```
%tensorflow_version 2.x
```

```
Colab only includes TensorFlow 2.x; %tensorflow_version has no effect.
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


import tensorflow as tf
import keras
```

Загрузка и подготовка датасета MNIST

```
df = pd.DataFrame(columns = ["optimizers", "accuracy"])


(train_x, train_y), (test_x, test_y) = tf.keras.datasets.mnist.load_data()


train_x = train_x.reshape(-1, 28, 28, 1).astype(np.float32)/ 255
test_x = test_x.reshape(-1, 28, 28, 1).astype(np.float32)/ 255

train_y = train_y.astype(np.int32)
test_y = test_y.astype(np.int32)

print(train_x.shape, train_x.dtype)
print(test_x.shape, test_x.dtype)
print(train_y.shape, train_y.dtype)
print(test_y.shape, test_y.dtype)
```

```
(60000, 28, 28, 1) float32
(10000, 28, 28, 1) float32
(60000,) int32
(10000,) int32
```

Визуализация датасета MNIST

```
some_samples = train_x[:10, ...]

fig = plt.figure(figsize=(10, 6))
for j in range(some_samples.shape[0]):
    ax = fig.add_subplot(4, 8, j+1)
    ax.imshow(some_samples[j,:,:,0], cmap='gray')
    plt.xticks([]), plt.yticks([])
plt.show()
```

```
NUM_EPOCHS = 5
BATCH_SIZE = 64
LEARNING_RATE = 0.001


train_ds = tf.data.Dataset.from_tensor_slices((train_x, train_y))
train_ds = train_ds.shuffle(buffer_size=train_x.shape[0])
train_ds = train_ds.repeat(NUM_EPOCHS)
train_ds = train_ds.batch(BATCH_SIZE)


class Model(tf.keras.Model):

    def __init__(self):
        super(Model, self).__init__()

        self.conv1 = tf.keras.layers.Conv2D(32, (5, 5), activation='relu', padding='same')
        self.conv2 = tf.keras.layers.Conv2D(64, (5, 5), activation='relu', padding='same')
        # self.conv3 = tf.keras.layers.Conv2D(128, (5, 5), activation='relu', padding='sam
        self.fc1 = tf.keras.layers.Dense(256, activation='relu')
        self.fc2 = tf.keras.layers.Dense(10, activation=None)
        self.max_pool = tf.keras.layers.MaxPooling2D((2, 2), (2, 2))
        self.flatten = tf.keras.layers.Flatten()

    def call(self, inp):

        out = self.conv1(inp)
        out = self.max_pool(out)
        out = self.conv2(out)
        out = self.max_pool(out)
        out = self.flatten(out)
        out = self.fc1(out)
        out = self.fc2(out)

        return out

model = Model()
model2 = Model()
model3 = Model()
model4 = Model()
model5 = Model()


def loss(logits, labels):
    return tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(
        logits=logits, labels=labels))

def accuracy(logits, labels):
```

```
    predictions = tf.argmax(logits, axis=1, output_type=tf.int32)
    return tf.reduce_mean(tf.cast(tf.equal(predictions, labels), dtype=tf.float32))

LEARNING_RATE = 0.001
optimizer = tf.keras.optimizers.SGD(LEARNING_RATE)
# optimizer = tf.keras.optimizers.RMSprop(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adamax(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adam(LEARNING_RATE)

writer = tf.summary.create_file_writer('logs/sgd')
# writer = tf.summary.create_file_writer('logs/RMSprop')
# writer = tf.summary.create_file_writer('logs/Adamax')
# writer = tf.summary.create_file_writer('logs/adam')
```

Цикл обучения модели

```
%%time

for iteration, (images, labels) in enumerate(train_ds):

    # Forward
    with tf.GradientTape() as tape:
        logits = model(images)
        loss_value = loss(logits, labels)

    # Backward
    grads = tape.gradient(loss_value, model.trainable_variables)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))

    # Calc and display loss/accuracy
    if iteration % 200 == 0:
        test_logits = model(test_x[:256, ...])
        accuracy_value = accuracy(test_logits, test_y[:256, ...])

