Домашнее задание №5

5.1 Обучить Агента решать Acrobot-v1, MountainCar-v0, или LunarLander-v2 (одну на выбор) методом DQN. Найти оптимальные гиперпараметры. Сравнить с алгоритмом Deep Cross-Entropy на графиках.

При выполнении данного задания файлы .py названы по названию алгоритмов.

Импорт необходимых библиотек. Алгоритмы реализованы в соответствующих .py файлах.

In [1]:
```
import gym
import matplotlib.pyplot as plt
import numpy as np
from DeepCrossEntropy import DeepCrossEntropy
from DQN import DQN
from DQN_HTN import DQN_HTN
from DQN_STN import DQN_STN
from DQN_DDQN import DQN_DDQN
from scipy.interpolate import make_interp_spline
```

```
Warning: Gym version v0.24.0 has a number of critical issues with `gym.make` such that the `reset` and `step` fu
nctions are called before returning the environment. It is recommend to downgrading to v0.23.1 or upgrading to v
0.25.1
```

In [2]:
```
dqn = DQN(env=gym.make('LunarLander-v2'), lr=0.0005, n_episode=300, n_neurons=128, eps_decay=0.95)
dqn.fit()
```

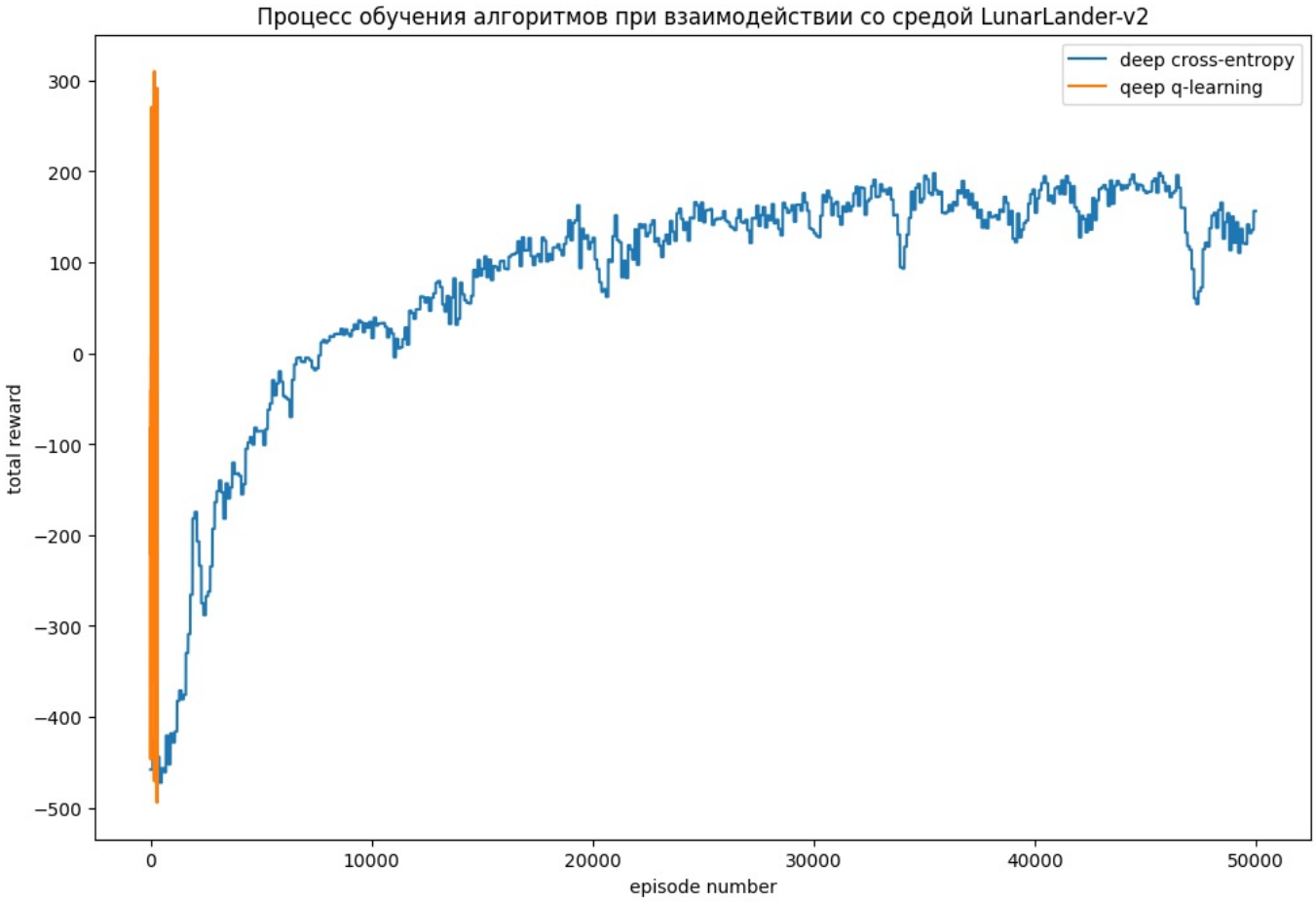
In [10]:
```
dce = DeepCrossEntropy(gym.make("LunarLander-v2"), q=0.8, n_trajectories=100, n_episode=500, n_neurons=128,
                       eps_discount=0.05, lr=0.01, is_print=False)
dce.fit()
```

