Q1:

	$Q(1)$	$Q(2)$	$Q(3)$	$Q(4)$	
0.	0	0	0	0	
$A_1 = 1, R_1 = -1$					$Q(1)=Q(2)=Q(3)=Q(4)$, ε may have occured
1.	-1	0	0	0	
$A_2 = 2, R_2 = 1$					$Q(2)=Q(3)=Q(4)>Q(1)$, ε may have occured
2.	-1	1	0	0	
$A_3 = 2, R_3 = -2.$					$Q(2)>Q(3)=Q(4)>Q(1)$, ε may have occured.
3.	-1	-0.5	0	0	
$A_4 = 2, R_4 = 2.$					$Q(3)=Q(4)>Q(2)>Q(1)$, ε definitely occured ✓
4.	-1	0.33	0	0	
$A_5 = 3, R_5 = 0$					$Q(2)>Q(3)=Q(4)>Q(1)$, ε definitely occured ✓
5.	-1	0.33	0	0	

Q2:

when α_n is non-stationary:

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha_n [R_n - Q_n] = \alpha_n R_n + (1-\alpha_n) Q_n \\
&= \alpha_n R_n + (1-\alpha_n)[\alpha_{n-1} R_{n-1} + (1-\alpha_{n-1}) Q_{n-1}] \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1}) Q_{n-1} \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\alpha_{n-2} R_{n-2} + \\
&\qquad \cdots + \prod_{i=2}^{n}(1-\alpha_i)\cdot \alpha_1 R_1 + \prod_{i=1}^{n}(1-\alpha_i)\cdot Q_1
\end{aligned}
$$

$$
= Q \cdot \prod_{i=1}^{n}(1-\alpha_i) + \sum_{j=1}^{n-1}\left[\alpha_j R_j \cdot \prod_{i=j+1}^{n}(1-\alpha_i)\right] + \alpha_n R_n
$$

Q3:

(a) sample-average estimate in Equation 2.1 is unbiased.

rewrite eq 2.1 as:

$$Q_n(a) = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

$\because E(R_i) = q^*$ for $i=1,2,\cdots n-1$

$\therefore E(Q_n(a)) = \frac{(n-1)\cdot q^*}{n-1} = q^*$

$\therefore$ equation 2.1 is unbiased

(b) when $\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = 1$, equation is unbiased, when $Q_i = 0$

eq 2.6:

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

take expectation of both sides of the equation

$$E(Q_{n+1}) = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} E(R_i)$$

when $E(Q_{n+1}) = q^*$, we say it's unbiased.

when $Q_1 = 0$.

$$E(Q_{n+1}) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} \cdot E(R_i)$$

$\because E(R_i) = q^*$ for $i=1,2,\cdots n$.

$$\therefore E(Q_{n+1}) = \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} q^*$$

$\therefore$ when $\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = 1$, equation is unbiased

(c) As derived in (b), when n is finite, Q_n will be unbiased if:

$$Q_1 = 0, \quad \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = 1$$

(d) when n is infinite:

first term $(1-\alpha)^n Q_1$ will decrease asymptotically to 0, since $0<\alpha<1$

second term $\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$:

when $n \to \infty$, $\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} = \alpha \cdot \frac{1-(1-\alpha)^{n-1}}{1-(1-\alpha)} = \alpha \frac{1-(1-\alpha)^{n-1}}{\alpha} = 1-(1-\alpha)^{n-1}$

again $(1-\alpha)^{n-i}$ will decrease asymptotically to 0, thus $1-(1-\alpha)^{n-1} \to 1$

