Chapter 3

Induction on Lattices

This chapter introduces the concept of induction on lattices. Two techniques backward induction and forward induction are introduced. We will explain these concepts by applying them to the problem of option pricing. As usual, we will assume that we have a lattice model \( LM = (V, E, W, time, price) \) of prices of \( n \) assets \( \{A_1, \cdots, A_n\} \) with time horizon \( T \). We will also assume that risk-neutral probabilities exist and \( p(u, v) \) is the risk-neutral probability of transitioning from node \( u \) to \( v \). Discount factor of \( R = 1 + r \) (\( r \) is the short rate and is always non-negative) is also assumed. Sometimes we will use state prices instead of risk-neutral probabilities, but it will always be clear from the context whether we are using state prices or risk-neutral probabilities. Connection between state-prices and risk-neutral probabilities was explored in the last chapter. Throughout this chapter the lattice \( LM \) and the set of assets \( \{A_1, \cdots, A_n\} \) will be assumed to be given to us.

3.1 Definitions

Given a lattice \( LM = (V, E, W, time, price) \), an option is a financial instrument that has two components: a payoff function and a set of exercise dates. The payoff function of an option (denoted by payoff) determines the payoff of the option at each node of the lattice. The payoff function is given in terms of the prices of assets at that node, i.e. \( payoff : \mathbb{R}^n \to \mathbb{R} \). The payoff at node \( v \) is
Chapter 3. Induction on Lattices

Payoff $\text{payoff}(\text{price}(v)[1], \ldots, \text{price}(v)[n])$, i.e., payoff of an option at a node only depends upon the price of the assets at that node. Additionally, payoff could also depend on time. In that case we will write the payoff function at time $t$ as $\text{payoff}_t$. Payoff of an option is received when the holder of the option exercises it. The set of exercise dates associated with the option indicates when the holder of the option is allowed to exercise the option. Once the holder of the option exercises it, the option ceases to exist. An European option can be exercised only at the time horizon (or the expiration date) or can be only exercised at the leaf nodes of the lattice. An American option can be exercised at any time or exercised at any node of the lattice.

Depending on the form of the payoff function, options are given different names. We discuss some examples of options in the context of the binomial model. Recall that $S_t$ denotes the stock price at time $t$.

- An option with $\text{payoff}(S_t) = (S_t - K)^+$ is the call option with strike price of $K$.

- An option with $\text{payoff}(S_t) = (K - S_t)^+$ is the put option with strike price of $K$.

- Let $M_t$ be the maximum of the stock price up-to time $t$, i.e., $M_t = \max\{S_0, \ldots, S_t\}$. A lookback option with strike price $K$ has $\text{payoff}(S_t, M_t) = (M_t - K)^+$. A knock-in-barrier option with barrier $B$ and strike price $K$ has $\text{payoff}(S_t, M_t)$ given by $1_{M_t \geq B}(S_t - K)^+$. $1_{M_t \geq B}$ is the indicator function and is 1 if $M_t \geq B$ and 0 otherwise. An up-and-out option with barrier $B$ and strike price $K$ has payoff $(S_t - K)^+ 1_{M_t < B}$.

- Let $A_t = (\sum_{i=0}^t S_i)/(t+1)$ be the arithmetic average of the stock prices up-to time $t$. $\text{payoff}(S_t, A_t) = (A_t - K)^+$ is the asian call option with strike price $K$. An asian put option with strike price $K$ has payoff $(K - A_t)^+$.

- Let $G_t = (\prod_{i=0}^t S_i)^{1/t}$ be the geometric average of the stock price. $\text{payoff}(S_t, G_t) = (G_t - K)^+$ is the geometric call option with strike price of $K$. A geometric put option with strike price $K$ has payoff $(K - G_t)^+$.
3.2 Pricing european options

Consider a lattice $LM = (V, E, W, \text{time}, \text{price})$ modeling prices of assets \{A_1, A_2, \ldots, A_n\}. We will price an european option with payoff function \text{payoff}. We will assume that the time horizon is $T$ is also the expiration date of the option. The value of the option is given by \textbf{backward induction} or a \textbf{backward equation}. Intuitively, a backward equation on a lattice is a recipe to compute a value at a node in terms of the values at its successors. Of course, in order to start the induction process one has to provide the values at the leaf nodes of the lattice. A node $v$ of $LM$ has two components $\text{time}(v)$ and $\text{price}(v)$. The value of the option at node $v$ is computed in the following manner:

- If $v$ is a leaf node (i.e., $\text{time}(v) = T$), then $\text{val}(v) = \text{payoff}(\text{price}(v))$. Value at the leaf nodes is simply the payoff of the option.

- Consider a node $v$ with $\text{time}(v) < T$ and let $\text{succ}(v) = \{v_1, \ldots, v_k\}$. The value $\text{val}(v)$ of the option at node $v$ is given by the following backward equation:

\[
\text{val}(v) = \sum_{i=1}^{k} \lambda(v, v_i) \text{val}(v_i)
\]

\[
= \frac{1}{1 + r} \sum_{i=1}^{k} p(v, v_i) \text{val}(v_i)
\]

Recall that $\lambda(v, v_i)$ is the state price and $p(v, v_i)$ is the risk-neutral probability. This is the classic form of backward equation used in option pricing.

The value at a non leaf node is the expectation of the discounted value at the successor nodes.

In this framework, the option value is simply the value at the root node, or $\text{val}(\text{root})$.

**A recursive procedure**

In Figure 3.1 we give a recursive procedure to price an option. The option value is calculated by calling the procedure with the parameter as $\text{root}$ or $\text{RecursiveEval}(\text{root})$. 
\begin{verbatim}
proc RecursiveEval(v) begin 
  if (time(v) = T) 
    then return(payoff(price(v))) 
  else 
    ( Assume v has k successors) 
    succ(v) = \{v_1, \ldots, v_k\} 
    returnVal = 0 
    for i := 1 to k step 1 do 
      returnVal = returnVal + p(v, v_i)RecursiveEval(v_i) 
    od 
    ( Discount the value) 
    returnVal = returnVal/(1 + r) 
  return(returnVal) 
end
\end{verbatim}

Figure 3.1: A recursive procedure to evaluate an european option

Exercise 3.2.1 Consider the binomial model with $T = 2$, $u = \frac{1}{d} = 2$, and $r = 0.25$. Evaluate an European call option with strike price $K = 1$ using \textit{RecursiveEval}. Show all your steps. How many calls to \textit{RecursiveEval} were required? Do you spot any redundant calls?

Exercise 3.2.2 \textit{RecursiveEval} is given in terms of risk-neutral probabilities. Modify the procedure to use state prices instead of risk-neutral probabilities.

Next, we evaluate the time complexity of \textit{RecursiveEval}. Assume that the number of successor nodes that a node can have is bounded by $m$. Let $G(t)$ be the upper bound on the time taken by \textit{RecursiveEval}(v) if $time(v) = T - t$, i.e., time taken to evaluate the option value at a node at time $t$. We have the following recurrence equation for $G(t)$

$$G(t) \leq \begin{cases} 
1 & \text{if } t = 0 \\
mg(t - 1) + m & \text{if } t > 0 
\end{cases}$$
3.2. PRICING EUROPEAN OPTIONS

Exercise 3.2.3 Justify the recurrence equation for $G(t)$.

Exercise 3.2.4 Prove using the guessing method that $G(t)$ satisfies the following equation (for $t > 1$)

$$G(t) \leq \sum_{i=1}^{t} m^i + m^t$$

Notice that the argument given above proves that the time complexity of RecursiveEval is $O(m^T)$ (exponential in $T$ the time horizon).

Backward Induction

A procedure to compute the value of an european option is shown in Figure 3.2. First, we give an explanation of the algorithm given in Figure 3.2. The algorithm works backward in time because the option value is given by a backward equation. Assume that we have a procedure Frontier($t$) which returns a queue of nodes $v$ such that time($v$) = $t$, and that each node $v$ has an additional field associated with it (denoted by val($v$)) to hold the option value.

- Each time in the body of the for loop we get all the nodes that correspond to time $t$ (using the call to Frontier($t$)).
- In the while loop we go through each node in the queue. If the node is a leaf node, its value is simply the payoff at that node. If the node is non-leaf node, its value is the current value of value of its successors (check that this is what we are computing). Although, we give the algorithm it terms of state prices, it could be easily changed to use risk-neutral probabilities. The computation of $val(v)$ in terms of risk-neutral probabilities is:

$$\frac{1}{1 + r} \sum_{i=1}^{k} p(v, v_i) val(v_i)$$

We explain the algorithm in the context of the binomial model. Consider the binomial model shown in Figure 3.3. We evaluate a call
proc \( \text{Eval}() \)
begin
( Go backward in time)
for \( i := T \) to \( 0 \) step \(-1\) do
\( q = \text{Frontier}(t) \)
while not Empty\( (q) \) do
\( v = \text{DeQueue}(q) \)
if \( (\text{time}(v) = T) \)
then \( \text{val}(v) = \text{payoff}(\text{price}(v)) \)
else
\( \text{succ}(v) = \{v_1, \ldots, v_k\} \)
\( \text{val}(v) = \sum_{i=1}^{k} \lambda(v, v_i) \text{val}(v_i) \)
fi
od
od
return(\( \text{val}(\text{root}) \))
end

Figure 3.2: Backward induction to price european option

Figure 3.3: Binomial model for \( T = 2 \)
3.2. PRICING EUROPEAN OPTIONS

option with strike price $K = 1$. The initial stock price is 1. The up-factor $u = 2$ and the down-factor $d = \frac{1}{u} = 0.5$. Assume the short rate $r = 0.25$. The risk-neutral probability $p$ of an up-tick is given by the following equation:

$$p = \frac{1.25 - 0.5}{1.5} = 0.5$$

Recall that in the context of the binomial model we write a node $v$ as $(\text{time}(v), \text{price}(v))$. First the nodes corresponding to time $t = 2$ are put in the queue. These nodes are $v_{uu} = (2, 4)$, $v_{ud} = (2, 1)$, $v_{dd} = (2, \frac{1}{4})$. Value of the call option at these nodes is 3, 0, and 0 respectively. These values are stored in the fields of the corresponding nodes. Next, the nodes corresponding to time $t = 1$ ($v_u = (1, 2)$, and $v_d = (1, 0.5)$) are put in the queue. In the while loop $v_u$ is dequeued and the option value at $v_u$ is computed using the following equation (notice that values for $v_{uu}$ and $v_{ud}$ have already been computed).

$$\text{val}(v_u) = \frac{1}{1 + r} \left( p \, \text{val}(v_{uu}) + (1 - p) \, \text{val}(v_{ud}) \right)$$

$$= \frac{1}{1.25} (0.5 \cdot 3 + 0.5 \cdot 0)$$

$$= \frac{1.5}{1.25}$$

$$= 1.2$$

Next, node $v_d$ is dequeued and its value $\text{val}(v_d)$ is computed. Notice that $\text{val}(v_d) = 0.0$. Finally, the initial node $v_0 = (0, 1)$ is put on the queue and the value at the root is 0.48.

**Exercise 3.2.5** Recall that a lookback option with strike price $K$ has a payoff of $(M_t - K)^+$ where $M_t$ is the maximum of the stock prices up-to time $t$. Show the lattice for pricing the lookback option for $T = 2$. Show how the algorithm $Eval$ works to price a lookback option (assume same parameters as the call option example shown before).

**Forward Induction**

We will first explain the concept in context of the binomial model. Consider a binomial model with time horizon $T$. Recall that a binomial
model is driven by $T$ iid variables $X_1, \cdots, X_T$ such that $P(X_i = 1) = p$ and $P(X_i = -1) = 1 - p$. Also assume that the up-factor is $u$ and the down-factor is $d$. Earlier it was shown that there are $O(T^2)$ nodes in the binomial tree. Suppose we want to price $m$ European call options with $m$ different strike prices $K_1 < \cdots < K_m$. A naive procedure is to call Eval for each option. The time complexity for this procedure is $O(mT^2)$. We will show a much more efficient procedure to find the value of all $m$ call options simultaneously. Recall that a node $v$ in the binomial model is completely characterized by two quantities: time $t$ (denoted by $time(t)$) and $U$, the number of up-ticks or the random variables equal to $+1$ from the root to that node. From here on we will write a node as $(t, U)$, where $t$ is the time corresponding to that node and $U$ is the number of random variables $X_i$ that are $+1$ (or number of up-ticks) on the path from the root to that node. Typically, there are many paths from the root to a node $(t, U)$. Consider a path $\pi = v_0v_1 \cdots v_t$ to the node $(t, U) = v_t$. Let $x_1, \cdots, x_t$ be the value of the random variables along the path. Let $D$ be the number of random variables that are $-1$, or $D$ is the number of down ticks. Notice that $D + U = t$, so $D = t - U$. Also the number of down-ticks and up-ticks along different paths from the root to a specific node $(t, U)$ are the same. Assuming that the initial stock price is $S_0$ the stock price at node $(t, U)$ is $S_0u^U d^D$. The probability of a path from the root to node $(t, U)$ is $p^U (1 - p)^D$ and is called the path probability (denoted by $PP(t, U)$). Every path from root to a node has the same probability. Next we determine how many paths there are from root to the node $(t, U)$. We have $t$ random variables $X_1, \cdots, X_t$. In order to reach node $(t, U)$ we have to set $U$ of them equal to $+1$ and the rest to $-1$. This is equivalent to choosing $U$ objects (the number of up-ticks) out of $t$ objects (the number of time periods). The formula for choosing $U$ objects out of $t$ objects is given below:

$$\binom{t}{U} = \frac{t!}{U!(t-U)!}$$

where $t!$ is factorial of $t$ and given by the following expression:

$$t! = t(t-1)(t-2) \cdots 2 \cdot 1$$
3.2. PRICING EUROPEAN OPTIONS

Therefore there are \( \binom{t}{U} \) paths from the root to the node \((t, U)\). The node probability of a node \((t, U)\) is the sum of probabilities of all paths from root to that node. Since there are \( \binom{t}{U} \) paths from root to node \((t, U)\) and each path has probability \( p^U d^{t-U} \), the node probability of \((t, U)\) (denoted by \( NP(t, U) \)) is:

\[
\binom{t}{U} p^U d^{t-U}
\]

