Lab 13: Reinforcement Learning (RL)

In [50]:

```
import importlib
from collections import defaultdict
import torch
import numpy

game_name = 'tictactoe'
game_module = importlib.import_module("games." + game_name)
env = game_module.Game()

env.reset()

# defining epsilon-greedy policy
def gen_epsilon_greedy_policy(n_action, epsilon):
    def policy_function(state, Q, available_actions):
        probs = torch.ones(n_action) * epsilon / n_action
        # print(probs)
        # print(state)
        # print(Q[state])
        best_action = torch.argmax(Q[state]).item()
        if not(best_action in available_actions):
            best_action = -1
            Q_max = -800000000
            for i in range(n_action):
                if i in available_actions and Q_max < Q[state][i]:
                    Q_max = Q[state][i]
                    best_action = i
        probs[best_action] += 1.0 - epsilon
        action = torch.multinomial(probs, 1).item()
        return action
    return policy_function

def q_learning(env, gamma, n_episode, alpha, player):
    """
    Obtain the optimal policy with off-policy Q-learning method
    @param env: OpenAI Gym environment
    @param gamma: discount factor
    @param n_episode: number of episodes
    @return: the optimal Q-function, and the optimal policy
    """
    n_action = 9
    Q = defaultdict(lambda: torch.zeros(n_action))
    for episode in range(n_episode):
        if episode % 10000 == 9999:
            print("episode: ", episode + 1)
        state = env.reset()
        state = hash(tuple(state.reshape(-1)))

        is_done = False
        while not is_done:
            if env.to_play() == player:
                available_action = env.legal_actions()
                action = epsilon_greedy_policy(state, Q, available_action)
                next_state, reward, is_done = env.step(action)
                next_state = hash(tuple(next_state.reshape(-1)))
                td_delta = reward + gamma * torch.max(Q[next_state]) - Q[state][
action]
                Q[state][action] += alpha * td_delta
            else:
                action = env.expert_agent()
```

```
                next_state, reward, is_done = env.step(action)
                next_state = hash(tuple(next_state.reshape(-1)))

                if is_done:
                    reward = -reward
                    td_delta = reward + gamma * torch.max(Q[next_state]) - Q[sta
te][action]
                    Q[state][action] += alpha * td_delta

            length_episode[episode] += 1
            total_reward_episode[episode] += reward

            if is_done:
                break
            state = next_state

    policy = {}
    for state, actions in Q.items():
        policy[state] = torch.argmax(actions).item()
    return Q, policy

gamma = 1

n_episode = 1000000

alpha = 0.4

epsilon = 0.1

available_action = env.legal_actions()
epsilon_greedy_policy = gen_epsilon_greedy_policy(9, epsilon)

length_episode = [0] * n_episode
total_reward_episode = [0] * n_episode

# agent play first
optimal_Q, optimal_policy = q_learning(env, gamma, n_episode, alpha, 1)


print('The optimal policy:\n', optimal_policy)
```

5: 0, -8921562862055878748: 0, 5804471335317639714: 6, 1222802750924
923823: 0, 8748956428649517688: 0, 2909964738678234170: 0, -64919435
97298662998: 0, 3447602624896303656: 0, -3304069188338600697: 0, -14
78061838276642523: 0, -8259897177948066666: 0, -6139116141089208671:
0, -5324109887964636169: 0, 1271978875132078937: 0, 5677256947168970
719: 3, 8047552283934953556: 0, 5617795650212269352: 0, 363084506863
6650362: 0, 3704031541910399844: 0, -1295474922522984264: 0, 1540338
3379866475: 0, 4028274238675309913: 0, 3075498732548716036: 0, 39690
3780980753027: 0, -2903387146842176953: 0, -6080136958132622774: 0,
-4026211331039043401: 0, 7814348627122961182: 0, 414303100954562032
6: 0, -13033546906471243: 0, 6560102334827197799: 0, 826003913789193
3802: 0, -4660198133650040: 0, -4666419504314519616: 0, -37235876942
3872095: 0, -5194136811097380033: 0, -668739533188762136: 0, 5386951
221386053076: 0, 9159537521389301381: 0, -4779982424788258116: 3, -7
263158734089982026: 0, -4026227037530479219: 0, -15378411585428130:
0, -8269159459780754036: 0, 3308925129741168895: 0, 5342677402683626
514: 0, 5828948080567245560: 1, 8446622707075320599: 0, 660458553884
7306990: 1, -1145682939129873512: 0, -4846057390810031810: 0, 216529
3781300246492: 0, 8553287994649665709: 0, -981126019776627712: 0, -8
757070215102897504: 0, 4621786767402815317: 0, 1975603450992337766:
0, 5767639462433361538: 0, 8924205010180370383: 0, 92837742460867439
0: 5, 8331956610605076860: 0, -2215355669324681905: 0, 7549135525562
998496: 0, 8617081838985646035: 0, -7237120461650976246: 0, 48886084
529310953: 0, -326391375348639236: 0, 8077249507517454347: 0, 356316
7829515681332: 0, 4812079022751436305: 0, -3967508713703253991: 0, -
2515041507972542695: 0, -1367933113750821692: 0, -906152025695805654
0: 0, -6059013499214052970: 0, -471688578611335233: 0, -149852710480
8571679: 0, -850275809433846549: 0, 3414396187977190249: 0, -4540813
430328811096: 0, 3209255665784735372: 0, -949468478807768467: 0, -26
09365527792145056: 0, -972018742077252468: 0, 52359391969125669: 0,
-3926396035812398586: 1, 6543958508189382961: 0, 571219849131437791
9: 0, -5055455548212540676: 0, -5683144597888785685: 0, 476354804650
7446541: 0, 2085290077170957283: 0, -1281278213911595679: 0, -533203
3608925429764: 0, -5693355343281714044: 0, 5502486068224611949: 0, -
5488355019241817610: 0, 4456382930536938953: 0, 2927020846375588549:
0, 1628700589685784361: 0, 7504929851343568680: 0, -1310531645204753
070: 0, -3057460126649455988: 0, 7641157285567556870: 0, -8710280544
57672748: 0, 5769320128541633370: 0, -332055465533698572: 1, -728362
8537258030145: 0, 2664264438123521940: 0, -3220284665337532430: 0, 9
189441106637710057: 0, 4748612290366311181: 0, 3685025083247947054:
0, -580633575929632717: 0, -575413625618009764: 0, -6754197797872131
734: 0, 3290248066833008167: 0, -4042155255943710819: 0, -7352270706
908053605: 0, -8780228510031034406: 0, -1722929421168848555: 0, -712
993081342878206: 0, 954267534604756098: 0, -3156594353622095496: 0,
-3931622924412944742: 0, 3260446948416954591: 0, -809190363398188275
3: 0, -833559731020831220: 0, 5526319747877379815: 0, -5716798583290
977043: 0, 1076351240999606855: 0, -6417593291663768303: 0, 72279698
10888269047: 0, 5319150127087211587: 5, 7877572623167876753: 0, 8128
83981369395669: 0, 1521497541696655711: 0, 5709898733329286031: 8, -
6165922760415035496: 0, 8548036176468603215: 0, 8413223634055102253:
0, 5386572191378914475: 0, -8783782692373267500: 0, -170966383781720
1041: 0, 8166165810107034701: 0, -5481833877605133126: 0, -774661547
4930899518: 0, 3792771629504751952: 0, 8607097284527297187: 0, -8238
696806309348287: 0, 5792357832908515980: 0, -4294804999098322119: 0,
291535262382579800: 0, -8730150229827278011: 0, -487283093995746815
7: 0, -3481500395897019739: 2, 2463885925289536562: 0, -614439411868
2224743: 0, 7368175754882597197: 0, 3699136679794013135: 0, -1833913
910544548850: 0, -95562277725525471: 0, 6686313364805781976: 0, 8654
516527824838426: 0, 8182523327023571680: 0, -8826740732927275411: 0,
-2569232471821800756: 0, 5475002501892732047: 0, 717642206585358333
3: 0, 4829852969850390047: 0, -5136510084262580977: 0, 8033515235073

462229: 0, 5863005988013794212: 0, 5310101758255242435: 0, -50341795
14029542936: 0, -76197593067371071: 0, 941001449247654526: 0, -69796
12625931651679: 0, 4004858378014229197: 0, 5634955652133389132: 0, -
5479417427604O905: 2, 7763807865172170084: 0, 5104743659233744912:
0, -3235828377586077022: 0, 9028956433192079849: 4, -548696219692561
0261: 0, 5944330630665664689: 0, -3610085284456822497: 0, 7433101383
713526630: 0, 4878932784723546720: 0, -5267931356888244491: 0, 706353
2048261667829: 0, 3082429219778665876: 0, -9005596384780072347: 0, -
6765946327870579065: 0, -1801315551566709607: 0, 171670498539950014
4: 0, -7062371166451156342: 0, -8288555244635140859: 0, -73581491109
07742393: 0, -7937998065048606222: 0, -1691699654204131048: 2, 27197
41621714646916: 0, -5470479662530512443: 0, -4101026172500118746: 0,
-3850698769448047950: 0, -1126760985932173221: 0, -46360155820010143
11: 0, -1247782924250455748: 0, 1183366925757752766: 0, 349981721307
5979559: 0, 8263824484716605885: 0, 6472474804219689402: 0, -6629440
102901593804: 0, 7767253885247609390: 0, 737893384218604035: 0, 4636
710528710138521: 0, 5704656842132786060: 3, 8559518406500841072: 0,
-4804646401990661792: 0, -5784825662824724652: 0, -63772655697244087
93: 0, 6851598929796216737: 0, 3626251703784550693: 0, 1194128355812
119348: 0, 3929013377432987537: 0, 8679827170624211639: 0, -28276259
59365903326: 0, 2502841287681628961: 0, -8629372588167166383: 0, 208
2242032420857584: 0, 2925591534776817502: 0, -1805930730880063971:
0, 6847267773824824647: 0, -3665584379836471419: 0, -242652400703906
9281: 0, 6898893510750879704: 0, -4498291553376804570: 0, -627779730
8404802197: 0, -5694401775209211226: 0, -748814007906790258: 0, 6886
737822067607081: 0, -8069385044424371123: 0, -589246265690550873: 0,
3824443315393697717: 0, -1077134614419406126: 0, 6758997822615594:
0, 8303573505220319637: 0, -2373983451790357313: 0, -692233237546740
965: 0, 4269422328345992842: 0, -837875331757344107: 0, -56882646460
31660172: 0, -3280929328042748766: 0, -9000227688747555520: 0, -7004
337923298702683: 0, 228592566287370653: 0, 137216171035580795: 0, 46
83095485585300166: 0, 5511685073030003948: 0, 5177286435762995304:
0, 9105058540468430806: 0, -4286146113701759808: 0, 3296990922626389
873: 0, -5325340029658120323: 0, 8118421551001767528: 0, -4710618119
547325887: 0, 4230311785610723010: 0, -375803052170456188: 1, 159349
3959281830354: 0, 1416630654161139324: 0, -3054894445110586302: 0, 5
615454456759450413: 0, 7599838033102885123: 0, -4897690599642169168:
0, -2783899286118642231: 0, 1688942014894074909: 0, 3637207924190054
581: 0, 212151624395558859: 0, -8913111487322332277: 0, 478892606144
9169714: 0, -1425439591493418498: 0, -4638312655580506220: 0, 666024
1992963328178: 0, -7404149686161719725: 0, -129102434736027162: 0, 6
779959271488244576: 0, -1110714494376061529: 0, -817244227655288538
1: 0, -5638658804010529573: 2, 3749241702554947153: 0, 3605744150835
036726: 0, -4181763360746656263: 0, 4079672579742722195: 0, -4475631
808440743676: 0, 482493140474814904: 0, -1525925938901758417: 2, -89
94196457381528109: 0, 4780350200927659419: 0, -2215631956183993224:
0, 8044808391545411139: 1, 900278348855119085: 0, -33330287203080383
20: 0, 6882892555732680603: 0, -6613642601416821726: 0, -32757611518
94281691: 1, -7558871507897608058: 0, 1934996128938843085: 0, 345189
242349441098: 0, -6337980659271612396: 0, 811038609603791741: 0, 100
229182979909762: 0, -8911725543075160076: 0, 391155797284698532: 0,
-9043993064000114886: 0, 5452310661949554477: 0, -322050265642020827
7: 0, -2853567128593277051: 0, -6632062876206126071: 0, 673397506394
4363471: 0, -8689618895843156754: 0, 8357307381806137407: 0, 3296991
701899195431: 0, -2998086808963391127: 0, 9040553844810790505: 0, -6
304396863015601428: 0, 7430088970744202393: 0, -3887600965931363709:
0, -8201948295161360959: 0, 9137096169600600585: 0, -90303832762081O
1148: 0, -4707104814796257301: 0, 73863953879740284: 2, 405161886543
1982037: 0, -5480152747471988889: 0, -2697354850433211426: 2, -78685
36540649541647: 0, -4883625605683717077: 0, 7921239464343868605: 0,
1818074489686889974: 2, 6269004061777604768: 0, 6104253700202267146:

