

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ**

**Олимпиада школьников СПбГУ по  
математическому моделированию и  
искусственному интеллекту**

**Примеры заданий заключительного этапа  
2022/2023 учебный год**

**9-11 классы**

## Задача 1: Криптарифмы (20 баллов)

Автор задачи: Александр Кривошеин

Автор разбора: Александр Кривошеин

### Формулировка (Вариант 1)

Криптарифмом называют математическую головоломку, которая представляет собой арифметическое тождество, где каждая цифра заменена буквой (одинаковые цифры заменяются одинаковыми буквами, разные цифры заменяются разными буквами).

#### Пример:

$$AXA + OXO = CMEX$$

**Решением** криптарифма является такая подстановка цифр вместо букв, при которой получается верное равенство. При этом запрещается, чтобы первые буквы слов соответствовали нулям.

Например, решение криптарифма выше:  $A = 2, O = 7, X = 9, C = 1, M = 0, E = 8$  или  $292 + 797 = 1089$ .

Подстановка вида  $A = 1, O = 5, X = 6, C = 0, M = 7, E = 2$  или  $161 + 565 = 0726$  решением не является.

**Криптарифм называют правильным**, если он имеет только одно решение.

Пример выше не является правильным, так как существуют другие решения, например,  $494 + 595 = 1089$ .

Ниже представлено 40 криптарифмов. Также, для удобства, они представлены в файле “Var1.xls”

ЛОКОН+ЛОКОН=ВОЛОСЫ

ТРИ+ДВА=ПЯТЬ

ЛЮБА+ЛЮБИТ=АРБУЗЫ

АХИНЕЯ+АХИНЕЯ=ЧЕПУХА

ПЛЮС+ПЛЮС=МИНУС

КОРОЛЬ+КОРОНА=МОНАРХ

НОТА+НОТА=ТАКТ

РЮМКА+РЮМКА=АВАРИЯ

ЛАДЬЯ+ЛАДЬЯ=ФЕРЗЬ

ТРЮК+ТРЮК=ЦИРК

МОШКА+МОШКА=КОМАРЫ

БИТ+БАЙТ=СЛОВО

ВОЛК+ЛИСА=ЗВЕРИ

СИНИЦА+СИНИЦА=ПТИЧКИ

МУХА+МУХА=СЛОН

ОГОНЬ+ВОДА=ПОХОД

ГОРОД+ГОРОД=СТРАНА  
 КРЕМ+КРЕМ=ЖЕЛЕ  
 РАЙОН+РАЙОН=ГОРОД  
 ОХОХО+АХАХА=АХАХАХ  
 ДРАМА+ДРАМА=ТЕАТР  
 ОДИН+ОДИН=МНОГО  
 ТРОС+ТРОС=КАНАТ  
 РЕКА+МОРЕ=ОКЕАН  
 ЧЕТЫРЕ+ЧЕТЫРЕ=ВОСЕМЬ  
 СПОРТ+СПОРТ=КРОСС  
 ГНОМ+ГНОМ=СКАЛА  
 НИТКА+НИТКА=ТКАНЬ  
 ХОЛОД+ДОЖДЬ=ОСЕНЬ  
 КАРТА+КАРТА=АТЛАС  
 НАТАША+ТОНЯ=СЁСТРЫ  
 МЕМ+МЕМ=ПОСТ  
 МАМА+ПАПА=ЧАДО  
 ШЛИТЕ+ШЛИТЕ=ДЕНЬГИ  
 СЕМЬ+ОДИН=ШЕСТЬ  
 ВАГОН+ВАГОН=СОСТАВ  
 ПИТЕР+МОСКВА=ГОРОДА  
 МИНУС+МИНУС=РАВНО  
 БАЛЕТ+БАЛЕТ=ТЕАТР  
 ЛЕТО+ЛЕТО=ПОЛЕТ

Файл можно скачать по ссылке

<https://disk.yandex.ru/i/S7v3rknK3cRcTw>

## Решение

Простейшее решение основано на переборе возможных вариантов. Рассмотрим криптарифм:  
 ЛОКОН+ЛОКОН=ВОЛОСЫ

Сначала можно посчитать число используемых символов  $N_{\text{sym}}$ . Для выбранного примера оно равно 7, это символы: Л,О,К,Н,В,С,Ы.

Если используемых символов больше 10, то решения криптарифма нет.

Далее, из набора цифр от 0 до 9 формируем все возможные упорядоченные наборы из  $N_{\text{sym}}$  цифр.

Число таких наборов будет равно

$$\frac{10!}{(10 - N_{\text{sym}})!}. \quad \text{В нашем примере } \frac{10!}{3!} = 604\,800 \text{ наборов.}$$

Осталось выделить из этих наборов те, которые решают краптирифм. Пусть набор цифр имеет вид  $\{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ . Тогда надо сделать подстановку

$$\text{Л} = d_1, \text{ О} = d_2, \text{ К} = d_3, \text{ Н} = d_4, \text{ В} = d_5, \text{ С} = d_6, \text{ Ы} = d_7$$

и проверить, решает ли она криптарифм. Если среди всех возможных наборов из 7 цифр нашлось лишь одно решение, то это правильный криптарифм.

Для примера, приведён код в Wolfram Mathematica, реализующий этот алгоритм.

Загрузим данные

```
In[6]:= data = First@Import["D:\\Var1.xls"]

Out[6]= {{ЛОКОН, ЛОКОН, ВОЛОСЫ}, {ТРИ, ДВА, ПЯТЬ},
        {ЛЮБА, ЛЮБИТ, АРБУЗЫ}, {АХИНЕЯ, АХИНЕЯ, ЧЕПУХА},
        {ПЛЮС, ПЛЮС, МИНУС}, {КОРОЛЬ, КОРОНА, МОНАРХ}, {НОТА, НОТА, ТАКТ},
        {РЮМКА, РЮМКА, АВАРИЯ}, {ЛАДЬЯ, ЛАДЬЯ, ФЕРЗЬ}, {ТРЮК, ТРЮК, ЦИРК},
        {МОШКА, МОШКА, КОМАРЫ}, {БИТ, БАЙТ, СЛОВО}, {ВОЛК, ЛИСА, ЗВЕРИ},
        {СИНИЦА, СИНИЦА, ПТИЧКИ}, {МУХА, МУХА, СЛОН}, {ОГОНЬ, ВОДА, ПОХОД},
        {ГОРОД, ГОРОД, СТРАНА}, {КРЕМ, КРЕМ, ЖЕЛЕ}, {РАЙОН, РАЙОН, ГОРОД},
        {ОХОХО, АХАХА, АХАХАХ}, {ДРАМА, ДРАМА, ТЕАТР},
        {ОДИН, ОДИН, МНОГО}, {ТРОС, ТРОС, КАНАТ}, {РЕКА, МОРЕ, ОКЕАН},
        {ЧЕТЫРЕ, ЧЕТЫРЕ, ВОСЕМЬ}, {СПОРТ, СПОРТ, КРОСС},
        {ГНОМ, ГНОМ, СКАЛА}, {НИТКА, НИТКА, ТКАНЬ}, {ХОЛОД, ДОЖДЬ, ОСЕНЬ},
        {КАРТА, КАРТА, АТЛАС}, {НАТАША, ТОНЯ, СЁСТРЫ}, {МЕМ, МЕМ, ПОСТ},
        {МАМА, ПАПА, ЧАДО}, {ШЛИТЕ, ШЛИТЕ, ДЕНЬГИ}, {СЕМЬ, ОДИН, ШЕСТЬ},
        {ВАГОН, ВАГОН, СОСТАВ}, {ПИТЕР, МОСКВА, ГОРОДА},
        {МИНУС, МИНУС, РАВНО}, {БАЛЕТ, БАЛЕТ, ТЕАТР}, {ЛЕТО, ЛЕТО, ПОЛЕТ}}
```

