Tutorial 7 - Options Intro

Please complete this tutorial to get an overview of options and an implementation of SMDP Q-Learning and Intra-Option Q-Learning.

References:

Recent Advances in Hierarchical Reinforcement Learning is a strong recommendation for topics in HRL that was covered in class. Watch Prof. Ravi's lectures on moodle or nptel for further understanding the core concepts. Contact the TAs for further resources if needed.

In [1]:
```
!pip install gym==0.15.3
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
s/public/simple/
Requirement already satisfied: gym==0.15.3 in /usr/local/lib/python3.9/dist-packag
es (0.15.3)
Requirement already satisfied: cloudpickle~=1.2.0 in /usr/local/lib/python3.9/dist
-packages (from gym==0.15.3) (1.2.2)
Requirement already satisfied: pyglet<=1.3.2,>=1.2.0 in /usr/local/lib/python3.9/d
ist-packages (from gym==0.15.3) (1.3.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (fr
om gym==0.15.3) (1.10.1)
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from
gym==0.15.3) (1.16.0)
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.9/dist-pack
ages (from gym==0.15.3) (1.22.4)
Requirement already satisfied: future in /usr/local/lib/python3.9/dist-packages (f
rom pyglet<=1.3.2,>=1.2.0->gym==0.15.3) (0.18.3)
```

In [2]:
```
'''
A bunch of imports, you don't have to worry about these
'''

import numpy as np
import random
import gym
from gym.wrappers import Monitor
import glob
import io
import matplotlib.pyplot as plt
from IPython.display import HTML
```

In [3]:
```
'''
The environment used here is extremely similar to the openai gym ones.
At first glance it might look slightly different.
The usual commands we use for our experiments are added to this cell to aid you
work using this environment.
'''

#Setting up the environment
from gym.envs.toy_text.cliffwalking import CliffWalkingEnv
env = CliffWalkingEnv()

env.reset()

#Current State
```

```
print(env.s)

# 4x12 grid = 48 states
print ("Number of states:", env.nS)

# Primitive Actions
action = ["up", "right", "down", "left"]
#correspond to [0,1,2,3] that's actually passed to the environment

# either go left, up, down or right
print ("Number of actions that an agent can take:", env.nA)

# Example Transitions
rnd_action = random.randint(0, 3)
print ("Action taken:", action[rnd_action])
next_state, reward, is_terminal, t_prob = env.step(rnd_action)
print ("Transition probability:", t_prob)
print ("Next state:", next_state)
print ("Reward recieved:", reward)
print ("Terminal state:", is_terminal)
env.render()
```

```
36
Number of states: 48
Number of actions that an agent can take: 4
Action taken: left
Transition probability: {'prob': 1.0}
Next state: 36
Reward recieved: -1
Terminal state: False
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
x  C  C  C  C  C  C  C  C  C  C  T
```

Options

We custom define very simple options here. They might not be the logical options for this settings deliberately chosen to visualise the Q Table better.

In [4]:

```
# We are defining two more options here
# Option 1 ["Away"] - > Away from Cliff (ie keep going up)
# Option 2 ["Close"] - > Close to Cliff (ie keep going down)

def Away(env,state):

    optdone = False
    optact = 0

    if (int(state/12) == 0):
        optdone = True

    return [optact,optdone]

def Close(env,state):

    optdone = False
    optact = 2

    if (int(state/12) == 2 or int(state/12)==3):
        optdone = True
```

```
    return [optact,optdone]


'''
Now the new action space will contain
Primitive Actions: ["up", "right", "down", "left"]
Options: ["Away","Close"]
Total Actions :["up", "right", "down", "left", "Away", "Close"]
Corresponding to [0,1,2,3,4,5]
'''
```

Out[4]:

```
'\nNow the new action space will contain\nPrimitive Actions: ["up", "right", "down", "left"]\nOptions: ["Away","Close"]\nTotal Actions :["up", "right", "down", "left", "Away", "Close"]\nCorresponding to [0,1,2,3,4,5]\n'
```

Task 1

Complete the code cell below

In [5]:

```
#Q-Table: (States x Actions) === (env.ns(48) x total actions(6))
q_values_SMDP = np.zeros((48,6))

#Update_Frequency Data structure? Check TODO 4
update_freq_SMDP = np.zeros((48,6))

actions=[0,1,2,3,4,5]

seed = 36
rg = np.random.RandomState(seed)

# TODO: epsilon-greedy action selection function
def egreedy_policy(q_values,state,epsilon):
    if rg.rand() < epsilon:
       return rg.choice(actions)
    else:
      #max = np.max(q_values[state])
      #return rg.choice(np.where(q_values[state] == max)[0])
      return np.argmax(q_values[state])
```

Task 2

Below is an incomplete code cell with the flow of SMDP Q-Learning. Complete the cell and train the agent using SMDP Q-Learning algorithm. Keep the **final Q-table** and **Update Frequency** table handy (You'll need it in TODO 4)