We have the following relationship between the node and path probabilities:

\[
NP(t, U) = \binom{t}{U} PP(t, U)
\]

Now we will find an explicit formula for evaluating a call option with strike price \( K \) and time horizon \( T \). There are \( T + 1 \) nodes at time \( T \) in the binomial model. Each node is of the form \((T, U)\), where \( U \) is the number of up-ticks on the path from the root to that node. Notice that \( 0 \leq U \leq T \). Let the value of the call option at node \((T, U)\) be \( val(T, U) \), which is given by the following formula:

\[
(S_0 u^U d^{T-U} - K)^+
\]

The value of the option at time zero is given by the following expression:

\[
\left( \frac{1}{1 + r} \right)^T \sum_{U=0}^{T} NP(T, U) val(T, U)
\]

where \( NP(T, U) \) is the node probability of node \((T, U)\) that was defined before (justify this!). In order to find the values of \( m \) call options whose strike price is \( K_1, \ldots, K_m \), we evaluate the expression given above \( m \) times. If we precompute the node probabilities and stock prices at the nodes (this takes \( O(T) \) time) and then evaluate the formula \( m \) times (this takes \( O(mT) \) time). Total time for this method is \( O(mT + T) \) which is significantly less than \( O(mT^2) \).

We generalize this result to an arbitrary lattice. Assume that we are given a lattice \( LM = (V, E, W, time, price) \) modeling prices of \( n \) assets.
Also we have an european option whose payoff is given by the function \textit{payoff}. Our goal is to find the price or value of this option at the root node. For this purpose we introduce the concept of \textbf{compound state price}. Recall that $\lambda(v, v')$ (where $v'$ is a successor of $v$), the state price along the edge $(v, v')$, is defined as price of the pure contingent claim $\delta(v')$ at node $v$. We want to extend this notion to arbitrary pairs of nodes. Specifically, we want to define $\lambda(v, v')$ where $v'$ is a descendant (but not necessarily an immediate successor) of $v$. For notational convenience, define $\lambda(v, v') = 0$ if $v'$ is not a descendant of $v$. We define $\lambda(v, v')$ using \textbf{forward induction} or a \textbf{forward equation}. Assume that we have defined $\lambda(v, v')$ for edges in the lattice or for pair of nodes such that $time(v') - time(v) = 1$. Notice that this is just the notion of state prices introduced earlier. Inductively, assume that we have defined state prices for all pair $(v, v')$ of nodes such that $time(v') - time(v) \leq k$ (where $k \geq 1$). We use forward induction to define $\lambda(v, v')$ for nodes such that $time(v') - time(v) = k + 1$. The forward equation defining $\lambda(v, v')$ is given below:

\[
\lambda(v, v') = \sum_{v'' \in \text{succ}(v)} \lambda(v, v'') \lambda(v'', v')
\]

Notice that price of an option with payoff function \textit{payoff} at the \textit{root} is given by the following equation:

\[
\sum_{v \in \text{leaves}(PG)} \lambda(\text{root}, v) \text{payoff} \left( \text{price}(v) \right)
\]

where \textit{leaves}(\textit{PG}) are the set of leaves or terminal nodes of the lattice \textit{LM}. Once we have calculated the compound state prices, we have a closed form expression to compute the price of an european option.

\textbf{Exercise 3.2.6} Here is an alternative forward equation defining the compound state price $\lambda(v, v')$

\[
\lambda(v, v') = \sum_{v'' \in \text{pred}(v)} \lambda(v, v'') \lambda(v'', v')
\]