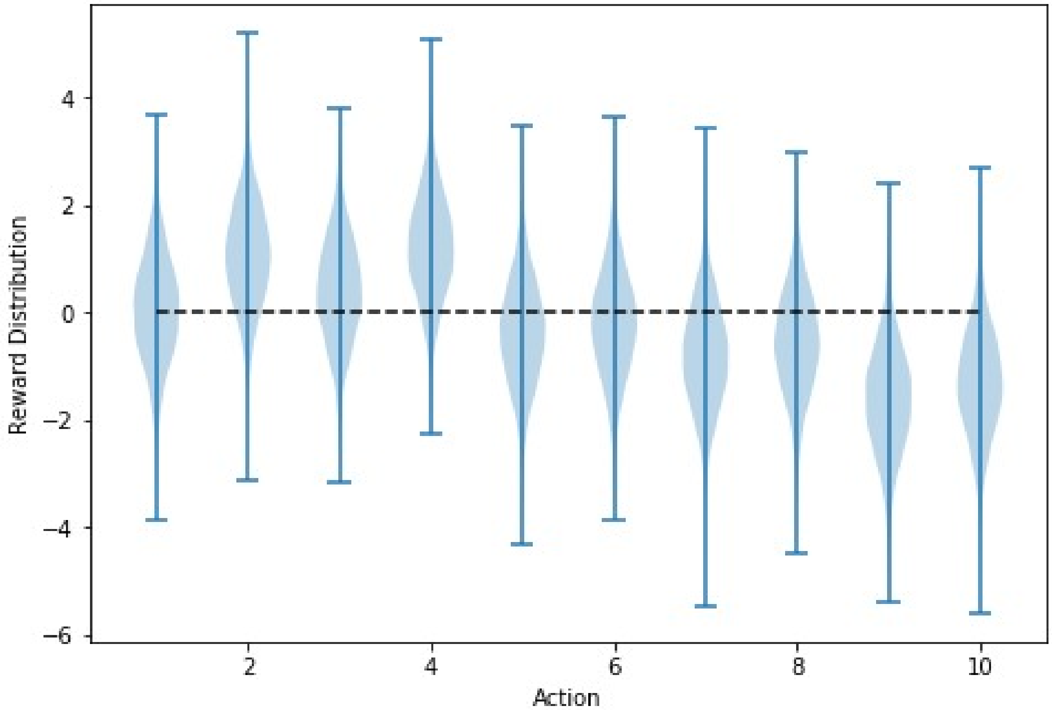
i.e. $\sum_{i=1}^{n} \alpha(1-\alpha)^{n-i}$ will converge to 1 asymptotically.

$\therefore$ the condition of Q_n to be unbiased, will be satisfied asymptotically

$\therefore$ Q_n is asymptotically unbiased as $n \to \infty$.

(e) By introducing hyperparameters like Q_1 and α, we can control the initial state conviniently. Also since we cannot approach asymptotic point by implementing $n \to \infty$, We can control learning rate by changing alpha to make it learning faster, and at the same time tracking the non stationary environment.

Q4: (All levers are chosen 10000 times in total)



Q5:

Q5:

In the long run, the method with $\varepsilon = 0.01$ will perform the best.

when step size $\rightarrow \infty$:

$\varepsilon = 0$: the agent will always choose the action it first discovered with positive reward, and then stick to it.

$\varepsilon = 0.01$: optimal action percentage:

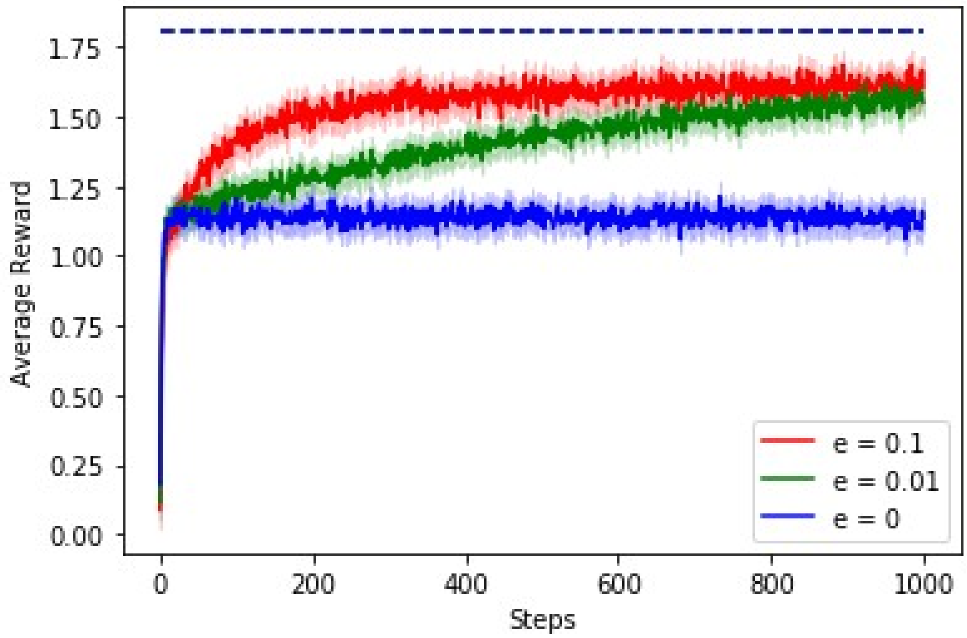
$$0.99 \times 1 + 0.01 \times \frac{1}{10} = 99.1\%$$

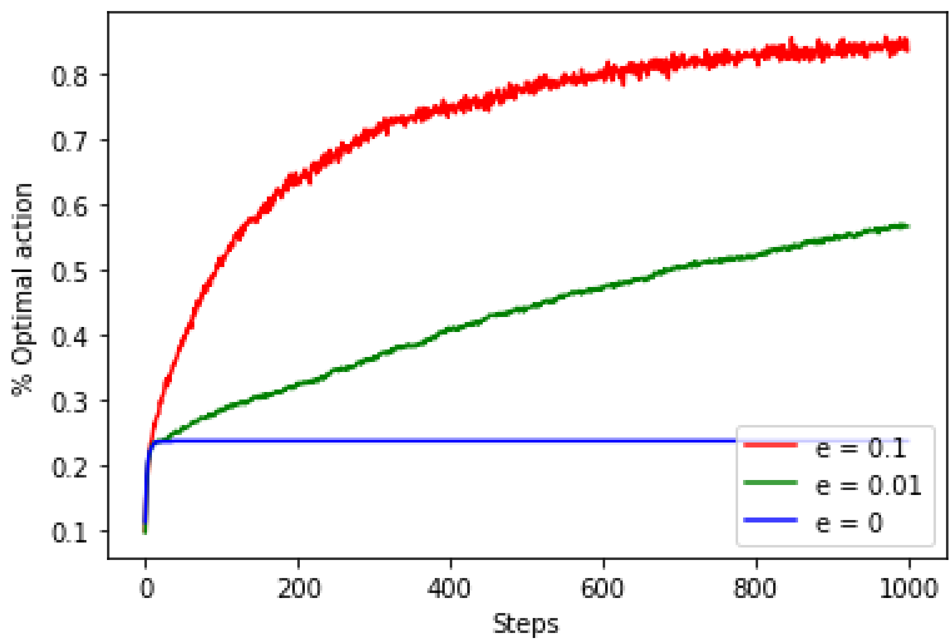
$\varepsilon = 0.1$: optimal action percentage:

$$0.9 \times 1 + 0.1 \times \frac{1}{10} = 91\%$$

$\therefore$ the optimal action percentage of $\varepsilon = 0.01$ will be 8.1% better than the method with $\varepsilon = 0.1$.

