
Algorithm 1 欧式期权二叉树模型定价

Input: S_0 初始股价, K 期权执行价, r 无风险利率, T 到期时间, N 期数, u, d 股价上涨或下跌的比率, is_put 是否看跌期权, is_am 是否美式期权, is_bar 是否障碍期权, is_down 是否向下期权, is_in 是否敲入期权, B 障碍价格

Output: $payoffs$ 期权价格

```

1: function STOCKOPTION( $S_0, K, r, T, N, u, d, is\_put, is\_am, is\_bar, is\_down, is\_in, B$ )
2:    $is\_call \leftarrow \text{not } is\_put; is\_eu \leftarrow \text{not } is\_am;$ 
3:    $is\_up \leftarrow \text{not } is\_down; is\_out \leftarrow \text{not } is\_in;$ 
4:    $dt \leftarrow T/N; df \leftarrow \exp(-r \times dt)$ 
5: end function
6:
7: function BINOMIALEUROPEANOPTION( $StockOption$ )
8:    $M \leftarrow N + 1; qu \leftarrow (\exp(r \times df) - d)/(u - d);$ 
9:    $qd \leftarrow 1 - qu; STs \leftarrow [];$ 
10:  for  $i = 0 \rightarrow M - 1$  do
11:     $STs[i] \leftarrow S_0 \times u^{N-i} \times d^i$ 
12:  end for
13:  if  $is\_call$  then
14:     $payoffs \leftarrow \max(0, STs - K)$ 
15:  else
16:     $payoffs \leftarrow \max(0, K - STs)$ 
17:  end if
18:  for  $i = 0 \rightarrow N - 1$  do
19:     $payoffs \leftarrow (payoffs[: -1] \times qu + payoffs[1 :] \times qd) \times df$ 
20:  end for
21: return  $payoffs[0]$ 
22: end function

```

Algorithm 2 美式 (含欧式) 期权二叉树模型定价

Input: S_0 初始股价, K 期权执行价, r 无风险利率, T 到期时间, N 期数, u, d 股价上涨或下跌的比率, is_put 是否看跌期权, is_am 是否美式期权, is_bar 是否障碍期权, is_down 是否向下期权, is_in 是否敲入期权, B 障碍价格

Output: $payoffs$ 期权价格

```

1: function STOCKOPTION( $S_0, K, r, T, N, u, d, is\_put, is\_am, is\_bar, is\_down, is\_in, B$ )
2:    $is\_call \leftarrow \text{not } is\_put; is\_eu \leftarrow \text{not } is\_am;$ 
3:    $is\_up \leftarrow \text{not } is\_down; is\_out \leftarrow \text{not } is\_in;$ 
4:    $dt \leftarrow T/N; df \leftarrow \exp(-r \times dt)$ 
5: end function
6:
7: function BINOMIALTREEOPTION( $StockOption$ )
8:    $qu \leftarrow (\exp(r \times df) - d)/(u - d); qd \leftarrow 1 - qu; STs \leftarrow [[S_0]]$ 
9:   for  $i = 0 \rightarrow N - 1$  do
10:     $prev\_branches \leftarrow STs[-1]$ 
11:     $st \leftarrow [prev\_branches \times u, [prev\_branches[-1] \times d]]$ 
12:     $STs.append(st)$ 
13:  end for
14:  if  $is\_call$  then
15:     $payoffs \leftarrow [\max(0, STs - K)]$ 
16:  else
17:     $payoffs \leftarrow [\max(0, K - STs)]$ 
18:  end if
19:  for  $i = N - 1 \rightarrow 0$  do
20:     $po \leftarrow (payoffs[N - i - 1][: -1] \times qu + payoffs[N - i - 1][1 :] \times qd) \times df$ 
21:     $payoffs.append(po)$ 
22:    if  $\text{not } is\_eu$  then
23:      if  $is\_call$  then
24:         $po \leftarrow \max(payoffs[N - i], STs[i] - K)$ 
25:      else
26:         $po \leftarrow \max(payoffs[N - i], K - STs[i])$ 
27:      end if
28:       $payoffs[N - i] \leftarrow po$ 
29:    end if
30:  end for
31: return  $payoffs[-1][0]$ 
32: end function

```

Algorithm 3 障碍期权二叉树模型定价

Input: S_0 初始股价, K 期权执行价, r 无风险利率, T 到期时间, N 期数, u, d 股价上涨或下跌的比率, is_put 是否看跌期权, is_am 是否美式期权, is_bar 是否障碍期权, is_down 是否向下期权, is_in 是否敲入期权, B 障碍价格

Output: $payoffs$ 期权价格

```

1: function STOCKOPTION( $S_0, K, r, T, N, u, d, is\_put, is\_am, is\_bar, is\_down, is\_in, B$ )
2:    $is\_call \leftarrow \text{not } is\_put; is\_eu \leftarrow \text{not } is\_am;$ 
3:    $is\_up \leftarrow \text{not } is\_down; is\_out \leftarrow \text{not } is\_in;$ 
4:    $dt \leftarrow T/N; df \leftarrow \exp(-r \times dt)$ 
5: end function
6:
7: function BINOMIALBARRIEROPTION( $StockOption$ )
8:    $qu \leftarrow (\exp(r \times df) - d)/(u - d); qd \leftarrow 1 - qu; STs \leftarrow [[S_0]]$ 
9:   for  $i = 0 \rightarrow N - 1$  do
10:     $prev\_branches \leftarrow STs[-1]$ 
11:     $st \leftarrow [prev\_branches \times u, [prev\_branches[-1] \times d]]$ ;  $STs.append(st)$ 
12:  end for
13:  if  $is\_call$  then  $payoffs \leftarrow [\max(0, STs - K)]$ 
14:  else  $payoffs \leftarrow [\max(0, K - STs)]$ 
15:  end if
16:  for  $i = N - 1 \rightarrow 0$  do
17:    if  $is\_bar$  then
18:      for  $tf = zip(STs[i + 1] > B, [0 : i + 2])$  do
19:        if  $tf[0]$  and  $is\_up$  and  $is\_out$  then  $payoffs[N - i - 1][tf[1]] = 0$ 
20:        else if  $!tf[0]$  and  $is\_up$  and  $!is\_out$  then  $payoffs[N - i - 1][tf[1]] = 0$ 
21:        else if  $!tf[0]$  and  $!is\_up$  and  $is\_out$  then  $payoffs[N - i - 1][tf[1]] = 0$ 
22:        else if  $tf[0]$  and  $!is\_up$  and  $!is\_out$  then  $payoffs[N - i - 1][tf[1]] = 0$ 
23:        end if
24:      end for
25:    end if
26:     $po \leftarrow (payoffs[N - i - 1][: -1] \times qu + payoffs[N - i - 1][1 :] \times qd) \times df$ ;  $payoffs.append(po)$ 
27:    if not  $is\_eu$  then
28:      if  $is\_call$  then  $po \leftarrow \max(payoffs[N - i], STs[i] - K)$ 
29:      else  $po \leftarrow \max(payoffs[N - i], K - STs[i])$ 
30:      end if
31:       $payoffs[N - i] \leftarrow po$ 
32:    end if
33:  end for
34: return  $payoffs[-1][0]$ 
35: end function

```