Please justify the equation given above.
3.2. PRICING EUROPEAN OPTIONS

Pricing Using Monte Carlo Simulation

Recall that the procedure $nextPath()$ generates a random path through the lattice. Given a path $\pi = v_0, \cdots, v_k$, define the payoff on path $\pi$ (denoted by $payoff(\pi)$) as the payoff at the terminal node $v_k$ or $payoff(price(v_k))$. Next we describe an algorithm to price an european option using the $nextPath()$ procedure. The algorithm works by generating a fixed number of random paths (using calls to $nextPath()$) and then averaging the payoffs on those paths. Formally, we generate $N$ random paths $\pi_1, \cdots, \pi_N$ by invoking $nextPath()$ $N$ times. An estimate for the price of the option is:

$$\frac{1}{N} \sum_{i=1}^{N} payoff(\pi_i)$$

Implied Binomial Trees

This section describes the technique of Mark Rubinstein’s to build binomial trees where the risk-neutral probabilities are implied by the option prices. This example further illustrates the use of backward induction. Consider a binomial model with time horizon $T$. Recall that the nodes in the binomial model are written as $(t, U)$ where $t$ is the time and $U$ is the number of up-ticks on a path from the root to node $(t, U)$. Suppose the node probabilities at the terminal nodes $\{(T, 0), \cdots, (T, T)\}$ are provided. We will show how to infer risk-neutral probabilities from the node probabilities given at the terminal nodes. Let $NP((t, U))$ and $S((t, U))$ be the node probability and stock price at the leaf node $(t, U)$. Assume that $NP((T, U))$ and $S((T, U))$ are given for $0 \leq U \leq T$. Recall that from the node probability of $(T, U)$ we can derive the path probability of a node (denoted by $PP(T, U)$) using the equation given below:

$$NP(T, U) = \binom{T}{U} PP(T, U)$$

The equation given above assumes that every path from the root to a node has equal probability. Hence the node probability associated with a node is simply the number of paths from the root to that node
times the path probability. For an arbitrary node \((t, U)\), \(PP((t, U))\) and \(S((t, U))\) satisfy the following backward equation:

\[
PP(t, U) = PP(t + 1, U + 1) + PP(t + 1, U - 1)
\]

\[
p(t, U) = \frac{PP(t + 1, U + 1)}{PP(t, U)}
\]

\[
S(t, U) = \frac{1}{1 + r} \left( p(t, U)S(t + 1, U + 1) + (1 - p(t, U))S(t + 1, U - 1) \right)
\]

where \(p((t, U))\) is the risk-neutral up-tick probability at node \((t, U)\). We have to check the following two things:

- **Lattice is still recombining**
  Given two nodes \(v\) and \(v'\) such that \(v'\) is a descendant of \(v\), we have to check that every path from \(v\) to \(v'\) has the same probability. Recall that while writing the backward equations we inherently assumed that each path from the root to a specific node has the same probability (check where did we use this). Consider an arbitrary path \(v = v_0, \ldots, v_k = v'\) from \(v\) to \(v'\). The probability of this path is given by:

\[
p(v_0, v_1) \cdots p(v_{k-1}, v_k) = \frac{PP(v_1)}{PP(v_0)} \cdots \frac{PP(v_k)}{PP(v_{k-1})} = \frac{PP(v_k)}{PP(v_0)}
\]

Hence the probability of a path from \(v\) to \(v'\) depends only on the path probability of the start node \(v\) and the end node \(v'\). Therefore, the probability of any path from \(v\) to \(v'\) is the same.

- **Still arbitrage free**
  One can check that the risk-neutral probabilities implied by the stock prices are equal to the given probabilities, i.e., \(S(t, U), S(t + 1, U + 1)\), and \(S(t + 1, U - 1)\) imply that the risk-neutral probability of going from node \((t, U)\) to \((t + 1, U + 1)\) is \(p(t, U)\). Since the risk-neutral probabilities exist, this implies that the fictitious economy that we created is arbitrage free.

Now the question is how do we infer the node probabilities at the terminal nodes? Assume that stock prices at the terminal nodes are given.
Suppose we have $m$ call options with strike prices $K_1, \cdots, K_m$ and there current bid and ask prices are $C_i^b$ and $C_i^a$ respectively. Let $S^b$ and $S^a$ be the current bid and ask prices of the stock. For ease of notation denote the node probability and stock price at node $(T, U)$ by $P_U$ and $S_U$ respectively. Assume that we know prior node probabilities $P_U^t$ for the leaf nodes $(t, U)$, where $0 \leq t \leq T$ (this might be obtained from the previous model). Node probabilities $P_U$ (for $0 \leq U \leq T$) are found by solving the following quadratic program:

$$\min \sum_{U=0}^{T}(P_U - P_U^t)^2$$

$$\sum_{U=0}^{T} P_U = 1 \text{ and } P_U \geq 0 \text{ for } U = 0, \cdots, T$$

$$S^b \leq S \leq S^a \text{ where } S = \frac{1}{(1+r)^U} \left( \sum_{U=0}^{T} P_U S_U \right)$$

and for $1 \leq i \leq m$ we have

$$C_i^b \leq C_i \leq C_i^a \text{ where } C_i = \frac{1}{(1+r)^U} \left( \sum_{U=0}^{T} P_U (S_U - K_i)^+ \right)$$

The objective function $\min \sum_{U=0}^{T}(P_U - P_U^t)^2$ finds new node probabilities $(P_U)$ that are closest to the prior node probabilities $(P_U^t)$. The first constraint states that $P_U$ are all non-negative and sum up-to one, i.e., $P_U$'s can be interpreted as probabilities. The second constraint states that the expected discount stock price is between the bid and the ask price. The last constraint states that for all the call options the initial price implied by the node probabilities $P_U$ is between the bid and the ask price.

**Exercise 3.2.7** Consider the binomial tree given in Figure 3.4. Let the node probabilities at nodes $(2, 0)$, $(2, 1)$, and $(2, 2)$ be 0.3, 0.4, and 0.3 and stock prices be 3, 6, and 8. Compute the risk-neutral probabilities and stock prices and the intermediate nodes. Use the backward and forward equation used for the implied binomial trees.

### 3.3 American Option

Recall that an american option can be exercised at any node in a lattice $LM = (V, E, W, \text{time}, \text{price})$. Assume that when the option is exercised the payoff is given by the function payoff. In order to talk about value
Figure 3.4: Binomial model for $T = 2$
3.3. AMERICAN OPTION

of an American option we will have to introduce the concept of **decision rules**. A decision rule \( \tau \) on a lattice \( PG \) is a function from \( V \) to \( \{0, 1\} \) such that every complete path of the lattice has exactly one node \( v \) such that \( \tau(v) = 1 \). Intuitively, \( \tau(v) = 1 \) means that the holder of the option exercises the option at node \( v \). Think of \( \tau \) as the rule the holder of the option uses to exercise the option. Suppose the holder exercises according to the decision rule \( \tau \), what is the value of the option? We proceed to answer this question. Denote the value of the option at node \( v \) by \( \text{val}_\tau(v) \). \( \text{val}_\tau(v) \) is given by the following backward equation:

\[
\text{val}_\tau(v) = \begin{cases} 
\text{payoff}(\text{price}(v))\tau(v) & \text{if } \text{time}(v) = T \\
\text{payoff}(\text{price}(v)) & \text{if } \text{time}(v) < T \text{ and } \tau(v) = 1 \\
\frac{1}{1+r} \sum_{v' \in \text{succ}(v)} p(v,v') \text{val}_\tau(v') & \text{if } \text{time}(v) < T \text{ and } \tau(v) = 0 
\end{cases}
\]

Initial price of the option (if the holder exercises according to rule \( \tau \)) is simply \( \text{val}_\tau(\text{root}) \), i.e., value of the option at the root. The holder of the option is obviously going to choose a decision rule that maximizes the initial value. In other words, the value of the American option is given by the following equation

\[
\max_\tau \text{val}_\tau(\text{root})
\]

Does this mean that to calculate the value of an American option we have to compute its value for all decision rules? If this is the case, that is bad news because there are exponential number of decision rules (check this!). Fortunately, the answer is no. Define \( \text{val}(v) \) according to the following backward equation:

\[
\text{val}(v) = \begin{cases} 
\text{payoff}(\text{price}(v)) & \text{if } \text{time}(v) = T \\
\max\{\text{payoff}(\text{price}(v)), D(v)\} & \text{otherwise} 
\end{cases}
\]

where \( D(v) \) is

\[
\frac{1}{1+r} \sum_{v' \in \text{succ}(v)} p(v,v') \text{val}(v').
\]

Notice that if we are at a terminal node \( v \) (or \( \text{time}(v) = T \)), we have to exercise, so \( \text{val}(v) = \text{payoff}(\text{price}(v)) \). If we are at an intermediate node \( v \), then by exercising the option at this node the holder of the option receives a payoff of \( \text{payoff}(\text{price}(v)) \). By not exercising at the node the
value of the option is \( \frac{1}{1+r} \sum_{v' \in \text{succ}(v)} p(v, v') \text{val}(v') \), i.e., the present value of the option at the successor nodes. Therefore, we only exercise at a node \( v \) iff
\[
\frac{1}{1+r} \text{payoff}(\text{price}(v)) \geq \frac{1}{1+r} \sum_{v' \in \text{succ}(v)} p(v, v') \text{val}(v')
\]
This explains taking the maximum in the definition of \( \text{val}(v) \) at an intermediate node \( v \). We find the exact decision rule that the value of the option corresponds to. For all terminal nodes \( v \) define \( \text{exercise}(v) = 1 \). For an intermediate node \( v \), \( \text{exercise}(v) = 1 \) iff the following equation is true
\[
\text{payoff}(\text{price}(v)) \geq \frac{1}{1+r} \sum_{v' \in \text{succ}(v)} p(v, v') \text{val}(v)
\]
In other words, \( \text{exercise}(v) = 1 \) iff we will exercise at node \( v \) if we ever reach it. Finally, the optimal rule \( \tau^* \) is given by \( \tau^*(v) = 1 \) iff \( \text{exercise}(v) = 1 \) and there is no ancestor \( v' \) of \( v \) such that \( \text{exercise}(v) = 1 \).

