In [76]:

```
from cartpole import CartPoleEnv
import numpy as np
import random
import matplotlib.pyplot as plt

env = CartPoleEnv()
env.reset()



def discretize(val,bounds,n_states):
    discrete_val = 0
    if val <= bounds[0]:
        discrete_val = 0
    elif val >= bounds[1]:
        discrete_val = n_states-1
    else:
        discrete_val = int(round((n_states-1)*((val-bounds[0])/(bounds[
1]-bounds[0]))))
    return discrete_val



def discretize_state(vals, s_bounds, n_s):
    discrete_vals = []
    for i in range(len(n_s)):
        discrete_vals.append(discretize(vals[i], s_bounds[i], n_s[i]))
    return np.array(discrete_vals, dtype=np.int)



def epsilon_greedy_action(env, Q, state, epsilon=0.3):
    n = random.uniform(0, 1)
    if n<= epsilon:
        return np.random.randint(env.action_space.n)
    else:
```

```
        return np.argmax(Q[tuple(state)])



# In[11]:


def Q_learning(env, episodes=1000, gamma=0.91, alpha=0.1):
    n_s = np.array([7, 7, 7, 7])
    n_a = env.action_space.n
    Q = np.zeros(np.append(n_s, n_a))

    # tablica zwierająca granice przedziałów
    s_bounds = np.array(list(zip(env.observation_space.low, env.observa
tion_space.high)))
    s_bounds[1] = (-1.0, 1.0)
    s_bounds[3] = (-1.0, 1.0)
    # konieczna konwersja typu
    s_bounds = np.dtype('float64').type(s_bounds)

    #tablica do której zapisujemy sumę nagród z każdego epizodu
    Rewards = []

    for i in range(episodes):
        finished = False
        obs = env.reset()



        S = discretize_state(obs, s_bounds, n_s)

        episode_reward = 0
        time_step=0

        #print("======================================== ", i)
        while not finished and not time_step==200:
            A = epsilon_greedy_action(env, Q, S)
            obs,R,finished,info = env.step(A)
```

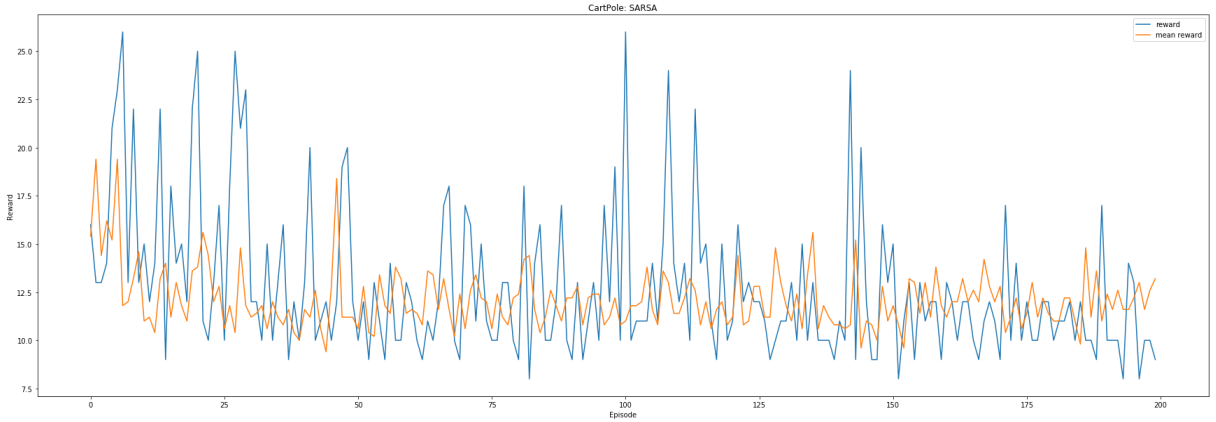
```
            next_S = discretize_state(obs,s_bounds,n_s)

            episode_reward += R

            indices = tuple(np.append(S,A))
            next_indices = tuple(np.append(next_S,A))


            Q[indices] += alpha * (R + gamma * np.max(Q[next_indices])
- Q[indices])


            S = next_S


            time_step += 1

        Rewards.append(episode_reward)

    return Q, Rewards
```

In [79]:
```
learning_episodes = 1000
Q, R = Q_learning(env,learning_episodes)
print(np.max(Q))  #zaokrąglamy do dwóch miejsc po przecinku
```

```
6.628139022491609
```

In [92]:
```
meanR= []
my_range = 200
for i in range(my_range):
    meanR.append(np.mean(R[int(learning_episodes/my_range)*i:int(learni
ng_episodes/my_range)*(i+1)]))


x_data = range(0,my_range)
plt.figure(figsize=(30,10))

plt.plot(x_data,R[0:my_range],label="reward")
```

```
plt.plot(x_data,meanR,label="mean reward")
plt.title('CartPole: SARSA')
plt.xlabel('Episode')
plt.ylabel('Reward')
plt.legend()

plt.show()



env.close()
```




In [ ]:

In [ ]: