомо плоский.

5.2. АЛГОРИТМЫ МАРШРУТИЗАЦИИ

Задача определения оптимального по заданному критерию маршрута (кратчайшего пути) решается с применением алгоритмов маршрутизации, реализуемых протоколами маршрутизации. Вообще говоря, задача поиска кратчайшего пути формализуется как задача линейного программирования. Однако для решения данной задачи разработаны специальные алгоритмы. В частности, широкое применение находят уже упоминавшиеся алгоритмы Дейкстры и Беллмана — Форда, которые позволяют вычислить кратчайшие пути от заданного узла до всех узлов сети. В качестве метрики при определении кратчайшего пути, как указывалось выше, могут выступать задержка, пропускная способность и др. В результате работы алгоритма маршрутизации в маршрутизаторе создается таблица маршрутизации, в которой хранятся маршрутные записи, содержащие информацию о маршрутах с указанием соответствующих им метрик.

5.2.1. АЛГОРИТМ ДЕЙКСТРЫ

Алгоритм, предложенный нидерландским ученым Э. Дейкстрой в 1959 г., позволяет находить кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов, не имеющих ребер с отрицательной метрикой (весом).

Работа алгоритма происходит поэтапно путем последовательной обработки всех вершин графа, начиная с вершины-источника и вычисления при этом кратчайших расстояний до k узлов за k шагов. В процессе выполнения алгоритма при переходе от узла i к узлу j используется специальная процедура пометки вершин графа. Метки бывают *временные* и *постоянные* и представляют собой метрику пути до данной вершины. Если узлу присвоена постоянная метрика, то это означает что данный путь до узла от вершины-источника — кратчайший. После того, как все вершины графа обработаны, и все метки стали постоянными, считается, что все кратчайшие пути от узла-источника до остальных узлов найдены, и алгоритм прекращает свою работу.

Пусть $G(V,E)$ — граф, описывающий моделируемую сеть, характеризуемую множеством узлов V и множеством ребер E; T — множество вершин графа, уже обработанных алгоритмом; c_{ij} — длина ребра между i-м (υ_i) и j-м (υ_j) смежными узлами ($c_{ij} \geq 0$). Метка j-го узла (u_j, i) определяется как

$$(u_j, i) = (u_i + c_{ij}, i),$$

где u_i — кратчайшее расстояние от узла-источника до i-го узла.

Формально алгоритм представляется в виде следующей последовательности этапов (2-й этап повторяется до окончания работы алгоритма).

1. Первый этап — инициализация

Исходное множество узлов состоит из одной вершины-источника $T = \{\upsilon_i\}$. Метка самой вершины υ_i полагается постоянной и равной $(0, -)$, метки остальных вершин — временные и равные бесконечности $(\infty, -)$. Это отражает то, что метрики от υ_i до других вершин пока неизвестны. Все вершины графа помечаются как необработанные.

2. Второй этап

2.1. Вычисляются временные метки

$$(u_j, i) = (u_i + c_{ij}, i)$$

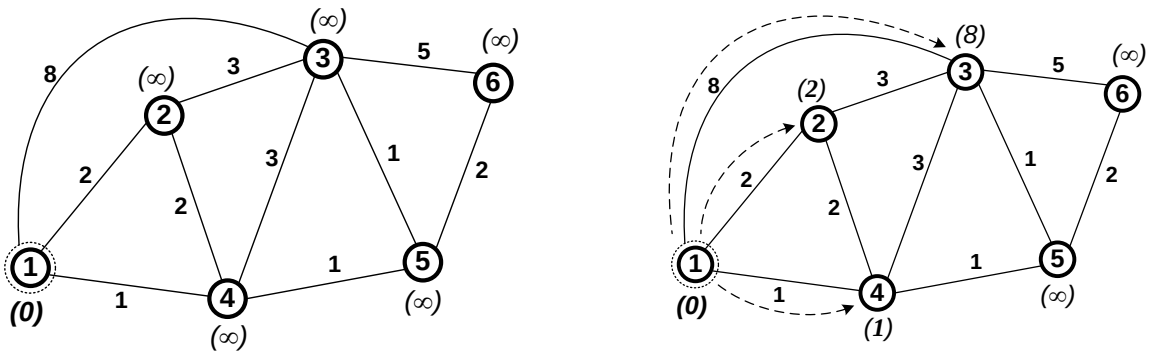
для всех узлов (соседей), которых можно достичь непосредственно из вершины υ_i и которые не имеют постоянных меток. Далее, если узел υ_j уже имеет временную метку (u_j, k), полученную от узла υ_k, то при $u_i + c_{ij} < u_j$ (т. е. когда значение новой метки меньше, чем старой) метка (u_j, k) заменяется на (u_j, i). В противном случае, когда $u_i + c_{ij} > u_j$, метка не меняется.

