MA374 Financial Engineering Lab
Assignment - 03

Name - Shivendu Mishra
Roll Number -200123050
email id - m.shivendu@iitg.ac.in

Question - 1

In this problem we have to write a program to find the inital price of an A***merican Call and Put*** option using the data given below and also do the sensitivity analysis with respect to the various parameters .

Given Data - S(0) = 100 , K = 100,T = 1,M = 100,r = 8%, $\sigma = 20\%$

In order to implement this I used the standard binomial pricing algorithm which we use for the pricing of European option with slight modification. Let C_i^A denote the price of an American option at the ith step of a binomial model starting with sate 0, clearly we have to find C_0^A . The formulae used with continuous compounding conventions is:-

$$C_i^A = max\left(e^{-\Delta t}\left(\hat{p} * C_{i+1}^A(H)\right) + \hat{q} * C_{i+1}^A(T), max\left(\left(S(t_i) - K\right), 0\right)\right), where\ \Delta t = T/M$$
$$u = e^{\sigma\sqrt{(\Delta)} + (r - \sigma^2/2)t}, d = u = e^{-\sigma\sqrt{(\Delta)} + (r - \sigma^2/2)t}$$
$$\hat{p} = e^{r\Delta} - d/u - d \quad \text{and q = 1-p}$$

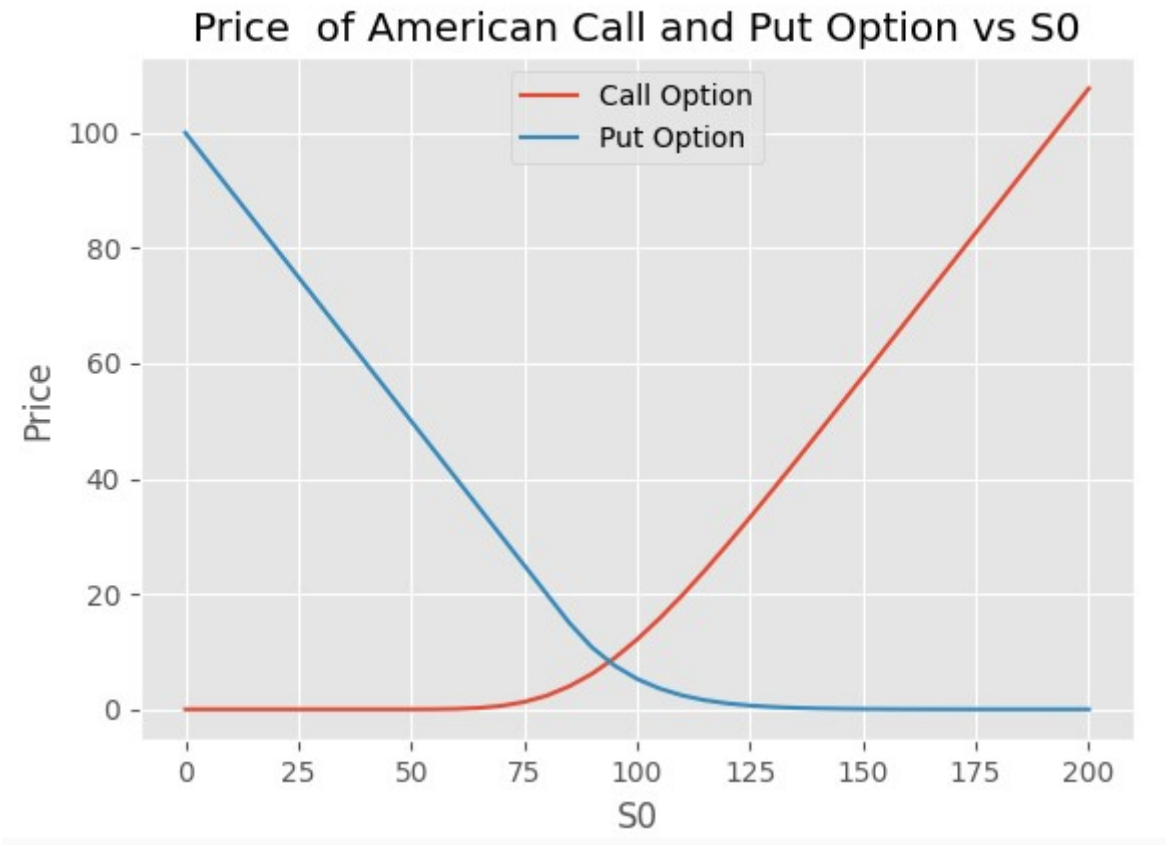
also $C_M^A = max\left(S(T) - K, 0\right)$. The formulae is similar for put option with obvious changes. I implemented the above algorithm in an iterative fashion and calculated the price of American Call and American Put option as follows:-

