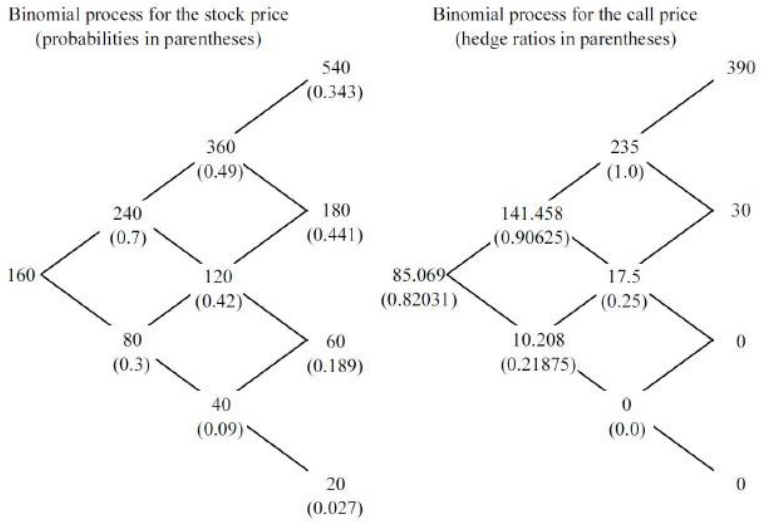
(1) Binomial Process for Stock Price



我利用 NumPy array 儲存 stock price 資訊，並參考上圖實作 Stock price 的 binomial tree：

```
Binomial tree for stock price:
 [[240. 360. 540.]
 [ 80. 120. 180.]
 [  0.  40.  60.]
 [  0.   0.  20.]]
```

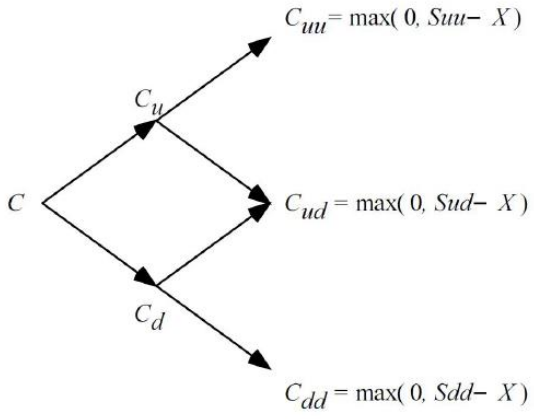
在我設計的 tree 中，資料呈現方式和上圖類似，但資料皆向上對齊，並將原始的 price 移除，只留下分枝的部分。

(2) Binomial Probability for Stock Price

我也實作了一個機率的 binomial tree，也就是上圖中的 binomial probability 的部分（標示在括上圖的括號中）

```
Binomial probability for stock price:
 [[0.7   0.49  0.343]
 [0.3   0.42  0.441]
 [0.    0.09  0.189]
 [0.    0.    0.027]]
```

(3) Binomial Process for Call Price



一樣使用 NumPy array，利用 backward induction，因此先處理 leaf 的資訊，處理完 leaf 資訊後 array 如下：

```
First start from the leaf:
 [[  0.   0. 390.]
 [  0.   0.  30.]
 [  0.   0.   0.]
 [  0.   0.   0.]]
```

接下來，利用以下公式一步一步回推得到 call price：

- The call values at time 1 can be obtained by applying the same logic:

$$C_u = \frac{pC_{uu} + (1-p)\,C_{ud}}{R},$$

$$C_d = \frac{pC_{ud} + (1-p)\,C_{dd}}{R}.$$

印出 call price 的 binomial tree 以及 final call price 結果：

```
Binomial tree for call price:
 [[141.458 235.    390.   ]
 [ 10.208  17.5    30.   ]
 [  0.      0.      0.   ]
 [  0.      0.      0.   ]]

Final Call Price:  85.06909477256853
```

(4) Binomial Process for Put Price

步驟與 call price 相同，只有公式有部分不同：

$$C = \frac{\sum_{j=0}^{n} \binom{n}{j} p^j (1-p)^{n-j} \times \max\left(0, Su^j d^{n-j} - X\right)}{R^n}.$$

$$P = \frac{\sum_{j=0}^{n} \binom{n}{j} p^j (1-p)^{n-j} \times \max\left(0, X - Su^j d^{n-j}\right)}{R^n}.$$

一樣利用 backward induction，逐步印出 tree 的資訊以及最終 put price 結果：

```
First start from the leaf:
 [[  0.   0.   0.]
 [  0.   0.   0.]
 [  0.   0.  90.]
 [  0.   0. 130.]]


Binomial tree for put price:
 [[  5.625   0.      0.   ]
 [ 34.375  22.5     0.   ]
 [ 49.583  85.     90.   ]
 [  0.      0.    130.   ]]

Final Put Price:  11.87506324480364
```