0, 6593816439551503576: 0, 6593239777625455160: 0, -7476538855986104
001: 0, 5048190628245314092: 0, 7671748574965308006: 0, -39406196238
25487051: 0, -6802739199125712296: 0, 8225806752570481780: 0, 601489
6329961232337: 0, -1945917281653620755: 0, 4975062261946432146: 0, -
3150112517937285810: 0, -5262236048975762314: 0, -517227626170158942
8: 0, 5237449175981079404: 0, 398078470691139738: 0, -68146488653773
04124: 0, 7914175676469688715: 0, 5302837978660890645: 0, 8951736040
137601723: 0, -8565339928347675360: 0, 3187068837164232554: 0, -8388
787563614635204: 0, 8419435750749179882: 0, -1011927528811278590: 0,
2310217442953270334: 0, 5804786923149464220: 0, -765969347221591446
6: 0, -8149624658633734348: 0, 2943626703266171190: 1, 7006361400422
155973: 0, 6130792217205571581: 0, -8047010666634756331: 0, -3001247
144861452920: 0, -5183706258396125433: 0, -4881759372412761243: 0, 7
268942071927049476: 0, -8927307037775026730: 1, 2219924211424425700:
0, 1504047097184520628: 0, -8942300787472958597: 0, 6722004674192654
720: 0, -4232803596530311272: 0, 1048607554222801311: 0, 58388218512
98548901: 0, 6310581112935707775: 0, 4832691929199406628: 0, 2176131
705323972559: 0, 632750898706174270: 0, 493438031620433181: 0, -1397
64749807304595: 0, -4558311563840080204: 0, -673922130738517918: 1,
6654391870623735121: 0, 6052404545257377684: 0, 6742121896218048841:
0, -7554430660503805748: 0, -6968586408063843654: 0, -92172707992755
09467: 0, 3971384785165626722: 0, 483283852188017242: 2, -7179206355
87552828: 0, -67010128931366863: 0, 5373195667355841225: 0, -2590070
371534023536: 0, 164306572798662601: 2, -6133199541179748491: 0, -11
98165932240312669: 1, 3992872885075485317: 0, 5315554216942821171:
0, 1648818532047489468: 0, -3474273173395108466: 0, 1975708737271537
658: 1, -8361934732990384507: 0, 1263360312058371090: 0, -1703894491
394904707: 0, -8647088629737320480: 0, 2691152232965179999: 0, 69513
07297373075374: 0, 1086674065085977402: 0, 5019309669717049872: 0, 8
463975799114138127: 0, -7671739229066647927: 0, -455096176333891931
3: 0, -5214733921771883611: 0, 6597658968285259427: 0, 2764279587416
024702: 0, -3383704013818382211: 0, -9152812026493762427: 0, 6596542
93342498322: 0, -8159416088185502155: 0, -2834956764951245: 0, -3148
699932499784647: 0, 5970027374461132047: 0, 6926343418120701405: 0,
-7681507434241390054: 0, -1621951746473000887: 0, -25223390103833430
99: 0, 7876312628304108064: 0, 5710093016685128319: 0, 8251535069573
873294: 0, 8183988198092676967: 0, 1438865477323328710: 0, 273386767
9064425291: 0, -1601171090203808047: 0, -5190390228068440986: 0, 292
2128315361844558: 0, -6175425482877156421: 0, 8223145215777691152:
0, 6487800533987302719: 0, 708021500259181022: 0, 77398016923898891
1: 0, -5407218711961898353: 0, -8613908131504660081: 0, -87896178749
63588470: 0, 248278117268463532: 0, 5156271485998919882: 1, -1950650
693300846178: 0, 8982762416091875840: 0, 162074709221695863: 0, -422
8426502705039120: 0, -1028410765361038489: 0, -1033750462555076480:
0, 2335767239684139047: 0, -6107234144200544203: 0, -566523753613454
3913: 0, -6292174494283101222: 0, 3578142640455560924: 0, 4019995843
292413411: 0, 5073499660688691680: 0, 8004796966510717320: 0, -54529
43064209901384: 0, -5862045625040241164: 4, 1891853566137605639: 0,
6355643230258603935: 0, -8804489547800969055: 0, -732786903440090792
9: 0, -5132246173960794134: 0, 2221316022898655487: 0, 4479219055051
575645: 0, 5163227089718517184: 0, -2390746319889074981: 0, -3155759
961749610198: 0, -6462131470853536012: 0, 9140062762552358562: 0, 80
6728243168518850: 0, 2840252404351713445: 0, 3729404016061678401: 0,
4426062506757088380: 0, 7351465297342478483: 0, 6166832940074773867:
0, -6540162879973660334: 0, -5911452782275541139: 0, 436993534138376
7297: 0, -4252090345028737936: 0, 5787206100593433971: 0, -599568598
3018577174: 0, -8977002464903704777: 0, 4141550267056066160: 0, -372
4113789218761280: 0, 8699336437707801295: 0, -7463087606570658639:
1, 6353042542029784303: 0, -7659019490673543403: 0, -685800871558627
2892: 2, 9147979552476530977: 0, -2047217927446299900: 0, 7356817509
744761735: 0, 6438307314424271384: 0, -1512045118752538920: 0, 68228