Награды для алгоритма кросс-энтропии необходимо преобразовать, поскольку для получения каждой оценки необходимо 400 траекторий.

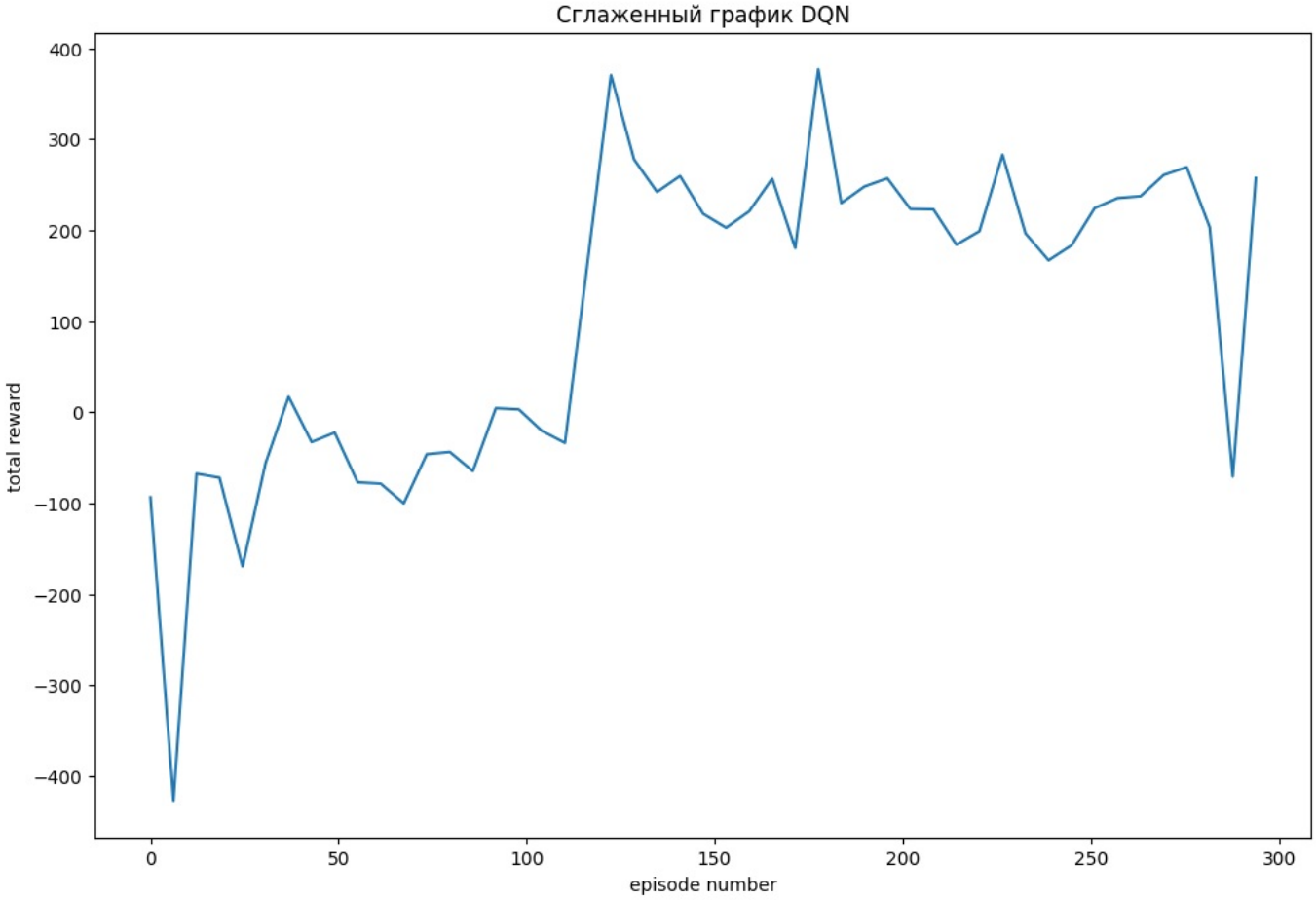
In [18]:
```
dce_mean_total_rewards_50 = []
for reward in dce.mean_total_rewards:
    for _ in range(100):
        dce_mean_total_rewards_50.append(reward)
```