Найдём число уникальных символов.

```
In[7]:= {w1, w2, w3} = data[[1]];
Chars = Union[Characters[StringJoin[w1, w2, w3]]];
NumOfChars = Length[Chars]
```

```
Out[9]= 7
```

Сформируем набор перестановок

```
In[10]:= AllPerms = Permutations[{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, {NumOfChars}];
```

Формируем правила подстановки

```
In[11]:= Permutation = AllPerms[[1]];
rule = Rule@@@Transpose[{Chars, Permutation}]
```

```
Out[12]= {B → 0, K → 1, L → 2, H → 3, O → 4, C → 5, Ы → 6}
```

Надо сразу отметить, что не все подстановки нам подходят, а именно, те подстановки, при которых первые буквы слов равны нулю, нам не подойдут. Сформируем правила, когда возникают эти проблемные перестановки.

```
In[13]:= firstChars =
  Union[{First[Characters[w1]], First[Characters[w2]], First[Characters[w3]]}];
badRules = Rule@@@Transpose[{firstChars, Table[0, Length[firstChars]]}]
```

```
Out[14]= {B → 0, L → 0}
```

Создадим метод, проверяющий одно сформированное правило.

```

In[15]:= CheckRule[w1_, w2_, w3_, rules_, badRules_] :=
  If[Length[Intersection[rules, badRules]] == 0,
    FromDigits[Characters[w1] /. rules] + FromDigits[Characters[w2] /. rules] ==
    FromDigits[Characters[w3] /. rules], False]
CheckRule[w1, w2, w3, rule, badRules]

Out[16]= False

```

Осталось сформировать список со значениями True или False для одного криптоарифма. И подсчитать количество значений True

```

In[17]:= CheckCrypto[w1_, w2_, w3_, Chars_, AllPerms_, badRules_] :=
  Table[CheckRule[w1, w2, w3, Rule @@@ Transpose[{Chars, AllPerms[k]}], badRules],
    {k, 1, Length[AllPerms]}]

In[*]:= Count[CheckCrypto[w1, w2, w3, Chars, AllPerms, badRules], True]

Out[*]= 0

```

Объединим шаги выше в единый цикл по всем криптоарифмам. На каждой итерации цикла будет проверяться один криптоарифм, если он правильный, то мы это отметим, выставив 1 на соответствующем месте в массиве из нулей isCorrect.

```

isCorrect = Table[0, Length[data]];
For[i = 1, i <= Length[data], i++,
  {w1, w2, w3} = data[[i]];
  Chars = Union[Characters[StringJoin[w1, w2, w3]]];
  NumOfChars = Length[Chars];
  If[NumOfChars > 10, Continue[]];
  AllPerms = Permutations[{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, {NumOfChars}];
  firstChars =
    Union[{First[Characters[w1]], First[Characters[w2]], First[Characters[w3]]}];
  badRules = Rule @@@ Transpose[{firstChars, Table[0, Length[firstChars]]}];
  If[Count[CheckCrypto[w1, w2, w3, Chars, AllPerms, badRules], True] == 1,
    isCorrect[[i]] = 1;
  ]

isCorrect
Total@isCorrect

{0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1}

16

```

Таким образом, итоговый ответ: в наборе 16 правильных криптоарифмов.

## Формулировка (Вариант 2)

Криптоарифмом называют математическую головоломку, которая представляет собой арифметическое тождество, где каждая цифра заменена буквой (одинаковые цифры заменяются одинаковыми буквами, разные цифры заменяются разными буквами).

**Пример:**

$$AXA + OXO = CMEX$$

**Решением** криптоарифма является такая подстановка цифр вместо букв, при которой получается верное равенство. При этом запрещается, чтобы первые буквы слов соответствовали нулям.

Например, решение криптоарифма выше:  $A = 2, O = 7, X = 9, C = 1, M = 0, E = 8$  или  $292 + 797 = 1089$ .

Подстановка вида  $A = 1, O = 5, X = 6, C = 0, M = 7, E = 2$  или  $161 + 565 = 0726$  решением не является.

**Криптоарифм называют правильным**, если он имеет только одно решение.

Пример выше не является правильным, так как существуют другие решения, например,  $494 + 595 = 1089$ .

Ниже представлено 40 криптоарифмов. Также, для удобства, они представлены в файле “Var6.xls”

$$\text{ОТВЕТ} + \text{ОЧЕНЬ} = \text{ПРОСТ}$$

$$\text{КАПЛЯ} + \text{КАПЛЯ} = \text{ДОЖДЬ}$$

$$\text{ПИТЕР} + \text{МОСКВА} = \text{ГОРОДА}$$

$$\text{АХИНЕЯ} + \text{АХИНЕЯ} = \text{ЧЕПУХА}$$

$$\text{ГОРА} + \text{ОГОНЬ} = \text{ВУЛКАН}$$

$$\text{НОТА} + \text{НОТА} = \text{ТАКТ}$$

$$\text{РЕКА} + \text{МОРЕ} = \text{ОКЕАН}$$

$$\text{ГНОМ} + \text{ГНОМ} = \text{СКАЛА}$$

$$\text{НАТАША} + \text{ТОНЯ} = \text{СЁСТРЫ}$$

$$\text{КИРПИЧ} + \text{КИРПИЧ} = \text{СТЕНКА}$$

$$\text{УТРО} + \text{ВЕЧЕР} = \text{СУТКИ}$$

$$\text{ВОЛК} + \text{ЛИСА} = \text{ЗВЕРИ}$$

$$\text{ВЕСНА} + \text{ЛЕТО} = \text{ТЕПЛО}$$

$$\text{МАГНИЙ} + \text{ТАНТАЛ} = \text{МЕТАЛЛЫ}$$

$$\text{ЛАПА} + \text{ШАЛЬ} = \text{ШЛЯПА}$$

$$\text{ЕЛЬ} + \text{ЕЛЬ} = \text{ЛЕС}$$

$$\text{ЛАДЬЯ} + \text{ЛАДЬЯ} = \text{ФЕРЗЬ}$$

$$\text{РЮМКА} + \text{РЮМКА} = \text{АВАРИЯ}$$

$$\text{СИНИЦА} + \text{СИНИЦА} = \text{ПТИЧКИ}$$

$$\text{СЕМЬ} + \text{ОДИН} = \text{ШЕСТЬ}$$

$$\text{МЕМ} + \text{МЕМ} = \text{ПОСТ}$$

$$\text{БАЛЕТ} + \text{БАЛЕТ} = \text{ТЕАТР}$$

$$\text{МАМА} + \text{ПАПА} = \text{ЧАДО}$$

$$\text{ТРЮК} + \text{ТРЮК} = \text{ЦИРК}$$

$$\text{ЧЕТЫРЕ} + \text{ЧЕТЫРЕ} = \text{ВОСЕМЬ}$$

$$\text{ВАГОН} + \text{ВАГОН} = \text{СОСТАВ}$$

НИТКА+НИТКА=ТКАНЬ  
 ТРИ+ДВА=ПЯТЬ  
 КУРСК+ГОРСК=ГОРОДА  
 СПОРТ+СПОРТ=КРОСС  
 БИТ+БАЙТ=СЛОВО  
 ПЛЮС+ПЛЮС=МИНУС  
 ОДИН+ОДИН=МНОГО  
 ЛЮБА+ЛЮБИТ=АРБУЗЫ  
 ОХОХО+АХАХА=АХАХАХ  
 МИНУС+МИНУС=РАВНО  
 СЛОВО+СЛОВО=ПЕСНЯ  
 ДЕТАЛЬ+ДЕТАЛЬ=ИЗДЕЛИЕ  
 УДАР+УДАР=ДРАКА  
 ТРОС+ТРОС=КАНАТ

Файл можно скачать по ссылке

<https://disk.yandex.ru/i/kaAqa2kuY7WsRw>

## Решение

Применяя описанный выше алгоритм, получим

```
In[*]:= data = First@Import["D:\\Var6.xls"]
```

```
Out[*]:= { {ОТВЕТ, ОЧЕНЬ, ПРОСТ}, {КАПЛЯ, КАПЛЯ, ДОЖДЬ},
  {ПИТЕР, МОСКВА, ГОРОДА}, {АХИНЕЯ, АХИНЕЯ, ЧЕПУХА},
  {ГОРА, ОГОНЬ, ВУЛКАН}, {НОТА, НОТА, ТАКТ}, {РЕКА, МОРЕ, ОКЕАН},
  {ГНОМ, ГНОМ, СКАЛА}, {НАТАША, ТОНЯ, СЁСТРЫ},
  {КИРПИЧ, КИРПИЧ, СТЕНКА}, {УТРО, ВЕЧЕР, СУТКИ}, {ВОЛК, ЛИСА, ЗВЕРИ},
  {ВЕСНА, ЛЕТО, ТЕПЛО}, {МАГНИЙ, ТАНТАЛ, МЕТАЛЛЫ},
  {ЛАПА, ШАЛЬ, ШЛЯПА}, {ЕЛЬ, ЕЛЬ, ЛЕС}, {ЛАДЬЯ, ЛАДЬЯ, ФЕРЗЬ},
  {РЮМКА, РЮМКА, АВАРИЯ}, {СЕНИЦА, СЕНИЦА, ПТИЧКИ},
  {СЕМЬ, ОДИН, ШЕСТЬ}, {МЕМ, МЕМ, ПОСТ}, {БАЛЕТ, БАЛЕТ, ТЕАТР},
  {МАМА, ПАПА, ЧАДО}, {ТРЮК, ТРЮК, ЦИРК}, {ЧЕТЫРЕ, ЧЕТЫРЕ, ВОСЕМЬ},
  {ВАГОН, ВАГОН, СОСТАВ}, {НИТКА, НИТКА, ТКАНЬ},
  {ТРИ, ДВА, ПЯТЬ}, {КУРСК, ГОРСК, ГОРОДА}, {СПОРТ, СПОРТ, КРОСС},
  {БИТ, БАЙТ, СЛОВО}, {ПЛЮС, ПЛЮС, МИНУС}, {ОДИН, ОДИН, МНОГО},
  {ЛЮБА, ЛЮБИТ, АРБУЗЫ}, {ОХОХО, АХАХА, АХАХАХ},
  {МИНУС, МИНУС, РАВНО}, {СЛОВО, СЛОВО, ПЕСНЯ},
  {ДЕТАЛЬ, ДЕТАЛЬ, ИЗДЕЛИЕ}, {УДАР, УДАР, ДРАКА}, {ТРОС, ТРОС, КАНАТ} }
```

```

In[ ]:= isCorrect = Table[0, Length[data]];
For[i = 1, i <= Length[data], i++,
  {w1, w2, w3} = data[[i]];
  Chars = Union[Characters[StringJoin[w1, w2, w3]]];
  NumOfChars = Length[Chars];
  If[NumOfChars > 10, Continue[]];
  AllPerms = Permutations[{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, {NumOfChars}];
  firstChars =
  Union[{First[Characters[w1]], First[Characters[w2]], First[Characters[w3]]}];
  badRules = Rule@@@ Transpose[{firstChars, Table[0, Length[firstChars]]}];
  If[Count[CheckCrypto[w1, w2, w3, Chars, AllPerms, badRules], True] == 1,
    isCorrect[[i]] = 1];
]

In[ ]:= isCorrect
Total@isCorrect

Out[ ]:= {0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1,
  0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0}

Out[ ]:= 18

```

Ответ: в наборе 18 правильных криптоарифмов.

### Формулировка (Вариант 3)

Криптоарифмом называют математическую головоломку, которая представляет собой арифметическое тождество, где каждая цифра заменена буквой (одинаковые цифры заменяются одинаковыми буквами, разные цифры заменяются разными буквами).

#### Пример:

AXA+OXO = CMEX

**Решением** криптоарифма является такая подстановка цифр вместо букв, при которой получается верное равенство. При этом запрещается, чтобы первые буквы слов соответствовали нулям.

Например, решение криптоарифма выше: A = 2, O = 7, X = 9, C = 1, M = 0, E = 8 или  $292 + 797 = 1089$ .

Подстановка вида A = 1, O = 5, X = 6, C = 0, M = 7, E = 2 или  $161 + 565 = 0726$  решением не является.

**Криптоарифм называют правильным**, если он имеет только одно решение.

Пример выше не является правильным, так как существуют другие решения, например,  $494 + 595 = 1089$ .

Ниже представлено 40 криптоарифмов. Также, для удобства, они представлены в файле “Var10.xls”

ЛУКА+ЛУКИЧ=ИВАНОВ

ПЕСОК+ВОДА=ОАЗИС



ПЛЮС+ПЛЮС=МИНУС  
ВОЛК+ЛИСА=ЗВЕРИ  
СЕМЬ+ОДИН=ШЕСТЬ  
ВАГОН+ВАГОН=СОСТАВ  
СЛОВО+СЛОВО=ПЕСНЯ  
ПАС+ПАС=ГОЛ  
КУРСК+ГОРСК=ГОРОДА  
КРЕМ+КРЕМ=ЖЕЛЕ  
СЕТ+ГЕЙМ=МАТЧ  
МАМА+ПАПА=ЧАДО  
КОРОЛЬ+КОРОНА=МОНАРХ  
НОТА+НОТА=ТАКТ  
КАРТА+КАРТА=АТЛАС  
ВЕСНА+ЛЕТО=ТЕПЛО  
НИТКА+НИТКА=ТКАНЬ  
ОХОХО+АХАХА=АХАХАХ  
СИНИЦА+СИНИЦА=ПТИЧКИ  
ВЕТКА+ВЕТКА=ДЕРЕВО  
ЛОКОН+ЛОКОН=ВОЛОСЫ  
НАУКА+УЧЁБА=РАБОТА  
МОШКА+МОШКА=КОМАРЫ  
НАТАША+ТОНЯ=СЁСТРЫ  
АЛЬДЕ+БАРАН=ТЕЛЕЦ  
ГОРА+ОГОНЬ=ВУЛКАН  
КАПЛЯ+КАПЛЯ=ДОЖДЬ  
ТРЮК+ТРЮК=ЦИРК  
ХОЛОД+ДОЖДЬ=ОСЕНЬ  
ЕЛЬ+ЕЛЬ=ЛЕС  
ЛЕТО+ЛЕТО=ПОЛЕТ  
КОСТИ+КОСТИ=СКЕЛЕТ  
КИРПИЧ+КИРПИЧ=СТЕНКА  
БОРТ+МАЧТА=ЛОДКА  
ТРИ+ДВА=ПЯТЬ  
ГОРОД+ГОРОД=СТРАНА  
МЕМ+МЕМ=ПОСТ  
БИТ+БАЙТ=СЛОВО  
МУХА+МУХА=СЛОН  
ОТВЕТ+ОЧЕНЬ=ПРОСТ

Файл можно скачать по ссылке

<https://disk.yandex.ru/i/Suu-Og4cPzD7sg>

## Решение

Применяя описанный выше алгоритм, получим

```
In[18]:= data = First@Import["D:\\Var10.xls"]

Out[18]:= {{ЛУКА, ЛУКИЧ, ИВАНОВ}, {ПЕСОК, ВОДА, ОАЗИС}, {ПЛЮС, ПЛЮС, МИНУС},
{ВОЛК, ЛИСА, ЗВЕРИ}, {СЕМЬ, ОДИН, ШЕСТЬ}, {ВАГОН, ВАГОН, СОСТАВ},
{СЛОВО, СЛОВО, ПЕСНЯ}, {ПАС, ПАС, ГОЛ}, {КУРСК, ГОРСК, ГОРОДА},
{КРЕМ, КРЕМ, ЖЕЛЕ}, {СЕТ, ГЕЙМ, МАТЧ}, {МАМА, ПАПА, ЧАДО},
{КОРОЛЬ, КОРОНА, МОНАРХ}, {НОТА, НОТА, ТАКТ}, {КАРТА, КАРТА, АТЛАС},
{ВЕСНА, ЛЕТО, ТЕПЛО}, {НИТКА, НИТКА, ТКАНЬ}, {ОХОХО, АХАХА, АХАХАХ},
{СИНИЦА, СИНИЦА, ПТИЧКИ}, {ВЕТКА, ВЕТКА, ДЕРЕВО},
{ЛОКОН, ЛОКОН, ВОЛОСЫ}, {НАУКА, УЧЁБА, РАБОТА},
{МОШКА, МОШКА, КОМАРЫ}, {НАТАША, ТОНЯ, СЁСТРЫ},
{АЛЬДЕ, БАРАН, ТЕЛЕЦ}, {ГОРА, ОГОНЬ, ВУЛКАН},
{КАПЛЯ, КАПЛЯ, ДОЖДЬ}, {ТРЮК, ТРЮК, ЦИРК}, {ХОЛОД, ДОЖДЬ, ОСЕНЬ},
{ЕЛЬ, ЕЛЬ, ЛЕС}, {ЛЕТО, ЛЕТО, ПОЛЕТ}, {КОСТИ, КОСТИ, СКЕЛЕТ},
{КИРПИЧ, КИРПИЧ, СТЕНКА}, {БОРТ, МАЧТА, ЛОДКА},
{ТРИ, ДВА, ПЯТЬ}, {ГОРОД, ГОРОД, СТРАНА}, {МЕМ, МЕМ, ПОСТ},
{БИТ, БАЙТ, СЛОВО}, {МУХА, МУХА, СЛОН}, {ОТВЕТ, ОЧЕНЬ, ПРОСТ}}
```

```
In[19]:= isCorrect = Table[0, Length[data]];
For[i = 1, i <= Length[data], i++,
  {w1, w2, w3} = data[[i]];
  Chars = Union[Characters[StringJoin[w1, w2, w3]]];
  NumOfChars = Length[Chars];
  If[NumOfChars > 10, Continue[]];
  AllPerms = Permutations[{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, {NumOfChars}];
  firstChars =
    Union[{First[Characters[w1]], First[Characters[w2]], First[Characters[w3]]}];
  badRules = Rule@@@Transpose[{firstChars, Table[0, Length[firstChars]]}];
  If[Count[CheckCrypto[w1, w2, w3, Chars, AllPerms, badRules], True] == 1,
    isCorrect[[i]] = 1];
]
```

```
In[21]:= isCorrect
Total@isCorrect

Out[21]:= {1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0}
```

```
Out[22]:= 14
```

Ответ: в наборе 14 правильных криптарифмов.

## Задача 2: Стихотворный размер (40 баллов)

Автор задачи: Александр Кривошеин

Автор разбора: Александр Кривошеин

### Формулировка (Вариант 1)

На уроке литературы Петя узнал, что любимым стихотворным размером русских классических поэтов 19-го века был ямб (среди пяти классических стихотворных размеров: хорей, ямб, дактиль, амфибрахий, анапест). Петя любил анализировать данные и подвергал всё сомнению. Он решил проверить, действительно ли это так. Для этого он случайным образом выбрал 800 стихотворений А.А. Фета и решил определить количество стихов, написанных ямбом.

Набор стихов находится в файле “Stihi1.xls”. Также, для удобства, стихи можно скачать архивом “Stihi1.zip”. В архиве каждый стих записан в отдельном текстовом файле.

**Задача: помогите Пете найти количество стихотворений, написанных ямбом.**

Для решения задачи может понадобиться орфоэпический словарь (словарь ударений)

Файл с таким словарём прикреплен здесь “all\_accents.zip”. Словарь является текстовым файлом, в файле в каждой строке записано слово и через запятую его версия с ударением, которое выделено символом ^.

Оригинал словаря в формате .tsv можно найти по ссылке:

[https://github.com/Koziev/NLP\\_Datasets/blob/master/Stress/all\\_accents.zip](https://github.com/Koziev/NLP_Datasets/blob/master/Stress/all_accents.zip)

Файл “Stihi1.xls” можно скачать по ссылке:

<https://disk.yandex.ru/i/B0E-5OHуYHLOKg>

Файл “all\_accents.zip” можно скачать по ссылке:

<https://disk.yandex.ru/d/0qKARuegQkatWQ>

### Комментарий

Набор стихотворений был выгружен поэтического корпуса Национального корпуса русского языка:

<https://ruscorpora.ru/new/search-poetic.html>

Этот корпус содержит базу данных с размеченными стихами, стихотворный размер указан для каждой строчки. Выбирались те стихи, в которых более 90% строк написаны одним из классических стихотворных размеров.

### Решение

Рассмотрим одно из стихотворений.

Поднялася пыль степная,  
Солнышко взошло,  
Всюду сбруя боевая  
Блещет как стекло.