83128385782857: 0, 8226797444230976375: 0, 2998287338421329445: 0, -
5262335342237635658: 0, -1488385440569208 83: 0, -240767766690381393
2: 0, 8856699033979993308: 6, -9058048487308367070: 0, -352235038601
8318782: 0, -1062045559862599550: 1, -8708587459928325418: 0, 535586
4754887818799: 0, 2922435559476828726: 0, -4047926444986504 15: 0, 65
88064823058366931: 0, 7014136991016943425: 0, 2762088427477837097:
0, -7998717642287968361: 0, 5080630937215096339: 0, 9127933702851729
309: 0, -6221141064806408609: 1, 6887885403614230406: 0, -2873628364
796243959: 0, 4458574469012493996: 0, 223512079936805273: 0, -740301
7035379837366: 0, 2623292179141091182: 0, 4719476582667954894: 0, 75
44217837828307694: 0, -942766961776748394: 0, -6819278778571023939:
0, -1309743425894648460: 0, 1598342762489897632: 0, 8000296392086258
843: 0, 4396270471672944386: 0, -8226975523401199275: 0, 73668674296
74817296: 0, -1052180273472724074: 0, -1318716143560469388: 0, 16645
01317651873294: 0, -312666622472491208: 0, -2406971415022579924: 0,
-1035733661113372642: 0, -3602123340093580418: 0, 565048874850319314
8: 0, 928643157352310400: 0, 6466744997369318990: 0, 649392462950734
9942: 0, 4932422611406245331: 0, 4397782763476390217: 0, -4959798727
654324477: 0, 8047979052150545354: 0, -3653452325043002689: 0, -8088
568969036932937: 0, -8976227240725825273: 0, 8491857808052370940: 0,
7281996616691859148: 0, 5001990609095479015: 0, -473986170369646808
7: 0, 568103843966567671: 0, -864680361955912542: 0, -74771996982989
31980: 0, -7212447109714619190: 0, -2576931780992261448: 0, 64756170
19423615473: 0, 6087194696482695496: 0, 158706666928875144: 0, -7876
649868145651560: 0, -2975890636971237740: 0, -8835102116417088085:
0, -2112950051960749743: 0, 451564349530295618: 0, -9087550968670782
297: 0, -4066505843306139594: 0, 6332594850328465974: 0, 67265619995
28176057: 0, 2712703767151739902: 0, -6126621536000756607: 0, -12486
951991649817: 0, 1068521741682112041: 0, 3161170418365679207: 0, 160
4981175388060027: 0, -704683288142809045: 0, 5416205902635908607: 0,
-3883129889842523406: 0, -1087484795163620470: 0, 440470635798181081
9: 0, -3550188402766863260: 0, 1097435781377668117: 0, -372781807820
8689261: 0, 8954150210577663095: 0, 5173462505539024634: 0, -1377534
331294947010: 0, -8520982487728599828: 0, -6055042965985033428: 0, -
5267339566800682332: 0, 3014624456093380946: 0, -550555372619896270
1: 0, 8418687035942854981: 0, 5919478840847623000: 0, -8574240234221
926219: 0, -2094841544693906755: 0, 8898028074783020859: 0, -7199818
045040073621: 0, 27150510447983163: 0, -1798716874013868713: 0, 5945
983064221703459: 0, -7025636848878432379: 3, -3846851488981388811:
0, -2996623882548316683: 0, -5614626709597289215: 0, -35514891736464
66248: 0, -1172436730268885899: 0, 1358416626712930751: 0, 361686935
4923228343: 0, -7420308182340661922: 0, -302948163249823070: 0, 6708
655079793636681: 0, -7225794920531460000: 1, 8851792423098679991: 0,
3439272043415265806: 0, -5417918107453395058: 0, 19217013244351527:
0, -9192646632135461500: 0, -418764415311399 31: 0, 47195338272395525
94: 0, 3796090529117825535: 0, -5211999080833329591: 0, -59810941531
74193285: 0, -3399432583333620552: 0, 3547353429757514798: 0, -15590
21919735700426: 1, 3616107916761109952: 0, -8223466967289089571: 0,
-3753447336270323476: 0, -1647996445431029647: 0, 867594702874925804
7: 0, 2520512555334920049: 0, -7447531866298527239: 0, -557340096892
0570690: 0, -2142261972033696989: 0, -3932601231087819585: 1, -56468
87663647871807: 0, 8391784887615319396: 0, -6138209347992530154: 0,
-1033081430678230274: 0, -2468859136927730194: 4, -61435274086794663
53: 0, -5285164094814222996: 0, -6724695945672375 57: 0, -74607170391
75514988: 0, -7421314663751741873: 0, -7671848235273724617: 0, 62175
47620867972385: 0, 9101951070377939893: 0, 5870068309317425606: 0, 5
231520007928499304: 0, -8854206103529193073: 0, -635563280170086343
9: 0, -2128468192997711912: 0, 3112042719631182891: 0, -545566146959
4158107: 0, -2196655686226148577: 0, -5091825614765119113: 0, -87865
94074161291576: 0, 5223396309590178252: 0, 1789740374448463988: 0, -
1531725273919955886: 0, 6531866248404734493: 0, -714236522911281708:

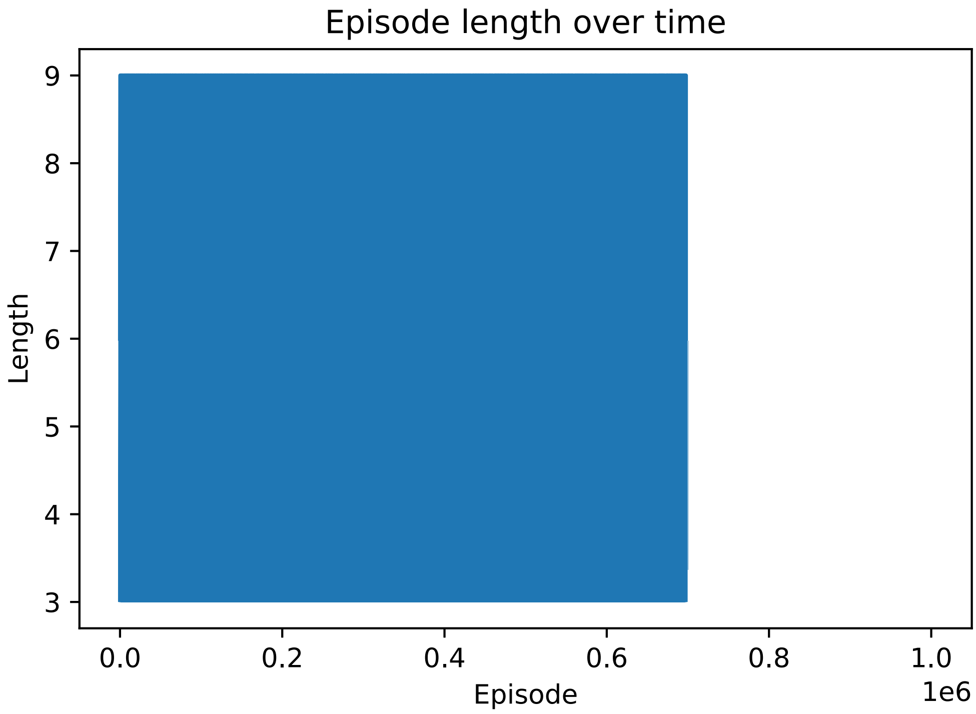
0, -2797919972975089947: 0, 7160025399696512238: 0, -373103687234606
9259: 0, 8487550052153149877: 0, 1406530271444472857: 0, -4889224038
46732905: 0, 6188079246549514741: 0, -8661304460079458954: 0, 372960
9797160779231: 0, -4317832716000192828: 0, -3051818190647672316: 0,
7931429250059047631: 0, 3012174304843787251: 0, -878999192850918516
4: 0, 8219548191555861096: 0, 5080414414082307 88: 0, -57627410268325
18650: 0, -1800172709801469957: 0, 6705830185397624173: 0, -11961222
96836791442: 0, 2638505144644789769: 0, 4669188629170701063: 0, 8302
928048542207780: 0, -7874119454290355996: 0, 6495001927104800336: 0,
-3420487631561019612: 0, 9618979279211366 71: 0, 4383376451193215718:
0, -1375570119374002657: 0, 5120604052192121857: 0, -769364093582461
9312: 0, 8605305087593506334: 0, -7842752799927479325: 0, -251305115
3409208731: 0, 4330472103638419129: 0, 8967632187622893357: 0, -3110
824323796129899: 0, 8746730475619360385: 0, -6146533625052131316: 0,
4426052302895354098: 0, -7702348011381709719: 0, -251559232895671301
9: 0, 4046232163081232513: 0, 3232915163494912506: 0, -9490655332872
70177: 0, 444839421744947 43: 0, 6615263159932059492: 0, -31845469813
15753663: 0, -2688098793466575996: 0, -2813471368328714975: 0, -5311
841414416177139: 0, -7522639794516304612: 0, -2473746362237026979:
0, 6554735663788078558: 0, -5640458121931261 95: 0, -4040129875838688
308: 0, 5188348341802652469: 0, -8899033286474298 34: 0, -69296986971
95216603: 0, -2916503976795199275: 0, -2209605701429611059: 0, -2554
554160774078262: 0, -3546200364325154453: 0, -1452372470625222386:
0, 4486429140916756737: 0, 1883238975969158507: 0, 47979994989622512
12: 0, 5395222060777555662: 0, -9027082860264106283: 1, -72932254968
24095362: 0, 7697881277219435753: 0, 2815042278863232271: 0, -640045
9936771325304: 0, -6089992047037646353: 1, -2217649070232285215: 0,
8236888099419773586: 0, 8123887155312025824: 0, 2710069639603399039:
0, -6067509577300354985: 0, 5298393196643841309: 0, 2328353759262716
239: 0, -1466956233050061567: 0, -4446239659750465364: 0, 5293736687
203953379: 0, -7573296002462827584: 0, 8346331179597878035: 0, 60510
3714466759593: 0, 5519888695921858193: 0, -3044218504807824277: 0, -
1028595790744220 37: 0, 1796469419638802564: 0, 8452536319253146470:
0, -7823709704442834417: 0, 7392315887191414395: 0, 8891363983589291
942: 1, 6394220426763568580: 0, -3498763117482120054: 0, -7817921314
64874442: 0, -4010535041999640 05: 0, 5278678187532310374: 0, -212021
625911994008: 0, -1536258589672032554: 0, 5060141612178602976: 0, -2
514575559886850352: 0, 8607982872515240230: 0, -6581507357315495943:
0, -1937763502700280675: 0, 4090141282549007219: 0, 1852089447501880
355: 0, -7128292799150948 07: 0, -6320265307051007606: 0, 67962108617
17697511: 0, 704651261563767 67: 0, 7505499839410593683: 0, 617623384
2208363929: 1, 1314704363649322562: 0, -4231453973189670380: 0, -895
814109630322181 4: 0, -321544199204128656: 0, -6038139043191912878:
0, -7359546343021362833: 0, -1239968635456229297: 0, -19950939898527
7198: 0, 5705484735013472070: 1, 4631069547645800482: 0, 65608377441
93724723: 0, -8036828767026790469: 0, 2060895355605192921: 0, 260255
8657279208253: 0, 3699246752843519 60: 0, -9432785916091533 83: 0, -12
05876681251943607: 0, 7981887807614060237: 0, 1326293696705300869:
0, 7398595387516165 17: 0, -6588196163369693171: 0, 66655386803095894
56: 0, -6215062646038970709: 1, 5514530176848975758: 0, 446478886787
2025996: 1}