In [6]:

```
#### SMDP Q-Learning

# Add parameters you might need here
gamma = 0.9
alpha = 0.4
epsilon = 0.1

Rewards = []

# Iterate over 1000 episodes
for _ in range(1000):
    state = env.reset()
    done = False
```

```
    episode_reward = 0

    # While episode is not over
    while not done:

        # Choose action
        action = egreedy_policy(q_values_SMDP, state, epsilon=0.1)

        # Checking if primitive action
        if action < 4:
            # Perform regular Q-Learning update for state-action pair
            next_state, reward, done,_ = env.step(action)

            q_values_SMDP[state, action] += alpha*(reward + gamma*np.max([q_values_

            update_freq_SMDP[state,action] += 1

            state = next_state

            episode_reward += reward

        # Checking if action chosen is an option
        reward_bar = 0
        if action == 4: # action => Away option

            optdone = False
            while (optdone == False):

                # Think about what this function might do?
                optact,optdone = Away(env,state)
                next_state, reward, done,_ = env.step(optact)

                # Is this formulation right? What is this term?
                reward_bar = gamma*reward_bar + reward

                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                q_values_SMDP[state, action] += alpha*(reward_bar + (gamma)*np.max
                update_freq_SMDP[state,action] += 1

                state = next_state

                episode_reward += reward

        if action == 5: # action => Close option

            optdone = False
            while (optdone == False):

                # Think about what this function might do?
                optact,optdone = Close(env,state)
                next_state, reward, done,_ = env.step(optact)

                # Is this formulation right? What is this term?
                reward_bar = gamma*reward_bar + reward

                #if next_state in [36, 37, 38, 39, 40, 41, 42, 43]: # check if next
                #    done = True

                # Complete SMDP Q-Learning Update
                # Remember SMDP Updates. When & What do you update?
                q_values_SMDP[state, action] += alpha*(reward_bar + (gamma)*np.max
                update_freq_SMDP[state,action] += 1
```

```
                state = next_state

                episode_reward += reward

                #if done: # terminate episode if agent falls in the cliff
                #    break


# Print final Q-table and Update Frequency table
print("Final Q-values:")
print(q_values_SMDP)
print("\nUpdate Frequency:")
print(update_freq_SMDP)
```

```
Final Q-values:
[[  -7.77614476   -7.70678309   -7.70780839   -7.82166587  -10.24870589
    -7.70874639]
 [  -7.62037387   -7.45531804   -7.4551474    -7.80709603   -8.99224897
    -7.45509322]
 [  -7.22877075   -7.17324595   -7.17388968   -7.5164836    -9.16401083
    -7.17417032]
 [  -7.103383     -6.8600376    -6.86026604   -7.23582774   -7.90868958
    -6.85992101]
 [  -6.63432707   -6.51200704   -6.51208921   -6.61930584   -8.47807394
    -6.51233529]
 [  -6.35208262   -6.12491838   -6.12476738   -6.38604833   -8.1644201
    -6.12516369]
 [  -6.09185206   -5.69474254   -5.69464982   -5.96407984   -7.83043477
    -5.69485716]
 [  -5.47113751   -5.21668471   -5.21668022   -5.56468018   -7.36143929
    -5.21664811]
 [  -4.76420788   -4.68539353   -4.68544557   -4.77980503   -6.92606388
    -4.68541088]
 [  -4.34446481   -4.09499413   -4.09498992   -4.16812741   -6.39484181
    -4.09496022]
 [  -3.61760669   -3.43896387   -3.43896537   -4.35919221   -5.60971132
    -3.43895604]
 [  -3.19571843   -3.17263276   -2.70999794   -2.77317449   -4.94501746
    -2.70999747]
 [  -7.70814366   -7.45798856   -7.4580404    -7.50252547   -9.50584756
    -8.28037392]
 [  -7.49392836   -7.17567344   -7.17566064   -7.49037666   -8.57940679
    -8.01533758]
 [  -7.35641942   -6.86188935   -6.86188735   -7.35205901   -8.354092
    -7.68409239]
 [  -6.87925822   -6.51321445   -6.5132144    -6.96544423   -7.94120776
    -7.41299227]
 [  -6.46316702   -6.1257947    -6.12579476   -6.57602763   -7.75375514
    -7.02569742]
 [  -6.06852065   -5.69532773   -5.69532778   -6.20905879   -7.40479834
    -6.54785183]
 [  -5.82431677   -5.21703094   -5.21703093   -5.8145905    -7.0213428
    -5.97039277]
 [  -5.10575392   -4.68558998   -4.68558998   -5.26537835   -6.54684175
    -5.47980415]
 [  -4.82268041   -4.09509999   -4.09509999   -4.65430969   -6.11659107
    -4.98927131]
 [  -4.5400528    -3.439         -3.439        -4.31111064   -5.58515682
    -4.31096354]
 [  -3.3015951    -2.71          -2.71         -3.000982     -4.90699599
    -3.60649252]
 [  -2.92203025   -2.0501162    -1.9          -3.22724488   -4.11241888
    -2.79989784]
 [  -7.71195054   -7.17570464   -7.71228651   -7.4581334    -8.45811951
    -8.1561706 ]
 [  -7.45735918   -6.86189404 -106.71229246   -7.45812328   -7.45766588
  -107.07000876]
 [  -7.17567415   -6.5132156  -106.71230866   -7.17570332   -7.17563526
  -107.4738146 ]
 [  -6.86182383   -6.12579511 -106.71212836   -6.86187185   -6.86163028
  -106.74598485]
 [  -6.51320396   -5.6953279  -106.70832053   -6.51316506   -6.51318971
  -107.43363011]
 [  -6.12569609   -5.217031   -106.71220588   -6.1257945    -6.12571861
  -107.983442  ]
 [  -5.69526104   -4.68559    -106.71143025   -5.69531021   -5.69454519
  -107.48253028]
 [  -5.21700239   -4.0951     -106.70829856   -5.21670773   -5.21684323
```

```
  -107.7629538 ]
 [  -4.68390049   -3.439      -106.69298697   -4.68558891   -4.68557888
  -107.71349121]
 [  -4.07514974   -2.71       -106.47648482   -4.09508914   -4.0950968
  -107.29255611]
 [  -3.43898401   -1.9        -106.57059272   -3.43897069   -3.43881401
  -107.02804135]
 [  -2.70999626   -1.8996486    -1.           -2.70998763   -2.70866674
    -1.8540403 ]
 [  -7.45813417 -106.71228386   -7.71232075   -7.71232075   -7.45813417
    -7.71232074]
 [   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.            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.        ]]

Update Frequency:
[[  38.  265.   86.   39.  388.   83.]
 [  37.  226.   58.   25.   63.   57.]
 [  32.  200.   50.   22.   53.   49.]
 [  32.  176.   43.   20.   47.   42.]
 [  27.  160.   40.   17.   48.   40.]
 [  26.  142.   38.   16.   44.   38.]
 [  26.  119.   36.   14.   38.   35.]
 [  20.   97.   34.   13.   36.   33.]
 [  16.   78.   33.   10.   36.   32.]
 [  14.   69.   30.    9.   37.   29.]
 [  11.   41.   29.   10.   30.   29.]
 [  10.   10.   32.    6.   22.   31.]
 [  26.  110.   41.   34.  383.   85.]
 [  27.  127.   44.   22.   52.   64.]
 [  28.  141.   50.   21.   47.   53.]
 [  25.  143.   53.   18.   39.   44.]
 [  23.  142.   53.   16.   42.   42.]
 [  21.  131.   53.   15.   39.   43.]
 [  18.  117.   51.   14.   31.   41.]
 [  15.  108.   51.   11.   33.   36.]
 [  14.   92.   51.   10.   33.   36.]
 [  14.   66.   49.    9.   35.   32.]
 [   8.   54.   53.    6.   26.   34.]
 [   7.    6.   93.    8.   21.   32.]
 [  75. 1564.   37.   57.  379.  127.]
 [  45. 1469.   30.   38.   45.   86.]
 [  51. 1360.   32.   41.   45.   77.]
 [  44. 1273.   26.   34.   36.   75.]
 [  42. 1204.   20.   31.   38.   72.]
```

```
[  35. 1137.   27.   39.   35.   58.]
[  32. 1076.   23.   31.   27.   64.]
[  32. 1046.   20.   24.   28.   47.]
[  22.  997.   17.   34.   32.   59.]
[  15.  968.   12.   28.   33.   47.]
[  27.  944.   13.   25.   22.   49.]
[  28.   18.  954.   25.   16.   46.]
[1673.   30.   74.   74.  345.   68.]
[   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.    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.]]
```

