Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

«Глубокие самообучающиеся агенты для мультиагентной системы маршрутизации»

Автор: Мухутдинов Дмитрий Вадимович _____
Направление подготовки (специальность): 01.03.02 Прикладная математика и информатика
Квалификация: Бакалавр
Руководитель: Фильченков А.А., к.ф.-м.н _____

К защите допустить
Зав. кафедрой Васильев В.Н., докт. техн. наук, проф. _____

«____» _____ 20_____ г.

Санкт-Петербург, 2017 г.

Студент Мухутдинов Д.В. **Группа** М3438 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования

Направленность (профиль), специализация Математические модели и алгоритмы разработки программного обеспечения

Консультанты:

а) Вяткин В.В., PhD, Luleå University of Technology _____

Квалификационная работа выполнена с оценкой _____

Дата защиты «____» _____ 20_____ г.

Секретарь ГЭК _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Задача пакетной маршрутизации - это задача поиска кратчайшего пути в графе в условиях, когда за каждый узел графа отвечает отдельный вычислительный процесс. Это означает, что каждый отдельный узел графа должен принять решение о том, какому из соседей следует отправить очередной пакет, чтобы тот достиг пункта назначения как можно быстрее.

Задача пакетной маршрутизации впервые обрела актуальность с появлением компьютерных сетей. Первые алгоритмы сетевой маршрутизации появились в процессе разработки сети ARPANet. Именно тогда были изобретены такие подходы к пакетной маршрутизации, как distance-vector[1] и link-state[2], которые и по сей день лежат в основе таких стандартных и широко применяемых алгоритмов сетевой маршрутизации, как Routing Information Protocol (RIP)[3] и Open Shortest Path First (OSPF)[4].

Оба этих подхода основаны на идее вычисления кратчайшего пути между текущим узлом сети и пунктом назначения пакета. Однако существуют и другие подходы к решению задачи маршрутизации, основанные на идее обучения с подкреплением (reinforcement learing). Первым таким алгоритмом стал алгоритм Q-routing[5], основанный на методе Q-learning[6]. Этот алгоритм, как и его модификации[7, 8], благодаря обучению с подкреплением оказался способен лучше адаптироваться к изменениям в интенсивности сетевого трафика, чем алгоритмы, основанные на вычислении кратчайшего пути, но из-за использования большего количества служебных сообщений применение таких алгоритмов в реальных сетях ограничено.

Но стоит отметить, что задача маршрутизации встречается не только в компьютерных сетях, но также и в более сложных средах, таких как киберфизические системы. Примером такой задачи, частично решаемой с помощью алгоритмов маршрутизации, является управление конвейерной системой (в частности, системой распределения багажа в аэропорту). Задача управления такой системой в большинстве случаев решается с помощью централизованных алгоритмов, и децентрализованное решение (основанное на алгоритма Беллмана-Форда) было пред-

ложено совсем недавно[9]. Такая задача отличается от задачи сетевого роутинга, с одной стороны, тем, что служебные сообщения и целевые сообщения являются разными сущностями (“целевое сообщение” — это чемодан, а служебные сообщения являются цифровыми), и служебные сообщения передаются мгновенно по сравнению с целевыми. С другой стороны, состояние каждого конвейера и всей системы в целом задается большим количеством параметров — такими как скорости конвейеров, положение, количество и масса чемоданов на каждом конвейере, и так далее. С третьей стороны, желательно, чтобы система оптимизировала не только скорость доставки чемоданов до точки назначения, но и, к примеру, собственное энергопотребление.

Первое обстоятельство снимает технические ограничения на количество служебных сообщений, что позволяет в полной мере применять алгоритмы на основе обучения с подкреплением. Второе и третьи обстоятельства усложняют написание оптимального детерминированного алгоритма и наталкивают на идею реализации приближенного решения, например, с использованием нейронных сетей.

В настоящее время в области обучения с подкреплением с применением нейронных сетей достигнуты впечатляющие успехи. Всплеск активности в этой области произошел после выхода статьи команды Google DeepMind об обучении глубокой сверточной нейронной сети игре на консоли Atari 2600[10]. Применению полученной модели к различным задачам в различных условиях посвящено множество исследований, часть из них посвящена проблеме мультиагентного обучения с подкреплением (multi-agent reinforcement learning). Так как задачу маршрутизации можно сформулировать как задачу мультиагентного обучения с подкреплением, имеет смысл применить данные наработки для ее решения.

В данной работе будет предложен алгоритм маршрутизации, основанный на методе Q-routing, но использующий нейронную сеть в качестве обучающегося агента. На данный момент не существует алгоритма маршрутизации, построенного по такому принципу.