Q6: (Medium setting)



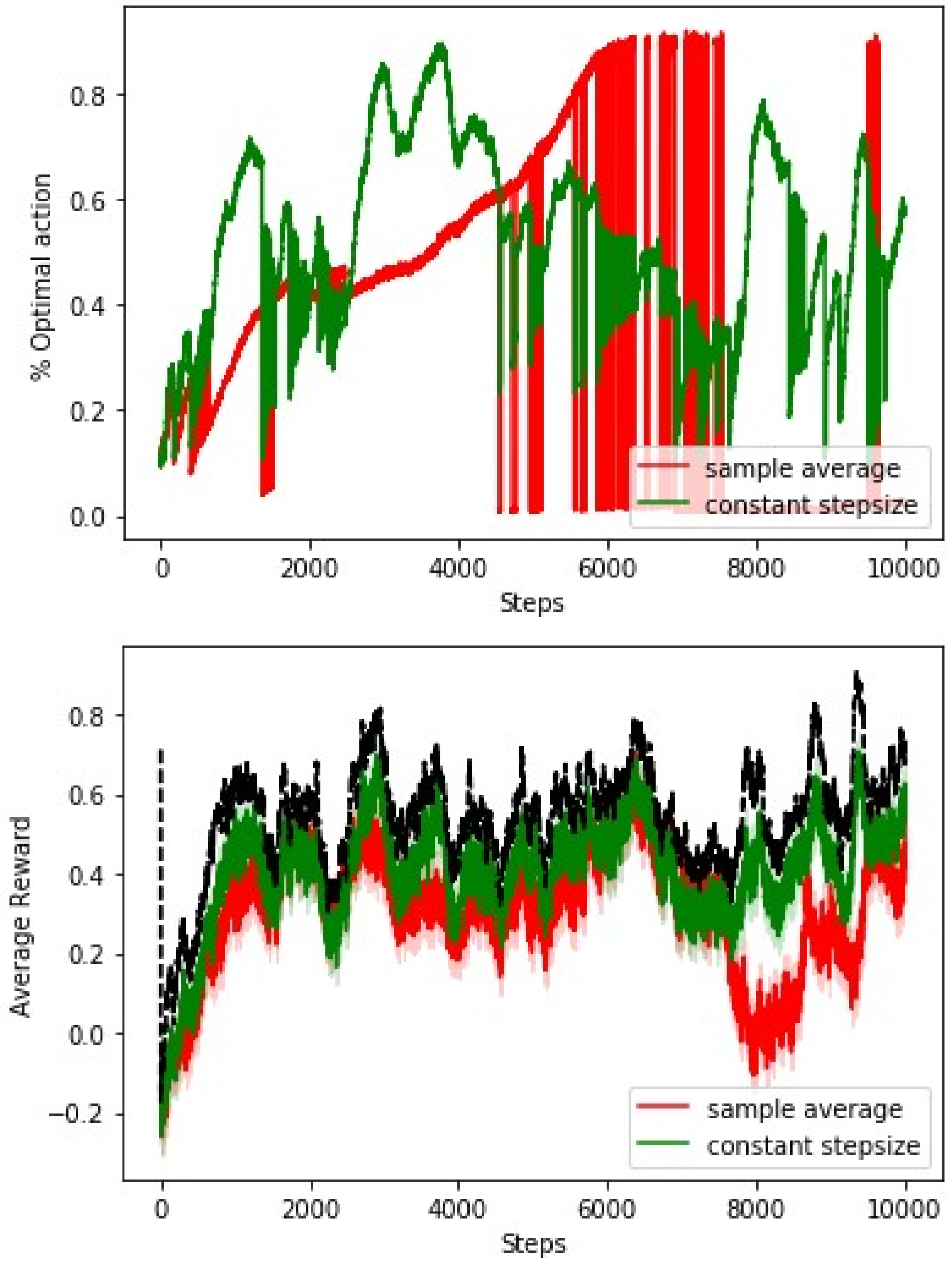


% Optimal action and Average reward graph for e-Greedy method with different e value

Q6: (Written)

It hasn't. Since those performance could only be achieved when $n \to \infty$, in our experiment the steps, no matter 10^3 or 10^4, is not large enough to asymptotically achieve it.

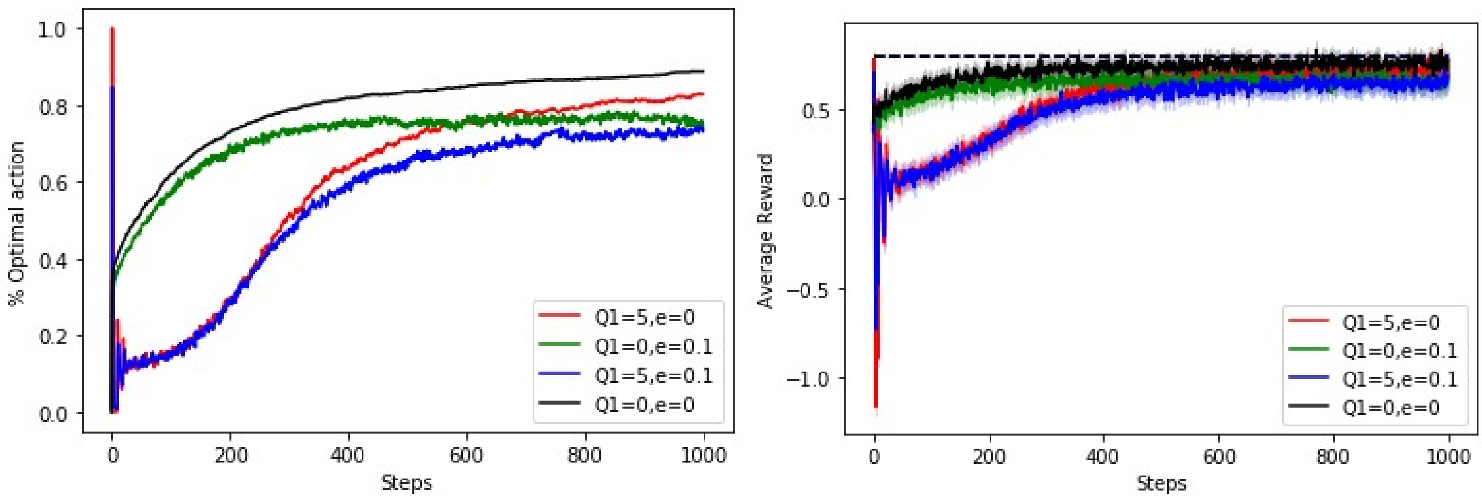
Q7: (Large setting)



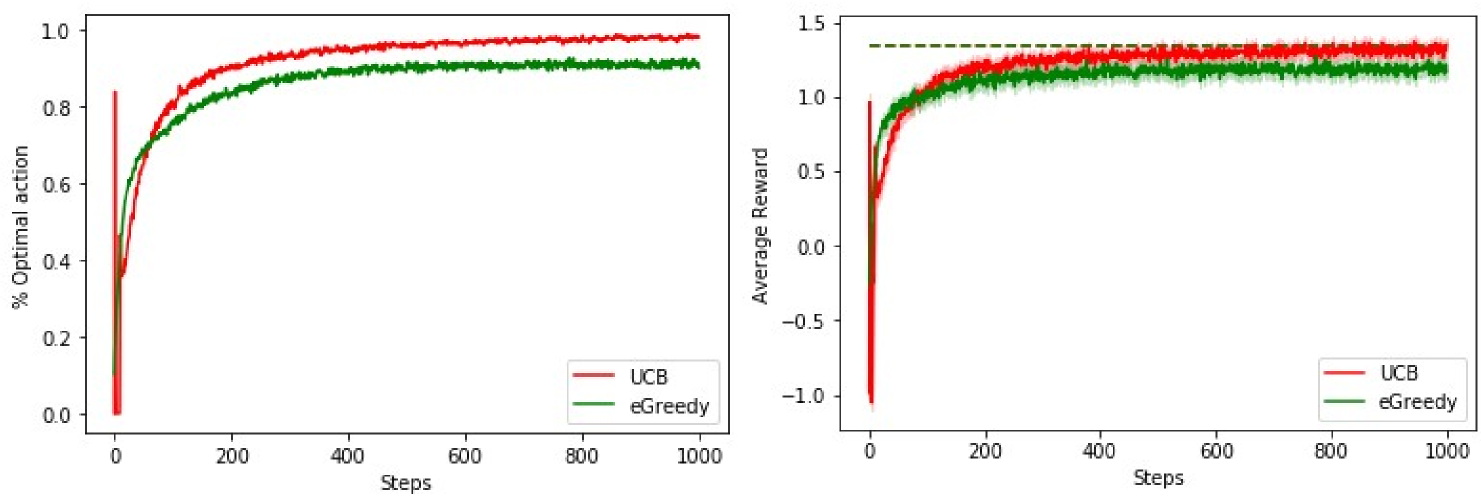
comparison between sample average and constant stepsize method