2.2. Узел υ_i отмечается как обработанный и добавляется во множество T, а его метка становится постоянной.

а). Если все узлы имеют постоянные метки, то процесс вычислений заканчивается. Или, что то же самое, когда $T = V$, т. е. когда все вершины графа обработаны и попали во множество T.

б). В противном случае из ещё не обработанных узлов (не имеющих постоянной метки) выбирается тот, который имеет минимальное значение u_r временной метки (u_r, s). Напомним, что u_r — это метрика пути от узла-источника до r-го узла через предпоследний узел υ_s на этом пути. Далее полагаем $i = r$ и возвращаемся ко второму шагу.

Рассмотрим работу алгоритма Дейкстры на примере сети на рис. 5.5. Здесь $V = \{\upsilon_1, \upsilon_2, ..., \upsilon_6\}$. Будем искать кратчайшие пути из узла 1 ко всем остальным узлам сети. В кружках обозначены номера вершин, над ребрами обозначена их «стоимость» — метрика ребра. Рядом с каждой вершиной в круглых скобках обозначается временная метка, а в квадратных — постоянная (длина кратчайшего пути в эту вершину из вершины 1). Пунктиром обведен тот узел, который в данный момент выбран для обработки. Затенены уже обработанные вершины графа.



а) *б*)

Рис. 5.4. Инициализация (*а*) и вычисление меток пути (*б*) в алгоритме Дейкстры

1. Первый этап

При инициализации (рис. 5.4, *а*) 1-й узел υ_1 отмечается как обрабатываемый, ему присваивается метка $(0,-)$, остальным узлам — метки $(\infty,-)$.

2. **Второй этап**

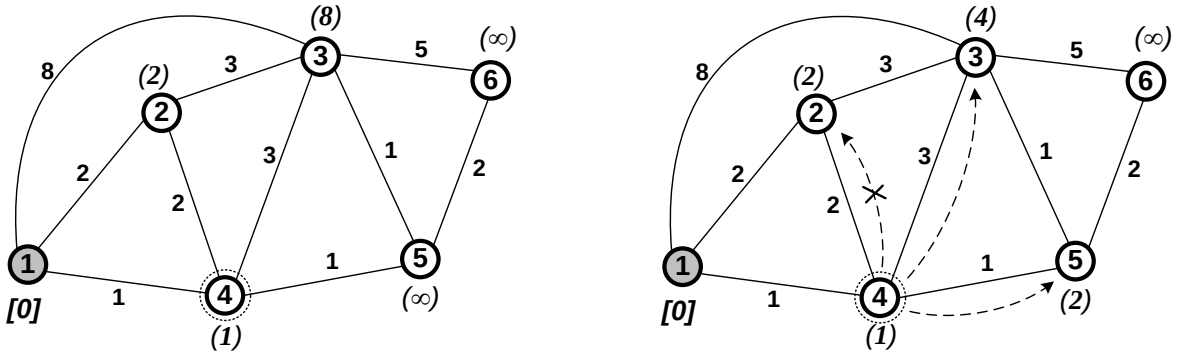
2.1. Определяем временные метки смежных с υ_1 узлов υ_2, υ_3 и υ_4. Они вычисляются как сумма значения метки 1-го узла и метрики ребра между υ_1 и соответствующим соседним узлом (рис. 5.4, *б*):

$$\text{для } \upsilon_2 \quad (u_2,1)=(0+2,1)=(2,1);$$

$$\text{для } \upsilon_3 \quad (u_3,1)=(0+8,1)=(8,1);$$

$$\text{для } \upsilon_4 \quad (u_4,1)=(0+1,1)=(1,1).$$

Так как все рассматриваемые узлы до этого имели значения меток, равные бесконечности, их метки заменяются на вычисленные $(u_2,1)$, $(u_3,1)$ и $(u_4,1)$.



в) *г*)

Рис. 5.4. Второй (*в*) и третий (*г*) шаги в алгоритме Дейкстры

2.2. 1-й узел υ_1 отмечается как обработанный (эта вершина на рис. 5.4, *в* затенена), а его метка приобретает постоянный статус (в квадратных скобках). В качестве следующего обрабатываемого узла выбирается 4-й узел υ_4, так как он имеет минимальную из оставшихся узлов временную метку (метрику пути из 1-го узла) $u_4 = 1$. Так как не всем узлам присвоены постоянные метки, на следующем шаге возвращаемся к началу второго этапа.

3. Второй этап (повторяется)

3.1. Определяем временные метки смежных с υ_4 узлов υ_2, υ_3 и υ_5 (рис. 5.4, *г*). Они вычисляются как сумма значения метки 4-го узла и метрики ребра между υ_4 и соответствующим соседним узлом:

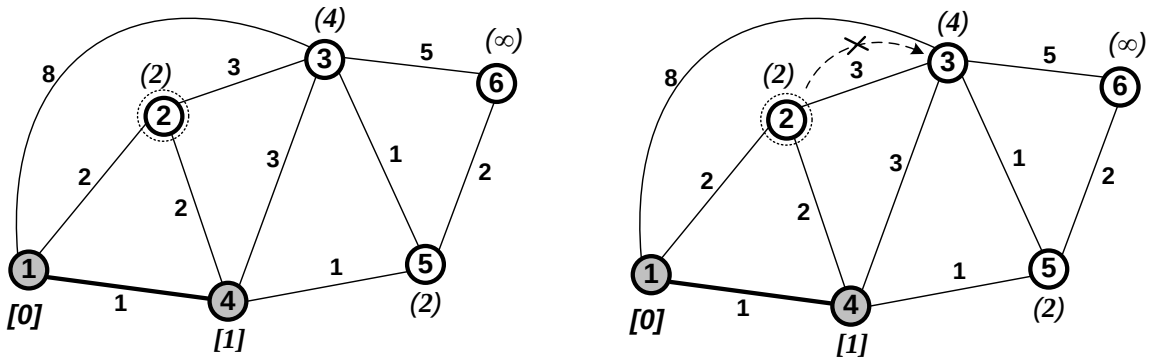
$$\text{для } \upsilon_2 \ \ (u_2, 4) = (1+2, 1) = (3, 1);$$

$$\text{для } \upsilon_3 \ \ (u_3, 4) = (1+3, 1) = (4, 1);$$

$$\text{для } \upsilon_5 \ \ (u_5, 4) = (1+1, 1) = (2, 1).$$

Значение текущей временной метки 2-го узла $(u_2, 1) = (2, 1)$ меньше вычисленной $(u_2, 4) = (3, 1)$, поэтому она не заменяется. Вычисленные значения меток для 3-го и 5-го узлов меньше текущих, поэтому метки этих узлов меняются на новые: $(u_3, 4) = (4, 1)$ и $(u_5, 4) = (2, 1)$.

3.2. 4-й узел υ_4 отмечается как обработанный (рис. 5.4д), а его метка приобретает постоянный статус (в квадратных скобках). Это означает, что найден кратчайший путь из 1-го узла во 4-й (это ребро e_{14}, оно выделено утолщенной линией). В качестве следующего обрабатываемого узла выбирается 2-й узел υ_2, так как он имеет минимальную из оставшихся узлов временную метку (метрику пути из 1-го узла) $u_2 = 2$. Так как не всем узлам присвоены постоянные метки, то на четвертом шаге возвращаемся в начало второго этапа.