In [56]:

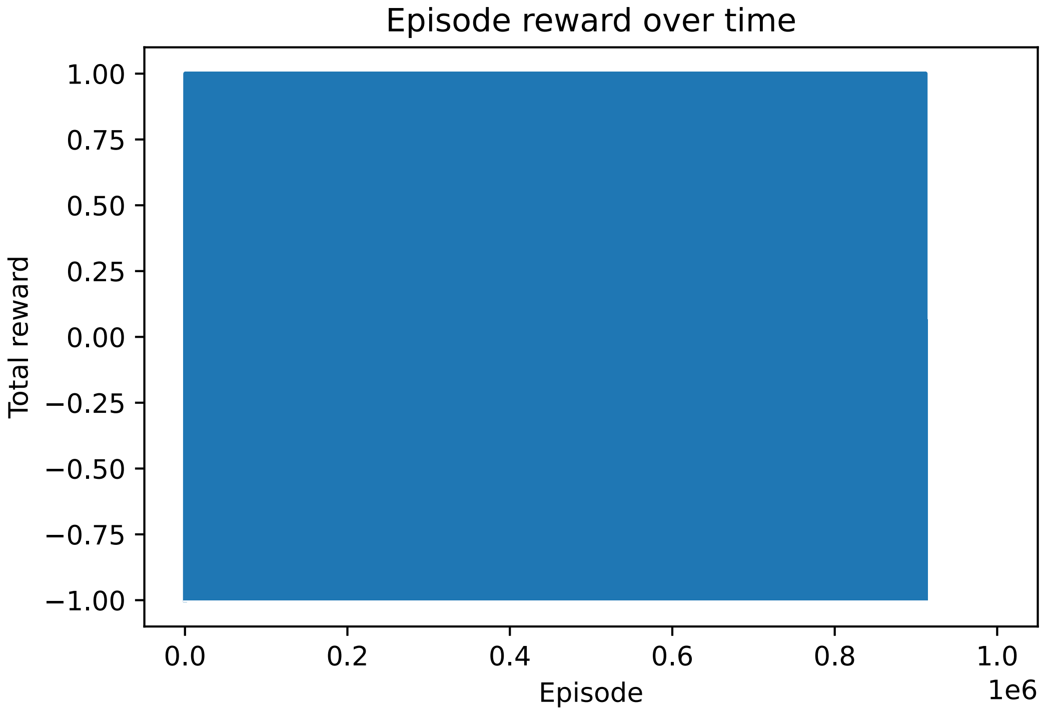
```python
import matplotlib.pyplot as plt

plt.plot(length_episode)
plt.title('Episode length over time')
plt.xlabel('Episode')
plt.ylabel('Length')
plt.show()


plt.plot(total_reward_episode)
print(total_reward_episode[-100:])
plt.title('Episode reward over time')
plt.xlabel('Episode')
plt.ylabel('Total reward')
plt.show()
```




```
[-1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, 0, -1, -1, 1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, 1, 1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1,
-1, 1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, -1, -1, -1, -1, -
1, -1, -1, 1, 0, -1, -1, 1, 1, -1, -1, -1, -1, -1]
```