В целях приемлемой визуализации выведем на отображение награду на каждой 10 итерации

In [20]:
```
plt.figure(figsize = (12, 8))
plt.plot(dce_mean_total_rewards_50, label='deep cross-entropy')
plt.plot(dqn.mean_total_rewards, label='qeep q-learning')
plt.title('Процесс обучения алгоритмов при взаимодействии со средой LunarLander-v2')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.legend()
plt.show()
```



In [62]:
```
plt.figure(figsize = (12, 8))
spl = make_interp_spline(list(range(300)), dqn.mean_total_rewards, k=3)
xnew = np.linspace(0, 300, 50)
dqn_smooth = spl(xnew)
plt.plot(xnew[:49], dqn_smooth[:49])
plt.title('Сглаженный график DQN')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.show()
```



Выводы по заданию 1:

Очевидно, что алгоритм DQN сходится за значительно меньшее количество обращений к среде (получении траекторий).

5.2 Реализовать с сравнить (на выбранной ранее среде) друг с другом и с обычным DQN следующие его модификации:

- DQN с Hard Target Update;
- DQN с Soft Target Update;
- Double DQN.

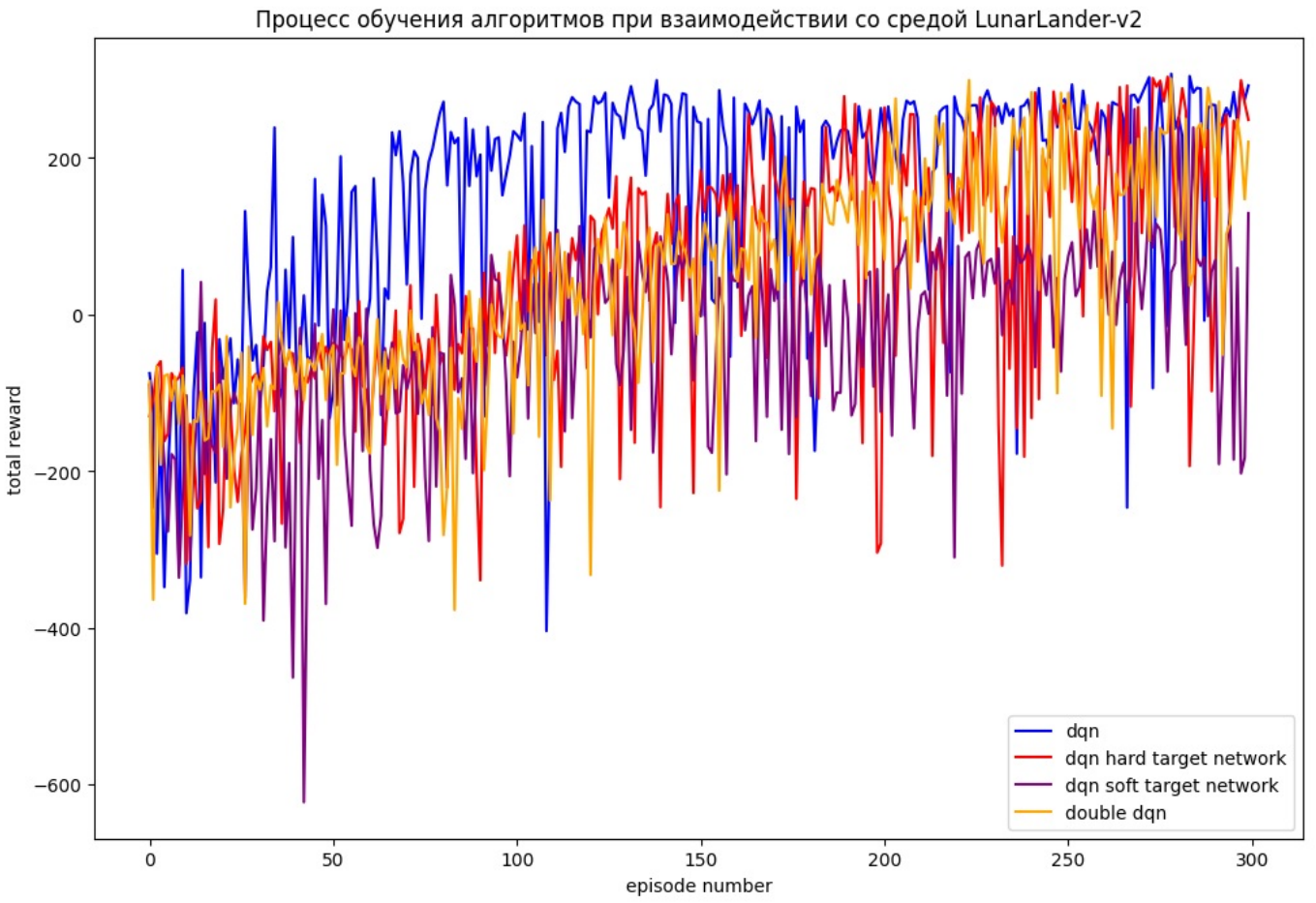
Алгоритмы реализованы в соответствующих файлах.