        print("[%4d] Accuracy: %5.2f %%" % (
            iteration, accuracy_value.numpy()*100))

        with writer.as_default():
            tf.summary.scalar('accuracy', accuracy_value, iteration)
            tf.summary.scalar('loss', loss_value, iteration)
```

```
[   0] Accuracy: 13.67 %
[ 200] Accuracy: 33.20 %
[ 400] Accuracy: 47.66 %
[ 600] Accuracy: 60.16 %
[ 800] Accuracy: 68.36 %
[1000] Accuracy: 76.95 %
[1200] Accuracy: 78.91 %
[1400] Accuracy: 80.08 %
[1600] Accuracy: 82.81 %
[1800] Accuracy: 85.16 %
[2000] Accuracy: 87.50 %
[2200] Accuracy: 89.45 %
[2400] Accuracy: 89.84 %
```

```
[2600] Accuracy: 91.02 %
[2800] Accuracy: 91.80 %
[3000] Accuracy: 91.80 %
[3200] Accuracy: 91.80 %
[3400] Accuracy: 92.58 %
[3600] Accuracy: 93.36 %
[3800] Accuracy: 91.80 %
[4000] Accuracy: 92.97 %
[4200] Accuracy: 92.58 %
[4400] Accuracy: 92.97 %
[4600] Accuracy: 94.14 %
CPU times: user 43.2 s, sys: 1.54 s, total: 44.7 s
Wall time: 48.1 s
```

Оценка качества модели

```
df = df.append({'accuracy': accuracy_value.numpy() * 100, 'optimizers': str(optimizer).spl
```

```
LEARNING_RATE = 0.001
# optimizer = tf.keras.optimizers.SGD(LEARNING_RATE)
optimizer = tf.keras.optimizers.RMSprop(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adamax(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adam(LEARNING_RATE)

# writer = tf.summary.create_file_writer('logs/sgd')
writer = tf.summary.create_file_writer('logs/RMSprop')
# writer = tf.summary.create_file_writer('logs/Adamax')
# writer = tf.summary.create_file_writer('logs/adam')
```

```
%%time

for iteration, (images, labels) in enumerate(train_ds):

    # Forward
    with tf.GradientTape() as tape:
        logits = model2(images)
        loss_value = loss(logits, labels)

    # Backward
    grads = tape.gradient(loss_value, model2.trainable_variables)
    optimizer.apply_gradients(zip(grads, model2.trainable_variables))

    # Calc and display loss/accuracy
    if iteration % 200 == 0:
        test_logits = model2(test_x[:256, ...])
        accuracy_value = accuracy(test_logits, test_y[:256, ...])

        print("[%4d] Accuracy: %5.2f %%" % (
            iteration, accuracy_value.numpy()*100))

        with writer.as_default():
            tf.summary.scalar('accuracy', accuracy_value, iteration)
            tf.summary.scalar('loss', loss_value, iteration)
```

```
[   0] Accuracy:  9.38 %
[ 200] Accuracy: 99.61 %
[ 400] Accuracy: 96.09 %
[ 600] Accuracy: 99.61 %
[ 800] Accuracy: 99.22 %
[1000] Accuracy: 99.61 %
[1200] Accuracy: 98.83 %
[1400] Accuracy: 99.61 %
[1600] Accuracy: 100.00 %
[1800] Accuracy: 100.00 %
[2000] Accuracy: 99.61 %
[2200] Accuracy: 99.22 %
[2400] Accuracy: 100.00 %
[2600] Accuracy: 99.22 %
[2800] Accuracy: 100.00 %
[3000] Accuracy: 100.00 %
[3200] Accuracy: 99.61 %
[3400] Accuracy: 99.22 %
[3600] Accuracy: 99.61 %
[3800] Accuracy: 100.00 %
[4000] Accuracy: 99.22 %
[4200] Accuracy: 99.22 %
[4400] Accuracy: 99.61 %
[4600] Accuracy: 99.61 %
CPU times: user 1min 11s, sys: 1.06 s, total: 1min 12s
Wall time: 1min 22s
```

```
df = df.append({'accuracy': accuracy_value.numpy() * 100, 'optimizers': str(optimizer).spl
```

```
LEARNING_RATE = 0.001
# optimizer = tf.keras.optimizers.SGD(LEARNING_RATE)
optimizer = tf.keras.optimizers.RMSprop(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adamax(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adam(LEARNING_RATE)

# writer = tf.summary.create_file_writer('logs/sgd')
writer = tf.summary.create_file_writer('logs/RMSprop')
# writer = tf.summary.create_file_writer('logs/Adamax')
# writer = tf.summary.create_file_writer('logs/adam')
```

```
%%time

for iteration, (images, labels) in enumerate(train_ds):

    # Forward
    with tf.GradientTape() as tape:
        logits = model3(images)
        loss_value = loss(logits, labels)

    # Backward
    grads = tape.gradient(loss_value, model3.trainable_variables)
    optimizer.apply_gradients(zip(grads, model3.trainable_variables))

    # Calc and display loss/accuracy
```

```
    if iteration % 200 == 0:
        test_logits = model3(test_x[:256, ...])
        accuracy_value = accuracy(test_logits, test_y[:256, ...])

        print("[%4d] Accuracy: %5.2f %%" % (
            iteration, accuracy_value.numpy()*100))

        with writer.as_default():
            tf.summary.scalar('accuracy', accuracy_value, iteration)
            tf.summary.scalar('loss', loss_value, iteration)
```

```
[   0] Accuracy: 12.50 %
[ 200] Accuracy: 99.61 %
[ 400] Accuracy: 98.44 %
[ 600] Accuracy: 100.00 %
[ 800] Accuracy: 100.00 %
[1000] Accuracy: 99.22 %
[1200] Accuracy: 100.00 %
[1400] Accuracy: 100.00 %
[1600] Accuracy: 100.00 %
[1800] Accuracy: 99.61 %
[2000] Accuracy: 100.00 %
[2200] Accuracy: 99.61 %
[2400] Accuracy: 100.00 %
[2600] Accuracy: 100.00 %
[2800] Accuracy: 100.00 %
[3000] Accuracy: 100.00 %
[3200] Accuracy: 99.61 %
[3400] Accuracy: 100.00 %
[3600] Accuracy: 99.61 %
[3800] Accuracy: 99.22 %
[4000] Accuracy: 99.22 %
[4200] Accuracy: 99.22 %
[4400] Accuracy: 99.61 %
[4600] Accuracy: 99.61 %
CPU times: user 1min 10s, sys: 991 ms, total: 1min 11s
Wall time: 1min 10s
```

```
df = df.append({'accuracy': accuracy_value.numpy() * 100, 'optimizers': str(optimizer).spl
```

```
LEARNING_RATE = 0.001
# optimizer = tf.keras.optimizers.SGD(LEARNING_RATE)
# optimizer = tf.keras.optimizers.RMSprop(LEARNING_RATE)
optimizer = tf.keras.optimizers.Adamax(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adam(LEARNING_RATE)

# writer = tf.summary.create_file_writer('logs/sgd')
# writer = tf.summary.create_file_writer('logs/RMSprop')
writer = tf.summary.create_file_writer('logs/Adamax')
# writer = tf.summary.create_file_writer('logs/adam')
```

```
%%time

for iteration, (images, labels) in enumerate(train_ds):
```

```
    # Forward
    with tf.GradientTape() as tape:
        logits = model4(images)
        loss_value = loss(logits, labels)

    # Backward
    grads = tape.gradient(loss_value, model4.trainable_variables)
    optimizer.apply_gradients(zip(grads, model4.trainable_variables))

    # Calc and display loss/accuracy
    if iteration % 200 == 0:
        test_logits = model4(test_x[:256, ...])
        accuracy_value = accuracy(test_logits, test_y[:256, ...])

        print("[%4d] Accuracy: %5.2f %%" % (
            iteration, accuracy_value.numpy()*100))

        with writer.as_default():
            tf.summary.scalar('accuracy', accuracy_value, iteration)
            tf.summary.scalar('loss', loss_value, iteration)
```

```
[   0] Accuracy: 12.50 %
[ 200] Accuracy: 97.27 %
[ 400] Accuracy: 99.61 %
[ 600] Accuracy: 99.22 %
[ 800] Accuracy: 99.61 %
[1000] Accuracy: 99.22 %
[1200] Accuracy: 98.05 %
[1400] Accuracy: 99.61 %
[1600] Accuracy: 100.00 %
[1800] Accuracy: 100.00 %
[2000] Accuracy: 99.61 %
[2200] Accuracy: 100.00 %
[2400] Accuracy: 99.61 %
[2600] Accuracy: 100.00 %
[2800] Accuracy: 99.61 %
[3000] Accuracy: 99.61 %
[3200] Accuracy: 100.00 %
[3400] Accuracy: 100.00 %
[3600] Accuracy: 100.00 %
[3800] Accuracy: 100.00 %
[4000] Accuracy: 100.00 %
[4200] Accuracy: 100.00 %
[4400] Accuracy: 100.00 %
[4600] Accuracy: 100.00 %
CPU times: user 42.7 s, sys: 595 ms, total: 43.3 s
Wall time: 42.5 s
```

```
df = df.append({'accuracy': accuracy_value.numpy() * 100, 'optimizers': str(optimizer).spl
```

```
LEARNING_RATE = 0.001
# optimizer = tf.keras.optimizers.SGD(LEARNING_RATE)
# optimizer = tf.keras.optimizers.RMSprop(LEARNING_RATE)
# optimizer = tf.keras.optimizers.Adamax(LEARNING_RATE)
optimizer = tf.keras.optimizers.Adam(LEARNING_RATE)
```

```
# writer = tf.summary.create_file_writer('logs/sgd')
# writer = tf.summary.create_file_writer('logs/RMSprop')
# writer = tf.summary.create_file_writer('logs/Adamax')
writer = tf.summary.create_file_writer('logs/adam')


%%time

for iteration, (images, labels) in enumerate(train_ds):

    # Forward
    with tf.GradientTape() as tape:
        logits = model5(images)
        loss_value = loss(logits, labels)

    # Backward
    grads = tape.gradient(loss_value, model5.trainable_variables)
    optimizer.apply_gradients(zip(grads, model5.trainable_variables))

    # Calc and display loss/accuracy
    if iteration % 200 == 0:
        test_logits = model5(test_x[:256, ...])
        accuracy_value = accuracy(test_logits, test_y[:256, ...])

        print("[%4d] Accuracy: %5.2f %%" % (
            iteration, accuracy_value.numpy()*100))

        with writer.as_default():
            tf.summary.scalar('accuracy', accuracy_value, iteration)
            tf.summary.scalar('loss', loss_value, iteration)
```

```
[   0] Accuracy: 25.78 %
[ 200] Accuracy: 98.83 %
[ 400] Accuracy: 100.00 %
[ 600] Accuracy: 98.44 %
[ 800] Accuracy: 99.22 %
[1000] Accuracy: 99.61 %
[1200] Accuracy: 100.00 %
[1400] Accuracy: 100.00 %
[1600] Accuracy: 100.00 %
[1800] Accuracy: 99.61 %
[2000] Accuracy: 100.00 %
[2200] Accuracy: 100.00 %
[2400] Accuracy: 99.22 %
[2600] Accuracy: 99.61 %
[2800] Accuracy: 99.61 %
[3000] Accuracy: 100.00 %
[3200] Accuracy: 100.00 %
[3400] Accuracy: 100.00 %
[3600] Accuracy: 100.00 %
[3800] Accuracy: 99.61 %
[4000] Accuracy: 98.83 %
[4200] Accuracy: 100.00 %
[4400] Accuracy: 100.00 %
[4600] Accuracy: 100.00 %
```

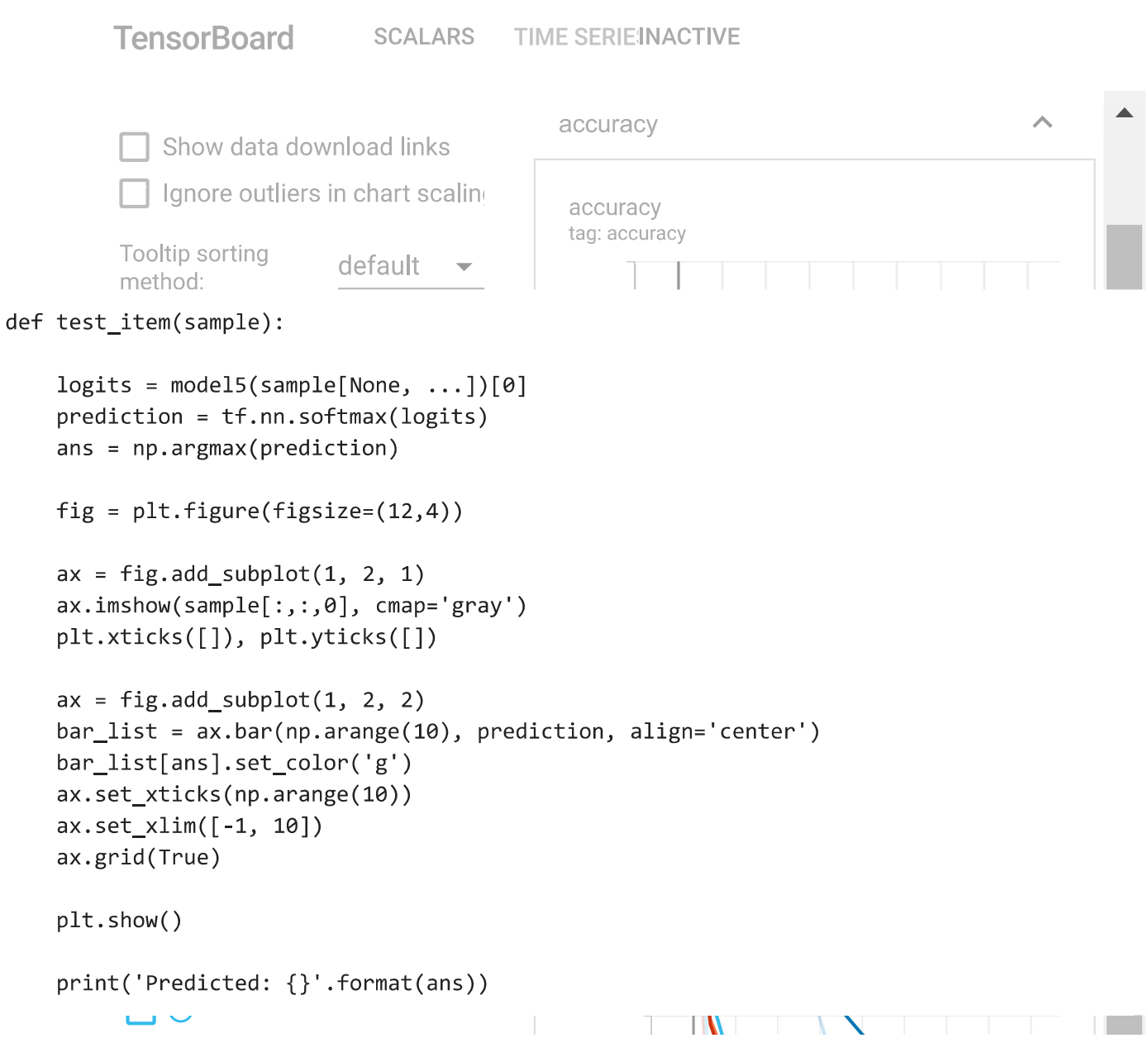
```
CPU times: user 42.5 s, sys: 591 ms, total: 43.1 s
Wall time: 42.3 s
```

```
df = df.append({'accuracy': accuracy_value.numpy() * 100, 'optimizers': str(optimizer).spl
```

```
df.head()
```

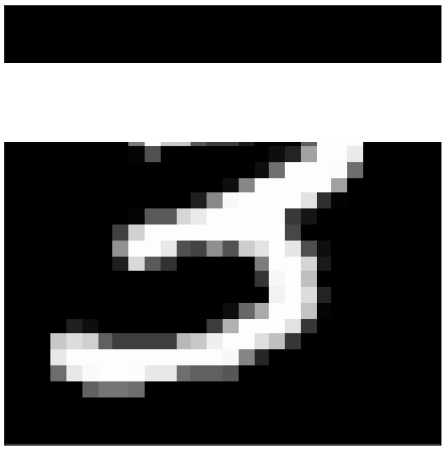
	optimizers	accuracy
0	SGD	94.140625
1	RMSprop	99.609375
2	RMSprop	99.609375
3	Adamax	100.000000
4	Adam	100.000000

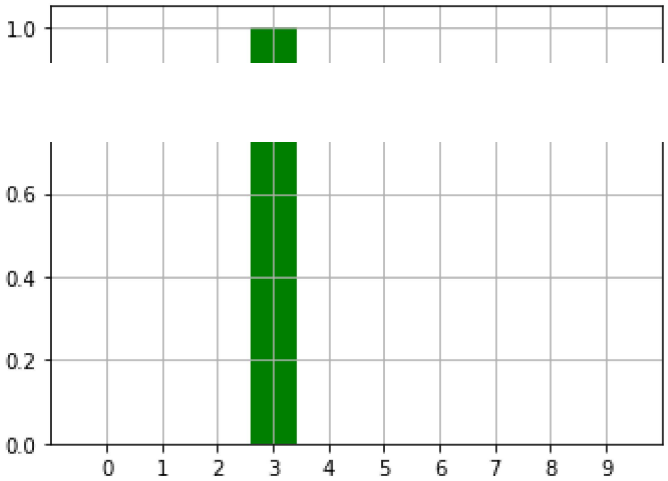
```
%load_ext tensorboard
%tensorboard --logdir logs
```




```
def test_item(sample):

    logits = model5(sample[None, ...])[0]
    prediction = tf.nn.softmax(logits)
    ans = np.argmax(prediction)

    fig = plt.figure(figsize=(12,4))

    ax = fig.add_subplot(1, 2, 1)
    ax.imshow(sample[:,:,0], cmap='gray')
    plt.xticks([]), plt.yticks([])

    ax = fig.add_subplot(1, 2, 2)
    bar_list = ax.bar(np.arange(10), prediction, align='center')
    bar_list[ans].set_color('g')
    ax.set_xticks(np.arange(10))
    ax.set_xlim([-1, 10])
    ax.grid(True)

    plt.show()

    print('Predicted: {}'.format(ans))
```

```
import random
idx = random.randint(0, test_x.shape[0])
sample = test_x[idx, ...]
test_item(sample)

print('True Answer: {}'.format(test_y[idx]))
```






```
Predicted: 3
True Answer: 3
```

Платные продукты Colab - Отменить подписку

✓ 0 сек. выполнено в 03:49