In [53]:

```
state = env.reset()
state = hash(tuple(state.reshape(-1)))

player = 1
is_done = False
while not is_done:
    if env.to_play() == player:
        available_action = env.legal_actions()
        print("available action:",available_action)
        action = epsilon_greedy_policy(state, optimal_Q, available_action)
        next_state, reward, is_done = env.step(action)
        next_state = hash(tuple(next_state.reshape(-1)))
        print("RL agent")
        print(env.action_to_string(action))
    else:
        action = env.expert_agent()
        next_state, reward, is_done = env.step(action)
        next_state = hash(tuple(next_state.reshape(-1)))
        print("Expert agent")
        print(env.action_to_string(action))

    state = next_state
    env.render()
```

```
available action: [0, 1, 2, 3, 4, 5, 6, 7, 8]
RL agent
Play row 1, column 1
 O |   |
---+---+---
   |   |
---+---+---
   |   |
Expert agent
Play row 2, column 1
 O |   |
---+---+---
 X |   |
---+---+---
   |   |
available action: [1, 2, 4, 5, 6, 7, 8]
RL agent
Play row 1, column 2
 O | O |
---+---+---
 X |   |
---+---+---
   |   |
Expert agent
Play row 1, column 3
 O | O | X
---+---+---
 X |   |
---+---+---
   |   |
available action: [4, 5, 6, 7, 8]
RL agent
Play row 2, column 2
 O | O | X
---+---+---
 X | O |
---+---+---
   |   |
Expert agent
Play row 3, column 3
 O | O | X
---+---+---
 X | O |
---+---+---
   |   | X
available action: [5, 6, 7]
RL agent
Play row 3, column 2
 O | O | X
---+---+---
 X | O |
---+---+---
   | O | X
```

In-lab exercise and homework: DQN

For any but trivial problems, tabular Q-Learning will fail due to the difficulty of storing the value of every state-action pair and the enormous amount of time it takes for every cell in the table to converge.

The first thing we'd like to do, then, is approximate Q[s,a] with a function q(s,a) using fewer than the 177,147 parameters we used with tabular Q-learning.

As a first attempt, we might convert the input state-action pairs to a binary representation and use a linear output for the Q value. We would use a one-hot representation for each of the 9 game cells in s (3 choices) and the 9 possible actions. This would give us 36 inputs. The neural network would be multi-layer. With two fully connected layers of 10 units each, we'd have 10x37+10x11+11=491 parameters. That's a lot less than tabular Q-learning!

However, take a look at DQN. Their model has a NN to process the input images from the Atari console and has one output for each possible action. With this approach, we would have just 27 inputs and 9 outputs. Two fully connected layers of 10 units each would give us 10x28+10x11+9x11=489 parameters, which is similar in size to the above and has the advantage of only having to be executed once to get the value of every action. We could go with this tiny network or make it a lot bigger and still beat the Q table's size by a wide margin.

So, the first step will be to replace the Q table with a Q network. Go ahead and write your network class in PyTorch. Deep Q-Learning

Alright, you have a neural network to replace the Q table, but how to learn its parameters?

Take a look again at Mnih et al. (2015). The method they recommend is experience replay in which they store recent state-action-reward tuples in a buffer and train on random subsamples from the buffer.

This is probably overkill for tic-tac-toe, but let's implement it anyway!

You'll have to decide some things, such as what to do if the agent samples an illegal move. Give a negative reward? Give a 0 reward? Think about it and try an approach.

Go ahead and write the DQN algorithm, using your simple fully connected network in place of the CNN. Get it learning!

Report on your experiments and results by next week.

In [1]:

```
import gym
import tensorflow as tf
import numpy as np
import math
import time
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torchvision.transforms as T
import random
device = torch.device("cuda" if torch.cuda.is_available() else "cpu ")
```

DQN

Note this DQN consists of two fully connected layers of 10 unit each

In [6]:

```
class DQN(nn.Module):

    def __init__(self, inputs, outputs):
        super(DQN, self).__init__()
        self.fc1 = nn.Linear(inputs, 10)
        self.fc2 = nn.Linear(10, outputs)
        self.relu = nn.ReLU()
        self.softmax = nn.Softmax()

    def forward(self, x):
        x = self.relu(self.fc1(x)) x = self.fc2(x)
        return x

    def act(self, state, epsilon):
        # Get an epsilon greedy action for given state
        if random.random() > epsilon: # Use argmax_a Q(s,a)
            state = autograd.Variable(torch.FloatTensor(state).unsqueeze(0), vol
atile=True).to(device)
            q_value = self.forward(state)
            q_value = q_value.cpu()
            action = q_value.max(1)[1].item()
        else: # get random action
            action = random.randrange(env.action_space.n)
        return action
```

