(1)CRR Binomial Tree & Bonus1: Option_Average_CRR.py

```
def simulate_stock_price(StInit, u, d,layers):
class Tree_Node:...
def average_CRR(StAve, StInit, K, time_elapsed, time_left_to_maturity, r, q, sigma,
M, layers_prev, layers, type, log_arrayed):...


# main
StInit = 50
StAve = 50
K = 50
r = 0.1
q = 0.05
sigma = 0.8
time_left_to_maturity = 0.25
M = 100
layers_prev = 100
layers = 100
```

直接調整 **#main** 下面的變數，呼叫函式存入變數，再印出即可(所有參數以及函式都已預先輸入好，直接執行即可以看到精美的結果)。

輸出看起來會是這樣 ：

```
============================================================
[ Save,t = 50 | time elapsed = 0 ]
[ log arrayed = False ]
------------------------------------------------------------
(CRR Binomial Tree) Price of European Average Call : 4.7354

============================================================
[ Save,t = 50 | time elapsed = 0 ]
[ log arrayed = True ]
------------------------------------------------------------
(CRR Binomial Tree) Price of European Average Call : 4.6949
...
```

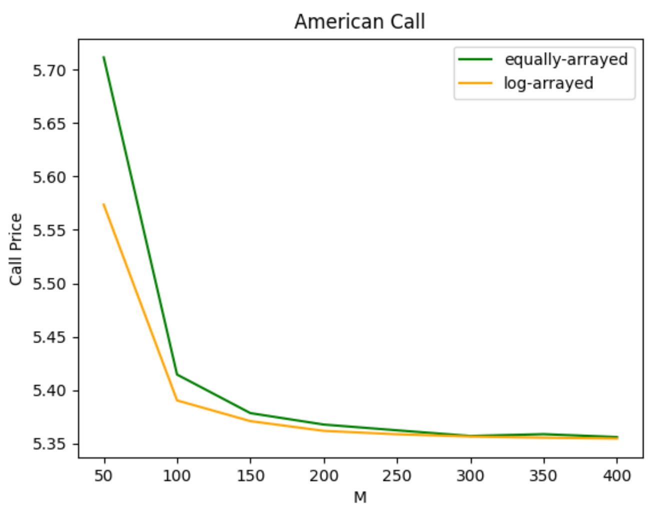
此外，下面還有使用 **multiprocessing** 套件執行的程式碼 （我已將其註解掉），可以用，但因為每個程序執行速度不同，輸出的結果跟原本的順序不同，會亂掉。所以想順便在此請問老師/助教如何解決這個問題。

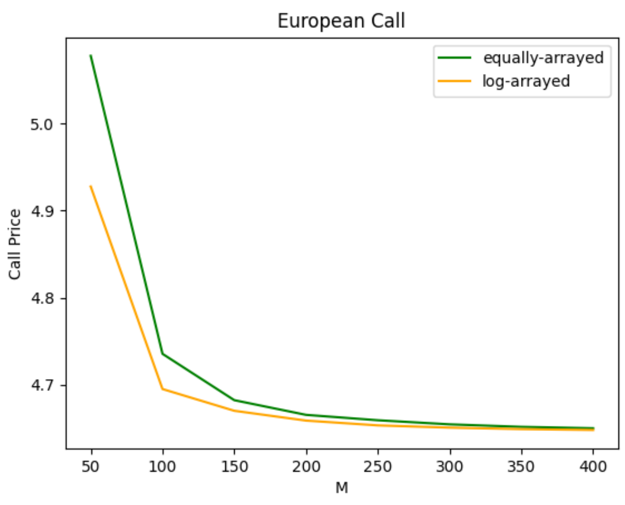
然後是 Bonus1 的表與圖表(使用 python 的 **matplotlib** 套件繪製)：
參數設定：

```
StInit = 50
StAve = 50
K = 50
r = 0.1
q = 0.05
sigma = 0.8
time_elapsed = 0
time_left_to_maturity = 0.25
layers_prev = 100
```

```
layers = 100


M = [50, 100, 150, 200, 250, 300, 350, 400]
# European Option
equal_EU = [5.0781, 4.7354, 4.6821, 4.6652, 4.659, 4.6543, 4.6514, 4.6498]
log_EU = [4.9278, 4.6949, 4.6699, 4.6586, 4.6531, 4.6505, 4.6488, 4.6478]
# American Option
equal_US = [5.7115, 5.4146, 5.3785, 5.3678, 5.3624, 5.3571, 5.3588, 5.3561]
log_US = [5.5737, 5.3904, 5.371, 5.3619, 5.3587, 5.3566, 5.3555, 5.3547]
```





圖表的部分，可以觀察到 `log-arrayed` 的收斂速度快一點點。

(2)Monte-Carlo: Option_Average_MonteCarlo.py

```
def average_MC(StAve, St, K, time_elapsed, time_left_to_maturity, r, q, sigma,
n_prev, n, sims, rep): ...


# main
```

```
St = 50
StAve = 50
K = 50
r = 0.1
q = 0.05
sigma = 0.8
time_left_to_maturity = 0.25
sims = 10000
rep = 20
n_prev = 100
n = 100
```

直接調整 **#main** 下面的變數，呼叫函式存入變數，再印出即可（所有參數以及函式都已預先輸入且用 **multiprocessing** 套件執行；直接執行即可以看到精美的結果）。

輸出看起來會是這樣 ：

```
============================================================
Average Option : European Call
[ Save,t = 50 | time elapsed = 0 ]
------------------------------------------------------------
平均 : 4.641001
標準誤 : 0.092334
九十五趴信賴區間 : [4.456332, 4.825669]
...
```

(3)Bonus2: Search_Algo_Comparison.py

```
def binary_search(array, x, low, high):...
def interpolation_search(array, x, low, high):...
class Tree_Node:...
def average_CRR(StAve, StInit, K, time_elapsed, time_left_to_maturity, r, q, sigma,
M, layers_prev, layers, type):...
def average_CRR_binary(StAve, StInit, K, time_elapsed, time_left_to_maturity, r, q,
sigma, M, layers_prev, layers, type):...
def average_CRR_interpolation(StAve, StInit, K, time_elapsed,
time_left_to_maturity, r, q, sigma, M, layers_prev, layers, type):...


# main
StInit = 50
StAve = 50
K = 50
r = 0.1
q = 0.05
sigma = 0.8
time_left_to_maturity = 0.25
M = 100
layers_prev = 100
layers = 100
```

直接調整 **#main** 下面的變數，呼叫函式存入變數，再印出即可（所有參數以及函式都已預先輸入且用 **multiprocessing** 套件執行；直接執行即可以看到精美的結果）。

輸出看起來會是這樣 ：

```
============================================================
{ Sequential Search }
============================================================
[ Save,t = 50 | time elapsed = 0 ]
------------------------------------------------------------
(CRR Binomial Tree) Price of European Average Call : 4.7354

============================================================
[ Save,t = 50 | time elapsed = 0.25 | previous layers = 100 ]
------------------------------------------------------------
(CRR Binomial Tree) Price of European Average Call : 2.3795

============================================================
[ Save,t = 50 | time elapsed = 0.25 | previous layers = 100 ]
------------------------------------------------------------
(CRR Binomial Tree) Price of American Average Call : 2.5079

============================================================
[ Save,t = 50 | time elapsed = 0 ]
------------------------------------------------------------
(CRR Binomial Tree) Price of American Average Call : 5.4146

Process finished in 5.4 second(s).



============================================================
{ Binary Search }
============================================================
[ Save,t = 50 | time elapsed = 0 ]
------------------------------------------------------------
(CRR Binomial Tree) Price of European Average Call : 4.7354
...
```