In [3]:
```
dqn_htn = DQN_HTN(env=gym.make('LunarLander-v2'), lr=0.001, n_episode=300, n_neurons=128, eps_decay=0.99)
dqn_htn.fit()
```

In [4]:
```
dqn_stn = DQN_STN(env=gym.make('LunarLander-v2'), lr=0.001, n_episode=300, n_neurons=128, eps_decay=0.99, tau=0
dqn_stn.fit()
```

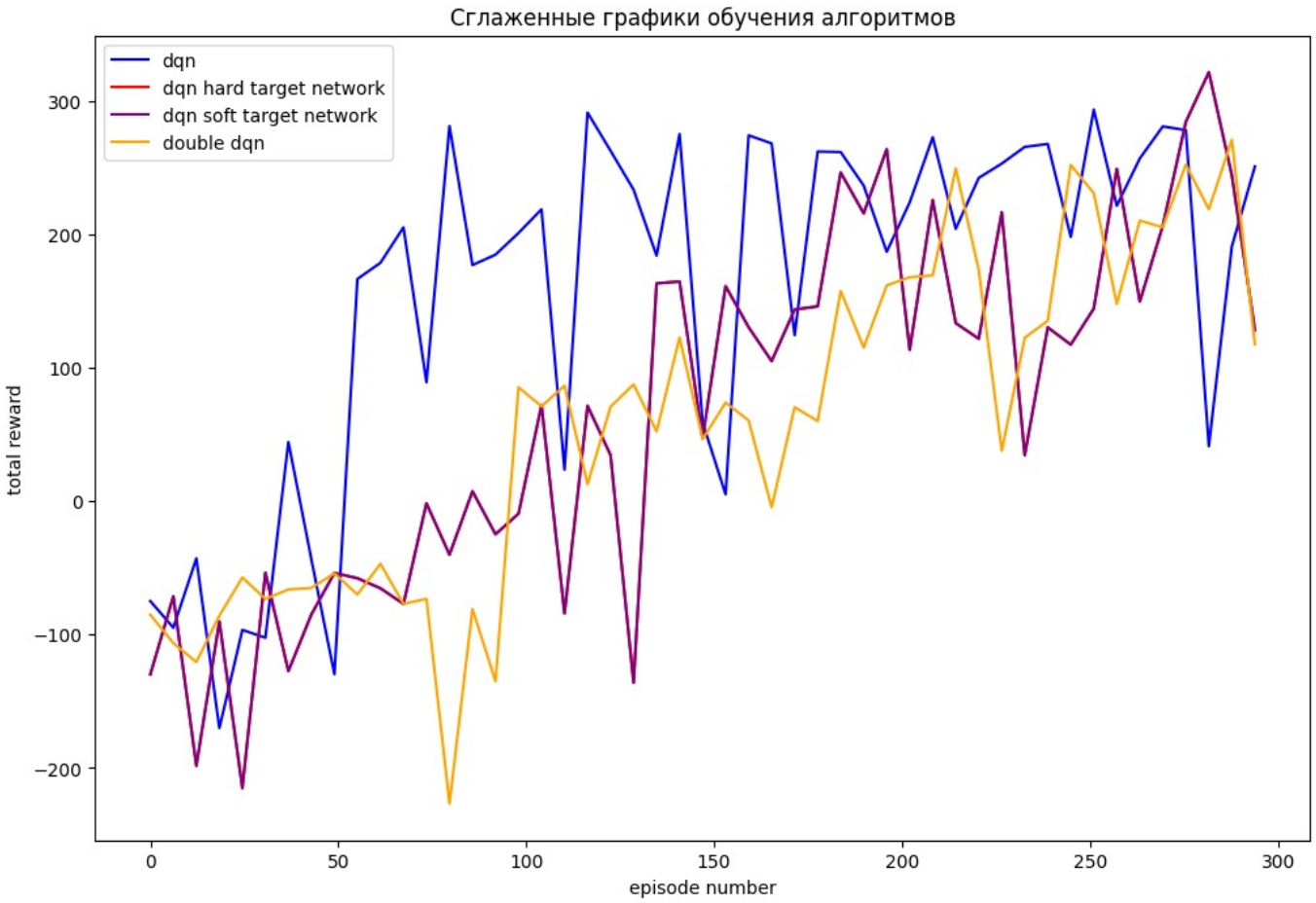
In [5]:
```
dqn_ddqn = DQN_DDQN(env=gym.make('LunarLander-v2'), lr=0.001, n_episode=300, n_neurons=128, eps_decay=0.99, tau=
dqn_ddqn.fit()
```

In [6]:
```
plt.figure(figsize = (12, 8))
plt.plot(dqn.mean_total_rewards, label='dqn', color='b')
plt.plot(dqn_htn.mean_total_rewards, label='dqn hard target network', color='r')
plt.plot(dqn_stn.mean_total_rewards, label='dqn soft target network', color='purple')
plt.plot(dqn_ddqn.mean_total_rewards, label='double dqn', color='orange')
plt.title('Процесс обучения алгоритмов при взаимодействии со средой LunarLander-v2')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.legend()
plt.show()
```



In [8]:
```
plt.figure(figsize = (12, 8))
