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

function StockOption($S_0, K, r, T, N, u, d, is\_put, is\_am, is\_bar, is\_down, is\_in, B$)
    $is\_call \leftarrow$ not is_put; $is\_eu \leftarrow$ not is_am;
    $is\_up \leftarrow$ not is_down; $is\_out \leftarrow$ not is_in;
    $dt \leftarrow T/N$; $df \leftarrow \exp(-r \times dt)$
end function

function BinomialEuropeanOption($StockOption$)
    $M \leftarrow N + 1$; $qu \leftarrow (\exp(r \times df) - d)/(u - d)$;
    $qd \leftarrow 1 - qu$; $STs \leftarrow []$;
    for $i = 0 \to M - 1$ **do**
        $STs[i] \leftarrow S_0 \times u^{N-i} \times d^i$
    end for
    if is_call **then**
        $payoffs \leftarrow \max(0, STs - K)$
    else
        $payoffs \leftarrow \max(0, K - STs)$
    end if
    for $i = 0 \to N - 1$ **do**
        $payoffs \leftarrow (payoffs[: -1] \times qu + payoffs[1 :] \times qd) \times df$
    end for
return $payoffs[0]$
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

function StockOption($S_0, K, r, T, N, u, d, is\_put, is\_am, is\_bar, is\_down, is\_in, B$)
 $is\_call \leftarrow$ not is_put; $is\_eu \leftarrow$ not is_am;
 $is\_up \leftarrow$ not is_down; $is\_out \leftarrow$ not is_in;
 $dt \leftarrow T/N$; $df \leftarrow \exp(-r \times dt)$
end function

function BinomialTreeOption($StockOption$)
 $qu \leftarrow (\exp(r \times df) - d)/(u - d)$; $qd \leftarrow 1 - qu$; $STs \leftarrow [[S_0]]$
 for $i = 0 \rightarrow N - 1$ **do**
  $prev\_branches \leftarrow STs[-1]$
  $st \leftarrow [prev\_branches \times u, [prev\_branches[-1] \times d]]$
  STs.append(st)
 end for
 if is_call **then**
  $payoffs \leftarrow [\max(0, STs - K)]$
 else
  $payoffs \leftarrow [\max(0, K - STs)]$
 end if
 for $i = N - 1 \rightarrow 0$ **do**
  $po \leftarrow (payoffs[N-i-1][:-1] \times qu + payoffs[N-i-1][1:] \times qd) \times df$
  $payoffs$.append(po)
  if not is_eu **then**
   if is_call **then**
    $po \leftarrow \max(payoffs[N-i], STs[i] - K)$
   else
    $po \leftarrow \max(payoffs[N-i], K - STs[i])$
   end if
   $payoffs[N-i] \leftarrow po$
  end if
 end for
return $payoffs[-1][0]$
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

```
function StockOption(S_0, K, r, T, N, u, d, is_put, is_am, is_bar, is_down, is_in, B)
    is_call ← not is_put; is_eu ← not is_am;
    is_up ← not is_down; is_out ← not is_in;
    dt ← T/N; df ← exp(−r × dt)
end function

function BinomialBarrierOption(StockOption)
    qu ← (exp(r × df) − d)/(u − d); qd ← 1 − qu; STs ← [[S_0]]
    for i = 0 → N − 1 do
        prev_branches ← STs[−1]
        st ← [prev_branches × u, [prev_branches[−1] × d]]; STs.append(st)
    end for
    if is_call then payoffs ← [max(0, STs − K)]
    else payoffs ← [max(0, K − STs)]
    end if
    for i = N − 1 → 0 do
        if is_bar then
            for tf = zip(STs[i + 1] > B, [0 : i + 2]) do
                if tf[0] and is_up and is_out then payoffs[N − i − 1][tf[1]] = 0
                else if !tf[0] and is_up and !is_out then payoffs[N − i − 1][tf[1]] = 0
                else if !tf[0] and !is_up and is_out then payoffs[N − i − 1][tf[1]] = 0
                else if tf[0] and !is_up and !is_out then payoffs[N − i − 1][tf[1]] = 0
                end if
            end for
        end if
        po ← (payoffs[N − i − 1][: −1] × qu + payoffs[N − i − 1][1 :] × qd) × df; payoffs.append(po)
        if not is_eu then
            if is_call then po ← max(payoffs[N − i], STs[i] − K)
            else po ← max(payoffs[N − i], K − STs[i])
            end if
            payoffs[N − i] ← po
        end if
    end for
return payoffs[−1][0]
end function
```