Price of American Call Option = 12.123047074012453
Price of Amercian Put Option = 5.279837145989147

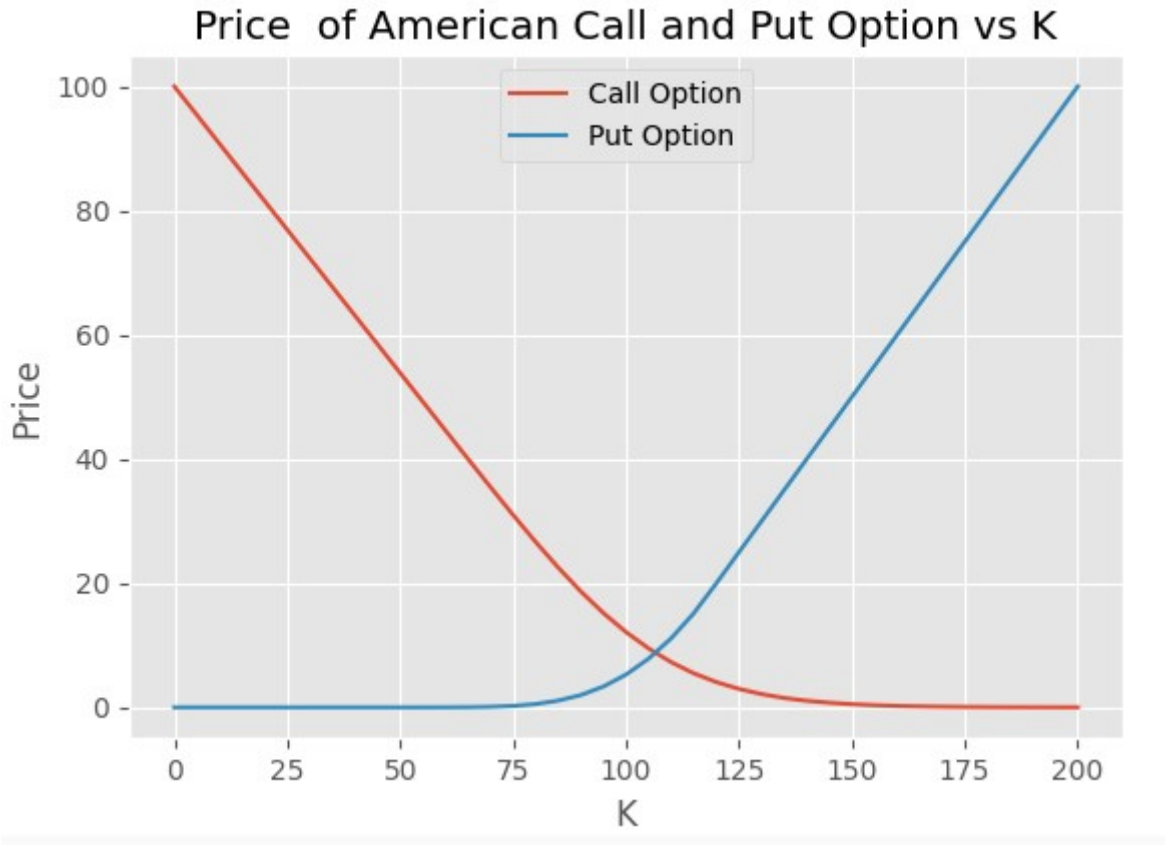
Sensitivity Analysis

I performed sensitivity analysis by changing the parameters S(0),K,r, σ and M and plotted the results . I have attached the screenshots of the plot below:-