We can see that in general constant stepsize method performs better than constant stepsize method, both in average reward and percentage of optimal action, and sample average method is more vulnerable in non-stationary case. For example, the red curve deviated a lot around 8000tth step while the green curve still tracks it well. This may probably because of sample average method won't 'forget' anything.

Q8: (Both medium setting)



% Optimal action and Average reward graph for Optimistic Initial Value method



% Optimal action and Average reward graph for UCB method

Q8: (Written).

In optimistic initial values, the initial value for all actions in above the reward that the optimal action could achieve. In first 10 steps, whenever after the agent chooses an action which hasn't been chosen, its value will decrease, and then the agent will choose another action which $Q(a) = Q_1$. After 10 steps, all ten actions have been chosen once, and the optimal action value is likely to be substracted the least amount among all action, thus at the 11th step the agent is likely to choose the optimal action, which result in a spike in the graph.

In UCB, the situation is similar to above: the second term in the equation — $c\sqrt{\frac{\ln t}{N_t(a)}}$ makes the agent likely to choose those actions which have choosen less frequently. During the first 10 steps, those unchosen actions will have infinity action value, resulted by the denominator $N_t(a)$ equals to 0. Because of that, the agent will choose all ten actions in the first 10 steps, and again at the 11th step the optimal action will likely to have higher action value, thus it will be chosen again, which result in a spike.

```
[5 5 5 5 5 5 5 5 5 5]
[4.211 5.    5.    5.    5.    5.    5.    5.    5.    5.   ]
[4.211 4.444 5.    5.    5.    5.    5.    5.    5.    5.   ]
[4.211 4.444 4.548 5.    5.    5.    5.    5.    5.    5.   ]
[4.211 4.444 4.548 4.371 5.    5.    5.    5.    5.    5.   ]
[4.211 4.444 4.548 4.371 4.609 5.    5.    5.    5.    5.   ]
[4.211 4.444 4.548 4.371 4.609 4.414 5.    5.    5.    5.   ]
[4.211 4.444 4.548 4.371 4.609 4.414 4.4   5.    5.    5.   ]
[4.211 4.444 4.548 4.371 4.609 4.414 4.4   4.481 5.    5.   ]
[4.211 4.444 4.548 4.371 4.609 4.414 4.4   4.481 4.384 5.   ]
[4.211 4.444 4.548 4.371 4.609 4.414 4.4   4.481 4.384 4.518]
[4.211 4.444 4.548 4.371 4.191 4.414 4.4   4.481 4.384 4.518]
```