Сначала заменим слова стихотворения на слова из словаря с ударениями. Далее, можно удалить все согласные буквы из стихотворения. Каждая строка теперь имеет вид

оаяа^ые^ая

<sup>^</sup>оыоо<sup>^</sup>о  
<sup>^</sup>юу<sup>^</sup>уяое<sup>^</sup>ая  
<sup>^</sup>ее<sup>^</sup>ае<sup>^</sup>о

Галочка находится перед ударной буквой, согласно словарю. Сразу можно отметить, что строки могут быть не равны по числу гласных. Кроме того, в словаре может не быть всех слов, например, нет слова “Поднялася” и у соответствующих гласных нет ударения, также в слове из 3-х слогов “Солнышко” указано 1 ударение, но в рамках стиха в этом слове два ударных слога. Поэтому по одной строке определить стихотворный размер автоматически не получится.

Полученные строчки мы заменим на бинарную строчку следующим образом: безударная гласная заменяется на 0, ударная на 1

0, 0, 0, 0, 1, 0, 1, 0  
 1, 0, 0, 0, 1  
 1, 0, 1, 0, 0, 0, 1, 0  
 1, 0, 1, 0, 1

Чтобы собрать информацию по всем строчкам, просуммируем полученные строки (причём короткие строки можно дополнить нулями до размера самой длинной строки).

3, 0, 2, 0, 3, 0, 2, 0

На тех позициях где часто встречается ударение будет локальный пик, то есть значение, которое больше соседних. Анализируя пики можно получить бинарную строку с расположением позиций, где чаще всего встречаются ударения. То есть пики заменяем на 1, прочие позиции заменяем на нули. В итоге получаем строчку

1, 0, 1, 0, 1, 0, 1, 0

Осталось сравнить эту строку с бинарными строками, соответствующими одному из 5 классических стихотворных размеров.

Хорей : 1, 0, 1, 0, 1, 0, 1, 0

Ямб : 0, 1, 0, 1, 0, 1, 0, 1

Дактиль : 1, 0, 0, 1, 0, 0, 1, 0


Амфибрахий : 0, 1, 0, 0, 1, 0, 0, 1


Анапест : 0, 0, 1, 0, 0, 1, 0, 0

Сравнивать строки можно с помощью расстояния Хэмминга, которое равно числу позиций, в которых соответствующие символы двух строк одинаковой длины различны. В данном случае размер стихотворения в точности совпадает с хореем.

Для примера приведён код в Wolfram Mathematica, реализующий описанный выше алгоритм. Формирование правил замены слов на слова с ударениями.

```
accents = Import["E:\\all_accents.txt"];
accents2 = StringSplit[accents, "
"];
accents3 = StringSplit[#, ","] & /@ accents2;
accRules = Dispatch[Rule @@@ accents3]
```

Dispatch [  Length: 1 680 535 ]

 Data not in notebook; Store now »

Генератор строк с 5-ю классическими размерами любого размера:

```
GenerateSize[n_] := Module[{r1, r2, r3, r4, r5},
  r1 = Take[Join @@ Table[{1, 0}, n], n];
  r2 = Take[Join @@ Table[{0, 1}, n], n];
  r3 = Take[Join @@ Table[{1, 0, 0}, n], n];
  r4 = Take[Join @@ Table[{0, 1, 0}, n], n];
  r5 = Take[Join @@ Table[{0, 0, 1}, n], n];
  {r1, r2, r3, r4, r5}
]
```

Функция для перевода строки стиха в бинарную строчку.

```
StrokeToNumbers[stroka_] := Module[
  {glas = {"a", "я", "o", "ё", "y", "e", "ю", "и", "ы", "э", "^"},
   glasStr = StringJoin[{"a", "я", "o", "ё", "y", "e", "ю", "и", "ы", "э"}],
   accStroka, onlyGlas},
  accStroka = TextWords[stroka] /. accRules;
  onlyGlas = Cases[Flatten@Characters[accStroka], Alternatives @@ glas] /. {"^" -> "Z"};
  ToExpression@
  Characters[StringReplace[StringReplace[StringJoin@onlyGlas,
    RegularExpression["Z[аяёуеюиыэ]" -> "1"],
    RegularExpression["[аяёуеюиыэ]" -> "0"]]]
]
```

Функция для определения размера стиха.

```
StihToSize[stih_] := Module[{stihList, finalRes},
  stihList = StringSplit[ToLowerCase[stih], "
"];
  Res = StrokeToNumbers /@ stihList;
  If[Total[Length /@ Res] == 0, {6, {0}},
    finalRes = Total@{PadRight[#, Max[Length /@ Res]] & /@ Res};
    finalRes = MaxDetect[finalRes];
    {First[
      Flatten@Position[HammingDistance[finalRes, #] & /@ GenerateSize[Length[finalRes]],
        Min[HammingDistance[finalRes, #] & /@ GenerateSize[Length[finalRes]]]]],
      finalRes}
]
```

Подгрузка стихов и запуск функции StihToSize[] для каждого стиха и выведение результата.

```
AllStih = Flatten@First@Import["E:\\Stihi1.xls"];
Print["Количество стихов: ", Length@AllStih]
Results = StihToSize /@ AllStih;
Tally[Transpose[Results][[1]]]

Количество стихов: 800

{{1, 159}, {2, 445}, {4, 62}, {3, 67}, {5, 67}}
```

Нас интересует число стихов написанных ямбом. Это второй элемент в полученном списке.

Значение, полученное в результате работы алгоритма равно 445.

Истинный верный ответ для данного набора стихов равен 451. Максимальное число баллов за задачу давалось, если полученный ответ был равен истинному с погрешностью в 7 стихов.

## Формулировка (Вариант 2)

На уроке литературы Петя узнал, что любимым стихотворным размером русских классических поэтов 19-го века был ямб (среди пяти классических стихотворных размеров: хорей, ямб, дактиль, амфибрахий, анапест). Петя любил анализировать данные и подвергал всё сомнению. Он решил проверить, действительно ли это так. Для этого он случайным образом выбрал 800 стихотворений А.А. Фета и решил определить количество стихов, написанных ямбом.

Набор стихов находится в файле “Stihi2.xls”. Также, для удобства, стихи можно скачать архивом “Stihi2.zip”. В архиве каждый стих записан в отдельном текстовом файле.

**Задача: помогите Пете найти количество стихотворений, написанных ямбом.**

Для решения задачи может понадобиться орфоэпический словарь (словарь ударений)

Файл с таким словарём прикреплен здесь “all\_accents.zip”. Словарь является текстовым файлом, в файле в каждой строке записано слово и через запятую его версия с ударением, которое выделено символом ^.

Оригинал словаря в формате .tsv можно найти по ссылке:

[https://github.com/Koziev/NLP\\_Datasets/blob/master/Stress/all\\_accents.zip](https://github.com/Koziev/NLP_Datasets/blob/master/Stress/all_accents.zip)

Файл “Stihi2.xls” можно скачать по ссылке:

<https://disk.yandex.ru/i/UhBbtdDLt8t1-A>

Файл “all\_accents.zip” можно скачать по ссылке:

<https://disk.yandex.ru/d/0qKARuegQkatWQ>

## Решение

Используя написанный выше код в Wolfram Mathematica, реализующий описанный выше алгоритм, получим:

```
AllStih = Flatten@First@Import["E:\\Stihi2.xls"];
Print["Количество стихов: ", Length@AllStih]
Results = StihToSize /@ AllStih;
Tally[Transpose[Results][[1]]]

Количество стихов: 800

{{1, 140}, {2, 472}, {3, 67}, {5, 68}, {4, 53}}
```

Нас интересует число стихов написанных ямбом. Это второй элемент в полученном списке.

Значение, полученное в результате работы алгоритма равно 472.

Истинный верный ответ для данного набора стихов равен 479. Максимальное число баллов за задачу давалось, если полученный ответ был равен истинному с погрешностью в 7 стихов.

### Формулировка (Вариант 3)

На уроке литературы Петя узнал, что любимым стихотворным размером русских классических поэтов 19-го века был ямб (среди пяти классических стихотворных размеров: хорей, ямб, дактиль, амфибрахий, анапест). Петя любил анализировать данные и подвергал всё сомнению. Он решил проверить, действительно ли это так. Для этого он случайным образом выбрал 800 стихотворений А.А. Фета и решил определить количество стихов, написанных ямбом.

Набор стихов находится в файле “Stihi3.xls”. Также, для удобства, стихи можно скачать архивом “Stihi3.zip”. В архиве каждый стих записан в отдельном текстовом файле.

**Задача: помогите Пете найти количество стихотворений, написанных ямбом.**

Для решения задачи может понадобиться орфоэпический словарь (словарь ударений)

Файл с таким словарём прикреплен здесь “all\_accents.zip”. Словарь является текстовым файлом, в файле в каждой строке записано слово и через запятую его версия с ударением, которое выделено символом ^.

Оригинал словаря в формате .tsv можно найти по ссылке:

[https://github.com/Koziev/NLP\\_Datasets/blob/master/Stress/all\\_accents.zip](https://github.com/Koziev/NLP_Datasets/blob/master/Stress/all_accents.zip)

Файл “Stihi3.xls” можно скачать по ссылке:

<https://disk.yandex.ru/i/yleslqBptBvTKg>

Файл “all\_accents.zip” можно скачать по ссылке:

<https://disk.yandex.ru/d/0qKARuegQkatWQ>

### Решение

Используя написанный выше код в Wolfram Mathematica, реализующий описанный выше алгоритм, получим:

```
AllStih = Flatten@First@Import["E:\\Stihi3.xls"];
Print["Количество стихов: ", Length@AllStih]
Results = StihToSize /@ AllStih;
Tally[Transpose[Results][[1]]]
```

Количество стихов: 800

```
{{1, 167}, {2, 431}, {4, 71}, {3, 67}, {5, 64}}
```

Нас интересует число стихов написанных ямбом. Это второй элемент в полученном списке.

Значение, полученное в результате работы алгоритма равно 431.

Истинный верный ответ для данного набора стихов равен 438. Максимальное число баллов за задачу давалось, если полученный ответ был равен истинному с погрешностью в 7 стихов.

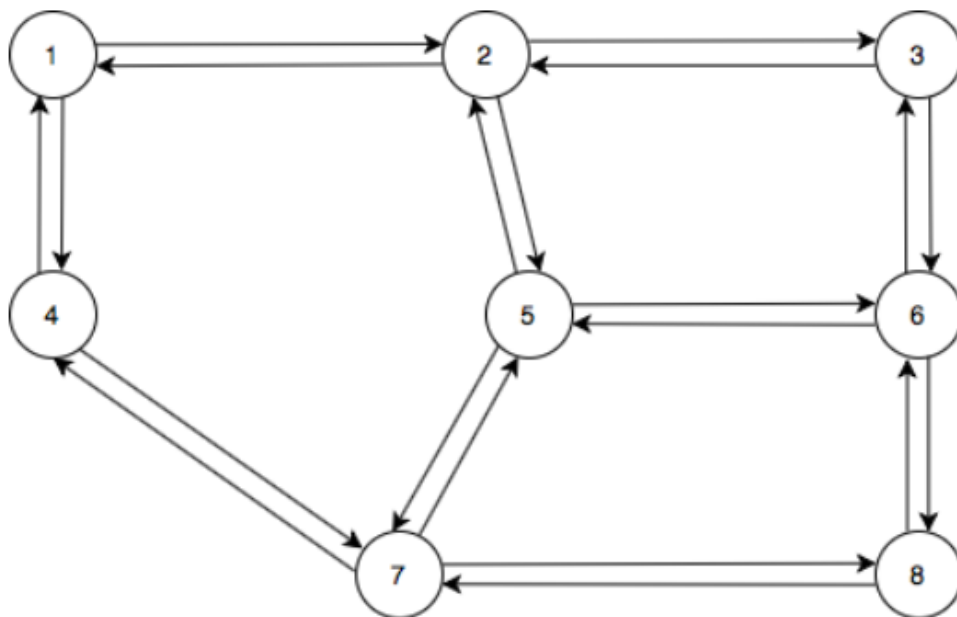
## Задача 3: Транспортный поток (40 баллов)

Автор задачи: Александр Крылатов

Авторы разбора: Александр Кривошеин

### Формулировка (Вариант 1)

Рассмотрим улично-дорожную сеть, состоящую из 8 узлов и 20 направленных дуг:



Время движения одного автомобиля по направленной дуге  $(i, j)$  зависит от количества всех автомобилей  $x_{ij}$ , желающих проехать по этой дуге, следующим образом:

$$t_{ij}(x_{ij}) = t_{ij}^0 \left( 1 + \frac{x_{ij}}{c_{ij}} \right),$$

где  $t_{ij}^0$  время свободного движения по дуге, а  $c_{ij}$  коэффициент, влияющий на время прохождения дуги.



Временем движения по некоторому маршруту из узла  $r$  в узел  $s$  назовём суммарное время движения по всем дугам, входящим в маршрут.

Пример. Пусть  $t_{36}^0 = 3$ ,  $t_{68}^0 = 2$ ,  $c_{36} = 2340$ ,  $c_{68} = 495$ . Тогда время свободного движения по маршруту из узла 3 через узел 6 до узла 8 равно  $2 + 3 = 5$ . Если же по этому маршруту желает проехать 1000 автомобилей, то время движения по маршруту равно

$$3 \left( 1 + \frac{1000}{2340} \right) + 2 \left( 1 + \frac{1000}{495} \right) = 10.32.$$

Пусть из узла отправления 1 в узел прибытия 8 желает проехать 9000 автомобилей. При этом эти автомобили распределятся по возможным маршрутам между узлами таким образом: время движения по выбранным маршрутам между парой узлов отправления-прибытия является одинаковым и это время меньше времени движения по оставшимся маршрутам между узлами 1 и 8.

**Найдите время движения из узла отправления 1 в узел прибытия 8 в загруженной указанным выше образом сети.** В качестве ответа введите число с точностью до двух знаков после запятой. Например, 25,03.

Значения параметров времени свободного движения  $t_{ij}^0$  и значения коэффициентов  $c_{ij}$  даны в таблицах 1 и 2 соответственно.