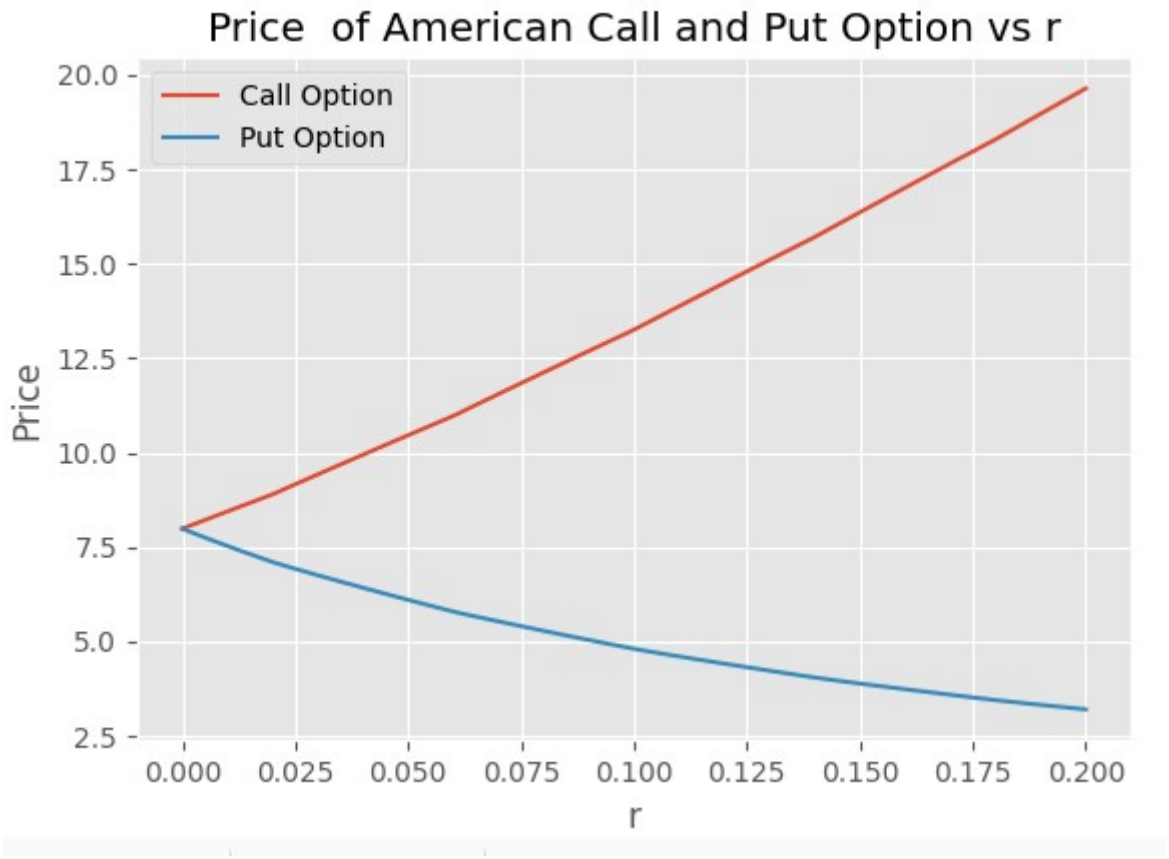
Variation with S(0)



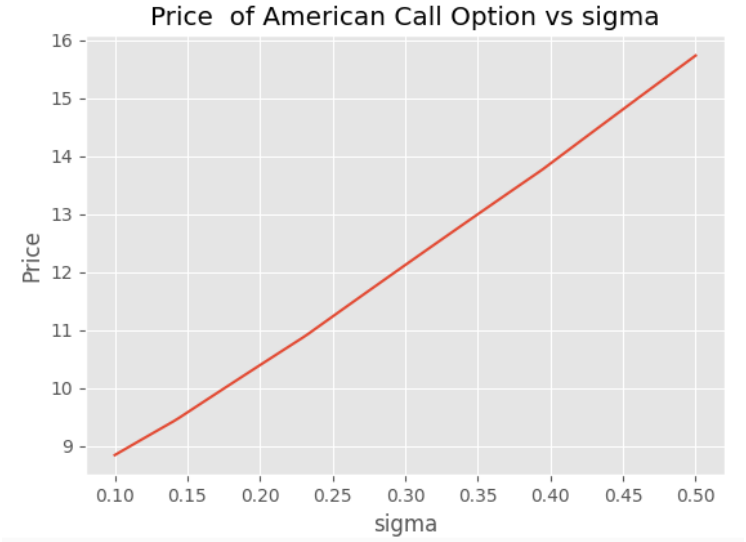
Variation with K

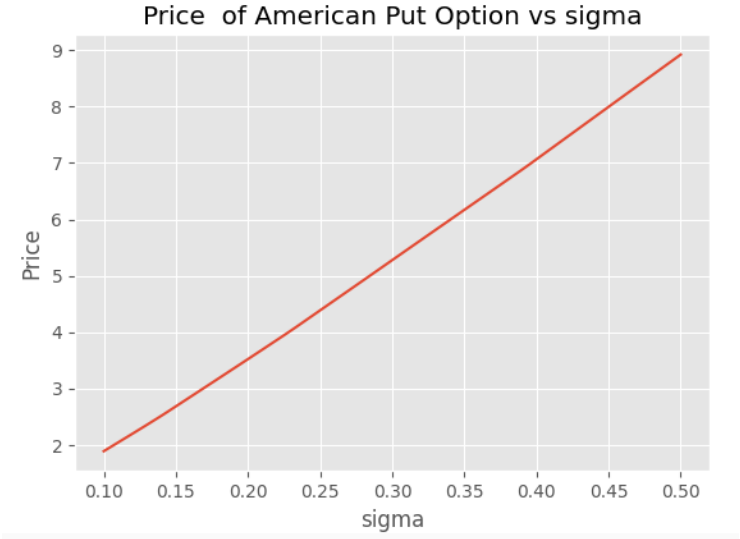


Variation with r



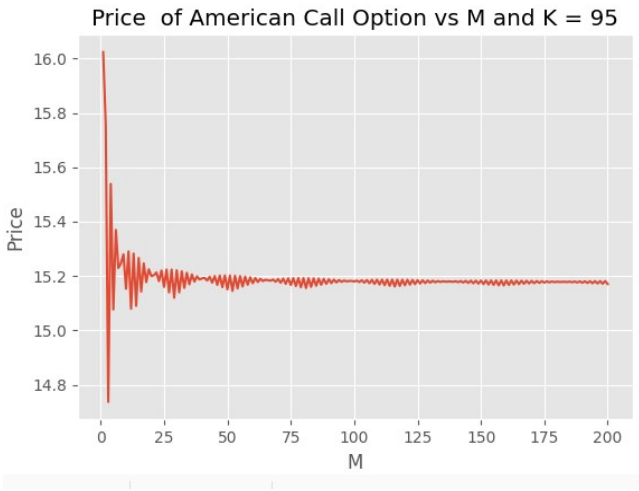
Variation with σ

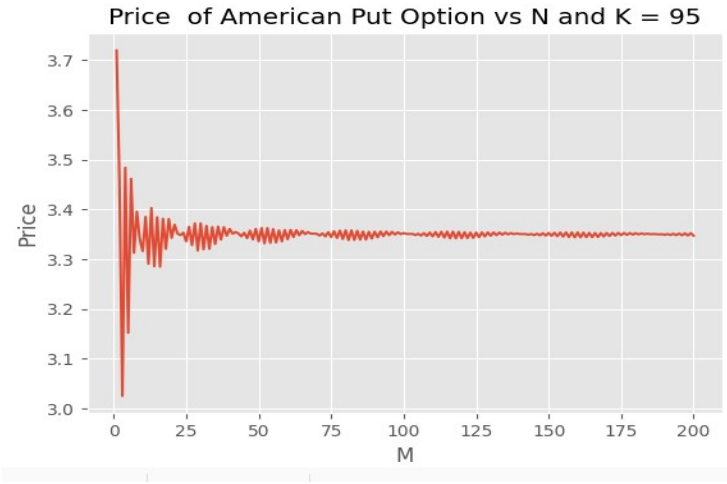




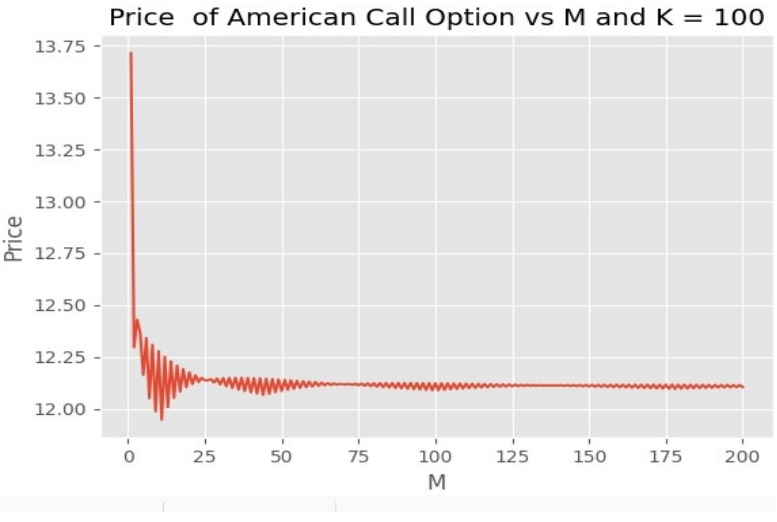
Variation with M

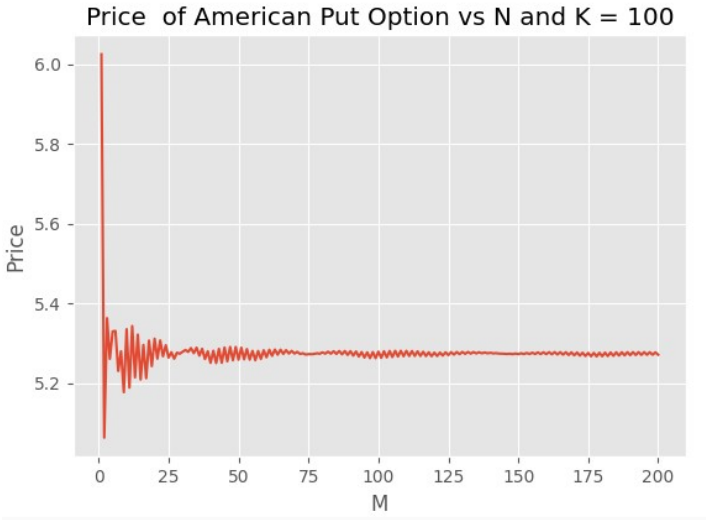
K = 95



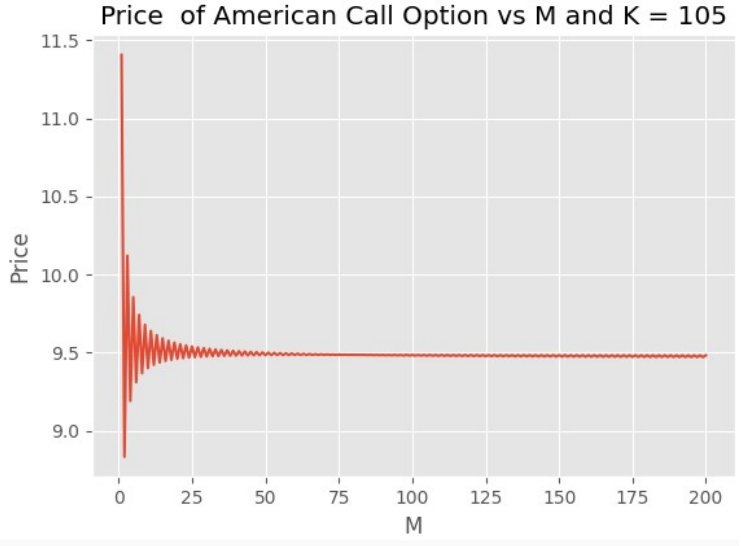


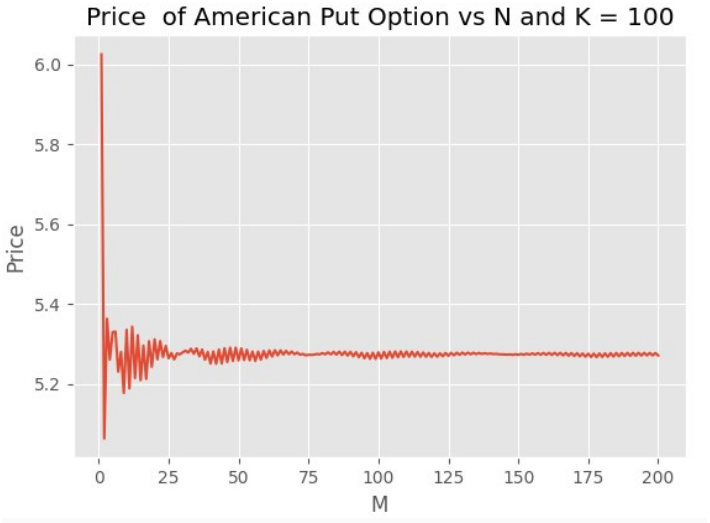
K = 100





K = 105





Question - 2

In this problem we have to calculate the price of a European Lookback option with the same S(0),r, σ as in the previous problem . Let L_i denote the price of a lookback option at step i of the binomial model. Then by using the standard binomial pricing algorithm we have:-

$$L_i = e^{-\Delta t}(\hat{p} * L_{i+1}(H)) + \hat{q} * L_{i+1}(T)$$

$$L_M = max(S(i))_{0 \le i \le m} - S(M)$$

I used a brute force algorithm by iterating through all paths and calculating the payoff at all paths then backtracking to calculate the prices and the summary of results are given below . Since we had to examine the computation time for the next problem, I took extra values of M like 26 and 27 in order to analyze the maximum value of M the algorithm can handle.

Value of M	**Lookback Option Price**	**Time of Computation (seconds)**
5	9.119298985864688	0.00025463104248046875
10	10.080582906831008	0.0033855438232421875
25	11.00349533564632	284.42060589790344
26	11.036041506227324	583.7798318862915
27	Did not terminate	xxxxxxxxxxxxxxxx
50	Did not terminate	xxxxxxxxxxxxxxxx

Note - Since this is a naive alorithm, it has a time complexity of O(2^M). So clearly calculating the option price at a value like M = 50 is not feasible.

We can observe from the table that as the value of M increases the lookback option prices increases very slowly to a value around 12

Intermediate option prices for M = 5

t = 0

9.119298985864688

t = 1

```
Option Prices at stage  1
Path =  H Price =  9.027951165547757
Path =  T Price =  9.504839866450858
```

t = 2

```
Option Prices at stage  2
Path =  HH Price =  8.548076183576446
Path =  HT Price =  9.79911875354703
Path =  TH Price =  7.1479157567747444
Path =  TT Price =  12.168664659721797
```

t = 3

```
Option Prices at stage  3
Path =  HHH Price =  7.416771005131012
Path =  HHT Price =  9.95527127295782
Path =  HTH Price =  6.201916453882752
Path =  HTT Price =  13.712862965988537
Path =  THH Price =  6.201916453882752
Path =  THT Price =  8.324614669633142
Path =  TTH Price =  7.1484182081901215
Path =  TTT Price =  17.582062714095425
```

t = 4

```
Option Prices at stage  4
Path =  HHHH Price =  5.501638813873981
Path =  HHHT Price =  9.57139153170023
Path =  HHTH Price =  4.600479677676438
Path =  HHTT Price =  15.631851880479829
Path =  HTHH Price =  4.600479677676438
Path =  HTHT Price =  8.003613780975444
Path =  HTTH Price =  6.680842999256647
Path =  HTTT Price =  21.1880893453456 52
Path =  THHH Price =  4.600479677676438
Path =  THHT Price =  8.003613780975444
Path =  THTH Price =  3.846928884415608
Path =  THTT Price =  13.07138097092879
Path =  TTHH Price =  3.846928884415608
Path =  TTHT Price =  10.680904426029972
Path =  TTTH Price =  10.680904426029972
Path =  TTTT Price =  25.05122945703703
```

t=5

```
Option Prices at stage  5
Path =  HHHHH Price =  0.0
Path =  HHHHT Price =  11.181413117784501
Path =  HHHTH Price =  0.0
Path =  HHHTT Price =  19.452691543130413
Path =  HHTHH Price =  0.0
Path =  HHTHT Price =  9.349916553291678
Path =  HHTTH Price =  6.374517470614265
Path =  HHTTT Price =  25.39456347506497
Path =  HTHHH Price =  0.0
Path =  HTHHT Price =  9.349916553291678
Path =  HTHTH Price =  0.0
Path =  HTHTT Price =  16.266373556657385
Path =  HTTHH Price =  0.0
Path =  HTTHT Price =  13.578002496522686
Path =  HTTTH Price =  13.578002496522686
Path =  HTTTT Price =  29.48259712227059
Path =  THHHH Price =  0.0
Path =  THHHT Price =  9.349916553291678
Path =  THHTH Price =  0.0
Path =  THHTT Price =  16.266373556657385
Path =  THTHH Price =  0.0
Path =  THTHT Price =  7.8184160295867144
Path =  THTTH Price =  5.330382286201839
Path =  THTTT Price =  21.234976911949744
Path =  TTHHH Price =  0.0
Path =  TTHHT Price =  7.8184160295867144
Path =  TTHTH Price =  2.9013504971397026
Path =  TTHTT Price =  18.805945122887607
Path =  TTTHH Price =  2.9013504971397026
Path =  TTTHT Price =  18.805945122887607
Path =  TTTTH Price =  18.805945122887607
Path =  TTTTT Price =  32.10539403853048
```

Question - 3

In this problem we have to find the price of the previous lookback option using a computationally efficient algorithm (Markov based) . The formuale used to calculate the prices using the Markov based algorithm is given below:-

$$L_i(S_i, S_{mi}) = e^{-\Delta t}(\hat{p} * L_{i+1}(uS_i, S_{mi} \vee uS_i) + \hat{q} * L_{i+1}(dS_i, S_{mi}))$$

Here $\vee$ denotes the max operation and S_{mi} denotes the maximum value of stock upto the state i. I implemented the above algorithm by a recursive procedure and used a hash table to store the price at each state as a cache in order to reduce the time complexity to going into exponential. The summary of the results are tabulated below:-

Value of M	Lookback Option Price	Time of Computation (seconds)
5	9.11929898586469	8.749961853027344e-05
10	10.08058290683101	0.0033855438232421875
25	11.00349533564632	0.03838396072387695
50	11.510862222177268	1.9405438899993896
75	11.745911763985793	51.930853605270386

Intermediate Option Prices

I printed the option prices at different time points in (S, S_m) format. Since the option price depend on these two values only at a particular time.

```
For t =  0
Stock Price      Max Price        Option Price
100.0000000000    100.0000000000       9.1192990

For t =  1
Stock Price      Max Price        Option Price
110.6766519994    110.6766519994       9.0279512
92.5480035208     100.0000000000       9.5048399

For t =  2
Stock Price      Max Price        Option Price
122.4932129779    122.4932129779       8.5480762
102.4290317891    110.6766519994       9.7991188
102.4290317891    102.4290317891       7.1479158
85.6513295568     100.0000000000      12.1686647

For t =  3
Stock Price      Max Price        Option Price
135.5713870504    135.5713870504       7.4167710
113.3650230595    122.4932129779       9.9552713
113.3650230595    113.3650230595       6.2019165
94.7960239464     110.6766519994      13.7128630
113.3650230595    113.3650230595       6.2019165
94.7960239464     102.4290317891       8.3246147
94.7960239464     100.0000000000       7.1484182
79.2685954938     100.0000000000      17.5820627

For t =  4
Stock Price      Max Price        Option Price
150.0458722566    150.0458722566       5.5016388
125.4686120606    135.5713870504       9.5713915
125.4686120606    125.4686120606       4.6004797
104.9170655324    122.4932129779      15.6318519
104.9170655324    113.3650230595       8.0036138
104.9170655324    110.6766519994       6.6808430
87.7318275795     110.6766519994      21.1880893
125.4686120606    125.4686120606       4.6004797
104.9170655324    113.3650230595       8.0036138
104.9170655324    104.9170655324       3.8469289
87.7318275795     102.4290317891      13.0713810
87.7318275795     100.0000000000      10.6809044
73.3615025485     100.0000000000      25.0512295
```

```
For t =  5
Stock Price     Max Price       Option Price
166.0657478768    166.0657478768      0.0000000
138.8644591388    150.0458722566      11.1814131
138.8644591388    138.8644591388      0.0000000
116.1186955073    135.5713870504      19.4526915
116.1186955073    125.4686120606      9.3499166
116.1186955073    122.4932129779      6.3745175
97.0986495029    122.4932129779      25.3945635
116.1186955073    116.1186955073      0.0000000
97.0986495029    113.3650230595      16.2663736
97.0986495029    110.6766519994      13.5780025
81.1940548771    110.6766519994      29.4825971
116.1186955073    125.4686120606      9.3499166
116.1186955073    116.1186955073      0.0000000
97.0986495029    113.3650230595      16.2663736
116.1186955073    116.1186955073      0.0000000
97.0986495029    104.9170655324      7.8184160
97.0986495029    102.4290317891      5.3303823
81.1940548771    102.4290317891      21.2349769
97.0986495029    100.0000000000      2.9013505
81.1940548771    100.0000000000      18.8059451
67.8946059615    100.0000000000      32.1053940
```

