

MA 374: Financial Engineering Lab

Lab 03



Jwalit Devalia (200123026)

# Question 1.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

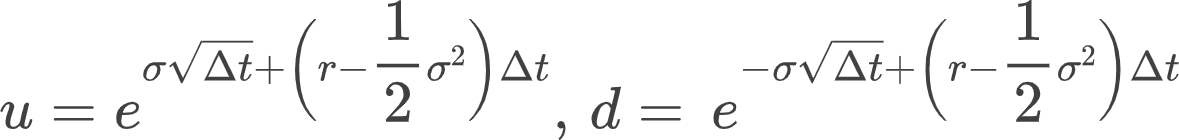
# 

# Question 2.

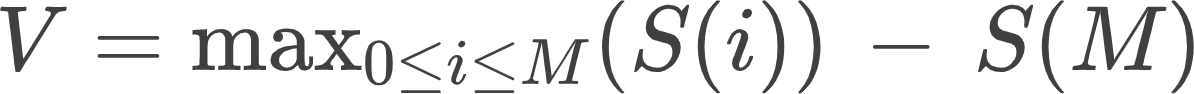
Data given to determine the initial price of a ***loopback*** (European) option using the binomial algorithm are:



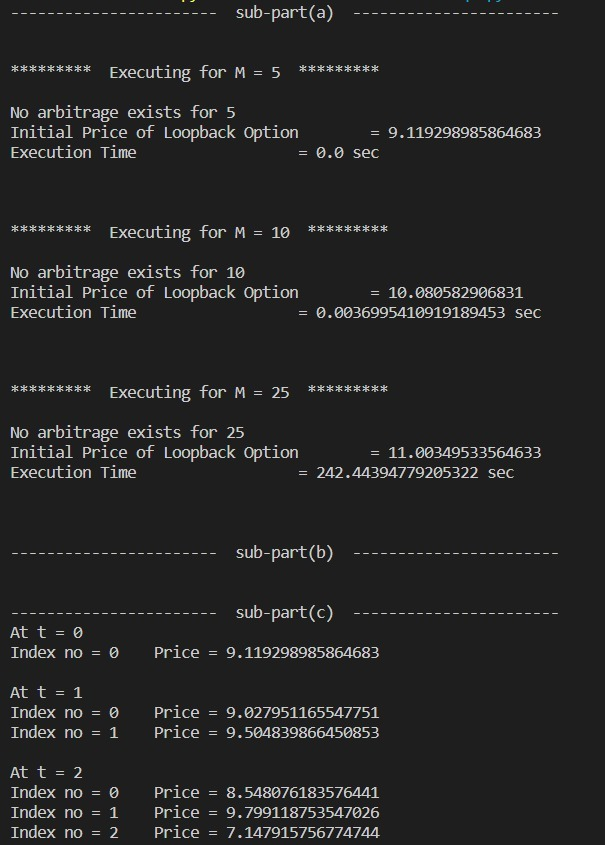
Also given u and d for this question:

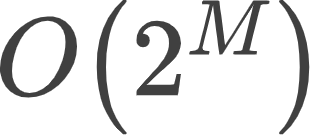


The payoff for the ***loopback*** option is given by:



1. Using the basic binomial algorithm, we obtain the initial option price for different values of M as follows:



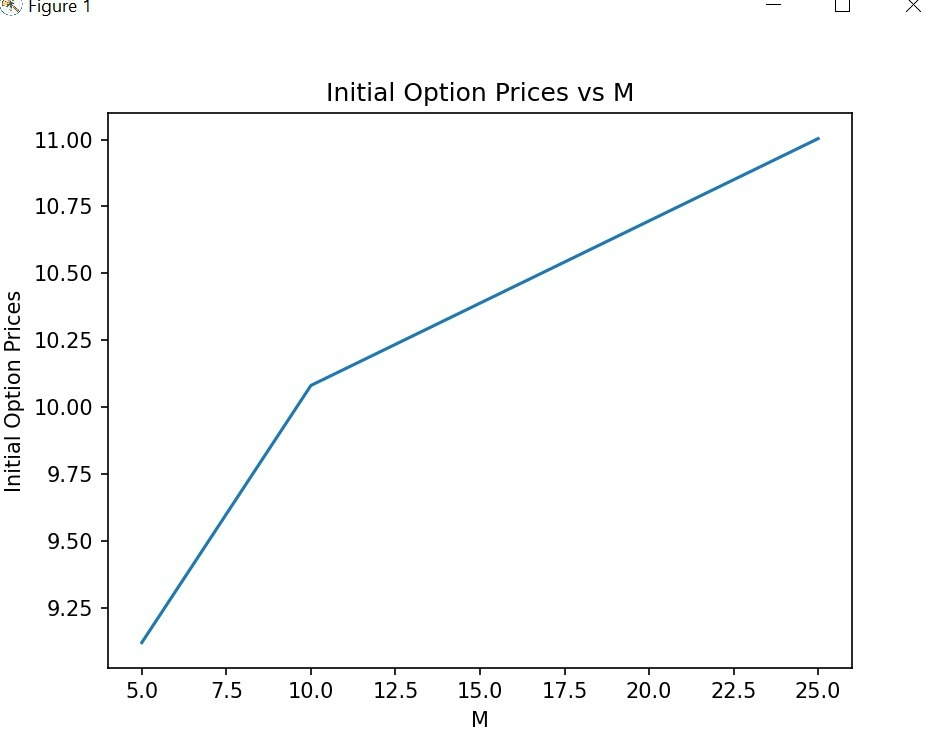
For  **M = 50**, the basic binomial model will scale in time complexity as it works in . And thus we can’t computationally handle this in python. An appropriate message is shown in the terminal to the user:

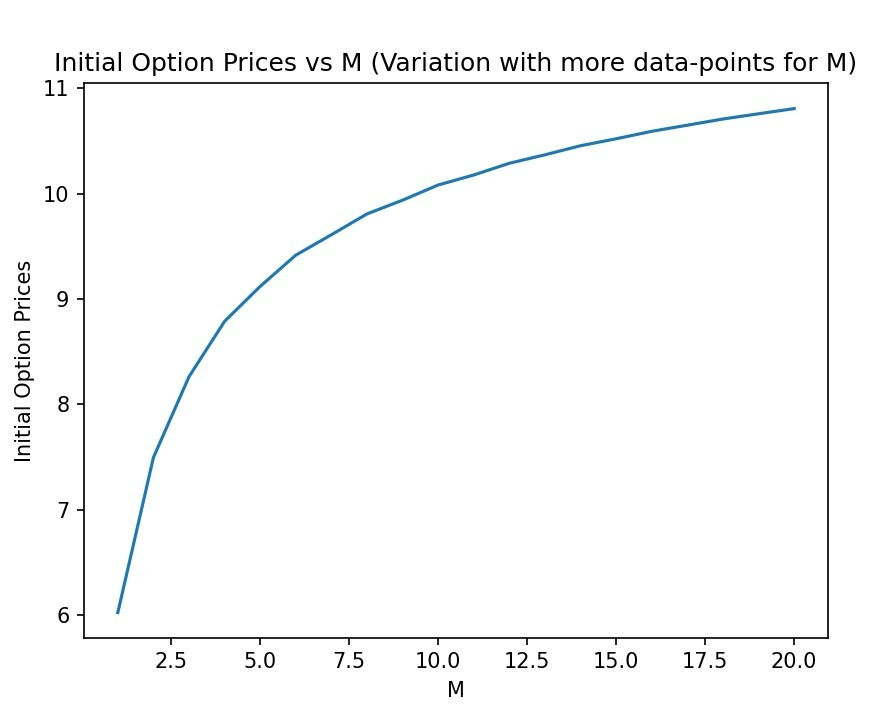
Due to complexity constraints, the initial value of the loopback option price cannot be calculated using the basic binomial algorithm for the case M = 25 and 50