**Exercise 3.3.1** Consider the binomial model given in Figure 3.3. Find the value of an American call option with strike price \( K = 1 \), \( u = \frac{3}{2} = 2 \), and short rate \( r = 0.25 \). Also indicate the optimal exercise rule for the holder of the option.

### 3.3.1 Bermuda Options

An American option can be exercised at any node in the lattice. An European option can only be exercised at the leaf nodes in the lattice. Therefore, and American option provides complete freedom to the holder of the option as far as exercise strategy is concerned. In contrast, an European option restricts the holder of the option to only exercise at the expiration date of the option. There are several cases where the holder can only exercise at some pre-specified time, e.g., exercise is only allowed in the last five time periods. These options are frequently called Bermuda options. Assume that the payoff function \( \text{payoff} \) of the option is given. In addition to the payoff function, a Bermuda option has a function
\[
\text{canExercise} : [0, 1, \cdots, T] \rightarrow \{0, 1\}.
\]
3.4 Hedging

Assume that we are given a lattice \( LM = (V, E, W, time, price) \) model of prices of \( n \) assets \( A_1, \ldots, A_n \). Suppose we are also given a function \( val : V \rightarrow \mathbb{R} \), or we are given a value at each node of the lattice. Our goal is to find an investing strategy (only using the \( n \) assets) such that our wealth at each node \( v \) is equal to \( val(v) \). A strategy has two components a portfolio process \( \Delta \) and a consumption process \( consume \). Recall that a portfolio is defined on nodes and consumption is defined on edges. Wealth \( X(v) \) at a node \( v \) is given by the following equation:

\[
X(v) = \Delta(v) \cdot price(v)
\]

In other words, wealth at a node \( v \) is simply the value of the portfolio at that node. For each edge \((v, v')\) we also have:

\[
X(v') = \Delta(v) \cdot price(v') - consume(v, v')
\]

In other words as we move from node \( v \) to \( v' \), the value of the portfolio at node \( v \) changes to \( \Delta(v) \cdot price(v') \) and \( consume(v, v') \) is consumed in moving from \( v \) to \( v' \). Given a function \( val \) our aim is to find a strategy \((\Delta, consume)\) such that \( X(v) = val(v) \), i.e., the wealth at each node matches the specified value at that node. Such a strategy is called a hedging strategy for the value function \( val \). The construction that we are about to provide assumes that we have complete markets. This construction embodies the essence of a complete market.

Every payoff process on the lattice has a corresponding hedging strategy.
We proceed to construct a hedging strategy for a value function \( \text{val} \). We construct a strategy at a node \( v \). To obtain a hedging strategy on the entire lattice, we simply repeat the process for each node of the lattice. Consider a node \( v \) with \( k \) successors \( v_1, \ldots, v_k \) and a portfolio \( \Delta'(v) \) such that the value of this portfolio at node \( v_i \) is \( \text{val}(v_i) \). \( \Delta'(v) \) is obtained by solving the following system of equations:

\[
\begin{align*}
\Delta'(v) \cdot \text{price}(v_1) &= \text{val}(v_1) \\
\Delta'(v) \cdot \text{price}(v_2) &= \text{val}(v_2) \\
& \vdots \\
\Delta'(v) \cdot \text{price}(v_k) &= \text{val}(v_k)
\end{align*}
\]

A solution to the system of equations given above exists because we are assuming that our lattice represents a complete market. Value of portfolio \( \Delta'(v) \) at the node \( v \) is given by:

\[
\Delta'(v) \cdot \text{price}(v) = \frac{1}{1 + r} \sum_{i=1}^{k} p(v, v') \Delta'(v) \cdot \text{price}(v_i)
\]

\[
= \frac{1}{1 + r} \sum_{i=1}^{k} p(v, v') \text{val}(v_i)
\]

In general, \( \Delta'(v) \cdot \text{price}(v) \) is not equal to \( \text{val}(v) \). Assume that asset \( A_1 \) has a positive price at node \( v \). Let \( \Delta''(v) \) be a portfolio whose value (\( \Delta''(v) \cdot \text{price}(v) \)) is equal to

\[
\text{val}(v) - \frac{1}{1 + r} \sum_{j=1}^{k} p(v, v_i) \text{val}(v_i).
\]