Comparing the two algorithms

Below is the table comparing the run time of the two algorithms for different values of M(**Note that all time values are in seconds**):-

Value of M	Naive Binomial Algorithm	Efficient Algorithm(Markov Based)
5	0.00025463104248046875	8.749961853027344e-05
10	0.0033855438232421875	0.0033855438232421875
25	284.42060589790344	0.03838396072387695
26	583.7798318862915	0.04182553291320801
50	Computationally Infeasible	1.9405438899993896
75	Computationally Infeasible	51.930853605270386
100	Computationally Infeasible	System crashed due to less memory

Time complexity - For the naive algorithm the time complexiy is O(2^M) whereas for the Markov based algorithm it is O(M^3)

Max Value of M - The naive algorithm requires exponential memory and time so it is impractical to use for high values like 50. We can use it to get results upto values

like 25 or 26. The time almost doubled on increasing M from 25 to 26 showing the limitations of the algorithm. The markov based algorithm has cubic memory and time constraints. It ran fine upto 75 however the pc crashed on M = 100 due to insufficient RAM.

Question - 4

I used a similar recursive algorithm like the 2nd problem for the inefficient algorithm which is present in ***q4_inefficient.py***. The efficient Markov based code is similar to the inefficent one just that we have cached the different states in order to reduce the compuation time . **Note that the parameters used to calculate the European Call Prices are same as in the first question of this assignment .**

Below is the comparison of computation time of the two algorithms :-(**Times are in seconds**)

Value of M	Call Option Price	TIme using Inefficient algorithm	Time using Markov algorithm
5	12.163185946764589	8.082389831542969e-05	3.337860107421875e-05
10	12.277327819222997	0.0010025501251220703	0.00012087821960449219
25	12.136745963232949	15.301544666290283	0.0012311935424804688
50	12.08536151007218	Computationally infeasible	0.009396791458129883
100	12.123047074012444	Computationally infeasible	0.0842740535736084
500	12.10864990170941	Computationally infeasible	16.866027116775513
1000	XXXXXXXXXXX	Computationally infeasible	Recursion depth exceeded

Time complexity - For the naive algorithm the time complexiy is O(2^M) whereas for the Markov based algorithm it is O(M^2)

Max Value of M - The naive algorithm requires exponential memory and time so it is impractical to use for high values like 50. We can use it to get results upto values like 25 or 26. The time almost doubled on increasing M from 25 to 26 showing the limitations of the algorithm. The markov based algorithm has quaadratic memory and time constraints. It ran fine upto values like M = 500 but the system gave ***recursion depth exceeded*** on M = 1000. However we can write an iterative version of the Markov algorithm(as done in 1st and 2nd assignments of this course) in order to obtain oprices till a value of M around M = 5000.