Ma374-LAB 03

Name: Harsh Yadav **Roll. No.**: 180123015 **Dept.**: Mathematics and Computing

Submission Date: 04-02-2021

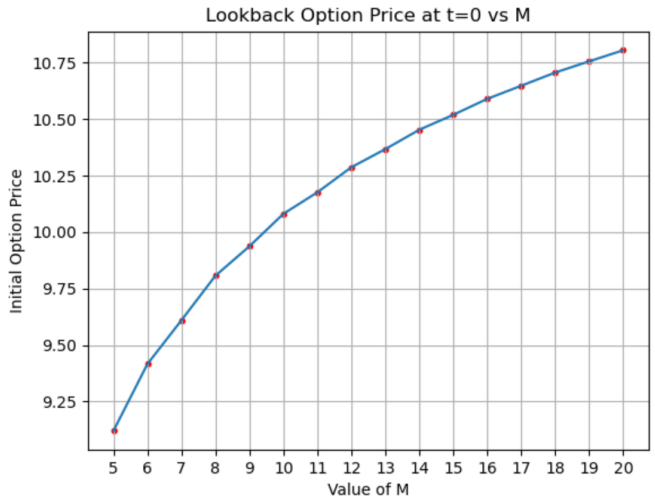
Question 1: S (0) = 100, T = 1, r = 8%, σ = 20%

a) The standard solution for finding the lookback option involves traversing over all known paths, hence leading to exponential time complexities (2^M). Python was able to calculate the initial lookback option prices only when M $\leq$ 20.

The values of initial lookback option prices are as follows:

M	Option Price	Computational Time
5	9.119298985864685	0.0015881061553955078
10	10.080582906831074	0.041570425033569336
15	10.519164595672047	0.7619645595550537
20	10.805118587177477	22.073245525360107

b) The initial lookback option prices were calculated for the values of M between the range 5 to 20. A graph of Option Price vs M was plotted out**. It can be seen that with increasing value of M, the option price also increasing.**



c) For each value of t, all possible case paths till time t were taken into consideration.
M=5

Time(t)	Possible Lookback option prices
0	9.119299
0.2	9.504840 9.027951
0.4	12.168665 9.799119 7.147916 8.548076
0.6	17.582063 13.712863 8.324615

	9.955271 7.148418 6.201916 6.201916 7.416771
0.8	25.051229 21.188089 13.071381 15.631852 10.680904 8.003614 8.003614 9.571392 10.680904 6.680843 3.846929 4.600480 3.846929 4.600480 4.600480 5.501639
1	32.105394 29.482597 21.234977 25.394563 18.805945 16.266374 16.266374 19.452692 18.805945 13.578002 7.818416 9.349917 7.818416 9.349917 9.349917 11.181413 18.805945 13.578002 5.330382 6.374517 2.901350 0.000000 0.000000 0.000000 2.901350 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

Question 2:

Markov-based binomial algorithm was used to calculate the Lookback Option Prices for various paths for this question. A recursive algorithm of dynamic programming also known as memoization was used with the help of container dictionary, which keeps track of the maximum Stock Price till now, and the current Stock Price as different paths are being explored. The time complexity of the program was reduced form O(2^m) to Polynomial Time Complexity.

M	Option Price	Computational Time
5	9.119298985864685	0.0009987354278564453
10	10.080582906831074	0.007241010665893555
25	11.003495335646338	0.08406758308410645
50	11.510862222177268	4.715372800827026

Comparision between markov based algorithm and algorithm used in Q1:

1. **Time Complexity:**
   Standard algorithm by exploring every path: Exponential time (O(2^m))
   Markov Based algorithm: Polynomial time

2. **Values of M algorithm can handle:**
   Standard algorithm by exploring every path: $1 \leq M \leq 20$
   Markov Based algorithm: $1 \leq M \leq 50$ (Maybe more than that but checked till 50 with efficient computational time.)

3. **Computational time:**
   Standard algorithm by exploring every path:

```
C:\Users\harshy\Desktop>python lab3.py
q1(a)
m = 5 ,lookback option price = 9.119298985864685
for m= 5, Computational time =  0.0015881061553955078
m = 10 ,lookback option price = 10.080582906831074
for m= 10, Computational time =  0.041570425033569336
m = 15 ,lookback option price = 10.519164595672047
for m= 15, Computational time =  0.7619645595550537
m = 20 ,lookback option price = 10.805118587177477
for m= 20, Computational time =  22.073245525360107
```

   Markov Based algorithm:

```
q2
m = 5 ,lookback option price = 9.11929898586469
for m= 5, Computational time for markov method =  0.0009987354278564453
m = 10 ,lookback option price = 10.08058290683101
for m= 10, Computational time for markov method =  0.007241010665893555
m = 25 ,lookback option price = 11.003495335646338
for m= 25, Computational time for markov method =  0.08406758308410645
m = 50 ,lookback option price = 11.510862222177268
for m= 50, Computational time for markov method =  4.715372800827026
```

4. **Differentiating Space Complexity:**
   Standard algorithm by exploring every path: Constant Space
   Markov Based algorithm: Polynomial time: Extra memory O(N^2) where N=max(S(i), $0 \leq i \leq M$)

Question 3:

In order to calculate the Initial European Call Option Price, two different Algorithms were used. First was the simple recursive algorithm by exploring every path having a time complexity of O(2^m). The second algorithm is a Markov based efficient binomial algorithm involving a recursive dynamic programming approach also called memoization. In this approach, a separate data structure (dictionary in python) was used to memorize already calculated answers to avoid recessive function calls. This algorithm has a time complexity of O(m^2).

M	Option Price (Exponential algo.)	Option Price (Markov based efficient algo.)	Computational time (Exponential algo.)	Computational time (Markov based efficient algo.)
5	12.163185946764589	12.163185946764589	0.0009250640869	0.001002073287
10	12.277327819222997	12.277327819222997	0.0118439197540	0.000996351242
15	12.052004991882896	12.052004991882896	0.0591905117034	0.001413345336
20	12.174708498955344	12.174708498955344	1.3586821556091	0.015305757522
25	-	12.136745963232963	-	0.041933059692
50	-	12.08536151007218	-	0.059981107711

Comparision on other factors:

1. **Time Complexity:**
   Simple Recursive algorithm by exploring every path: Exponential time (O(2^m))
   Markov Based algorithm: O(m^2).

2. **Values of M algorithm can handle:**
   Simple Recursive algorithm by exploring every path: $1 \leq M \leq 20$
   Markov Based algorithm: $1 \leq M \leq 1000$

3. **Differentiating Space Complexity:**
   Simple Recursive algorithm by exploring every path: Constant Space
   Markov Based algorithm: Polynomial time: Extra memory O (M^2)

4. **Computational time:** tabulated above