Task 3

Using the same options and the SMDP code, implement Intra Option Q-Learning (In the code cell below). You *might not* always have to search through options to find the options with similar policies, think about it. Keep the **final Q-table** and **Update Frequency** table handy (You'll need it in TODO 4)

In [7]:

```
#Q-Table: (States x Actions) === (env.ns(48) x total actions(6))
q_values_IOQL = np.zeros((48,6))

#Update_Frequency Data structure? Check TODO 4
update_freq_IOQL = np.zeros((48,6))
```

In [8]:

```
#### Intra-Option Q-Learning

# Add parameters you might need here
gamma = 0.9
alpha = 0.4
epsilon = 0.1

Rewards = []

# Iterate over 1000 episodes
for _ in range(1000):
    state = env.reset()
    done = False

    episode_reward = 0

    # While episode is not over
    while not done:

        # Choose action
        action = egreedy_policy(q_values_IOQL, state, epsilon=0.1)

        # Checking if primitive action
        if action < 4:
            # Perform regular Q-Learning update for state-action pair
            next_state, reward, done,_ = env.step(action)
```

```
                q_values_IOQL[state, action] += alpha*(reward + gamma*np.max([q_values_

                update_freq_IOQL[state,action] += 1

                episode_reward += reward
                state = next_state

            # Checking if action chosen is an option
            reward_bar = 0
            if action == 4: # action => Away option

                optdone = False
                while (optdone == False):

                    # Think about what this function might do?
                    optact,optdone = Away(env,state)
                    next_state, reward, done,_ = env.step(optact)

                    # Is this formulation right? What is this term?
                    reward_bar = gamma*reward_bar + reward

                    # Complete SMDP Q-Learning Update
                    # Remember SMDP Updates. When & What do you update?
                    q_values_IOQL[state, action] += alpha*(reward_bar + (gamma)*np.max
                    update_freq_IOQL[state,action] += 1

                    state = next_state

                    episode_reward += reward

                    q_values_IOQL[state, optact] += alpha*(reward + gamma*np.max([q_va
                    update_freq_IOQL[state,optact] += 1

                    if not optdone:
                        q_values_IOQL[state, action] += alpha*(reward + gamma*q_values_
                        update_freq_IOQL[state,action] += 1
                    else:
                        q_values_IOQL[state, action] += alpha*(reward + gamma*np.max([
                        update_freq_IOQL[state,action] += 1
                    state = next_state

            if action == 5: # action => Close option

                optdone = False
                while (optdone == False):

                    # Think about what this function might do?
                    optact,optdone = Close(env,state)
                    next_state, reward, done,_ = env.step(optact)

                    # Is this formulation right? What is this term?
                    reward_bar = gamma*reward_bar + reward

                    #if next_state in [36, 37, 38, 39, 40, 41, 42, 43]: # check if next
                    #    done = True

                    #else:
                    # Complete SMDP Q-Learning Update
                    # Remember SMDP Updates. When & What do you update?
                    q_values_IOQL[state, action] += alpha*(reward_bar + (gamma)*np.max
                    update_freq_IOQL[state,action] += 1

                    state = next_state
```

```
                episode_reward += reward

                if not optdone:
                    q_values_IOQL[state, action] += alpha*(reward + gamma*q_values_
                    update_freq_IOQL[state,action] += 1
                else:
                    q_values_IOQL[state, action] += alpha*(reward + gamma*np.max([
                    update_freq_IOQL[state,action] += 1
                state = next_state

                #if done: # terminate episode if agent falls in the cliff
                #    break

# Print final Q-table and Update Frequency table
print("Final Q-values:")
print(q_values_IOQL)
print("\nUpdate Frequency:")
print(update_freq_IOQL)
```

```
Final Q-values:
[[  -7.93750497   -7.70895505   -7.70886581   -7.86059746   -8.61844635
    -7.71000423]
 [  -7.70923135   -7.45554987   -7.45586182   -7.64044017   -8.38676692
    -7.45654681]
 [  -7.45499373   -7.17353366   -7.17352195   -7.51525673   -8.13761588
    -7.17397042]
 [  -7.1724547    -6.86019907   -6.86070716   -7.13731749   -7.86183326
    -6.86048795]
 [  -6.85957558   -6.51210687   -6.51200553   -6.82478969   -7.47034476
    -6.51189224]
 [  -6.51242032   -6.12517757   -6.12515172   -6.4416638    -7.2153511
    -6.12530889]
 [  -6.12528676   -5.69495008   -5.69496896   -5.80688155   -6.83768187
    -5.69490511]
 [  -5.69490314   -5.21672731   -5.21683879   -5.2381969    -6.41015416
    -5.21683287]
 [  -5.21659269   -4.68535952   -4.68539195   -5.29603985   -5.9480378
    -4.68545122]
 [  -4.68545481   -4.09500155   -4.09498879   -5.03298659   -5.4285836
    -4.09500903]
 [  -4.09495383   -3.43895414   -3.43894612   -3.59189437   -4.85198626
    -3.43895817]
 [  -3.43899678   -3.01982191   -2.70999804   -3.12371557   -4.21017894
    -2.70999845]
 [  -7.71207053   -7.45786897   -7.45794926   -7.46985224   -9.001496
    -8.40735053]
 [  -7.45862798   -7.17569248   -7.17569095   -7.54207657   -8.68693386
    -8.17612957]
 [  -7.22504504   -6.86189245   -6.86189208   -7.17322218   -8.44514142
    -7.88823185]
 [  -6.9867101    -6.51321531   -6.51321531   -6.68690513   -8.18102099
    -7.5437138 ]
 [  -6.51285177   -6.12579497   -6.12579498   -6.51155449   -7.64691211
    -7.18581547]
 [  -6.12670538   -5.69532783   -5.69532781   -6.33431473   -7.53212725
    -6.73885239]
 [  -5.69710954   -5.21703097   -5.21703098   -5.75692314   -7.19348594
    -6.23380022]
 [  -5.25827245   -4.68558999   -4.68558999   -5.59519257   -6.77806541
    -5.83543274]
 [  -4.68486543   -4.0951       -4.0951       -4.66790814   -6.30603908
    -5.27831559]
 [  -4.18029059   -3.439        -3.439        -3.44203124   -5.8060765
    -4.65827443]
 [  -3.44333591   -2.71         -2.71         -3.42243411   -5.27156817
    -3.9709963 ]
 [  -2.74784564   -2.37327493   -1.9          -2.90856726   -4.64155729
    -3.20731558]
 [  -7.4870487    -7.17570464   -7.71232067   -7.45813415   -7.85756884
    -8.1508221 ]
 [  -7.4580167    -6.86189404 -106.71220319   -7.45813326   -7.45806441
  -106.41110137]
 [  -7.17569817   -6.5132156  -106.71229472   -7.17570459   -7.17495302
  -105.91968905]
 [  -6.86187066   -6.12579511 -106.71076385   -6.86188863   -6.86187776
  -106.26251278]
 [  -6.51321019   -5.6953279  -106.71144191   -6.51321494   -6.51278522
  -105.01887424]
 [  -6.12528422   -5.217031   -106.71220746   -6.12578801   -6.12578188
  -106.22781299]
 [  -5.69524074   -4.68559    -106.71198789   -5.69524294   -5.69531316
  -104.95458855]
 [  -5.21685953   -4.0951     -106.68077648   -5.21689833   -5.21702117
```

```
  -104.55757924]
 [  -4.68558043   -3.439      -106.71225416   -4.68557636   -4.68286111
  -105.87895005]
 [  -4.09455948   -2.71       -106.71084035   -4.09509646   -4.09509696
  -105.41962861]
 [  -3.43867424   -1.9        -106.71089958   -3.43887422   -3.43781627
  -103.74335911]
 [  -2.70479407   -1.89999924   -1.           -2.70871549   -2.70993485
    -2.15101457]
 [  -7.45813417 -106.71231087   -7.7123205    -7.71232003   -7.45813417
  -106.64074834]
 [   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.            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.
    -1.        ]]

Update Frequency:
[[ 682.  222.   72.   40. 1019.   69.]
 [ 140.  203.   45.   23.  206.   45.]
 [ 102.  180.   39.   21.  148.   39.]
 [ 106.  168.   37.   19.  155.   36.]
 [  86.  148.   35.   17.  129.   33.]
 [  91.  132.   33.   16.  129.   32.]
 [  81.  118.   32.   13.  111.   31.]
 [  88.  105.   31.   12.  127.   30.]
 [  61.   81.   29.   12.   88.   29.]
 [  82.   64.   28.   13.  114.   28.]
 [  49.   38.   27.    8.   70.   27.]
 [  54.    9.   31.    7.   77.   31.]
 [ 337.   80.   36.   34.  671.  140.]
 [  65.  116.   45.   25.  128.   94.]
 [  52.  137.   52.   20.   83.   79.]
 [  57.  136.   56.   17.   92.   76.]
 [  43.  137.   56.   16.   72.   71.]
 [  49.  125.   54.   16.   79.   69.]
 [  49.  115.   55.   13.   69.   66.]
 [  42.  100.   53.   15.   72.   61.]
 [  27.   91.   52.   10.   46.   60.]
 [  40.   75.   52.    7.   68.   58.]
 [  25.   54.   53.    7.   40.   58.]
 [  28.    7.   83.    6.   47.   63.]
 [ 328. 1603.   49.   65.  635.  179.]
 [  58. 1477.   27.   42.   60.  130.]
 [  56. 1366.   30.   48.   39.  107.]
 [  43. 1293.   22.   37.   43.  104.]
 [  43. 1230.   23.   40.   33.   90.]
```

```
 [  31. 1153.   27.   34.   38.  103.]
 [  31. 1102.   25.   28.   34.   84.]
 [  28. 1063.   16.   26.   33.   80.]
 [  32. 1016.   28.   29.   20.   82.]
 [  22.  966.   22.   30.   32.   78.]
 [  21.  937.   22.   22.   18.   77.]
 [  14.   30.  949.   16.   22.   83.]
 [1671.   32.   61.   63.  301.  735.]
 [   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.    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.   51.]]
```