Replay Memory

In [8]:

```
class ReplayMemory(object):
    def __init__(self, capacity):
        self.capacity = capacity
        self.memory = [] # Queue-like self.position = 0
    def push(self, experience):
        """Saves a experience."""
        if len(self.memory) < self.capacity:
            self.memory.append(None) # if we haven't reached full c apacity, we
 append a new transition
            self.memory[self.position] = experience
            self.position = (self.position + 1) % self.capacity # e.g i f the ca
pacity is 100, and our position is now 101, we don't append to
                       # position 101 (impossible), but to position 1 (its remai
nder), overwriting old data
    def sample(self, batch_size):
        return random.sample(self.memory, batch_size)
    def __len__(self):
        return len(self.memory)
```

Losses

In [ ]:

```python
def compute_td_loss(model, batch_size, gamma=0.99):

    # Get batch from replay buffer
    state, action, reward, next_state, done = replay_buffer.sample(batch_size)

    # Convert to tensors. Creating Variables is not necessary with more recent PyTorch versions.
    state      = autograd.Variable(torch.FloatTensor(np.float32(state))).to(device)
    next_state = autograd.Variable(torch.FloatTensor(np.float32(next_state)), volatile=True).to(device)
    action     = autograd.Variable(torch.LongTensor(action)).to(device)
    reward     = autograd.Variable(torch.FloatTensor(reward)).to(device)
    done       = autograd.Variable(torch.FloatTensor(done)).to(device)

    # Calculate Q(s) and Q(s')
    q_values      = model(state)
    next_q_values = model(next_state)

    # Get Q(s,a) and max_a' Q(s',a')
    q_value          = q_values.gather(1, action.unsqueeze(1)).squeeze(1)
    next_q_value     = next_q_values.max(1)[0]
    # Calculate target for Q(s,a): r + gamma max_a' Q(s',a')
    # Note that the done signal is used to terminate recursion at end of episode.
    expected_q_value = reward + gamma * next_q_value * (1 - done)

    # Calculate MSE loss. Variables are not needed in recent PyTorch versions.
    loss = (q_value - autograd.Variable(expected_q_value.data)).pow(2).mean()

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    return loss
```

In [ ]:

```
def train(env, model, eps_by_episode, optimizer, replay_buffer, episodes = 10000
, batch_size=32, gamma = 0.99):
    losses = []
    all_rewards = []
    episode_reward = 0
    tot_reward = 0
    tr = trange(episodes+1, desc='Agent training', leave=True)

    # Get initial state input
    state = env.reset()

    # Execute episodes iterations
    for episode in tr:
        tr.set_description("Agent training (episode{}) Avg Reward {}".format(epi
sode+1,tot_reward/(episode+1)))
        tr.refresh()

        # Get initial epsilon greedy action
        epsilon = eps_by_episode(episode)
        action = model.act(state, epsilon)

        # Take a step
        next_state, reward, done, _ = env.step(action)

        # Append experience to replay buffer
        replay_buffer.push(state, action, reward, next_state, done)

        tot_reward += reward
        episode_reward += reward

        state = next_state

        # Start a new episode if done signal is received
        if done:
            state = env.reset()
            all_rewards.append(episode_reward)
            episode_reward = 0

        # Train on a batch if we've got enough experience
        if len(replay_buffer) > batch_size:
            loss = compute_td_loss(model, batch_size, gamma)
            losses.append(loss.item())

    plot(episode, all_rewards, losses)
    return model,all_rewards, losses
```

In [ ]:

```
import importlib
from collections import defaultdict
import torch
import numpy

game_name = 'tictactoe'
game_module = importlib.import_module("games." + game_name)
env = game_module.Game()
```

In [ ]:

```
state = env.reset()
available_action = env.legal_actions()
BATCH_SIZE = 5 # original = 128
GAMMA = 0.999 # original = 0.999
N_EPISODES = 50000 # total episodes to be run MEMORY_SIZE = 100000 # original =
 10000 EPS_START = 0.9 # original = 0.9
EPS_END = 0.01 # original = 0.05
EPS_DECAY = 3000 # original = 200
policy_net = DQN(state.flatten().shape[0], len(available_action)).t
o(device)
optimizer = optim.RMSprop(policy_net.parameters())
memory = ReplayMemory(MEMORY_SIZE)
player = 1

for episode in range(N_EPISODES):
    if episode % 10000 == 9999:
        print("episode: ", episode + 1)
    state = env.reset()
    state = torch.FloatTensor(state).view(27).to(device)
    is_done = False while not is_done:
    if env.to_play() == player:
        available_action = env.legal_actions()
        action = select_action(state)
        next_state, reward, is_done = env.step(action)
        next_state = torch.FloatTensor(next_state).view(27).to(device)
    else:
        action = env.expert_agent()
        next_state, reward, is_done = env.step(action)
        next_state = torch.FloatTensor(next_state).view(27).to(
    if is_done:
        reward = -reward
        optimize_model(state, False) if is_done:
        break
    memory.push((state, reward, is_done))
    state = next_state
print('Complete')
```

I found this lab to be difficult. I spent some time trying to understand it and started to get a gist of how it works.