Таблица 1: Значения параметров времени свободного движения

$t_{12}^0$	$t_{23}^0$	$t_{14}^0$	$t_{25}^0$	$t_{36}^0$	$t_{56}^0$	$t_{47}^0$	$t_{57}^0$	$t_{68}^0$	$t_{78}^0$
4	5	5	4	3	4	6	3	2	5

Таблица 2: Значения коэффициентов  $c_{ij}$

$c_{12}$	$c_{23}$	$c_{14}$	$c_{25}$	$c_{36}$	$c_{56}$	$c_{47}$	$c_{57}$	$c_{68}$	$c_{78}$
2590	496	2340	2340	2340	1778	1778	495	495	2340

Отметим, что  $t_{ij}^0 = t_{ji}^0$  и  $c_{ij} = c_{ji}$  для всех пар узлов  $i, j$ , соединённых дугами.

## Решение

Простейший способ решения, это выпускать по одной машине из узла отправления 1 в узел прибытия 8, каждая машина будет двигаться по самому короткому маршруту. С увеличением загрузки улично-дорожной сети самый короткий маршрут будет изменяться. Выпустив в цикле все 9000 машин, мы получим приближённое решение задачи.

Для примера, приведён код в Wolfram Mathematica, реализующий этот алгоритм. Запишем входные данные:

**segments:** список названий дуг

**timesInit:** значения параметров времени свободного движения по этим дугам

**coefs:** значения коэффициентов, влияющих на время прохождения дуг

**carsCurrent:** текущее распределение машин по дугам, инициализация нулями.

```

segments = ToString /@ {s14, s41, s12, s21, s47, s74, s23, s32, s25, s52, s36, s63,
    s56, s65, s57, s75, s78, s87, s68, s86};
timesInit = {5, 5, 4, 4, 6, 6, 5, 5, 4, 4, 3, 3, 4, 4, 3, 3, 5, 5, 2, 2};
coefs = {2340, 2340, 2590, 2590, 1778, 1778, 496, 496, 2340, 2340, 2340, 2340,
    1778, 1778, 495, 495, 2340, 2340, 495, 495};
carsInit = Table[0, {i, 1, 20}];

```

Маршруты будем формировать как списки проходимых узлов по маршруту, например, маршрут из 1 в 8 через узлы 2,5,6 имеет вид:

```
route = {1, 2, 5, 6, 8};
```

Найти сегменты по маршруту можно так:

```

routeBySegments = StringJoin["s", ToString#[[1]], ToString#[[2]]] & /@
    Partition[route, 2, 1]
{s12, s25, s56, s68}

```

Далее, потребуется функция, обновляющая распределение машин по дугам, если известно количество новых машин на маршруте. А также потребуется функция для вычисления текущих затрат времени на дугах при известном распределении машин по дугам.

```

UpdateDistribution[carsCurrent_, route_, numOfCars_] :=
Module[{routeBySegments, segmentsPos, carsCurrentNew = carsCurrent},
    routeBySegments = StringJoin["s", ToString#[[1]], ToString#[[2]]] & /@
        Partition[route, 2, 1];
    segmentsPos = Flatten[Position[segments, #] & /@ routeBySegments];
    (carsCurrentNew[[#]] = carsCurrent[[#]] + numOfCars) & /@ segmentsPos;
    carsCurrentNew
]
timesBySegments[carsCurrent_] := timesInit (1 + carsCurrent/coefs)

```

Например, запустим по маршруту из 1 в 8 через узлы 2,5,6 900 машин.

```

route = {1, 2, 5, 6, 8};
carsCurrent = UpdateDistribution[carsInit, route, 900]
timesCurrent = timesBySegments[carsCurrent]

{0, 0, 900, 0, 0, 0, 0, 0, 900, 0, 0, 0, 900, 0, 0, 0, 0, 0, 900, 0}

{5, 5,  $\frac{1396}{259}$ , 4, 6, 6, 5, 5,  $\frac{72}{13}$ , 4, 3, 3,  $\frac{5356}{889}$ , 4, 3, 3, 5, 5,  $\frac{62}{11}$ , 2}

```

Далее, можно было бы в полу-ручном режиме запускать какое-то количество машин по кратчайшим маршрутам и обновлять время движения по этим маршрутам, до тех пор пока не будут запущены все 9000 машин. А затем отбалансировать количество машин по маршрутам, так чтобы время движения по всем маршрутам от узла 1 до узла 8 было бы примерно одинаковым. Это время и будет ответом.

В автоматическом режиме эту схему решения можно реализовать так: построим граф улично-дорожной сети и визуализируем кратчайший текущий маршрут

```

panelLabel[lbl_] := Panel[lbl, FrameMargins → 0, Background → Lighter[Yellow, 0.7]];

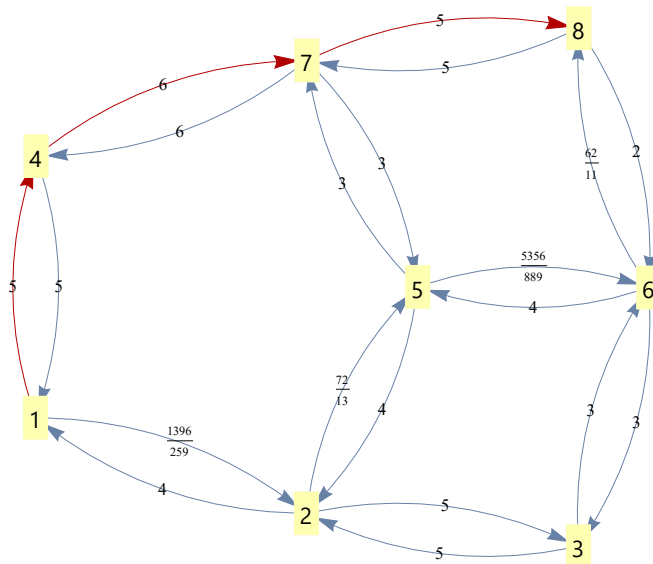
g = Graph[{1 ↔ 4, 4 ↔ 1, 1 ↔ 2, 2 ↔ 1, 4 ↔ 7, 7 ↔ 4, 2 ↔ 3, 3 ↔ 2, 2 ↔ 5, 5 ↔ 2, 3 ↔ 6,
  6 ↔ 3, 5 ↔ 6, 6 ↔ 5, 5 ↔ 7, 7 ↔ 5, 7 ↔ 8, 8 ↔ 7, 6 ↔ 8, 8 ↔ 6}, EdgeWeight → timesCurrent,
  VertexLabels → "Name", EdgeLabels → "EdgeWeight"];

route = FindShortestPath[g, 1, 8];
Print["Кратчайший текущий маршрут: ", route]
Grid[
  {HighlightGraph[g, PathGraph[route, DirectedEdges → True],
    VertexLabels → Table[i → Placed[ToString[i], Center, panelLabel], {i, 8}],
    PlotTheme → {"Web", "Wide", "Tall"}, ImageSize → Medium]},
  {Print["Время кратчайшего маршрута: ", GraphDistance[g, 1, 8]]}]

```

Кратчайший текущий маршрут: {1, 4, 7, 8}

Время кратчайшего маршрута: 16.



Осталось организовать цикл, в котором можно запускать по одной машине за итерацию.

```

timesCurrent = timesInit;
carsCurrent = carsInit;
For[i = 1, i ≤ 9000, i++,
  g = Graph[{1 ↔ 4, 4 ↔ 1, 1 ↔ 2, 2 ↔ 1, 4 ↔ 7, 7 ↔ 4, 2 ↔ 3, 3 ↔ 2, 2 ↔ 5, 5 ↔ 2, 3 ↔ 6,
    6 ↔ 3, 5 ↔ 6, 6 ↔ 5, 5 ↔ 7, 7 ↔ 5, 7 ↔ 8, 8 ↔ 7, 6 ↔ 8, 8 ↔ 6}, EdgeWeight → timesCurrent,
    VertexLabels → "Name", EdgeLabels → "EdgeWeight"];
  route = FindShortestPath[g, 1, 8];
  carsCurrent = UpdateDistribution[carsCurrent, route, 1];
  timesCurrent = N@timesBySegments[carsCurrent];
]

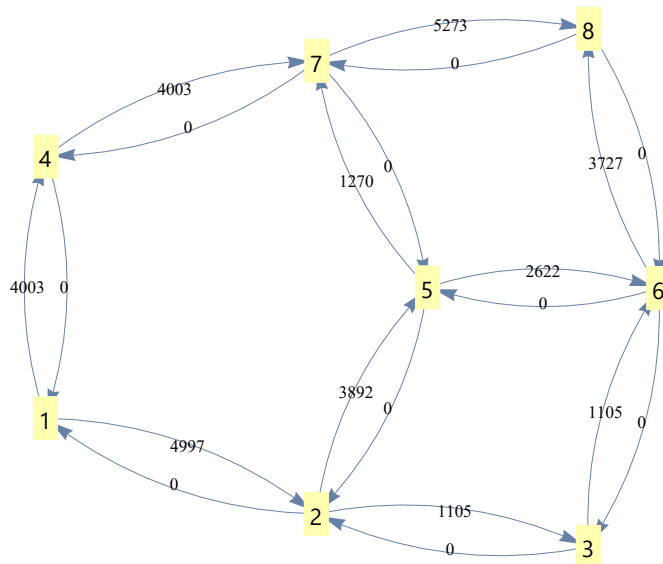
```

Далее, выведем итоговое распределение машин и найдём время движения по используемым маршрутам

```

Graph[{1 → 4, 4 → 1, 1 → 2, 2 → 1, 4 → 7, 7 → 4, 2 → 3, 3 → 2, 2 → 5, 5 → 2, 3 → 6,
      6 → 3, 5 → 6, 6 → 5, 5 → 7, 7 → 5, 7 → 8, 8 → 7, 6 → 8, 8 → 6}, EdgeWeight → carsCurrent,
      EdgeLabels → "EdgeWeight",
      VertexLabels → Table[i → Placed[ToString[i], Center, panelLabel], {i, 8}]]

```



```

GraphDistance[g, 1, 4] + GraphDistance[g, 4, 7] + GraphDistance[g, 7, 8]
GraphDistance[g, 1, 2] + GraphDistance[g, 2, 5] + GraphDistance[g, 5, 6] +
  GraphDistance[g, 6, 8]
GraphDistance[g, 1, 2] + GraphDistance[g, 2, 5] + GraphDistance[g, 5, 7] +
  GraphDistance[g, 7, 8]
GraphDistance[g, 1, 2] + GraphDistance[g, 2, 3] + GraphDistance[g, 3, 6] +
  GraphDistance[g, 6, 8]

```

49.3268

49.3245

49.323

49.3302

В качестве итогового ответа можно было взять среднее по этим числам и округлить до 2 знаков после запятой: 49.33.

Максимальный балл давался за ответ в интервале [49.30, 49.35]

## Формулировка (Вариант 2 и 3)

Варианты 2 и 3 отличаются от Варианта 1 количеством автомобилей, желающих проехать по маршруту от пункта 1 до пункта 8.

Вариант 2: Пусть из узла отправления 1 в узел прибытия 8 желает проехать 10000 автомобилей.

Вариант 3: Пусть из узла отправления 1 в узел прибытия 8 желает проехать 11000 автомобилей.

## Решение

Запуская описанный выше алгоритм, получим:

```

timesCurrent = timesInit;
carsCurrent = carsInit;
For[i = 1, i ≤ 10000, i++,
  g = Graph[{1 ↔ 4, 4 ↔ 1, 1 ↔ 2, 2 ↔ 1, 4 ↔ 7, 7 ↔ 4, 2 ↔ 3, 3 ↔ 2, 2 ↔ 5, 5 ↔ 2, 3 ↔ 6,
    6 ↔ 3, 5 ↔ 6, 6 ↔ 5, 5 ↔ 7, 7 ↔ 5, 7 ↔ 8, 8 ↔ 7, 6 ↔ 8, 8 ↔ 6}, EdgeWeight → timesCurrent,
    VertexLabels → "Name", EdgeLabels → "EdgeWeight"];
  route = FindShortestPath[g, 1, 8];
  carsCurrent = UpdateDistribution[carsCurrent, route, 1];
  timesCurrent = N@timesBySegments[carsCurrent];
]

Mean[{GraphDistance[g, 1, 4] + GraphDistance[g, 4, 7] + GraphDistance[g, 7, 8],
  GraphDistance[g, 1, 2] + GraphDistance[g, 2, 5] + GraphDistance[g, 5, 6] +
  GraphDistance[g, 6, 8],
  GraphDistance[g, 1, 2] + GraphDistance[g, 2, 5] + GraphDistance[g, 5, 7] +
  GraphDistance[g, 7, 8],
  GraphDistance[g, 1, 2] + GraphDistance[g, 2, 3] + GraphDistance[g, 3, 6] +
  GraphDistance[g, 6, 8]}]

53.1176

```

Итоговый ответ для Варианта 2: 53.12

Максимальный балл давался за ответ в интервале [53.10, 53.15].

```

timesCurrent = timesInit;
carsCurrent = carsInit;
For[i = 1, i ≤ 11000, i++,
  g = Graph[{1 ↔ 4, 4 ↔ 1, 1 ↔ 2, 2 ↔ 1, 4 ↔ 7, 7 ↔ 4, 2 ↔ 3, 3 ↔ 2, 2 ↔ 5, 5 ↔ 2, 3 ↔ 6,
    6 ↔ 3, 5 ↔ 6, 6 ↔ 5, 5 ↔ 7, 7 ↔ 5, 7 ↔ 8, 8 ↔ 7, 6 ↔ 8, 8 ↔ 6}, EdgeWeight → timesCurrent,
    VertexLabels → "Name", EdgeLabels → "EdgeWeight"];
  route = FindShortestPath[g, 1, 8];
  carsCurrent = UpdateDistribution[carsCurrent, route, 1];
  timesCurrent = N@timesBySegments[carsCurrent];
]

Mean[{GraphDistance[g, 1, 4] + GraphDistance[g, 4, 7] + GraphDistance[g, 7, 8],
  GraphDistance[g, 1, 2] + GraphDistance[g, 2, 5] + GraphDistance[g, 5, 6] +
  GraphDistance[g, 6, 8],
  GraphDistance[g, 1, 2] + GraphDistance[g, 2, 5] + GraphDistance[g, 5, 7] +
  GraphDistance[g, 7, 8],
  GraphDistance[g, 1, 2] + GraphDistance[g, 2, 3] + GraphDistance[g, 3, 6] +
  GraphDistance[g, 6, 8]}]

56.9091

```

Итоговый ответ для Варианта 3: 56.91

Максимальный балл давался за ответ в интервале [56.89, 56.94].