## This issue will be addressed using a Markov based, computationally efficient binomial algorithm, in question 2.

1. The following conclusions can be drawn from the comparison of initial loopback option prices:

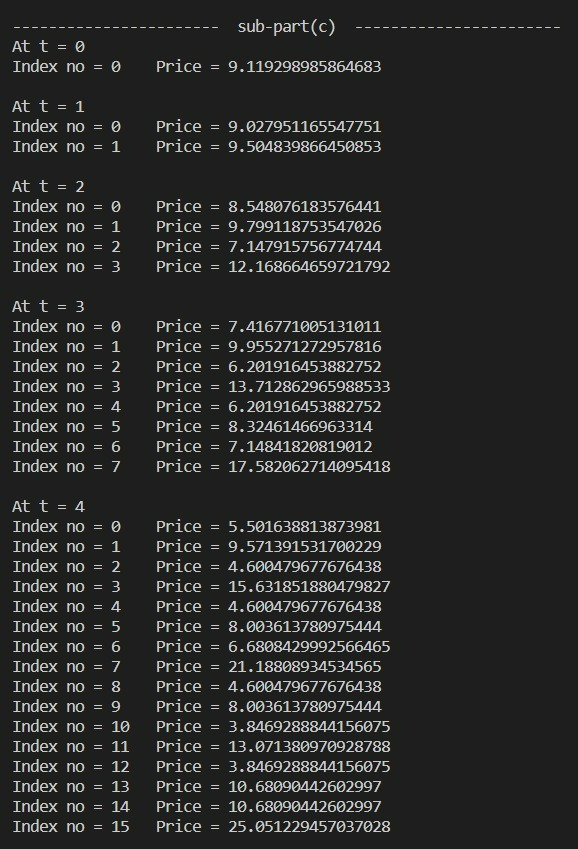
* From the graph below, ***it is seen that the initial values for the loopback option tend to converge***.

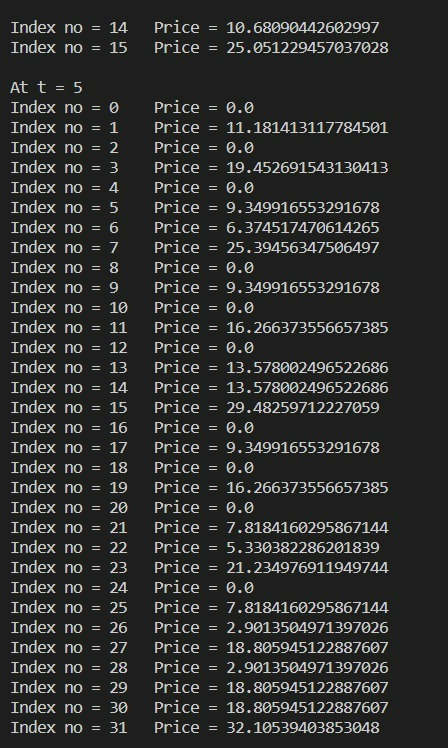




* Also, for the initial values of M (the values have been observed as far as 15), ***an increasing pattern of the initial option value with M is observed***.

