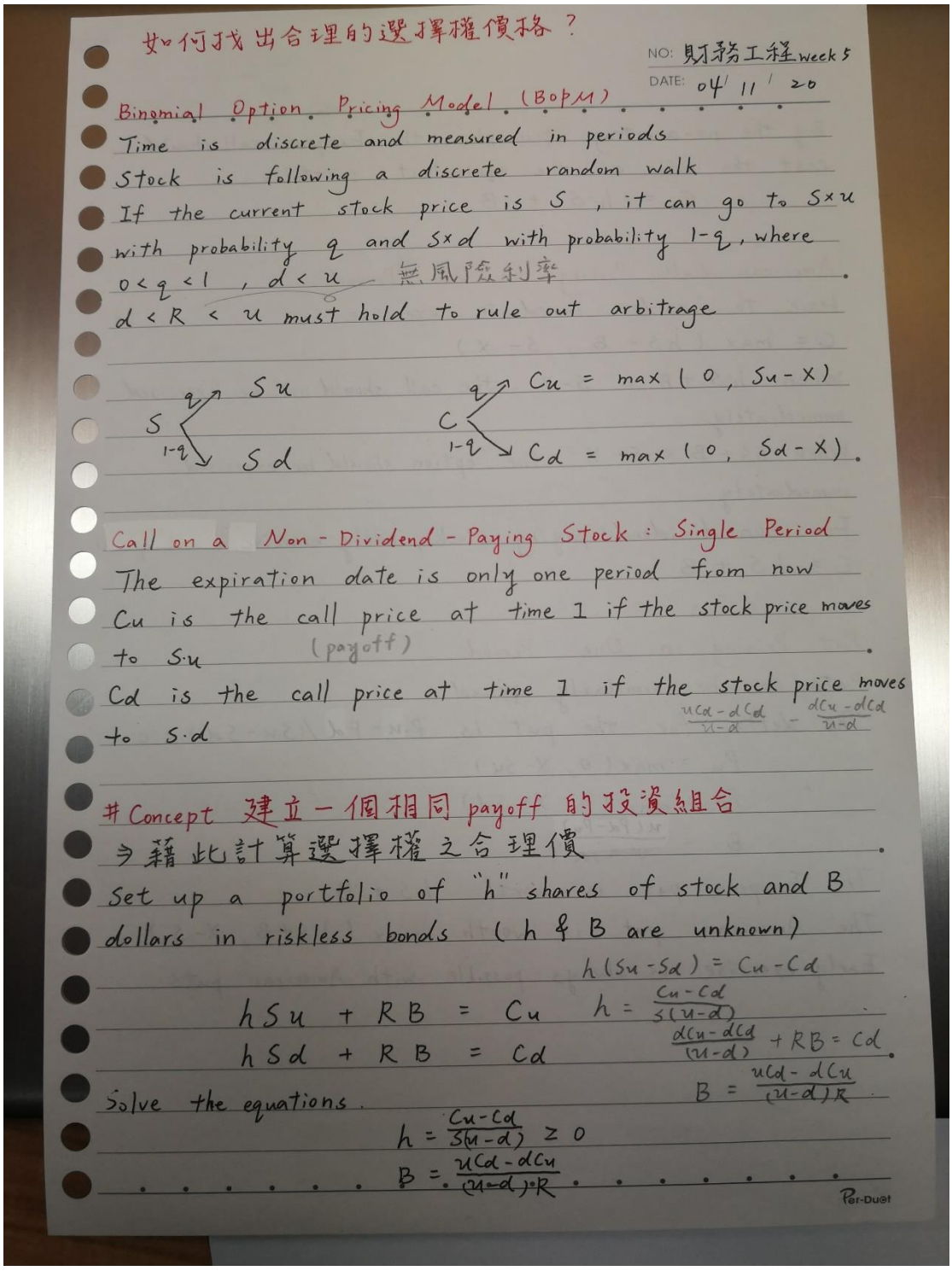
Step 1 釐清計算概念與投影片中觀念

其中 American call 為什麼不選擇兌現花了一點時間理解，無風險利率的部分也費了一點時間查資料，後來發現都是時間價值的問題



DATE: / /

By the no-arbitrage principle, the European call should cost the same as the equivalent portfolio,

$$C = hS + B$$

American Call Pricing in One Period

Have to consider immediate exercise

$C = \max(hS + B, S - X)$

When $hS + B \geq S - X$, the call should not be exercised immediately

When $hS + B < S - X$, the option should be exercised immediately.