В главе 1 будет сформулирована обобщенная постановка задачи маршрутизации в терминах мультиагентного обучения с подкреплени-

ем. Будут рассмотрены существующие алгоритмы маршрутизации, их сильные и слабые стороны. Также будут рассмотрены существующие методы обучения нейронных сетей с подкреплением, в том числе в мультиагентном случае.

В главе 2 будет рассмотрен предложенный алгоритм и обоснованы решения, принятые в ходе его разработки.

В главе 3 будут приведены экспериментальные результаты работы алгоритма для задач пакетной маршрутизации в компьютерной сети и управления системой багажных конвейеров. Будут приведены результаты работы в условиях неравномерной нагрузки на сеть (или конвейерную систему) и изменения топологии сети. Также будет проведено сравнение с существующими алгоритмами маршрутизации.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Постановка задачи

Пусть задана сеть в виде графа: $G = (V, E)$. Каждому узлу и каждому ребру в сети приписывается некоторое *состояние*: узел v имеет состояние $s_v \in S_V$, ребро e имеет состояние $s_e \in S_E$. Состояние всей сети, таким образом, описывается как $s = (s_{v_1}, ..., s_{v_n}, s_{e_1}, ..., s_{e_m}) \in S$.

На узел сети $s \in V$ приходят пакеты $p = (d, prop)$, где $d \in V$ - узел назначения пакета, а $prop$ - произвольная дополнительная информация о нем. Обозначим множество всевозможных свойств пакета как $Prop$ ($prop \in Prop$), а множество пакетов в целом, таким образом, как $P = (V, Prop)$. Также заданы некоторые функции *стоимости* прохождения данного пакета через ребра и узлы сети, зависящие от их состояний: $R_v : V \times S_V \times P \to \mathbb{R}^+$ - стоимость прохождения пакета через узел, $R_e : E \times S_E \times P \to \mathbb{R}^+$. Мы явно указываем, что стоимости неотрицательны, чтобы избежать таких постановок задач, в которых существуют пути стоимости $-\infty$ (циклы отрицательной стоимости).

Задача пакетной маршрутизации заключается в том, чтобы определить, какому из соседей $n \in \{v|(s, v) \in E\}$ узел s должен перенаправить пакет $p = (d, prop)$, чтобы ожидаемая стоимость пути от s до d была минимальной.

1.2. Существующие алгоритмы маршрутизации

Почти все существующие алгоритмы маршрутизации были разработаны для маршрутизации пакетов в компьютерных сетях. Большинство алгоритмов маршрутизации в компьютерных сетях, применяемых на практике, относятся к одному из двух семейств алгоритмов — дистанционно-векторные (distance-vector)[1] или состояния каналов связи (link-state)[2]. Концептуально все алгоритмы внутри каждого из этих семейств одинаковы, и различаются только техническими деталями реализации, обусловленными спецификой конкретной узкой сферы применения. Поэтому мы не будем рассматривать алгоритмы каждого семейства по отдельности, а рассмотрим только концепции в целом.

1.2.1. Дистанционно-векторные алгоритмы

Идея дистанционно-векторных алгоритмов (distance-vector algoritms) заключается в следующем:

— Каждый маршрутизатор s в сети хранит таблицу, в которой для каждого другого узла сети d хранится следующая информация:
  – Предполагаемое кратчайшее расстояние от s до d
  – Сосед n, которому нужно отправить пакет, чтобы пакет прошел по кратчайшему пути до узла d.
— Периодически каждый маршрутизатор рассылает свою версию таблицы кратчайших расстояний всем своим соседям
— При получении вектора кратчайших расстояний от соседа маршрутизатор s сравнивает его поэлементно с текущей версией. Если оказывается, что наименьшая стоимость пути от соседа n до узла d, сложенная с оценкой стоимости ребра (s, d) меньше, чем наименьшая стоимость пути от s до d в текущей таблице, то значение в текущей таблице обновляется, и наилучшим соседом для отправки пакета в узел d становится сосед n.

Как можно видеть, дистанционно-векторный алгоритм является, в сущности, распределенной версией алгоритма Беллмана-Форда поиска кратчайшего пути в графе[11].