Task 4

Compare the two Q-Tables and Update Frequencies and provide comments.

In [9]:

```
# Use this cell for Task 4 Code
```

In [10]:

```
np.set_printoptions(threshold=np.inf, suppress=True, linewidth=np.inf)
print("Q-Table SMDP:")
print(q_values_SMDP)
```

```
Q-Table SMDP:
[[  -7.77614476   -7.70678309   -7.70780839   -7.82166587  -10.24870589   -7.70874
639]
 [  -7.62037387   -7.45531804   -7.4551474    -7.80709603   -8.99224897   -7.45509
322]
 [  -7.22877075   -7.17324595   -7.17388968   -7.5164836    -9.16401083   -7.17417
032]
 [  -7.103383     -6.8600376    -6.86026604   -7.23582774   -7.90868958   -6.85992
101]
 [  -6.63432707   -6.51200704   -6.51208921   -6.61930584   -8.47807394   -6.51233
529]
 [  -6.35208262   -6.12491838   -6.12476738   -6.38604833   -8.1644201    -6.12516
369]
 [  -6.09185206   -5.69474254   -5.69464982   -5.96407984   -7.83043477   -5.69485
716]
 [  -5.47113751   -5.21668471   -5.21668022   -5.56468018   -7.36143929   -5.21664
811]
 [  -4.76420788   -4.68539353   -4.68544557   -4.77980503   -6.92606388   -4.68541
088]
 [  -4.34446481   -4.09499413   -4.09498992   -4.16812741   -6.39484181   -4.09496
022]
 [  -3.61760669   -3.43896387   -3.43896537   -4.35919221   -5.60971132   -3.43895
604]
 [  -3.19571843   -3.17263276   -2.70999794   -2.77317449   -4.94501746   -2.70999
747]
 [  -7.70814366   -7.45798856   -7.4580404    -7.50252547   -9.50584756   -8.28037
392]
 [  -7.49392836   -7.17567344   -7.17566064   -7.49037666   -8.57940679   -8.01533
758]
 [  -7.35641942   -6.86188935   -6.86188735   -7.35205901   -8.354092     -7.68409
239]
 [  -6.87925822   -6.51321445   -6.5132144    -6.96544423   -7.94120776   -7.41299
227]
 [  -6.46316702   -6.1257947    -6.12579476   -6.57602763   -7.75375514   -7.02569
742]
 [  -6.06852065   -5.69532773   -5.69532778   -6.20905879   -7.40479834   -6.54785
183]
 [  -5.82431677   -5.21703094   -5.21703093   -5.8145905    -7.0213428    -5.97039
277]
 [  -5.10575392   -4.68558998   -4.68558998   -5.26537835   -6.54684175   -5.47980
415]
 [  -4.82268041   -4.09509999   -4.09509999   -4.65430969   -6.11659107   -4.98927
131]
 [  -4.5400528    -3.439        -3.439        -4.31111064   -5.58515682   -4.31096
354]
 [  -3.3015951    -2.71         -2.71         -3.000982     -4.90699599   -3.60649
252]
 [  -2.92203025   -2.0501162    -1.9          -3.22724488   -4.11241888   -2.79989
784]
 [  -7.71195054   -7.17570464   -7.71228651   -7.4581334    -8.45811951   -8.15617
06 ]
 [  -7.45735918   -6.86189404 -106.71229246   -7.45812328   -7.45766588 -107.07000
876]
 [  -7.17567415   -6.5132156  -106.71230866   -7.17570332   -7.17563526 -107.47381
46 ]
 [  -6.86182383   -6.12579511 -106.71212836   -6.86187185   -6.86163028 -106.74598
485]
 [  -6.51320396   -5.6953279  -106.70832053   -6.51316506   -6.51318971 -107.43363
011]
 [  -6.12569609   -5.217031   -106.71220588   -6.1257945    -6.12571861 -107.98344
2  ]
 [  -5.69526104   -4.68559    -106.71143025   -5.69531021   -5.69454519 -107.48253
028]
 [  -5.21700239   -4.0951     -106.70829856   -5.21670773   -5.21684323 -107.76295
```

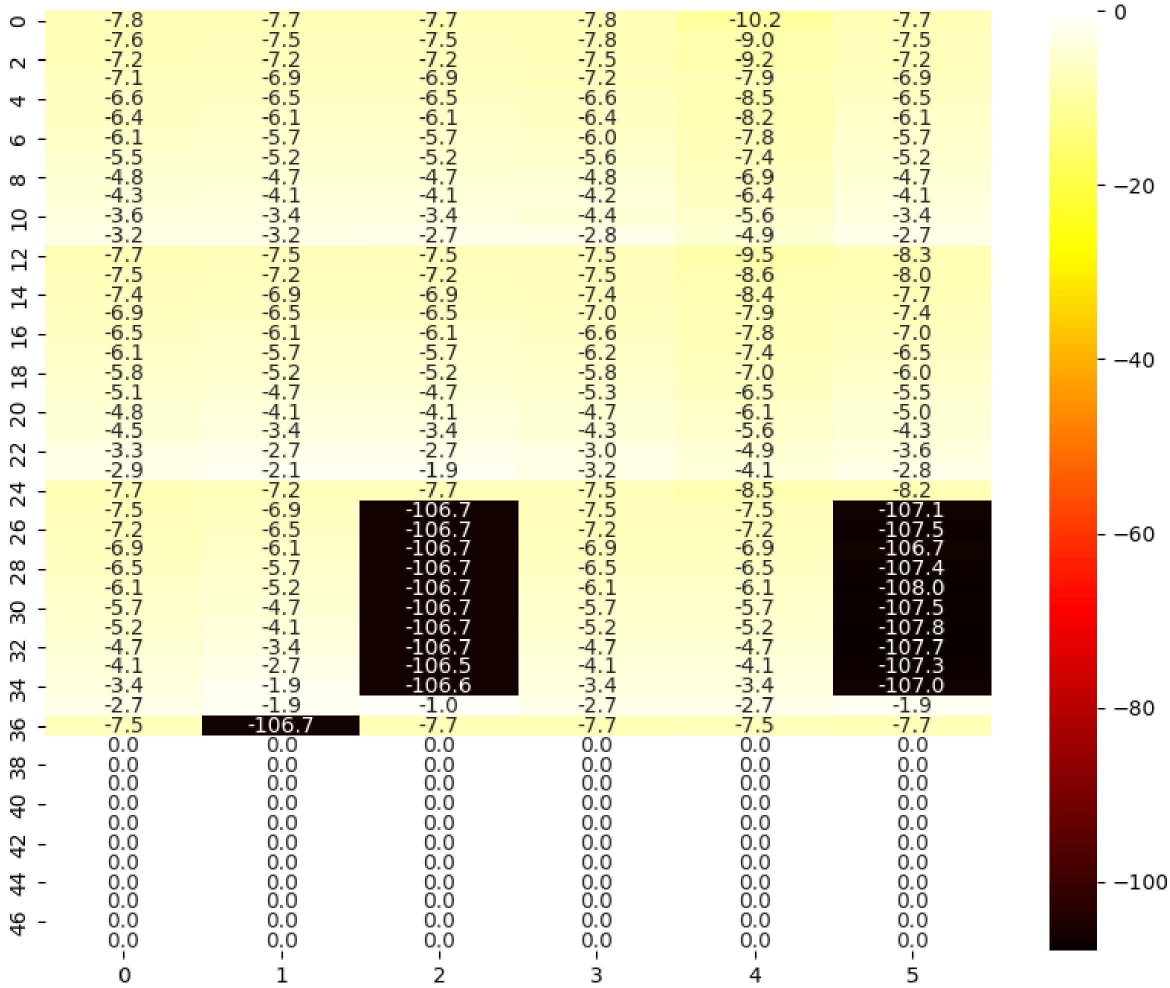
```
38 ]
 [  -4.68390049   -3.439      -106.69298697   -4.68558891   -4.68557888 -107.71349
121]
 [  -4.07514974   -2.71       -106.47648482   -4.09508914   -4.0950968  -107.29255
611]
 [  -3.43898401   -1.9        -106.57059272   -3.43897069   -3.43881401 -107.02804
135]
 [  -2.70999626   -1.8996486    -1.           -2.70998763   -2.70866674   -1.85404
03 ]
 [  -7.45813417 -106.71228386   -7.71232075   -7.71232075   -7.45813417   -7.71232
074]
 [   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.           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.
]]
```

In [11]:

```
import seaborn as sns
```

In [12]:

```
plt.figure(figsize=(10, 8))
sns.heatmap(q_values_SMDP, cmap='hot', annot=True, fmt='.1f')
plt.show()
```



In [13]:

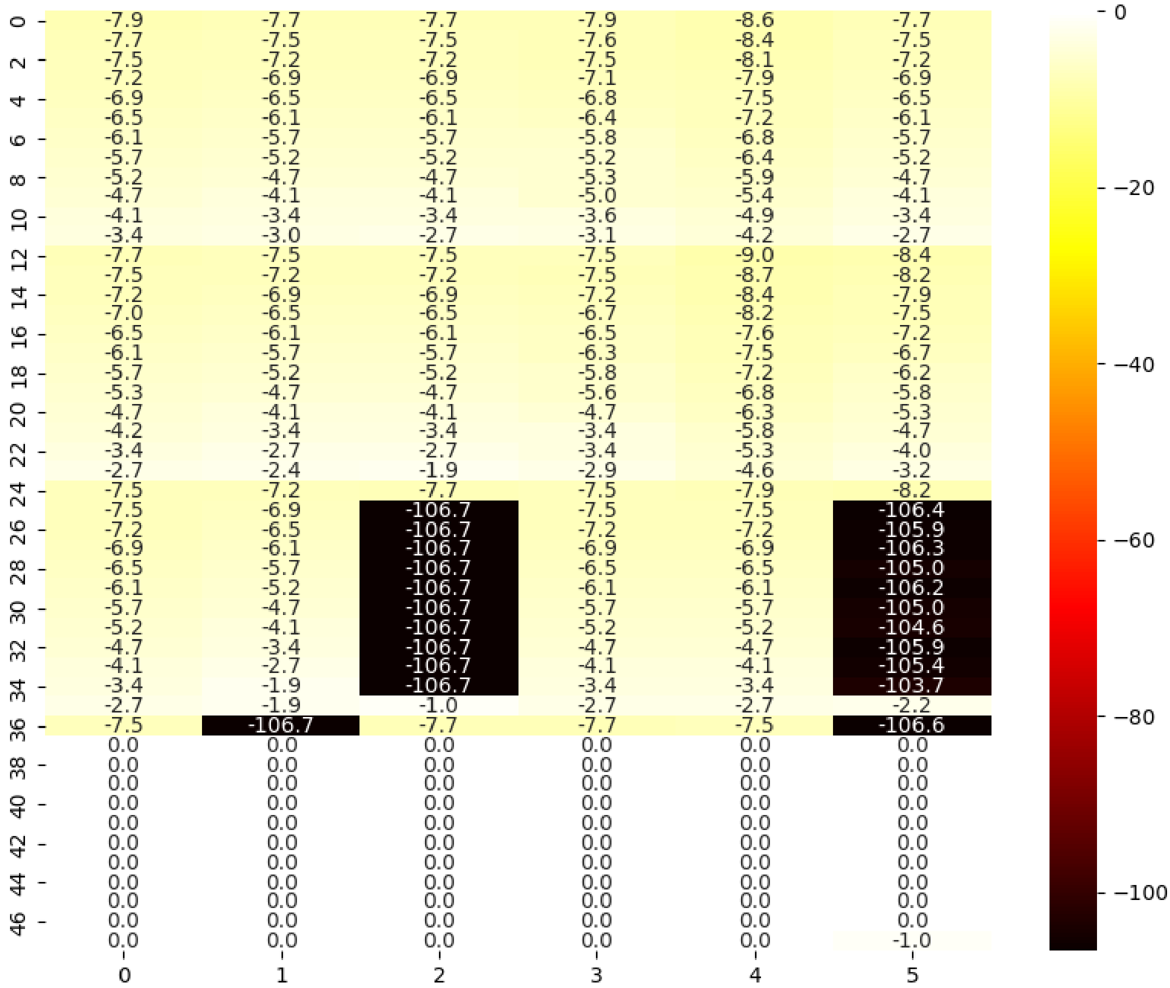
```
np.set_printoptions(threshold=np.inf, suppress=True, linewidth=np.inf)
print("Q-Table IOQL:")
print(q_values_IOQL)
```

```
Q-Table IOQL:
[[  -7.93750497   -7.70895505   -7.70886581   -7.86059746   -8.61844635   -7.71000
423]
 [  -7.70923135   -7.45554987   -7.45586182   -7.64044017   -8.38676692   -7.45654
681]
 [  -7.45499373   -7.17353366   -7.17352195   -7.51525673   -8.13761588   -7.17397
042]
 [  -7.1724547    -6.86019907   -6.86070716   -7.13731749   -7.86183326   -6.86048
795]
 [  -6.85957558   -6.51210687   -6.51200553   -6.82478969   -7.47034476   -6.51189
224]
 [  -6.51242032   -6.12517757   -6.12515172   -6.4416638    -7.2153511    -6.12530
889]
 [  -6.12528676   -5.69495008   -5.69496896   -5.80688155   -6.83768187   -5.69490
511]
 [  -5.69490314   -5.21672731   -5.21683879   -5.2381969    -6.41015416   -5.21683
287]
 [  -5.21659269   -4.68535952   -4.68539195   -5.29603985   -5.9480378    -4.68545
122]
 [  -4.68545481   -4.09500155   -4.09498879   -5.03298659   -5.4285836    -4.09500
903]
 [  -4.09495383   -3.43895414   -3.43894612   -3.59189437   -4.85198626   -3.43895
817]
 [  -3.43899678   -3.01982191   -2.70999804   -3.12371557   -4.21017894   -2.70999
845]
 [  -7.71207053   -7.45786897   -7.45794926   -7.46985224   -9.001496     -8.40735
053]
 [  -7.45862798   -7.17569248   -7.17569095   -7.54207657   -8.68693386   -8.17612
957]
 [  -7.22504504   -6.86189245   -6.86189208   -7.17322218   -8.44514142   -7.88823
185]
 [  -6.9867101    -6.51321531   -6.51321531   -6.68690513   -8.18102099   -7.54371
38 ]
 [  -6.51285177   -6.12579497   -6.12579498   -6.51155449   -7.64691211   -7.18581
547]
 [  -6.12670538   -5.69532783   -5.69532781   -6.33431473   -7.53212725   -6.73885
239]
 [  -5.69710954   -5.21703097   -5.21703098   -5.75692314   -7.19348594   -6.23380
022]
 [  -5.25827245   -4.68558999   -4.68558999   -5.59519257   -6.77806541   -5.83543
274]
 [  -4.68486543   -4.0951       -4.0951       -4.66790814   -6.30603908   -5.27831
559]
 [  -4.18029059   -3.439        -3.439        -3.44203124   -5.8060765    -4.65827
443]
 [  -3.44333591   -2.71         -2.71         -3.42243411   -5.27156817   -3.97099
63 ]
 [  -2.74784564   -2.37327493   -1.9          -2.90856726   -4.64155729   -3.20731
558]
 [  -7.4870487    -7.17570464   -7.71232067   -7.45813415   -7.85756884   -8.15082
21 ]
 [  -7.4580167    -6.86189404 -106.71220319   -7.45813326   -7.45806441 -106.41110
137]
 [  -7.17569817   -6.5132156  -106.71229472   -7.17570459   -7.17495302 -105.91968
905]
 [  -6.86187066   -6.12579511 -106.71076385   -6.86188863   -6.86187776 -106.26251
278]
 [  -6.51321019   -5.6953279  -106.71144191   -6.51321494   -6.51278522 -105.01887
424]
 [  -6.12528422   -5.217031   -106.71220746   -6.12578801   -6.12578188 -106.22781
299]
 [  -5.69524074   -4.68559    -106.71198789   -5.69524294   -5.69531316 -104.95458
855]
 [  -5.21685953   -4.0951     -106.68077648   -5.21689833   -5.21702117 -104.55757
```

```
924]
 [  -4.68558043   -3.439       -106.71225416   -4.68557636   -4.68286111 -105.87895
005]
 [  -4.09455948   -2.71        -106.71084035   -4.09509646   -4.09509696 -105.41962
861]
 [  -3.43867424   -1.9        -106.71089958   -3.43887422   -3.43781627 -103.74335
911]
 [  -2.70479407   -1.89999924   -1.            -2.70871549   -2.70993485   -2.15101
457]
 [  -7.45813417 -106.71231087   -7.7123205    -7.71232003   -7.45813417 -106.64074
834]
 [   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.           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.          -1.
]]
```

In [14]:

```
plt.figure(figsize=(10, 8))
sns.heatmap(q_values_IOQL, cmap='hot', annot=True, fmt='.1f')
plt.show()
```



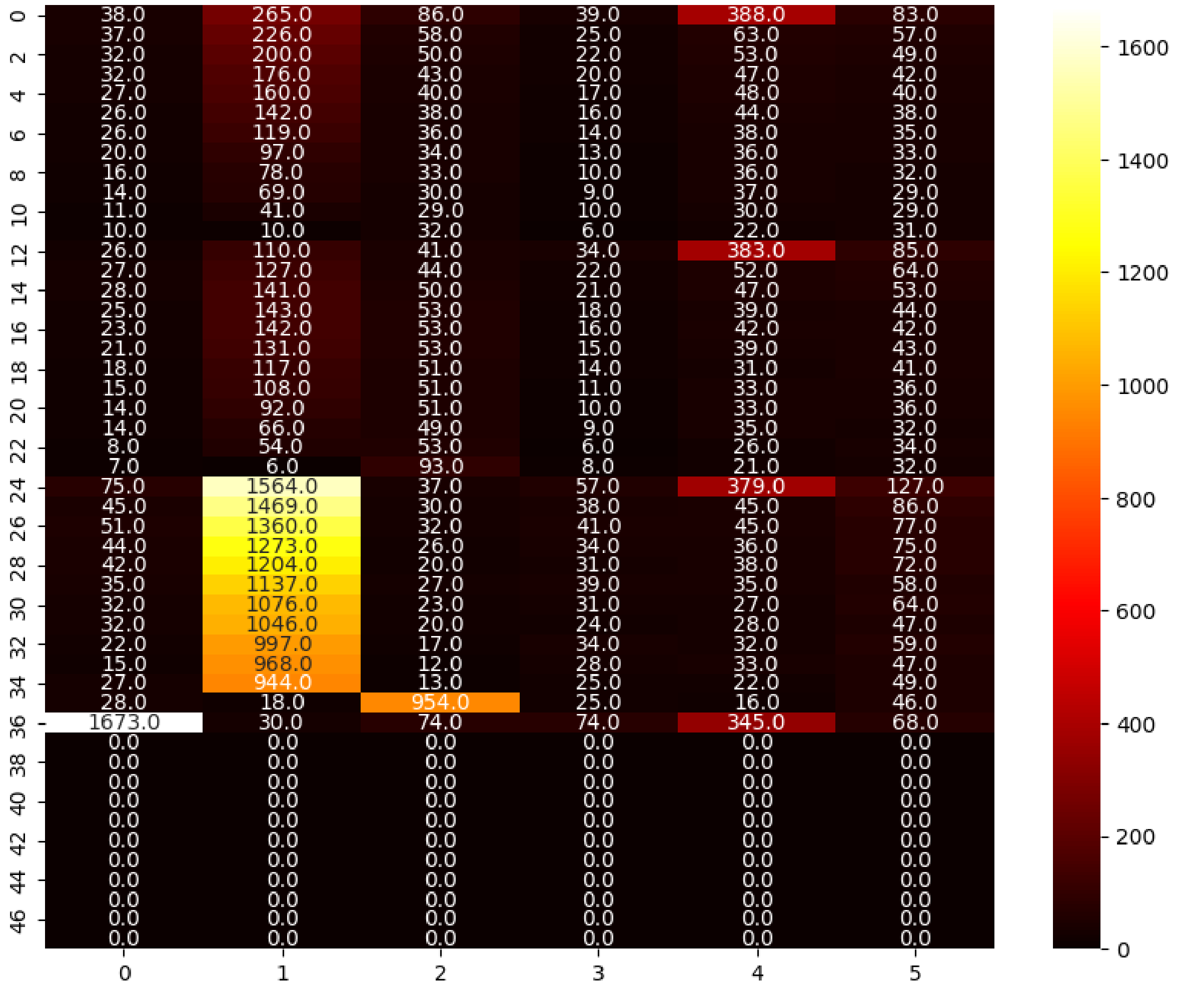
In [15]:

```
np.set_printoptions(threshold=np.inf, suppress=True, linewidth=np.inf)
print("Update Frequency:")
print(update_freq_SMDP)
```

```
Update Frequency:
[[  38.  265.   86.   39.  388.   83.]
 [  37.  226.   58.   25.   63.   57.]
 [  32.  200.   50.   22.   53.   49.]
 [  32.  176.   43.   20.   47.   42.]
 [  27.  160.   40.   17.   48.   40.]
 [  26.  142.   38.   16.   44.   38.]
 [  26.  119.   36.   14.   38.   35.]
 [  20.   97.   34.   13.   36.   33.]
 [  16.   78.   33.   10.   36.   32.]
 [  14.   69.   30.    9.   37.   29.]
 [  11.   41.   29.   10.   30.   29.]
 [  10.   10.   32.    6.   22.   31.]
 [  26.  110.   41.   34.  383.   85.]
 [  27.  127.   44.   22.   52.   64.]
 [  28.  141.   50.   21.   47.   53.]
 [  25.  143.   53.   18.   39.   44.]
 [  23.  142.   53.   16.   42.   42.]
 [  21.  131.   53.   15.   39.   43.]
 [  18.  117.   51.   14.   31.   41.]
 [  15.  108.   51.   11.   33.   36.]
 [  14.   92.   51.   10.   33.   36.]
 [  14.   66.   49.    9.   35.   32.]
 [   8.   54.   53.    6.   26.   34.]
 [   7.    6.   93.    8.   21.   32.]
 [  75. 1564.   37.   57.  379.  127.]
 [  45. 1469.   30.   38.   45.   86.]
 [  51. 1360.   32.   41.   45.   77.]
 [  44. 1273.   26.   34.   36.   75.]
 [  42. 1204.   20.   31.   38.   72.]
 [  35. 1137.   27.   39.   35.   58.]
 [  32. 1076.   23.   31.   27.   64.]
 [  32. 1046.   20.   24.   28.   47.]
 [  22.  997.   17.   34.   32.   59.]
 [  15.  968.   12.   28.   33.   47.]
 [  27.  944.   13.   25.   22.   49.]
 [  28.   18.  954.   25.   16.   46.]
 [1673.   30.   74.   74.  345.   68.]
 [   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.    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.]]
```

In [16]:

```
plt.figure(figsize=(10, 8))
sns.heatmap(update_freq_SMDP, cmap='hot', annot=True, fmt='.1f')
plt.show()
```

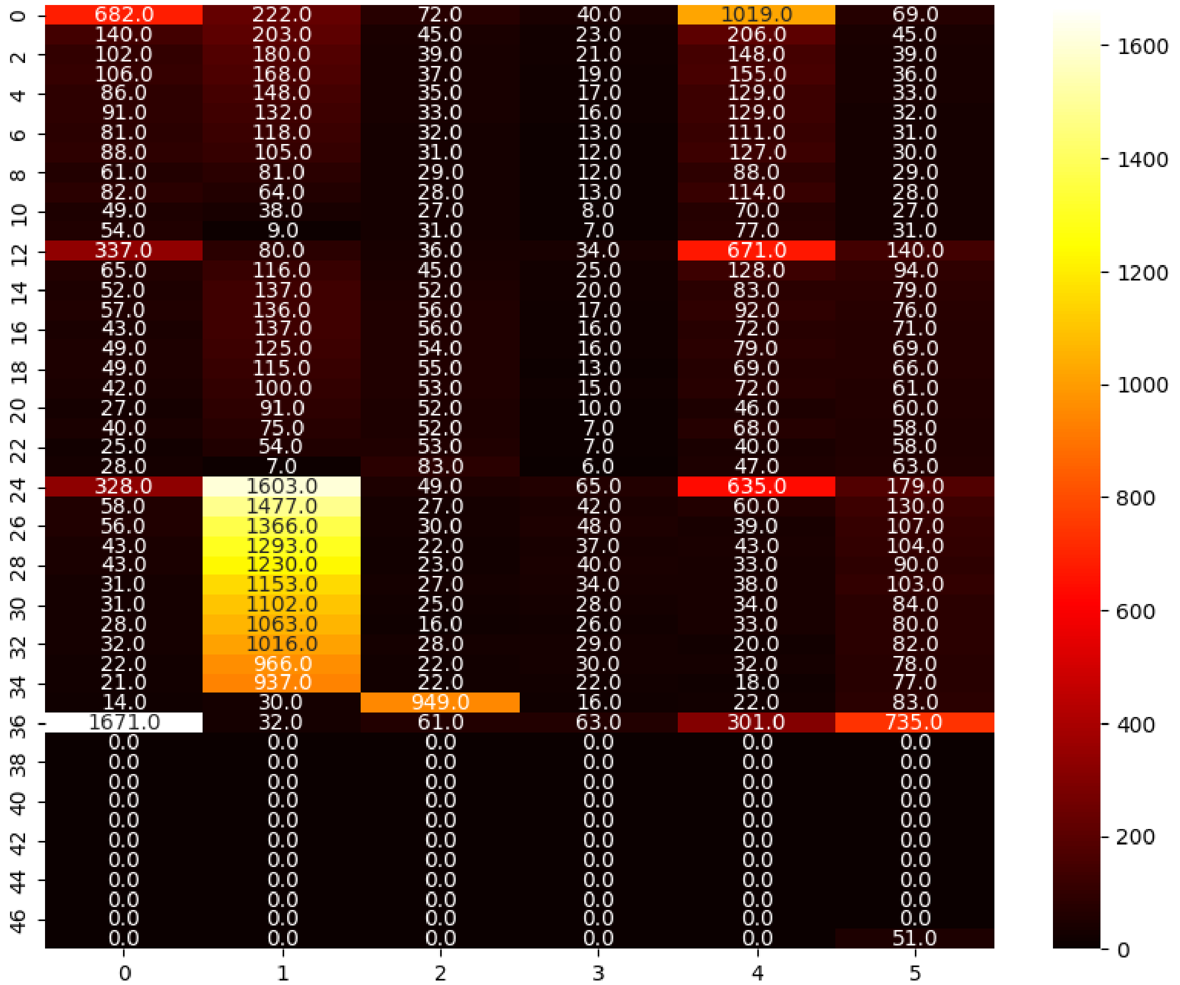


In [17]:
```
np.set_printoptions(threshold=np.inf, suppress=True, linewidth=np.inf)
print("Update Frequency:")
print(update_freq_IOQL)
```

```
Update Frequency:
[[ 682.  222.   72.   40. 1019.   69.]
 [ 140.  203.   45.   23.  206.   45.]
 [ 102.  180.   39.   21.  148.   39.]
 [ 106.  168.   37.   19.  155.   36.]
 [  86.  148.   35.   17.  129.   33.]
 [  91.  132.   33.   16.  129.   32.]
 [  81.  118.   32.   13.  111.   31.]
 [  88.  105.   31.   12.  127.   30.]
 [  61.   81.   29.   12.   88.   29.]
 [  82.   64.   28.   13.  114.   28.]
 [  49.   38.   27.    8.   70.   27.]
 [  54.    9.   31.    7.   77.   31.]
 [ 337.   80.   36.   34.  671.  140.]
 [  65.  116.   45.   25.  128.   94.]
 [  52.  137.   52.   20.   83.   79.]
 [  57.  136.   56.   17.   92.   76.]
 [  43.  137.   56.   16.   72.   71.]
 [  49.  125.   54.   16.   79.   69.]
 [  49.  115.   55.   13.   69.   66.]
 [  42.  100.   53.   15.   72.   61.]
 [  27.   91.   52.   10.   46.   60.]
 [  40.   75.   52.    7.   68.   58.]
 [  25.   54.   53.    7.   40.   58.]
 [  28.    7.   83.    6.   47.   63.]
 [ 328. 1603.   49.   65.  635.  179.]
 [  58. 1477.   27.   42.   60.  130.]
 [  56. 1366.   30.   48.   39.  107.]
 [  43. 1293.   22.   37.   43.  104.]
 [  43. 1230.   23.   40.   33.   90.]
 [  31. 1153.   27.   34.   38.  103.]
 [  31. 1102.   25.   28.   34.   84.]
 [  28. 1063.   16.   26.   33.   80.]
 [  32. 1016.   28.   29.   20.   82.]
 [  22.  966.   22.   30.   32.   78.]
 [  21.  937.   22.   22.   18.   77.]
 [  14.   30.  949.   16.   22.   83.]
 [1671.   32.   61.   63.  301.  735.]
 [   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.    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.   51.]]
```

In [18]:

```
plt.figure(figsize=(10, 8))
sns.heatmap(update_freq_IOQL, cmap='hot', annot=True, fmt='.1f')
plt.show()
```



Use this text cell for your comments - Task 4

Update frequency of primitive actions is almost similar for both the algorithms but Intra-Option Q-learning has a higher update frequency for both the Option actions.

Q-values for both the algorithms are pretty similar for most of the state action pairs. But in Intra-Option Q-learning algorithm there is more exploration and the agent moves closer to the cliff and takes a more negative reward.

In [21]:
```
pip install nbconvert
```