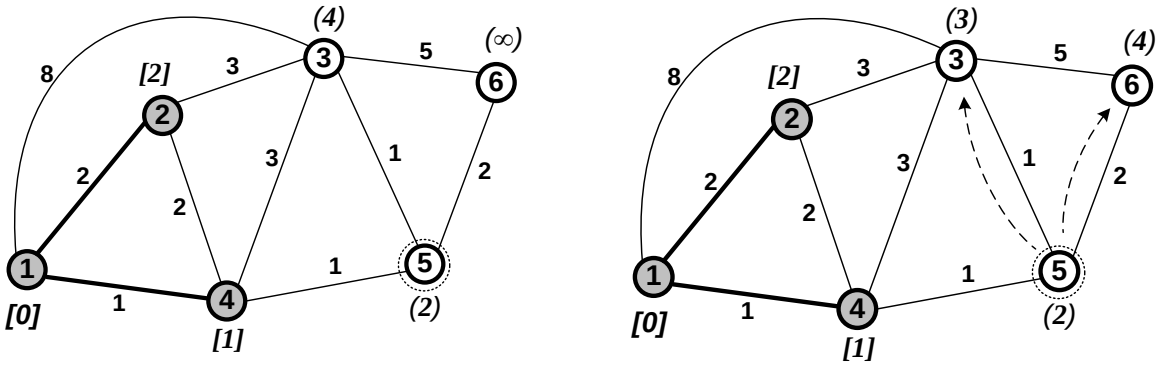
д) *е*)

Рис. 5.4. Третий (*д*) и четвёртый (*е*) шаги в алгоритме Дейкстры

4. Второй этап (повторяется)

4.1. Определяем временные метки соседних с υ_2 узлов, которым ещё не присвоены постоянные метки (рис. 5.4, *е*). Это 3-й узел υ_3: $(u_3, 2) = (2+3, 1) = (5, 1)$. Значение вычисленной метки $(u_3, 2)$ больше значения текущей метки 3-го узла, поэтому замена метки не осуществляется: $(u_3, 4) = (4, 1)$.

4.2. 2-й узел υ_2 отмечается как обработанный (рис. 5.4, *ж*), а его метка становится постоянной. Это означает, что найден кратчайший путь из 1-го узла во 2-й (это ребро e_{12}, оно выделено утолщённой линией). В качестве следующего обрабатываемого узла выбирается 5-й узел υ_5, так как он имеет минимальную из оставшихся узлов временную метку (метрику пути из 1-го узла) $u_5 = 2$. Так как не всем узлам присвоены постоянные метки, на пятом шаге возвращаемся ко второму этапу.



ж) *з*)

Рис. 5.4. Четвёртый (*ж*) и пятый (*з*) шаги в алгоритме Дейкстры

5. Второй этап (повторяется)

5.1. Определяем временные метки соседних с υ_5 узлов, которым ещё не присвоены постоянные метки (рис. 5.4, *з*). Это 3-й и 6-й узлы:

$$\text{для } \upsilon_3\text{: } (u_3, 5) = (2+1, 1) = (3, 1);$$

$$\text{для } \upsilon_6\text{: } (u_6, 5) = (2+2, 1) = (4, 1).$$

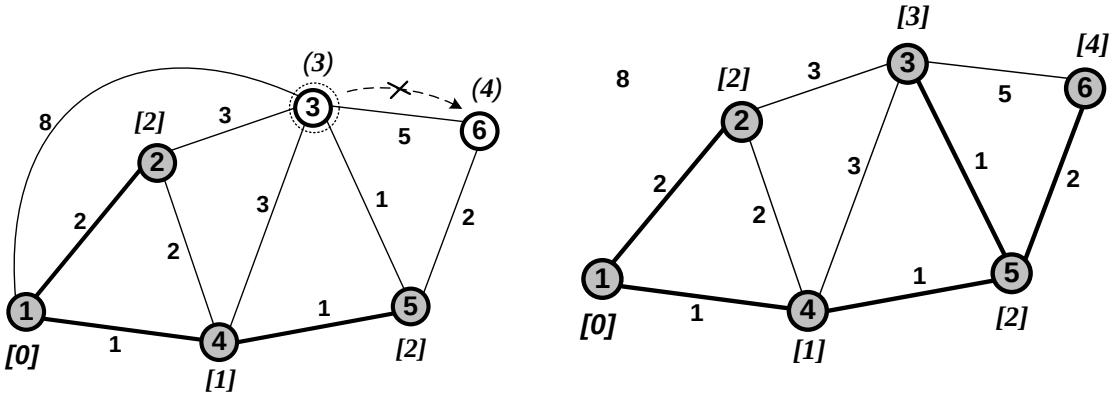
Вычисленные значения меток для 3-го и 6-го узлов меньше текущих. Поэтому метки этих узлов меняются на новые: $(u_3, 5) = (3, 1)$ и $(u_6, 5) = (4, 1)$.