spl = make_interp_spline(list(range(300)), dqn.mean_total_rewards, k=3)
xnew = np.linspace(0, 300, 50)
dqn_smooth = spl(xnew)
plt.plot(xnew[:49], dqn_smooth[:49], label='dqn', color='b')

spl = make_interp_spline(list(range(300)), dqn_htn.mean_total_rewards, k=3)
dqn_htn_smooth = spl(xnew)
```

```
plt.plot(xnew[:49], dqn_htn_smooth[:49], label='dqn hard target network', color='r')

spl = make_interp_spline(list(range(300)), dqn_stn.mean_total_rewards, k=3)
dqn_stn_smooth = spl(xnew)
plt.plot(xnew[:49], dqn_htn_smooth[:49], label='dqn soft target network', color='purple')

spl = make_interp_spline(list(range(300)), dqn_ddqn.mean_total_rewards, k=3)
dqn_ddqn_smooth = spl(xnew)
plt.plot(xnew[:49], dqn_ddqn_smooth[:49], label='double dqn', color='orange')

plt.title('Сглаженные графики обучения алгоритмов')
plt.xlabel('episode number')
plt.ylabel('total reward')
plt.legend()
plt.show()
```



Выводы по заданию 2:

Очевидно, что все алгоритмы сходятся достаточно быстро и их применение может зависеть от решаемой задачи и точного подбора гиперпараметров.

In [ ]: