Защищено: Гапанюк Ю.Е.		Демонстрация: Гапанюк Ю.Е.			
""	2024 г.	""	2024 г.		
	по лабораторной работ Методы машинного об		y		
Тема работы	: '' Обучение на осново	е временны'х ј	различий ''		
	11 (количество листов) <u>Вариант № 15</u>				
	ИСПОЛНИТЕЛЬ: студент группы ИУ5-22М Чиварзин А.Е.	(подпис	ъ)2024 г.		
	Москва, МГТУ - 202	4			

Задание:

На основе рассмотренного на лекции примера реализуйте следующие алгоритмы:

- SARSA
- Q-обучение
- Двойное Q-обучение

для любой среды обучения с подкреплением (кроме рассмотренной на лекции среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки).

Ход выполнения работы

```
In [3]:
import numpy as np
import matplotlib.pyplot as plt
import gym
from tqdm import tqdm
class BasicAgent:
  Базовый агент, от которого наследуются стратегии обучения
  # Наименование алгоритма
  ALGO_NAME = '---'
  def__init__(self, env, eps=0.1):
    #Среда
    self.env = env
    \# Размерности Q-матрицы
    self.nA = env.action_space.n
    self.nS = env.observation space.n
    #и сама матрица
    self.Q = np.zeros((self.nS, self.nA))
    #Значения коэффициентов
    #Порог выбора случайного действия
    self.eps=eps
    #Награды по эпизодам
    self.episodes_reward = []
  def print q(self):
    print('Вывод Q-матрицы для алгоритма ', self.ALGO NAME)
    print(self.Q)
  def get_state(self, state):
    Возвращает правильное начальное состояние
    if type(state) is tuple:
      #Если состояние вернулось с виде кортежа, то вернуть только номер состояния
      return state[0]
      return state
  def greedy(self, state):
    <<Жадное>> текущее действие
```

```
для состояния state
    return np.argmax(self.Q[state])
  def make action(self, state):
    Выбор действия агентом
    if np.random.uniform(0,1) < self.eps:
      #Если вероятность меньше ерѕ
      # то выбирается случайное действие
      return self.env.action space.sample()
    else:
      # иначе действие, соответствующее максимальному Q-значению
      return self.greedy(state)
  def draw episodes reward(self):
    #Построение графика наград по эпизодам
    fig, ax = plt.subplots(figsize = (15,10))
    y = self.episodes reward
    x = list(range(1, len(y)+1))
    plt.plot(x, y, '-', linewidth=1, color='green')
    plt.title('Награды по эпизодам')
    plt.xlabel('Номер эпизода')
    plt.ylabel('Награда')
    plt.show()
  def learn():
    Реализация алгоритма обучения
    pass
In [5]:
class SARSA_Agent(BasicAgent):
 Реализация алгоритма SARSA
  # Наименование алгоритма
  ALGO NAME = 'SARSA'
  def__init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
    #Вызов конструктора верхнего уровня
    super().__init__(env, eps)
    #Learning rate
    self.lr=lr
    #Коэффициент дисконтирования
    self.gamma = gamma
    #Количество эпизодов
    self.num episodes=num episodes
    #Постепенное уменьшение ерѕ
    self.eps_decay=0.00005
    self.eps_threshold=0.01
  def learn(self):
    Обучение на основе алгоритма SARSA
```

Возвращает действие, соответствующее максимальному Q-значению

```
self.episodes reward = []
    #Цикл по эпизодам
    for ep in tqdm(list(range(self.num episodes))):
      # Начальное состояние среды
      state = self.get state(self.env.reset())
      # Флаг штатного завершения эпизода
      done = False
      # Флаг нештатного завершения эпизода
      truncated = False
      # Суммарная награда по эпизоду
      tot rew = 0
      #По мере заполнения Q-матрицы уменьшаем вероятность случайного выбора действия
      if self.eps > self.eps threshold:
        self.eps -= self.eps decay
      #Выбор действия
      action = self.make action(state)
      #Проигрывание одного эпизода до финального состояния
      while not (done or truncated):
        #Выполняем шаг в среде
        next state, rew, done, truncated, = self-env.step(action)
        #Выполняем следующее действие
        next action = self.make action(next state)
        #Правило обновления Q для SARSA
        self.Q[state][action] = self.Q[state][action] + self.lr * \
           (rew + self.gamma * self.Q[next state][next action] - self.Q[state][action])
        # Следующее состояние считаем текущим
        state = next_state
        action = next action
        # Суммарная награда за эпизод
        tot rew += rew
        if (done or truncated):
           self.episodes_reward.append(tot_rew)
In [6]:
     class QLearning_Agent(BasicAgent):
 Реализация алгоритма Q-Learning
  # Наименование алгоритма
  ALGO NAME = 'Q-обучение'
  def__init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
    #Вызов конструктора верхнего уровня
    super(). init (env, eps)
    #Learning rate
    self.lr=lr
    #Коэффициент дисконтирования
    self.gamma = gamma
    #Количество эпизодов
    self.num episodes=num episodes
    #Постепенное уменьшение ерѕ
    self.eps decay=0.00005
    self.eps_threshold=0.01
```

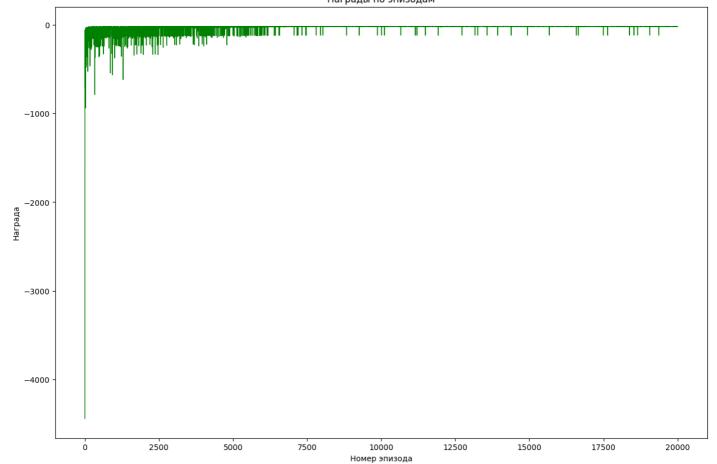
```
def learn(self):
    Обучение на основе алгоритма Q-Learning
    self.episodes reward = []
    #Цикл по эпизодам
    for ep in tqdm(list(range(self.num episodes))):
      # Начальное состояние среды
      state = self.get state(self.env.reset())
      # Флаг штатного завершения эпизода
      done = False
      # Флаг нештатного завершения эпизода
      truncated = False
      # Суммарная награда по эпизоду
      tot rew = 0
      #По мере заполнения О-матрицы уменьшаем вероятность случайного выбора действия
      if self.eps > self.eps threshold:
        self.eps -= self.eps decay
      #Проигрывание одного эпизода до финального состояния
      while not (done or truncated):
        #Выбор действия
        # B SARSA следующее действие выбиралось после шага в среде
        action = self.make action(state)
        #Выполняем шаг в среде
        next_state, rew, done, truncated, _ = self.env.step(action)
        #Правило обновления Q для SARSA (для сравнения)
        # self.Q[state][action] = self.Q[state][action] + self.lr * \
           (rew + self.gamma * self.Q[next_state][next_action] - self.Q[state][action])
        #Правило обновления для Q-обучения
        self.Q[state][action] = self.Q[state][action] + self.lr * \
           (rew + self.gamma * np.max(self.Q[next state]) - self.Q[state][action])
        # Следующее состояние считаем текущим
        state = next state
        # Суммарная награда за эпизод
        tot rew += rew
        if (done or truncated):
           self.episodes_reward.append(tot_rew)
In [7]:
class DoubleQLearning_Agent(BasicAgent):
  Реализация алгоритма Double Q-Learning
  # Наименование алгоритма
  ALGO_NAME = 'Двойное Q-обучение'
  def__init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
    #Вызов конструктора верхнего уровня
    super(). init (env, eps)
    #Вторая матрица
    self.Q2 = np.zeros((self.nS, self.nA))
    #Learning rate
    self.lr=lr
    #Коэффициент дисконтирования
    self.gamma = gamma
    #Количество эпизодов
```

```
self.num episodes=num episodes
  #Постепенное уменьшение ерѕ
  self.eps decay=0.00005
  self.eps threshold=0.01
def greedy(self, state):
  <<Жадное>> текущее действие
  Возвращает действие, соответствующее максимальному Q-значению
  для состояния state
  temp q = self.Q[state] + self.Q2[state]
  return np.argmax(temp q)
def print q(self):
  print('Вывод Q-матриц для алгоритма', self.ALGO NAME)
  print('Q1')
  print(self.Q)
  print('Q2')
  print(self.Q2)
def learn(self):
  Обучение на основе алгоритма Double Q-Learning
  self.episodes reward = []
  #Цикл по эпизодам
  for ep in tqdm(list(range(self.num_episodes))):
    # Начальное состояние среды
    state = self.get state(self.env.reset())
    # Флаг штатного завершения эпизода
    done = False
    # Флаг нештатного завершения эпизода
    truncated = False
    # Суммарная награда по эпизоду
    tot rew = 0
    #По мере заполнения Q-матрицы уменьшаем вероятность случайного выбора действия
    if self.eps > self.eps threshold:
       self.eps -= self.eps_decay
    #Проигрывание одного эпизода до финального состояния
    while not (done or truncated):
       #Выбор действия
       # B SARSA следующее действие выбиралось после шага в среде
       action = self.make action(state)
       #Выполняем шаг в среде
       next_state, rew, done, truncated, _ = self.env.step(action)
       if np.random.rand() < 0.5:
         # Обновление первой таблицы
         self.Q[state][action] = self.Q[state][action] + self.lr * \
            (rew + self.gamma * self.Q2[next state][np.argmax(self.Q[next state])] - self.Q[state][action])
       else:
         # Обновление второй таблицы
         self.Q2[state][action] = self.Q2[state][action] + self.lr * \
            (rew + self.gamma * self.Q[next_state][np.argmax(self.Q2[next_state])] - self.Q2[state][action])
       # Следующее состояние считаем текущим
       state = next state
```

```
# Суммарная награда за эпизод
          tot rew += rew