5.2. 5-й узел υ_5 отмечается как обработанный (рис. 5.4, *и*), а его метка становится постоянной. Это означает, что найден кратчайший путь из 1-го узла во 5-й (это путь $\upsilon_1 - \upsilon_4 - \upsilon_5$, включающий ребра e_{14} и e_{45}). В качестве следующего обрабатываемого узла выбирается 3-й узел υ_3, так как из

оставшихся не обработанными узлов υ_3 и υ_6 он имеет минимальную временную метку: $(u_3, 5) = (3, 1)$, значение метки – $u_3 = 3$. Так как не всем узлам присвоены постоянные метки, на шестом шаге возвращаемся ко второму этапу.

6. Второй этап (повторяется)

6.1. Определяем временные метки соседних с υ_3 узлов, которым ещё не присвоены постоянные метки (также иллюстрируется на рис. 5.4, *и*). Это 6-й узел υ_6: $(u_6, 3) = (3 + 5, 1) = (8, 1)$. Значение вычисленной метки $(u_6, 3)$ больше значения текущей метки 3-го узла, поэтому замена метки не осуществляется: $(u_6, 5) = (4, 5)$.



и) *к*)

Рис. 5.4. Пятый (*и*) и шестой (*к*) шаги в алгоритме Дейкстры

6.2. 3-й узел υ_3 отмечается как обработанный (рис. 5.4, *к*), а его метка становится постоянной. Это означает, что найден кратчайший путь из 1-го узла во 3-й (это путь $\upsilon_1 - \upsilon_4 - \upsilon_5 - \upsilon_3$, включающий ребра e_{14}, e_{45} и e_{53}). В качестве следующего обрабатываемого узла выбирается один не обработанный 6-й узел υ_6. Так как у него нет соседей с временными метками, статус его временной метки меняется на постоянный (рис. 5.4, *к*). Кратчайшим путем 1-го узла в 6-й является путь $\upsilon_1 - \upsilon_4 - \upsilon_5 - \upsilon_6$. Все вершины графа обработаны, и на этом алгоритм заканчивает работу.

В рассмотренном примере за шесть шагов алгоритм Дейкстры нашел все кратчайшие пути от 1-го узла до остальных пяти узлов.

Приведенный выше пример работы алгоритма Дейкстры можно представить в табличной форме (табл. 5.2). Жирным шрифтом выделены кратчайшие пути, находимые алгоритмом на каждом шаге работы. Значения постоянных меток узлов (метрик путей до этих узлов), как и ранее, обозначены в квадратных скобках. Из таблицы видно, что на каждом шаге, начиная со второго, множество *T* обработанных узлов увеличивается на один узел, и, соответственно, находится один кратчайший путь до какого-либо узла.

Таблица 5.2

Пример пошаговой работы алгоритма Дейкстры

Шаг	Множество T	Метрика пути/маршрут от узла υ_1 до узлов									
		υ_2		υ_3		υ_4		υ_5		υ_6	
1	$\{\upsilon_1\}$	2	1-2	8	1-3	1	1-4	—	—	—	—
2	$\{\upsilon_1,\upsilon_4\}$	2	1-2	4	1-4-3	[1]	**1-4**	2	1-4-5	—	—
3	$\{\upsilon_1,\upsilon_4,\upsilon_2\}$	[2]	**1-2**	4	1-4-3	[1]	**1-4**	2	1-4-5	—	—
4	$\{\upsilon_1,\upsilon_4,\upsilon_2,\upsilon_5\}$	[2]	**1-2**	3	1-4-5-3	[1]	**1-4**	[2]	**1-4-5**	4	1-4-5-6
5	$\{\upsilon_1,\upsilon_4,\upsilon_2,\upsilon_5,\upsilon_3\}$	[2]	**1-2**	[3]	**1-4-5-3**	[1]	**1-4**	[2]	**1-4-5**	4	1-4-5-6
6	$\{\upsilon_1,\upsilon_2,\upsilon_3,\upsilon_4,\upsilon_5,\upsilon_6\}$	[2]	**1-2**	[3]	**1-4-5-3**	[1]	**1-4**	[2]	**1-4-5**	[4]	**1-4-5-6**

Время работы алгоритма Дейкстры пропорционально $|V|^2$, так как алгоритм выполняет $|V|-1$ итераций, а количество операций, выполняемых на каждой итерации, пропорционально $|V|$.

5.2.2. АЛГОРИТМ БЕЛЛМАНА — ФОРДА

Алгоритм Беллмана — Форда был использован в 1969 г. в протоколе маршрутизации RIP сети ARPANET. Он, как и алгоритм Дейкстры, находит кратчайшие пути от одной вершины графа до всех остальных, но в отличие от последнего алгоритм Беллмана-Форда допускает наличие в графе рёбер с отрицательным весом.

Этот алгоритм также работает поэтапно. Но идея его состоит в том, что сначала находятся кратчайшие пути от заданной вершины, состоящие из одного ребра, затем находятся кратчайшие пути при условии, что они состоят максиму из двух рёбер, затем — максимум из трёх и т. д.

Пусть по-прежнему $G(V,E)$ — граф, описывающий моделируемую сеть; $c_{ij}=c(i,j)$ — метрика ребра между i-м (υ_i) и j-м (υ_j) смежными узлами ($c_{ij}=\infty$, если вершины υ_i и υ_j не соединены напрямую); h — максимальное количество рёбер в пути на текущем шаге алгоритма; $L_h(i)$ — минимальная метрика пути от узла-источника s до узла υ_i.

Алгоритм состоит из двух этапов. Этап 2 повторяется до тех пор, пока

метрики путей не перестанут изменяться, но для значений h не более $h \le |V| - 1$, поскольку кратчайший путь не может содержать большее число ребер, иначе он будет содержать цикл, который точно можно выкинуть.

1. Этап 1 — инициализация

$L_0(i) = \infty$ для всех $i \neq s$; $L_h(s) = 0$ для всех h.

2. Этап 2.

Вычисление для каждого узла *j* метрик минимальных путей до узла-источника *s*, включающих не более h рёбер (начиная с $h = 0$ и далее при повторении 2-го этапа: $h = 1, 2, \ldots$):

$$L_{h+1}(\mathrm{j}) = \min_k \{L_h(k) + c(k, j\}. \qquad (5.1)$$

Согласно (5.1) находится проходящий через соседний с узлом *j* узел *k* путь, состоящий из $h+1$ рёбер и имеющий минимальную метрику среди всех возможных соседних узлов *k*, пути $L_h(k)$ до которых от вершины-источника s были найдены на предыдущей итерации. При $h = k$ для каждой вершины *j* алгоритм сравнивает путь от вершины-источника *s* до вершины *j* длиной $k+1$ с путем, существовавшим к концу предыдущей итерации. Если предыдущий путь имеет меньшую метрику, то он сохраняется. В противном случае сохраняется новый путь от вершины-источника *s* до узла *j* длиной $k+1$.

В табл. 5.3 и на рис. 5.5 показано применение этого алгоритма для поиска кратчайших путей от 1-го узла υ_1 до всех остальных узлов графа.

1. Первый этап

При инициализации метрики путей до всех узлов от вершины-источника (от 1-го узла) полагаются равными бесконечности:

$$L_0(i) = \infty,\ i \neq 1,\ i = \overline{2, |V|}.$$

2. Второй этап

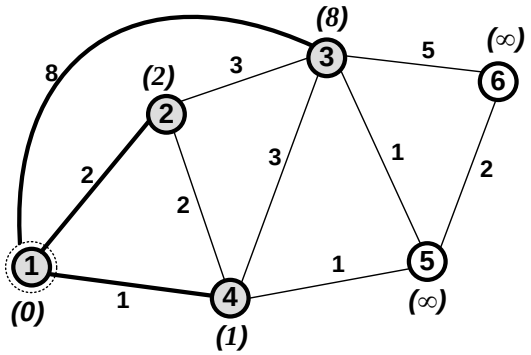
При первой итерации вычисляются метрики путей, состоящих из одного ребра (при $h = 1$), от 1-го узла до смежных узлов. Результат показан на рис. 5.5, *а* и в соответствующей строке табл. 5.3.

Таблица 5.3

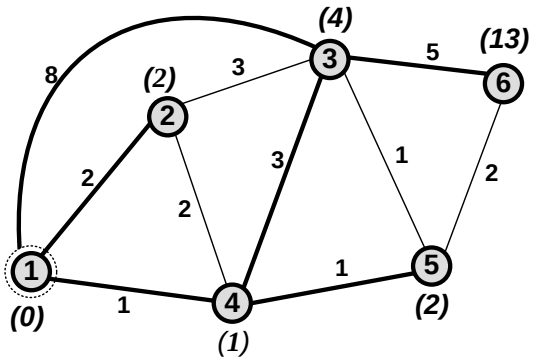
Пример пошаговой работы алгоритма Беллмана — Форда

h	$L_h(2)$	Путь	$L_h(3)$	Путь	$L_h(4)$	Путь	$L_h(5)$	Путь	$L_h(6)$	Путь
0	∞	—	∞	—	∞	—	∞	—	∞	—
1	2	1–2	8	1–3	1	1–4	∞	—	∞	—
2	2	1–2	4	1–4–3	1	1–4	2	1–4–5	10	1–3–6
3	2	1–2	3	1–4–5–3	1	1–4	2	1–4–5	4	1–4–5–6
4	2	1–2	3	1–4–5–3	1	1–4	2	1–4–5	4	1–4–5–6

На рис. 5.5, *а* в круглых скобках рядом с вершинами графа записаны значения метрик минимальных путей на каждом шаге от 1-го узла до данного узла.

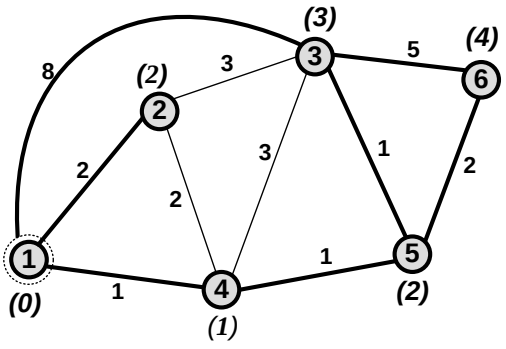


а) $h = 1$

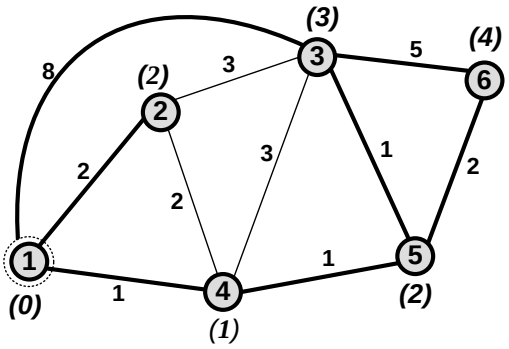


б) $h = 2$

Рис. 5.5. Пошаговая работа алгоритма Беллмана — Форда



в) $h = 3$



г) $h = 4$

Рис. 5.5. Пошаговая работа алгоритма Беллмана — Форда

Далее этап 2 повторяется для значений $h = 2$, 3 и 4. На каждом шаге алгоритм находит путь с минимальной метрикой, максимальное число ретрансляционных участков в котором не превышает h. Результаты после каждой итерации показаны на рис. 5.5, *б–г* и в соответствующих строках табл. 5.3. В конечном итоге определяются кратчайшие пути от 1-го узла до всех остальных (последняя строка табл. 5.3), и построено так называемое *остовное дерево* (рис. 5.5, *г*).

Время работы алгоритма Беллмана-Форда пропорционально $|V| \times |E|$, так как алгоритм выполняет $|V| - 1$ итераций, а каждая итерация включает определение веса каждого ребра.