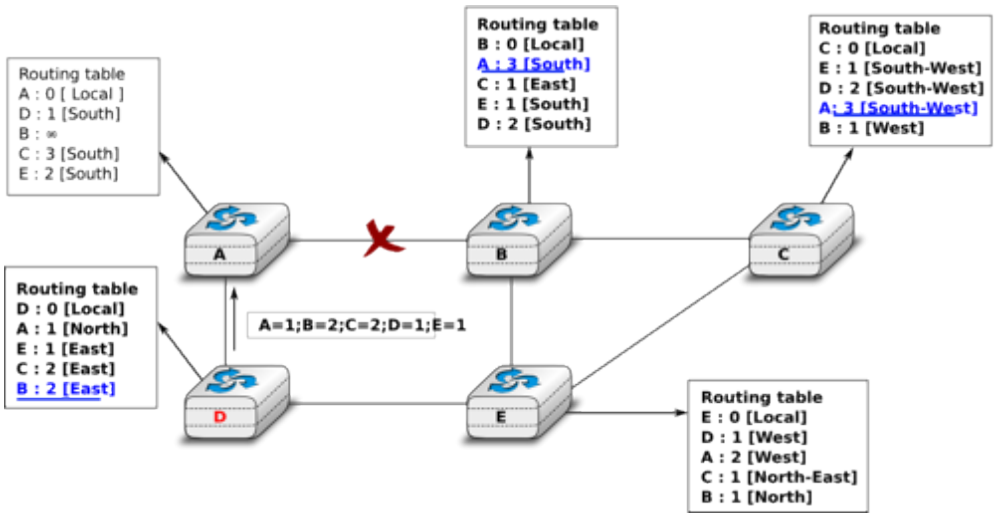

Рисунок 1 – Иллюстрация работы distance-vector алгоритма

Различные реализации дистанционно-векторного метода различаются, в частности, оценками стоимости соединений в сети. Так, например, протокол RIP[3] просто оценивает стоимость каждого соединения в 1, а IGRP[12] оценивает стоимость соединений исходя из оценок задержки и пропускной способности.

Преимуществами дистанционно-векторных алгоритмов являются простота реализации и низкие требования к памяти и вычислительной мощности. Недостатками же являются низкая скорость распространения информации по сети и сложности с приспособлением под изменяющуюся топологию (проблема count-to-infinity). Этих проблем удается избежать при применении другого распространенного подхода - алгоритмов на основе состояния канала связи.

1.2.2. Алгоритмы состояния канала связи

В отличие от дистанционно-векторных алгоритмов, в алгоритмах состояния канала связи (link-state) каждый узел сети хранит у себя модель всей сети в виде графа. Рассмотрим шаги алгоритма подробнее:

— Каждый маршрутизатор периодически проверяет состояние соединений до соседей
— При обнаружении обрыва какого-либо соединения алгоритм удаляет это соединение из собственного графа и рассылает соседям новую версию состояния соединений до них
— Соседи обновляют собственные версии графов в соответствии с полученной информацией и пересылают сообщение дальше
— Чтобы избежать зацикливания сообщений об обновлении состояния, каждое сообщение снабжается *номером версии*. Маршрутизатор n игнорирует сообщение от маршрутизатора n, если номер версии этого сообщения меньше или равен предыдущему.

Имея информацию обо всей сети в целом, маршрутизатор может рассчитать кратчайшие пути до всех остальных узлов. Обычно для этого используется алгоритм Дейкстры[13].

Link-state алгоритмы обладают способностью адаптироваться под изменения топологии сети гораздо быстрее, чем distance-vector алгоритмы за счет довольно несколько более сложной реализации и чуть больших затрат по памяти и вычислительной мощности. Это обуславливает то, что на данный момент именно link-state протоколы, такие как OSPF[4], доминируют в сетевой маршрутизации. Однако даже в решении задачи сетевой маршрутизации link-state алгоритмы в чистом виде не лучшим образом адаптируются к повышению нагрузки в сети. Рассматриваемые в дальнейшем другие алгоритмы, основанные на принципе

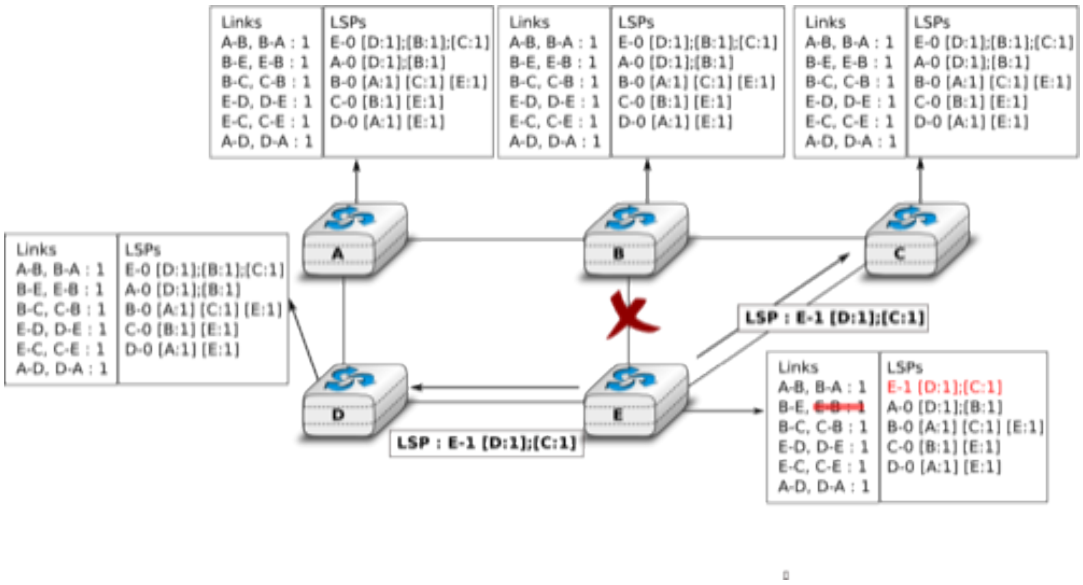

Рисунок 2 – Иллюстрация работы link-state алгоритма

обучения с подкреплением, справляются с задачей адаптации к изменчивой нагрузке лучше.

1.2.3. Другие подходы

Среди других подходов особый интерес представляют подходы на основе обучения с подкреплением. Первым алгоритмом маршрутизации, основанным на этой идее, стал алгоритм Q-routing[5]. Принцип его работы таков:

а) Каждый маршрутизатор x хранит $Q_x(d, y)$ — оценку минимального времени в пути до узла d, если следующим узлом на пути является сосед y. Очевидно, что $\forall y : Q_x(x, y) = 0$

б) Пакет, который необходимо доставить в узел d, отправляется соседу $y = \operatorname{argmin}_{(x,y)\in E} Q_x(d, y)$

в) При получении пакета узел y отправляет узлу x время получения t_r и собственную оценку оставшегося времени в пути $t = \min\limits_{(y,z)\in E} Q_y(d, z)$

г) Зная время отправления пакета t_s и получив t_r и t, узел x обновляет собственную оценку по формуле: $Q_x(d, y) = \alpha((t_r - t_s) + t - Q_x(d, y)) + Q_x(d, y)$, где α — это learning rate, параметр алгоритма.

1.3. Термины и понятия

Обучение с подкреплением - вид машинного обучения, в котором

а) Множества

б) Q-learning

в)

1.4. Постановка задачи в терминах обучения с подкреплением

1.5. Обзор методов обучения нейросетей с подкреплением

Таблица 1 – Таблица умножения (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Есть еще такое окружение `tabu`, его можно аккуратно растянуть на всю страницу. Приведем пример (таблица 2).

Таблица 2 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

1.6. Рисунки

Пример рисунка (с помощью `TikZ`) приведен на рисунке 3. Под `pdflatex` можно также использовать `*.jpg`, `*.png` и даже `*.pdf`, под `latex` можно использовать Metapost. Последний можно использовать и под `pdflatex`, для чего в стилевике продекларированы номера картинок от 1 до 20.

1.7. Листинги

В работах студентов кафедры «Компьютерные технологии» часто встречаются листинги. Листинги бывают двух основных видов — исходный код и псевдокод. Первый оформляется с помощью окружения

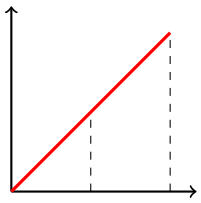

Рисунок 3 – Пример рисунка

`lstlisting` из пакета `listings`, который уже включается в стилевике и немного настроен. Пример Hello World на Java приведен на листинге 1.

Листинг 1 – Пример исходного кода на Java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

1.7.1. Тест

Псевдокод можно оформлять с помощью разных пакетов. В данном стилевике включается пакет `algorithmicx`. Сам по себе он не генерирует флоатов, поэтому для них используется пакет `algorithm`. Пример их совместного использования приведен на листинге 2. Обратите внимание, что флоаты разные, а нумерация — общая!

Листинг 2 – Пример псевдокода

function IsPrime(N)
    for $t \leftarrow [2; \lfloor\sqrt{N}\rfloor]$ **do**
        if $N \bmod t = 0$ **then**
            return false
        end if
    end for
    return true
end function

Наконец, листинги из `listings` тоже можно подвешивать с помощью `algorithm`, пример на листинге 3.

Листинг 3 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

ГЛАВА 2. ПРОВЕРКА СКВОЗНОЙ НУМЕРАЦИИ

Листинг 4 должен иметь номер 4.