For non-dividend-paying stocks, early exercise is not optimal

$C = hS + B$

h: delta

Put Pricing in One Period

Puts can be similarly priced.

The delta for the put is $Pu - Pd / (Su - Sd) \leq 0$

$Pu = \max(0, X - Su)$

$Pd = \max(0, X - Sd)$

$B = \frac{uPd - dPu}{(u-d)R}$

The European put is worth $hS + B$

The American put is worth $\max(hS + B, X - S)$

Early exercise is always possible with American puts.

NO: 財務工程 week5
DATE: 04 / 11 / 20

Risk

Surprisingly, the option value is independent of q.
Hence it is independent of the expected value of the stock,

$$q\,Su + (1-q)\,Sd$$

The option value depends on the sizes of price changes, u and d, which the investors must agree upon.
Then the set of possible stock prices is the same whatever q is.

Pseudo Probability

After substitution and rearrangement,

$$hS + B = \frac{\left(\frac{R-d}{u-d}\right)Cu + \left(\frac{u-R}{u-d}\right)Cd}{R}$$

$$hS + B = \frac{p\,Cu + (1-p)\,Cd}{R}$$

p 怎麼來的 ⇒ $p \times S \times u + (1-p) \times S \times d = R \times S$

$$p(u-d) + d = R$$

$$p = \frac{R-d}{u-d}$$

(p) $\triangleq \frac{R-d}{u-d}$

Risk - Neutral Probability

The expected rate of return for the stock is equal to the riskless r under p as 〈期望報酬率〉

$$p\,Su + (1-p)\,Sd = RS$$

The expected rates of return of all securities must be the riskless rate when investors are risk-neutral. For this reason, p is called the risk-neutral probability.
The value of an option is the expectation of its discounted future payoff in a risk-neutral economy. So the rate used for discounting the FV is the riskless rate in a risk-neutral economy.

Per-Duet

NO:

DATE: / /

Option on a Non-Dividend-Paying Stock: Multi Period

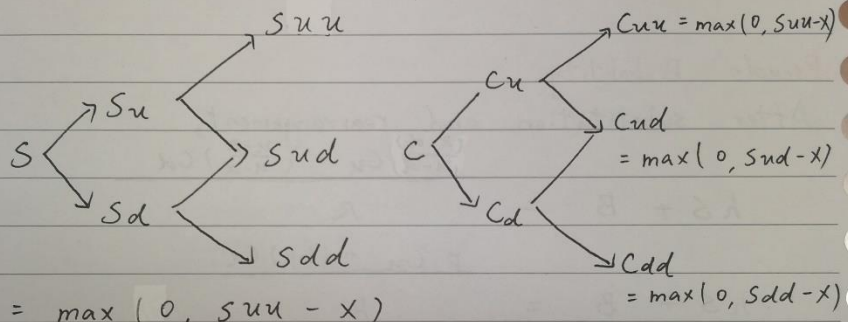
Consider a call with two periods remaining before expiration

Under the binomial model, the stock can take on three possible prices at time two: Suu, Sud, and Sdd

There are 4 paths

But the tree combines or recombines

At any note, the two stock prices only depend on the current price, not the prices of earlier times



$Cuu = \max(0, Suu - X)$

$Cud = \max(0, Sud - X)$

$Cdd = \max(0, Sdd - X)$

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

$$Cu = \frac{p\,Cuu + (1-p)\,Cud}{R}$$

$$Cd = \frac{p\,Cud + (1-p)\,Cdd}{R}$$

We now reach the current period. Compute

$$\frac{p\,Cu + (1-p)\,Cd}{R}$$

as the option price

NO: 財務工程
DATE: 04/12/20

Early Exercise

Since the call will not be exercised at time 1 even if it is American, $Cu = Su - X$ and $Cd = Sd - X$. Therefore,

$$hS + B = \frac{pCu + (1-p)Cd}{R} = \frac{[pu + (1-p)d]S - X}{R}$$

($pu + (1-p)d = R$)

$$= S - \frac{X}{R} > S - X$$

$$C = hS + B = \frac{pCu + (1-p)Cd}{R}$$

Backward Induction

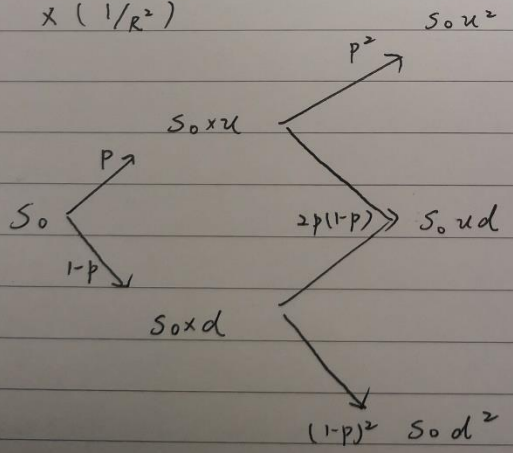
The above expression calculates C from the two successor nodes Cu and Cd and none beyond. The same computation happened at Cu and Cd, too

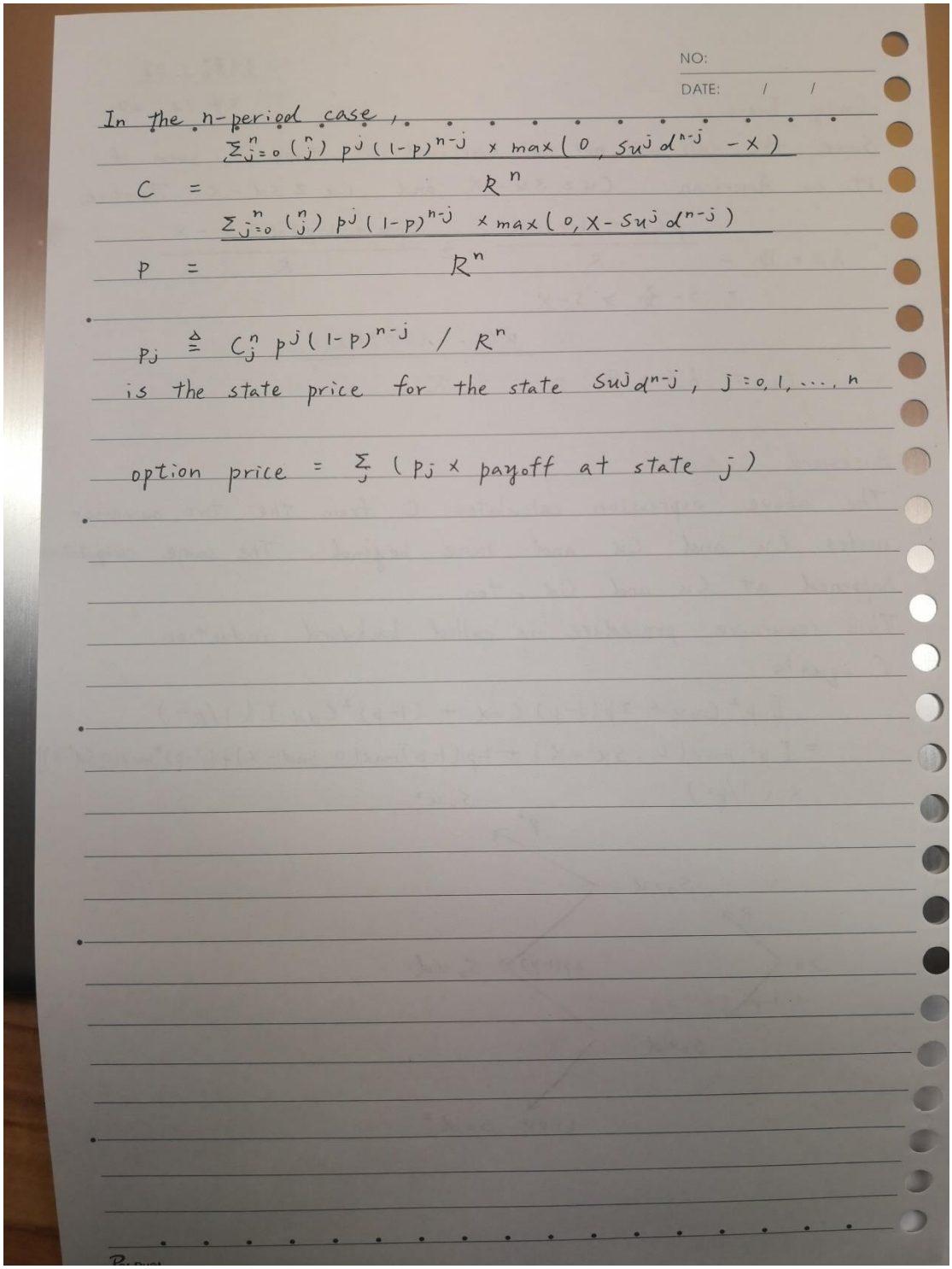
This recursive procedure is called backward induction.

C equals

$$[p^2 Cuu + 2p(1-p)Cud + (1-p)^2 Cdd](1/R^2)$$

$$= [p^2 \max(0, Su^2 - X) + 2p(1-p)\max(0, Sud - X) + (1-p)^2 \max(0, Sd^2 - X)] \times (1/R^2)$$





Step 2 程式實作

這次作業整體來說概念弄懂之後難度普通，最麻煩的是要怎麼不使用 pandas 讓出來的數值排列整齊，花了一點時間之後決定使用 format 取到小數點後第二位，

其他部分基本上就照著 PPT 做。

一開始看到作業 Assign 的以為要自己做一個巴斯塔三角形，並畫成樹狀圖，重新回去看老師影片聽起來是不用？而且坐起來有點花時間所以暫時的成品就長這樣了。

In [1]:
```
import math
#輸入基本資料
print("請輸入當前股價")
S = float(input())
print("請輸入選擇權約定價格")
X = float(input())
print("請輸入每期股票上漲幅度 ex:當期股價10元 下期股價15元 則請輸入1.5 (15/1
0)")
U = float(input())
print("請輸入每期股票下跌幅度 ex:當期股價10元 下期股價8元 則請輸入0.8 (8/10)")
D = float(input())
print("請輸入無風險利率 ex:當期資產10元 下期資產12元 則請輸入1.2")
R = float(input())
print("請輸入選擇權期數(整數)")
N = int(input())

#計算最後一級outcomes_call
a = []
S_X = []
strS_X = []
for i in range (N+1):
    a.append(S * U**(N-i) * D**(i))
    S_X.append( max(a[i]-X, 0) )
    strS_X.append( format(S_X[i], '.2f'))

print("T = " + str(N) + " ==>", end = '    ')
for i in range (N+1):
    print(strS_X[i], end = "    ")


#把計算前一級Option 寫成 function
P = (R - D) / (U - D)
Price1 = list(S_X)
count1 = N

def previous_price(P, Price, R):
    Previous_price = []
    for i in range (len(Price) - 1):
        Previous_price.append( (P * Price[i] + (1-P) * Price[i+1]) / R )

    return Previous_price

#Backward Induction
if count1 >= 0:
    for i in range(count1):
        Price1 = previous_price(P, Price1, R)
        print("")
        print("T = " + str(N-i-1) + " ==>", end = "    ")
        for j in range(count1):
            print(format(Price1[j], '.2f'), end ="    ")

        count1-=1

    print("")
    print("當期call value = " + format(Price1[0], '.2f') )
```

```
請輸入當前股價
160
請輸入選擇權約定價格
150
請輸入每期股票上漲幅度 ex:當期股價10元 下期股價15元 則請輸入1.5 (15/10)
1.5
請輸入每期股票下跌幅度 ex:當期股價10元 下期股價8元 則請輸入0.8 (8/10)
0.5
請輸入無風險利率 ex:當期資產10元 下期資產12元 則請輸入1.2
1.2
請輸入選擇權期數(整數)
3
T = 3 ==>    390.00    30.00    0.00    0.00
T = 2 ==>    235.00    17.50    0.00
T = 1 ==>    141.46    10.21
T = 0 ==>    85.07
當期call value = 85.07
```

In [2]:
```
#計算最後一期outcomes_call_put
X_S = []
strX_S = []

for i in range (N+1):
    X_S.append( max(X - a[i], 0) )
    strX_S.append( format(X_S[i], '.2f'))

print("T = " + str(N) + " ==>", end = '    ')
for i in range (N+1):
    print(strX_S[i], end = "     ")


#Backward Induction
Price2 = list(X_S)
count2 = N
if count2 >= 0:
    for i in range(count2):
        Price2= previous_price(P, Price2, R)
        print("")
        print("T = " + str(N-i-1) + " ==>", end = "    ")
        for j in range(count2):
            print(format(Price2[j], '.2f'), end = "     ")

        count2-=1

    print("")
    print("當期put value = " + format(Price2[0], '.2f') )
```

```
T = 3 ==>    0.00     0.00     90.00     130.00
T = 2 ==>    0.00     22.50     85.00
T = 1 ==>    5.63     34.38
T = 0 ==>    11.88
當期put value = 11.88
```

In [1]:
```
import math
#輸入基本資料
print("請輸入當前股價")
S = float(input())
print("請輸入選擇權約定價格")
X = float(input())
print("請輸入每期股票上漲幅度 ex:當期股價10元 下期股價15元 則請輸入1.5 (15/10)")
U = float(input())
print("請輸入每期股票下跌幅度 ex:當期股價10元 下期股價8元 則請輸入0.8 (8/10)")
D = float(input())
print("請輸入無風險利率 ex:當期資產10元 下期資產12元 則請輸入1.2")
R = float(input())
print("請輸入選擇權期數(整數)")
N = int(input())

#計算最後一期outcomes_call
a = []
S_X = []
strS_X = []
for i in range (N+1):
    a.append(S * U**(N-i) * D**(i))
    S_X.append( max(a[i]-X, 0) )
    strS_X.append( format(S_X[i], '.2f'))

print("T = " + str(N) + " ==>", end = '   ')
for i in range (N+1):
    print(strS_X[i], end = "    ")


#把計算前一期Option 價格寫成 function
P = (R - D) / (U - D)
Price1 = list(S_X)
count1 = N

def previous_price(P, Price, R):
    Previous_price = []
    for i in range (len(Price) - 1):
        Previous_price.append( (P * Price[i] + (1-P) * Price[i+1]) / R )

    return Previous_price

#Backward Induction
if count1 >= 0:
    for i in range(count1):
        Price1 = previous_price(P, Price1, R)
        print("")
        print("T = " + str(N-i-1) + " ==>", end = "   ")
        for j in range(count1):
            print(format(Price1[j], '.2f'), end ="    ")

        count1-=1

    print("")
    print("當期call value = " + format(Price1[0], '.2f') )
```

```
請輸入當前股價
200
請輸入選擇權約定價格
250
請輸入每期股票上漲幅度 ex:當期股價10元 下期股價15元 則請輸入1.5 (15/10)
1.7
請輸入每期股票下跌幅度 ex:當期股價10元 下期股價8元 則請輸入0.8 (8/10)
0.9
請輸入無風險利率 ex:當期資產10元 下期資產12元 則請輸入1.2
1.2
請輸入選擇權期數(整數)
5
T = 5 ==>   2589.71    1253.38    545.91    171.36    0.00    0.00
T = 4 ==>   1462.09    676.01    259.85    53.55    0.00
T = 3 ==>   808.99    346.59    109.09    16.73
T = 2 ==>   433.32    165.13    42.81
T = 1 ==>   221.42    73.90
T = 0 ==>   107.68
當期call value = 107.68
```

In [2]:
```
#計算最後一期outcomes_call_put
X_S = []
strX_S = []

for i in range (N+1):
    X_S.append( max(X - a[i], 0) )
    strX_S.append( format(X_S[i], '.2f'))

print("T = " + str(N) + " ==>", end = '   ')
for i in range (N+1):
    print(strX_S[i], end = "    ")


#Backward Induction
Price2 = list(X_S)
count2 = N
if count2 >= 0:
    for i in range(count2):
        Price2= previous_price(P, Price2, R)
        print("")
        print("T = " + str(N-i-1) + " ==>", end = "   ")
        for j in range(count2):
            print(format(Price2[j], '.2f'), end = "    ")

        count2-=1

    print("")
    print("當期put value = " + format(Price2[0], '.2f') )
```

```
T = 5 ==>   0.00    0.00    0.00    0.00    26.93    131.90
T = 4 ==>   0.00    0.00    0.00    14.02    77.11
T = 3 ==>   0.00    0.00    7.30    44.55
T = 2 ==>   0.00    3.80    25.48
T = 1 ==>   1.98    14.46
T = 0 ==>   8.15
當期put value = 8.15
```