Recall that \( p(v, v') \) is the risk-neutral probability on the edge \((v, v')\). Such a portfolio can be constructed by holding

\[
\frac{1}{\text{price}(v)[1]} \left( \text{val}(v) - \frac{1}{1 + r} \sum_{j=1}^{k} p(v, v_i) \text{val}(v_i) \right)
\]

units of asset \( A_1 \) and zero units of assets \( A_2, \ldots, A_n \). Let \( \Delta(v) = \Delta'(v) + \Delta''(v) \) and \( \text{consume}(v, v_i) = \Delta''(v) \cdot \text{price}(v_i) \). One can check that \( X(v) = \text{val}(v) \) and \( X(v_i) = \text{val}(v_i) \) for \( 1 \leq i \leq k \). Following observations are easy to make:
3.4. HEDGING

- If \( \text{val} \) corresponds to the value of an European option with payoff function \( \text{payoff} \). We have the following equation:

\[
\text{val}(v) = \frac{1}{1 + r} \sum_{v' \in \text{succ}(v)} \text{val}(v') p(v, v')
\]

It is easily seen that the hedging strategy for \( \text{val}(v) \) has the consumption process identically zero, or \( \text{consume}(v, v') = 0 \) for each edge \( (v, v') \).

- Let \( \text{val} \) correspond to the value of an American option with payoff function \( \text{payoff} \). We have the following equation:

\[
\text{val}(v) \geq \frac{1}{1 + r} \sum_{v' \in \text{succ}(v)} \text{val}(v') p(v, v')
\]

It is easily seen that the hedging strategy for \( \text{val}(v) \) has the consumption process non-negative, or \( \text{consume}(v, v') \geq 0 \) for each edge \( (v, v') \).

Assume that we have a seller and a buyer or holder of the option. The seller uses the hedging strategy so that if the buyer of the option exercises at a node, the seller has enough wealth to cover the payoff of the option. Reader should check that the hedging strategy just constructed satisfies this property.

We will illustrate the construction of our hedging strategy for the specific case of the binomial model. Consider a node \( v \) with two successors \( v_u \) and \( v_d \). We have two assets: a stock and bond. Price of the bond is 1 at node \( v \) and \( 1 + r \) at nodes \( v_u \) and \( v_d \). Price of the stock is \( S \) at node \( v \) and \( Su \) and \( Sd \) at nodes \( v_u \) and \( v_d \) respectively. Probability of an up-tick or transitioning from node \( v \) to \( v_u \) is \( p \). Suppose we are given values \( \text{val}(v) \), \( \text{val}(v_u) \), and \( \text{val}(v_d) \) at each node. We will construct a hedging strategy at node \( v \). First, we will construct a portfolio \( \Delta'(v) \) that satisfies the following equations:

\[
\Delta'(v)[1](1 + r) + \Delta'(v)[2]Su = \text{val}(v_u)
\]

\[
\Delta'(v)[1](1 + r) + \Delta'(v)[2]Sd = \text{val}(v_d)
\]
Solving these equations we have the following answer:

\[
\Delta'(v)[1] = \frac{u \cdot \text{val}(v_u) - d \cdot \text{val}(v_d)}{(1 + r)(u - d)}
\]

\[
\Delta'(v)[2] = \frac{\text{val}(v_u) - \text{val}(v_d)}{S(u - d)}
\]

Next we will construct a portfolio \(\Delta''(v)\) such that its value at node \(v\) is

\[
\text{val}(v) = \frac{1}{1 + r} (p \cdot \text{val}(v_u) + (1 - p) \text{val}(v_d))
\]

\(\Delta''(v)\) has 0 units of stock and

\[
\text{val}(v) = \frac{1}{1 + r} (p \cdot \text{val}(v_u) + (1 - p) \text{val}(v_d))
\]

units of bond. Portfolio \(\Delta(v)\) is equal to \(\Delta'(v) + \Delta''(v)\). Also, we define the consumption process using the following equations:

\[
\text{consume}(v, v_u) = \Delta''(v) \cdot \text{price}(v_u)
\]

\[
= \text{val}(v)(1 + r) - (p \cdot \text{val}(v_u) + (1 - p) \text{val}(v_d))
\]

\[
\text{consume}(v, v_d) = \Delta''(v) \cdot \text{price}(v_d)
\]

\[
= \text{val}(v)(1 + r) - (p \cdot \text{val}(v_u) + (1 - p) \text{val}(v_d))
\]