          if (done or truncated):
             self.episodes reward.append(tot rew)
def play agent(agent):
  Проигрывание сессии для обученного агента
  env2 = gym.make('CliffWalking-v0', render mode='human')
  state = env2.reset()[0]
  done = False
  while not done:
     action = agent.greedy(state)
     next state, reward, terminated, truncated, = env2.step(action)
     env2.render()
     state = next state
     if terminated or truncated:
       done = True
def run sarsa():
  env = gym.make('CliffWalking-v0')
  agent = SARSA Agent(env)
  agent.learn()
  agent.print q()
  agent.draw episodes reward()
  play agent(agent)
def run q learning():
  env = gym.make('CliffWalking-v0')
  agent = QLearning Agent(env)
  agent.learn()
  agent.print q()
  agent.draw episodes reward()
  play agent(agent)
def run double q learning():
  env = gym.make('CliffWalking-v0')
  agent = DoubleQLearning Agent(env)
  agent.learn()
  agent.print_q()
  agent.draw episodes reward()
  play agent(agent)
Запустим код
In [10]:
run_sarsa()
100%
                                                                                                                      20000/20000 [00:07<00:00, 2855.
25it/s]
Вывод Q-матрицы для алгоритма SARSA
[[ -13.28458418 -12.46793538 -14.22885418 -13.27657094]
[ -12.45936175 -11.66533596 -13.34660858 -13.45906628]
[-11.71638403 -10.88101524 -13.39477295 -12.65541078]
[-10.86737962 -10.11829038 -11.93552502 -11.92231622]
[-10.07705676 -9.37897128 -11.23305926 -11.06916088]
[ -9.2474099 -8.4283109 -10.18159789 -10.29406844]
[ -8.52879815 -7.47575527 -9.32193519 -9.43540966]
[ -7.55275583 -6.59704964 -8.722669 -8.63843327]
 -6.72157012 -5.70996807 -7.58792355 -7.72135969]
[ -5.75754608 -4.83684192 -5.44219841 -6.86500434]
[ -4.83800566 -4.03257156 -4.27658203 -5.94011482]
 -3.93797347 -3.91535093 -2.9404 -5.0490357 ]
[-13.2441641 -13.80960639 -14.87261324 -14.00541517]
[-12.42772767 -16.72399429 -20.83693558 -17.26277688]
[-11.87122455 -15.62293863 -19.44586793 -19.62173403]
[ 10.02590105 14.50095000 00.49297071 16.05194509]
```

```
-10.50104.50-14.50004.52- 56000506.41- 661000666.01-
[-10.1791077 -14.34966729 -19.60171914 -17.05191173]
[ -9.25034438 -13.2957176 -19.04127402 -15.47119261]
 -8.39161463 -12.51353545 -23.58611451 -14.26746483]
 -10.64160318 -7.63532978 -16.42032824 -14.50368699]
[ -6.64496295 -10.69126268 -13.03071696 -13.13939572]
 -7.04693285 -3.89162074 -23.314999 -9.67996485]
 -5.94566789 -2.9489369 -11.50318523 -6.49022603]
[ -4.14565422 -2.95381323 -1.98
                                     -3.94771702]
[-13.97411982 -14.32478585 -17.60119391 -14.68641294]
[-13.95947997 -26.97814415 -129.14492458 -20.79811959]
[-16.93797311 -19.79159385 -120.95044482 -22.32781021]
[-17.42446452 -22.84868818 -108.13557952 -19.16393337]
[-15.81244104 -19.68757679 -113.51401088 -18.30851315]
[-14.43815521 -36.3994923 -98.36182787 -18.80266114]
[-12.46281401 -17.23458648 -125.75819787 -20.94416146]
[-11.9020116 -19.1675061 -116.6303524 -16.13398853]
[-10.40603623 -19.12342799 -99.95915931 -22.66556518]
 -8.92808051 -15.89724535 -122.6162213 -14.99129734]
[ -8.6603934 -1.98006908 -126.94319544 -11.89128585]
 -3.16623855 -1.98480229 -1.
                                    -2.94889043]
[-14.68308311-114.49262873 -15.62346404 -15.65281831]
  0.
          0.
                  0.
                          0.
  0.
          0.
                  0.
                           0.
  0.
          0.
                  0.
                           0.
  0.
          0.
                  0.
                           0.
  0.
          0.
                  0.
                           0.
  0.
  0.
          0.
                  0.
                           0.
  0.
          0.
                  0.
                           0.
  0.
  0.
          0
                           0
                  0
[ 0.
          0.
                  0.
                           0.
                                 ]]
```