Листинг 4 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Рисунок 4 должен иметь номер 2.

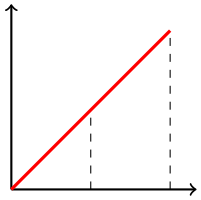

Рисунок 4 – Пример рисунка

Таблица 3 должна иметь номер 3.

Таблица 3 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Выводы по главе 2

В конце каждой главы желательно делать выводы. Вывод по данной главе — нумерация работает корректно, ура!

ЗАКЛЮЧЕНИЕ

TBD

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *McQuillan J. M.*, *Walden D. C.* The ARPA network design decisions // Computer Networks. — 1977. — 1(5). — С. 243–289.

2 *McQuillan J. M.*, *Richer I.*, *Rosen E. C.* The New Routing Algorithm for the ARPANet // IEEE Trans. on Comm. — 1980. — 28(5). — С. 711–719.

3 *Hendrick C.* Routing Information Protocol: RFC. — Июнь 1988. — № 1058.

4 *Moy J.* OSPF Version 2: RFC. — Апр. 1998. — № 1058.

5 *Boyan J. A.*, *Littman M. L.* Packet routing in dynamically changing networks: a reinforcement learning approach // Advances in Neural Information Processing Systems. — 1994. — No. 6. — P. 671–678.

6 *Watking C.* Learning from Delayed Rewards: дис. ... канд. / Watking C. — Cambridge : King's College, 1989.

7 *Choi S. P. . M.*, *Yeung D.-Y.* Predictive Q-Routing: A Memory-based Reinforcement Learning Approach to Adaptive Traffic Control // Advances in Neural Information Processing Systems. — 1996. — No. 8. — P. 945–951.

8 *Kumar S.*, *Miikkulainen R.* Dual reinforcement Q-routing: An on-line adaptive routing algorithm // Artificial Neural Networks in Engineering. — 1997. — No. 7. — P. 231–238.

9 *Yan J.*, *Vyatkin V.* Distributed Software Architecture Enabling Peer to Peer Communicating Controllers // IEEE Transactions on Industrial Informatics. — 2013. — 9(4). — P. 2200–2209.

10 Human-level control through deep reinforcement learning / V. Mnih [et al.] // Nature. — 2015. — No. 518. — P. 529–533.

11 *Bellman R.* On a routing problem // Quarterly of Applied Mathematics. — 1958. — № 16. — С. 87–90.

12 *Bosack L.* Method and apparatus for routing communications among computer networks. — Февр. 1992. — URL: `https://www.google.com/patents/US5088032` ; US Patent 5,088,032.

13 *Dijkstra E. W.* A note on two problems in connexion with graphs // Numerische Mathematik. — 1959. — № 1. — С. 269–271.

ПРИЛОЖЕНИЕ А. ПРИМЕР ПРИЛОЖЕНИЯ

В приложениях рисунки, таблицы и другие подобные элементы нумеруются по приложениям с соответствующим префиксом. Проверим это.

Листинг А.1 должен иметь номер А.1.

Листинг А.1 – Исходный код и флоат `algorithm`

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Рисунок А.1 должен иметь номер А.1.

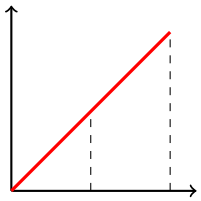

Рисунок А.1 – Пример рисунка

Таблица А.1 должна иметь номер А.1.

Таблица А.1 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68

Заодно проверим нумерованные и ненумерованные перечисления. Ненумерованные:

— пункт А;
— пункт Б;
— пункт В.

Нумерованные списки нескольких уровней:

а) первый элемент;
б) второй элемент с подэлементами:

1) первый подэлемент;
2) второй подэлемент;
3) третий подэлемент.

в) третий элемент;
г) четвертый элемент;
д) пятый элемент;
е) шестой элемент;
ж) седьмой элемент;
и) восьмой элемент;
к) девятый элемент;
л) десятый элемент.

ПРИЛОЖЕНИЕ Б. ЕЩЕ ОДИН ПРИМЕР ПРИЛОЖЕНИЯ С НЕИМОВЕРНО ДЛИННЮЩИМ НАЗВАНИЕМ ДЛЯ ТЕСТИРОВАНИЯ ПЕРЕНОСОВ

Проверим на примере таблиц, что нумерация в приложениях — по приложениям. Таблица Б.1 должна иметь номер Б.1.

Таблица Б.1 – Таблица умножения с помощью `tabu` (фрагмент)

–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68