1. The option values at all intermediate time points for M = 5 are shown in the table below:

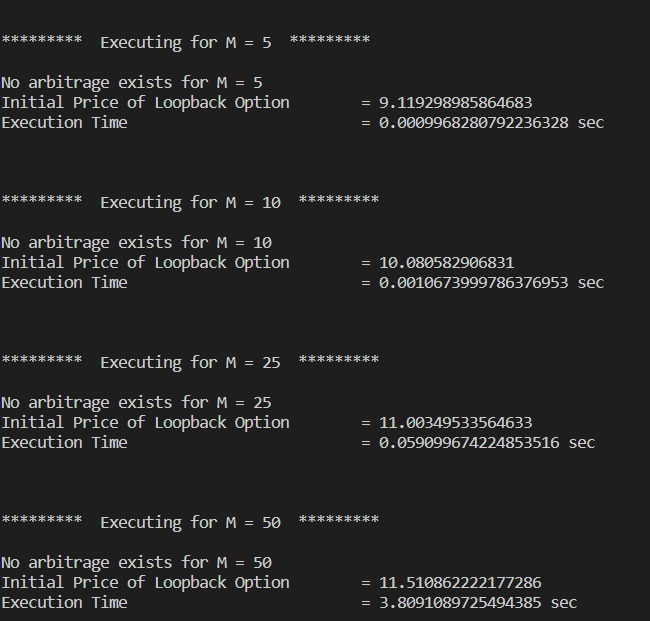


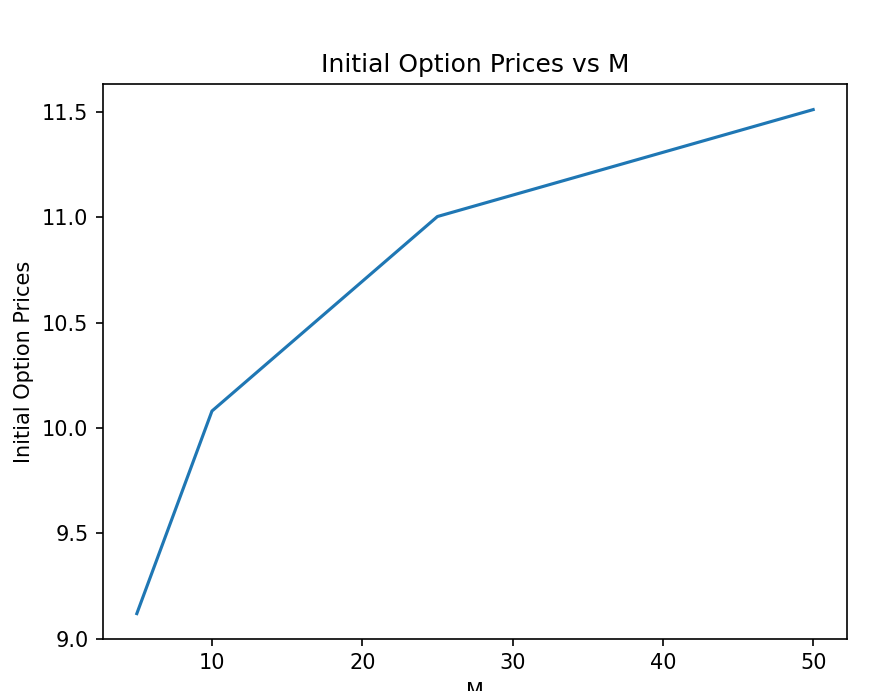


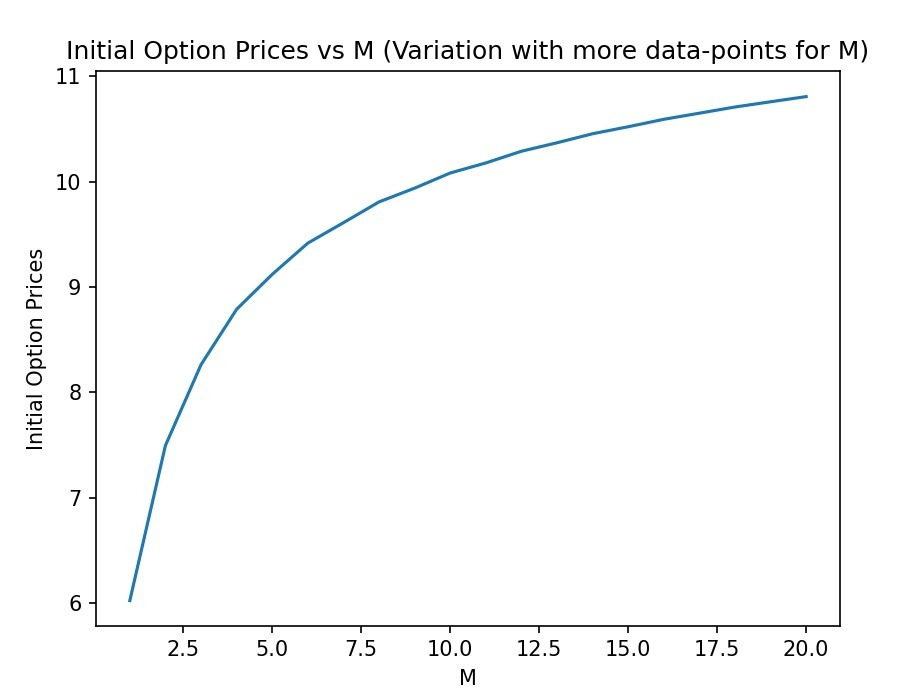
**Question 3**

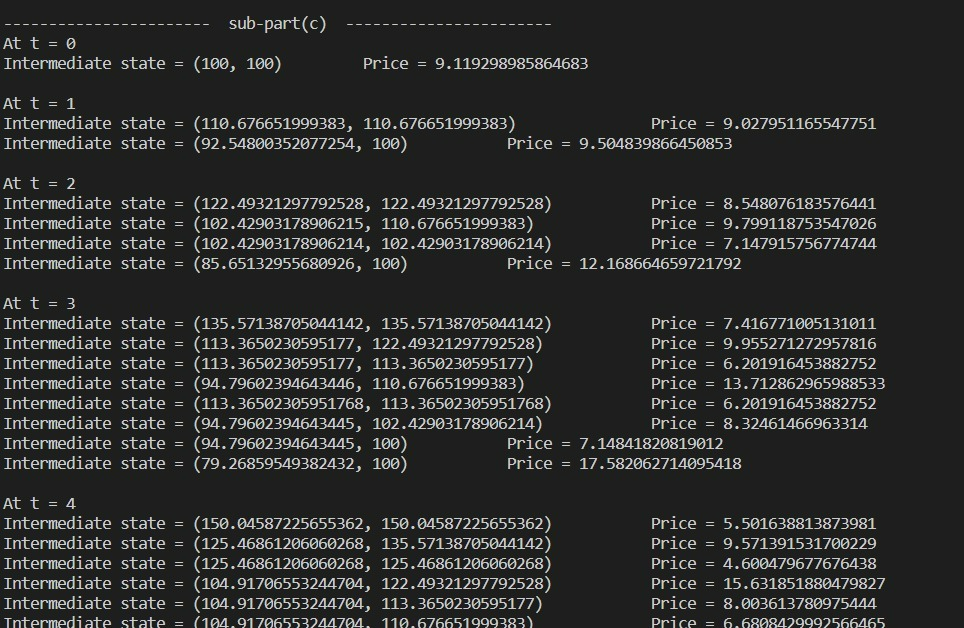
Problem 1 is repeated using a Markov based computationally efficient algorithm. **In this case, we make use of dynamic programming, and we use a map (in C++) or a dictionary (in python), to store the payoffs and keep a track of the max Stock price, even as we explore all the paths in our binomial model using a recursive function.**

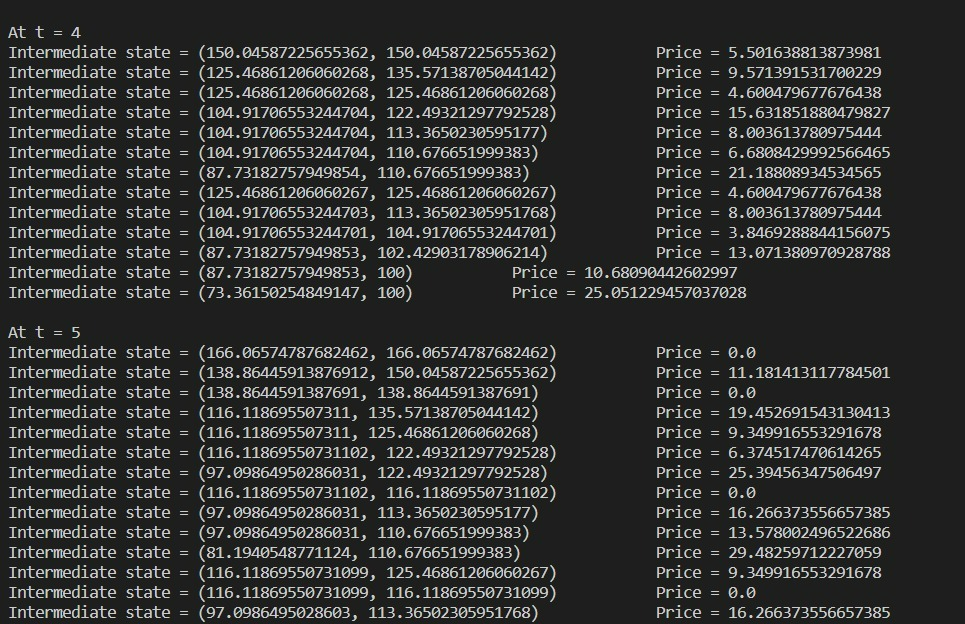
**a)**

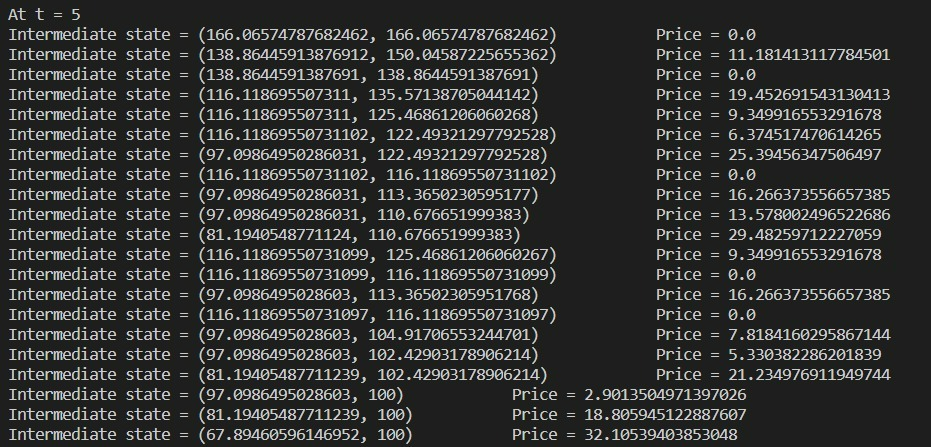


****



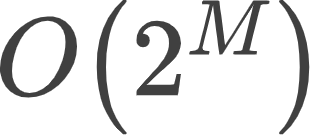






The 2 algorithms, i.e, Basic Binomial and Efficient Binomial (Markov Based) can be compared as follows:

## Time complexity:

* + The basic binomial algorithm has a time complexity of the order 
  + The Markov based has **polynomial** time complexity.

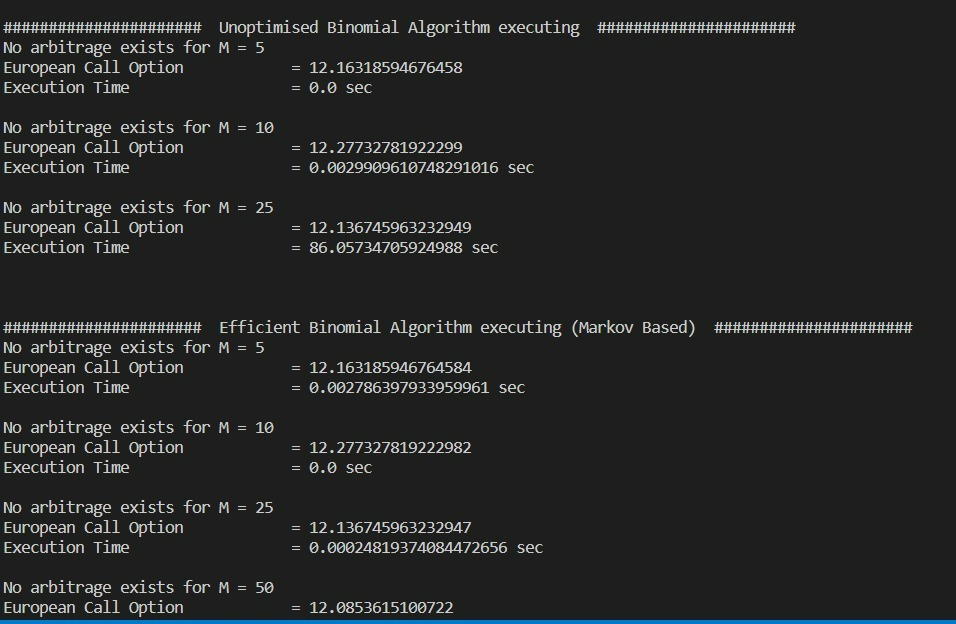
## Permissible values of M:

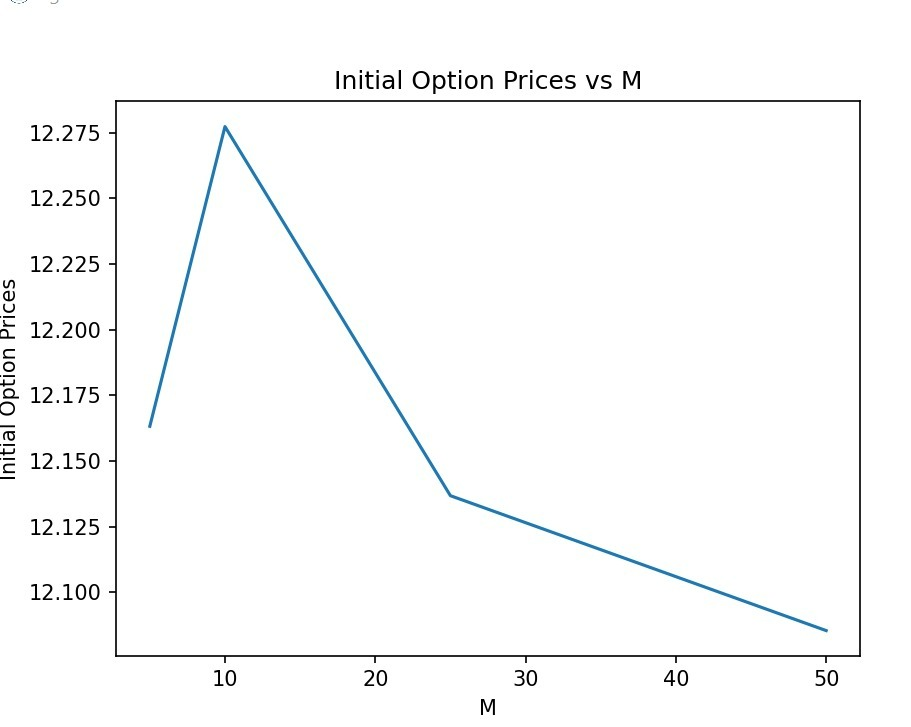
* + The basic binomial algorithm can take values of **M up to 20** (**or up to 25 in C++)**, after which it becomes computationally inefficient for calculations.
  + The Markov based algorithm can handle **M values up to 50** or more, because of its computational efficiency.

## Computational time (measured for M = 15 in python):

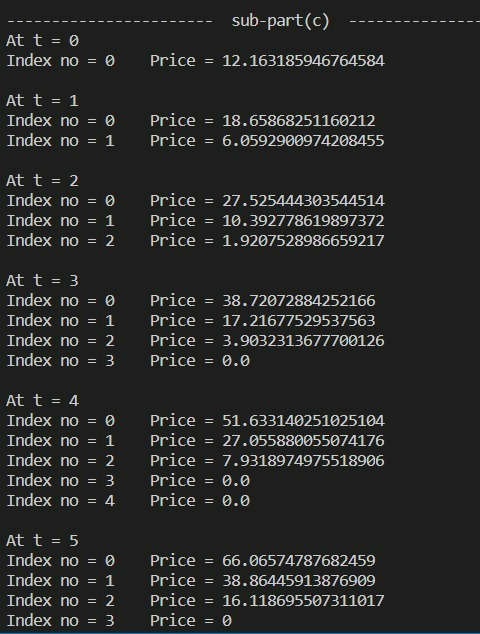
* + The basic binomial algorithm takes around **0.7-0.8 seconds** to run once, given all the input values with M took 15.
  + The Markov based algorithm takes around **0.004 seconds** to run once given all the input values and M took 15.

**Question 4**









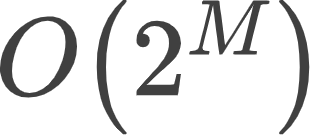
Similar to problems 1 and 2 we compute the initial option price of the European Call Option with 2 different algorithms - Basic Binomial and Markov-based efficient algorithm.

## In the case of Basic Binomial, we use recursion to take explore all possible paths for the Stock price using the Binomial model.

* **In the case of the Markov-based algorithm, we use dynamic programming as in the case of Question 2 and make changes only to the payoff and the key for the map/dictionary which now is {n, count of ups}.**

The 2 algorithms, i.e, Basic Binomial and Efficient Binomial (Markov Based) can be compared as follows:

## Time complexity:

* + The basic binomial algorithm has a time complexity of the order .
  + The Markov based has a time complexity of the order .

## Permissible values of M:

* + The basic binomial algorithm can take values of **M up to 20 (or up to 25 in C++)**, after which it becomes computationally inefficient for calculations.
  + The Markov based algorithm can handle **M values up to 1000 in C++ (or up to 500 in python after which the max recursion depth is exceeded)**, because of its computational efficiency.

## Computational time (measured for M = 15 in python):

* + The Basic Binomial algorithm takes around **0.85 seconds** to run once given all the input values and M took 15.
  + The Markov based algorithm takes around **0.20 seconds** to run once given all the input values and M took 15.