Награды по эпизодам



/home/chivarzinae/.local/lib/python3.10/site-packages/gym/utils/passive_env_checker.py:233: DeprecationWarning: `np.bool8` is a deprecated alias for `np.bool_`. (Deprecated NumPy 1.24)

if not is instance(terminated, (bool, np.bool8)):

In [12]:

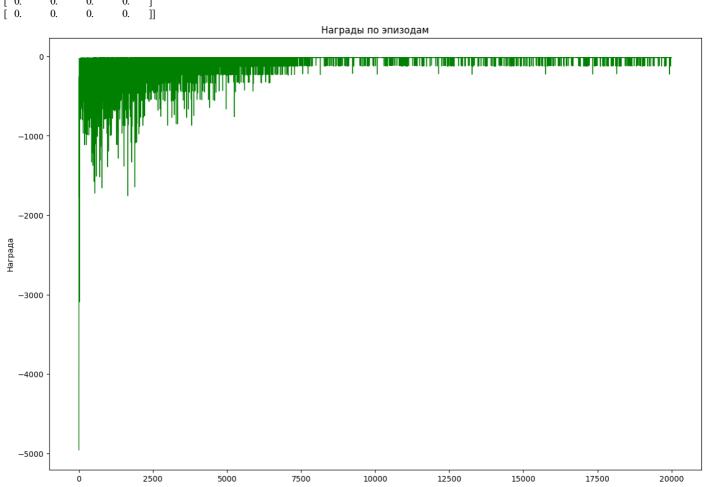
run_q_learning()

```
100% 20000/20000 [00:07<00:00, 2554. 52it/s] Вывод Q-матрицы для алгоритма Q-обучение [[ -12.49766339 -12.31070005 -12.31118573 -12.48057388]
```

[-12.04992222 -11.54861888 -11.54860106 -12.54603438] [-11.46678618 -10.76412877 -10.76412905 -12.16937408]

```
[-10.7141511 -9.96342959 -9.96342936 -11.46688772]
 -9.93713951 -9.14635911 -9.14635906 -10.68894613]
 -9.11813207 -8.31261184 -8.31261185 -9.94427009]
[ -8.29806773 -7.46184886 -7.46184886 -9.13158981]
 -7.45170816 -6.59372334 -6.59372334 -8.28755561]
 -6.59019291 -5.70788096 -5.70788096 -7.45437305]
[ -5.69600763 -4.80396016 -4.80396016 -6.5784164 ]
[\begin{array}{cccc} -4.7856999 & -3.881592 & -3.881592 & -5.66666797 ] \end{array}
 -3.87170624 -3.85779722 -2.9404
                                    -4.75994889]
[-13.05646141 -11.54888054 -11.54888054 -12.31788719]
[-12.31703959 -10.76416381 -10.76416381 -12.31790068]
[-11.54871861 -9.96343246 -9.96343246 -11.54888039]
 -10.76415436 -9.14635966 -9.14635966 -10.76416381]
 -9.96342579 -8.31261189 -8.31261189 -9.96343241]
 -9.14635939 -7.46184887 -7.46184887 -9.14635965]
 -8.31261182 -6.59372334 -6.59372334 -8.31261186]
[ -7.46184883 -5.70788096 -5.70788096 -7.46184671]
[ -6.59372332 -4.80396016 -4.80396016 -6.59372319]
 -5.70788095 -3.881592 -3.881592 -5.70788095]
 -4.80396014 -2.9404
                       -2.9404
                                  -4.80396015]
                       -1.98
 -3.881592 -2.9404
                                -3.88159199]
[-12.31790293 -10.76416381 -12.31790293 -11.54888054]
[-11.54888054 -9.96343246-111.31790293 -11.54888054]
[\ -10.76416381 \ \ -9.14635966 - 111.31790293 \ \ -10.76416381]
[ -9.96343246 -8.31261189 -111.31790293 -9.96343246]
 -9.14635966 -7.46184887 -111.31790293 -9.14635966]
 -8.31261189 -6.59372334 -111.31790293 -8.31261189]
 -7.46184887 -5.70788096 -111.31790293 -7.46184887]
 -6.59372334 -4.80396016 -111.31790293 -6.59372334]
 -5.70788096 -3.881592 -111.31790293 -5.70788096]
 -4.80396016 -2.9404 -111.31790293 -4.80396016]
[ -3.881592 -1.98
                    -111.31790293 -3.881592 ]
                              -2.9404 ]
 -2.9404
           -1.98
                     -1.
[-11.54888054-111.31790293 -12.31790293 -12.31790293]
  0.
                  0.
                          0.
  0.
                           0.
  0.
                           0.
  0.
                           0.
  0.
Γ
  0.
                          0.
  0.
                           0.
[ 0.
                           0.
[ 0.
                  0.
                           0.
[ 0.
                  0.
                           0.
[ 0.
                  0.
                          0.
```





```
In [11]: run_double_q_learning()
```

[-12.31790293 -10.76416381 -12.31790293 -11.54888054]

```
20000/20000 [00:06<00:00, 2969.
40it/s]
Вывод Q-матриц для алгоритма Двойное Q-обучение
[[ -22.37789595 -25.62062474 -12.39039485 -24.71573578]
[-24.1091759 -24.49194872 -11.98006061 -22.78512369]
[-22.62859519 -18.15277024 -10.84750376 -22.53090117]
[ -16.84208179 -10.22899569 -22.63069044 -24.14874837]
[-14.79470827 -11.09195446 -9.16553957 -15.00128277]
[-13.9227748 -13.76944309 -8.32878493 -13.03514537]
[-13.81883274 -13.09098109 -7.81537107 -11.41258451]
[-11.58672958 -12.4383191 -12.65458447 -11.9771373]
[-11.24180165 -5.7079225 -5.15213654 -12.2569396]
[-5.33932226 -4.80396227 -5.66995023 -11.56009641]
[ -4.76171913 -3.881592 -3.89780229 -5.59064732]
[ -3.8598237 -3.86727098 -2.9404 -4.71912904]
[-13.25958484 -11.61431522 -11.54888054 -12.33181141]
[\ -12.66950945\ -10.83529407\ -10.76416381\ -12.45569165]
[-13.88326168 -10.33615197 -9.96343246 -12.20620985]
[-12.79966567 -9.7658957 -9.14635966 -10.94037081]
[-10.53340717 -8.31261189 -8.50763127 -10.54129221]
[ -9.22968496 -7.50688511 -7.46184887 -9.19553911]
[-10.16592719 -7.45203939 -6.59372334 -8.72232692]
[-13.47945415 -5.70788096 -6.07250266 -7.64133735]
[-12.31555366 -4.80396016 -4.9084411 -7.14281158]
[ -5.67713261 -3.881592 -3.8825064 -5.69718511]
[ -4.80333524 -2.9404 -2.94516399 -4.80534725]
[ -3.88157603 -2.94039775 -1.98
                                  -3.88157198]
[-12.31790293 -10.76416381 -12.31790293 -11.54888054]
[-11.54888054 -9.96343246 -111.31790293 -11.54888054]
[-10.76416381 -9.14635966 -111.31790293 -10.76416381]
[ -9.96343246 -8.31261189 -111.31790293 -9.96343246]
[ -9.14635966 -7.46184887 -111.31790293 -9.14635966]
[\ -8.31261189\ \ -6.59372334-111.31790293\ \ -8.31261189]
 -7.46184887 -5.70788096 -111.31790293 -7.46184887]
[ -6.69547538 -4.80396016 -111.31790293 -6.59372334]
[ -5.70788096 -3.881592 -111.31790293 -5.70788096]
[ -4.80396016 -2.9404 -111.31790291 -4.80396016]
-2.9404 ]
[ -2.9404 -1.98
                     -1
[-11.54888054-111.31790293 -12.31790293 -12.31790293]
[ 0.
                          0.
                  0.
                                1
  0.
          0.
                  0.
                          0.
Γ
  0.
ſ
  0.
          0.
                  0.
                          0.
[ 0.
          0.
                  0.
                          0.
[ 0.
          0.
                  0
                          0.
[ 0.
          0.
                  0.
                          0.
                          0.
  0.
                                ]
Γ 0.
          0.
                  0.
                          0.
[ 0.
                  0.
                          0.
[ 0.
                  0.
                          0.
                                ]]
O2
[[ -22.11359144 -24.08695095 -12.34648022 -20.58376148]
[-24.3772731 -20.23936943 -11.60015664 -20.68966162]
[-20.78442806 -22.33907093 -11.32435794 -23.79702714]
[-21.40948399 -10.52538298 -11.8607127 -19.4275484]
[-14.41432868 -14.19766013 -9.15410114 -19.89932976]
[-12.73742277 -12.34065824 -8.31784586 -13.03936663]
[-12.21831473 -13.8877325 -7.87461485 -13.22091431]
[-13.02798502 -6.61729505 -9.58935746 -11.18921565]
[ -7.78057713 -6.67925808 -10.90617805 -12.56648911]
[ -7.20641231 -4.80396014 -4.68174345 -6.94005041]
[ -4.69247715 -3.881592 -3.98608298 -7.38218033]
[ -3.79611695 -3.86943326 -2.9404 -4.77927439]
[-13.88980285 -11.59500634 -11.54888054 -12.51232918]
[ -13.76011821 -11.53302271 -10.76416381 -12.34451383]
[ -12.16494067 -10.17350557 -9.96343246 -11.81916679]
[-15.97823657 -9.3125217 -9.14635966 -11.14385817]
[-10.41644216 -8.31261189 -8.33855528 -10.0650769]
[ -9.22029759 -7.47603917 -7.46184887 -9.1833926 ]
[ -9.32584615 -6.92989503 -6.59372334 -8.43118825]
[-13.52840395 -5.81039313 -6.03702695 -8.19836957]
[ -6.63599923 -4.80396016 -4.79143801 -7.15782243]
[ -6.67385549 -3.881592 -3.89020301 -6.16591246]
[ -4.80252192 -2.9404 -2.94057964 -4.80173266]
[ -3.88155886 -2.94039095 -1.98
                                  -3.881598961
```

L			,			
[-11.54888	3054 -9.9	96343246	-111.3	31790293	-11.54888054]
[-10.76416	6381 -9.	14635966	-111.3	31790293	-10.76416381]
ĺ	-9.96343	246 -8.3	1261189	-111.3	1790293	-9.96343246]
Ī	-9.14635	966 -7.4	6184887	-111.3	1790293	-9.14635966]
Ī	-8.31261	189 -6.5	9372334	-111.3	1790293	-8.31261189]
Ī	-7.46184	887 -5.7	0788096	-111.3	1790293	-7.46184887]
Ī	-6.59372	334 -4.8	0396016	-111.3	1790293	-6.59372334]
Ĭ	-5.81072	552 -3.8	81592 -	111.31	790293 -	5.70788096]
Ĭ						.80396016]
Ĭ					292 -3.8	-
Ĭ			-1.			•
Ĭ						-12.31790293]
Ĭ	0.	0.	0.	0.	1	-
Ĭ	0.	0.	0.	0.	í	
Ĭ	0.	0.	0.	0.	í	
Ĭ	0.	0.	0.	0.	í	
Ī	0.	0.	0.	0.	í	
Ĭ	0.	0.	0.	0.	í	
Ĭ	0.	0.	0.	0.	í	
Ĭ	0.	0.	0.	0.	í	
ĺ	0.	0.	0.	0.	i	
Ĭ	0.	0.	0.	0.	i	
Ī	0.	0.	0.	0.	ii .	