First 11 iteration of updating Q value using Optimistic Initial Value

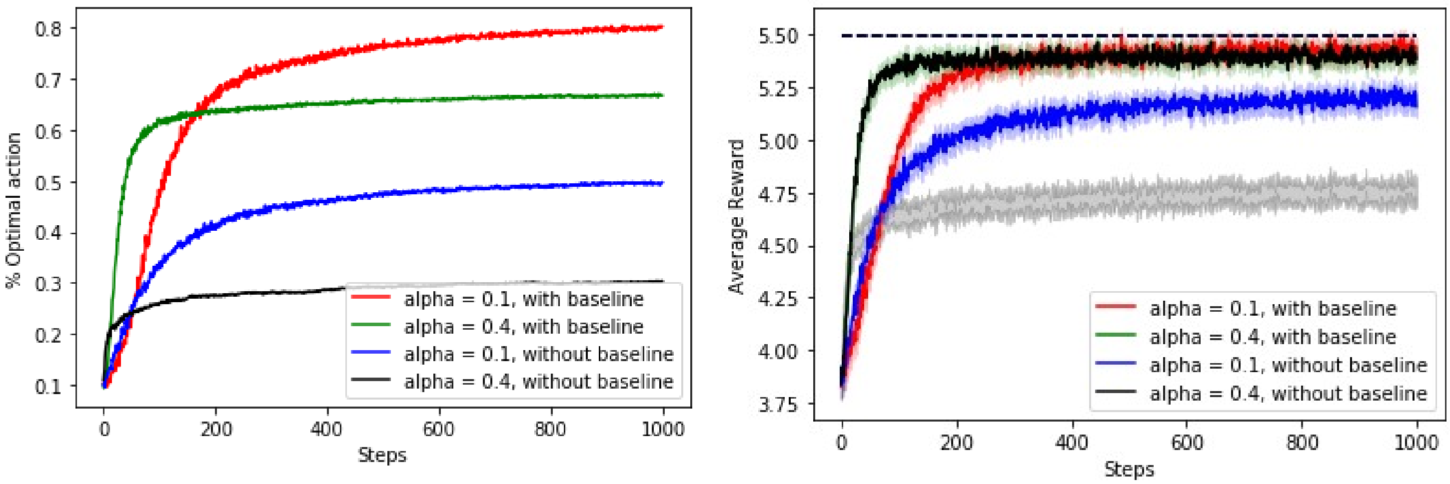
```
[0 0 0 0 0 0 0 0 0 0]
[-0.585  0.     0.     0.     0.     0.     0.     0.     0.     0.   ]
[-0.585  0.36   0.     0.     0.     0.     0.     0.     0.     0.   ]
[-0.585  0.36   0.476  0.     0.     0.     0.     0.     0.     0.   ]
[-0.585  0.36   0.476  0.889  0.     0.     0.     0.     0.     0.   ]
[-0.585  0.36   0.476  0.889 -0.431  0.     0.     0.     0.     0.   ]
[-0.585  0.36   0.476  0.889 -0.431  1.294  0.     0.     0.     0.   ]
[-0.585  0.36   0.476  0.889 -0.431  1.294  0.56   0.     0.     0.   ]
[-0.585  0.36   0.476  0.889 -0.431  1.294  0.56   0.317  0.     0.   ]
[-0.585  0.36   0.476  0.889 -0.431  1.294  0.56   0.317  1.108  0.   ]
[-0.585  0.36   0.476  0.889 -0.431  1.294  0.56   0.317  1.108  3.106]
[-0.585  0.36   0.476  0.889 -0.431  1.294  0.56   0.317  1.108  2.52 ]
```

First 11 iteration of updating Q value using UCB

Q9: (Medium setting)

I implemented Gradient Bandit method and managed to reproduce the **Figure 2.5** from the book. All the parameters are the same as the book: every initial Q value shifted up by 4, and 4 curves drew regarding alpha = 0.1 or 0.4, with or without baseline.

The result looks like this:



% Optimal action and Average reward graph for Gradient Bandit Method
(the black curve in average reward graph was misdirect on the green line, and it should be within the gray shade originally)

I found that gradient bandit algorithm with baseline performs significantly better than the ones without baseline, and the baseline could be easily calculated incrementally, without spending too much memory space.