Assignment 2 Report

Aw Sheng Xiang | 31259030

Parser BNF

<mem> ::= <hand> <separator><listAction> <separator><lastBid> <separator><numCards> <separator><runningCount>
<hand> ::= “[“ <listCard> “]” | “[””]”
<listCard> := <card> “,” | <card>
<card> := <suit> <rank>
<suit> := “S” | “C” | “D” | “H”
<rank> := “A” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” | “10” | “J” | “Q” | “K”
<listAction> := “[“ <actions> “]” | “[” ”]”
<actions> := <action> “,” | <action>
<action> := “st’ | “ht”| “dd”| “sp”| “ins”|
<lastBId> := <int>
<numCards> := <int>
<runningCount> := <int>
<int> := <digit>
<digit> := “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” | “-”
<separator> ::= “_”

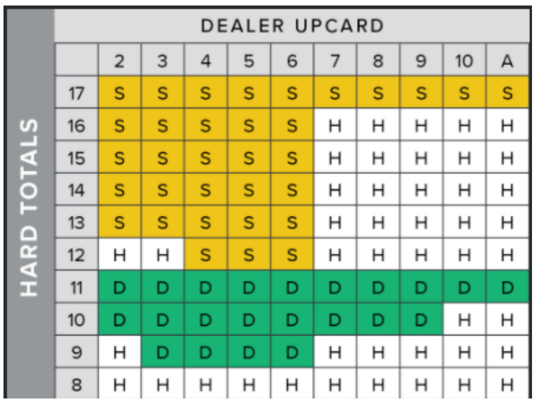
Player implementation

Main strategy consists of 2 parts:

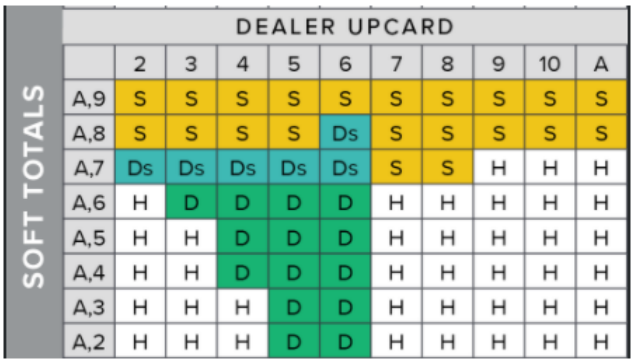
A. Playing basic strategy
B. Adjusting bids based on true count obtained from counting cards

A. Basic Strategy

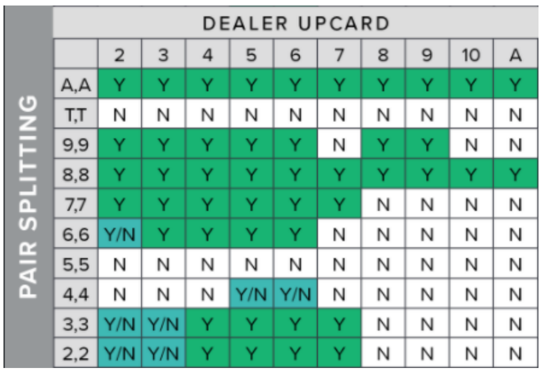
Basic strategy is a basic set of actions that is to be returned based on different combinations of the dealer upcard and the player’s hand that is implemented using simple conditionals such as guards and if-else statements.
The combinations and the action to return are shown in the diagram below:



HARD TOTALS	DEALER UPCARD									
	2	3	4	5	6	7	8	9	10	A
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	D	D	D	D	D	D	D	D	D	D
10	D	D	D	D	D	D	D	D	H	H
9	H	D	D	D	D	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H



SOFT TOTALS	DEALER UPCARD									
	2	3	4	5	6	7	8	9	10	A
A,9	S	S	S	S	S	S	S	S	S	S
A,8	S	S	S	S	Ds	S	S	S	S	S
A,7	Ds	Ds	Ds	Ds	Ds	S	S	H	H	H
A,6	H	D	D	D	D	H	H	H	H	H
A,5	H	H	D	D	D	H	H	H	H	H
A,4	H	H	D	D	D	H	H	H	H	H
A,3	H	H	H	D	D	H	H	H	H	H
A,2	H	H	H	D	D	H	H	H	H	H



PAIR SPLITTING	DEALER UPCARD									
	2	3	4	5	6	7	8	9	10	A
A,A	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
T,T	N	N	N	N	N	N	N	N	N	N
9,9	Y	Y	Y	Y	Y	N	Y	Y	N	N
8,8	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
7,7	Y	Y	Y	Y	Y	Y	N	N	N	N
6,6	Y/N	Y	Y	Y	Y	N	N	N	N	N
5,5	N	N	N	N	N	N	N	N	N	N
4,4	N	N	N	Y/N	Y/N	N	N	N	N	N
3,3	Y/N	Y/N	Y	Y	Y	Y	N	N	N	N
2,2	Y/N	Y/N	Y	Y	Y	Y	N	N	N	N

Blackjack strategy charts - how to play Perfect blackjack. Blackjack Apprenticeship. (2021, April 25). Retrieved October 25, 2021, from https://www.blackjackapprenticeship.com/blackjack-strategy-charts/.

Action such as double down which has to be followed by a sequence of specific actions is implemented using the memory such that there is a list stored in memory and it should remain empty unless a double down action is returned which will then be stored in the list Returning a Hit action after Double Down will be handled by the function hardMove and softMove which always checks if list action in memory is empty, if it is not empty then it will proceed to retrieve the first action and check if it is a double down action if it is then it will return Hit and Stand otherwise. Furthermore, actions that require an additional bid to be returned together such as Double Down and Split also stands to benefit from the use of parser as the bid amount from previous round can be easily converted into string to be stored as memory and retrieved using the parser when required

B. Bidding strategy derived from card counting

This is where the need for parsers comes into play again as for card counting we have to store a running count which is always updated before a player places a bet to provide the player with the most updated count in order to be passed into the bidding function to adjust the bid correctly. Second is the number of cards left in the deck. This is so that the player will know when the deck is reshuffled as this information is not provided to the playCard function therefore it is crucial to keep count of the cards left in the deck so that when it hits 0 the player will know that the deck will be reshuffled and proceed to reset the running count to 0 as leaving the running count not resetted will result in an inaccurate reflection of information. Next, having a count of the cards left in the deck also provides us the information needed to estimate the deck left which can in turn be used to calculate the true running count, this is needed as more than 1 deck is used in the play.

The tracking of cards left and running count is divided into 2 parts which is during the bidding turn and during the playing turn. This is due to the pattern of how the information about players is fed into the playCard function where during a bidding round only the most updated information of the other players are passed into the playCard function, this means that information for the other players that succeeds the current player in the play order are the latest from last round as they had not bid yet. Therefore we can use this information to update the counter that we are tracking while in the play round, the other players that precede the current player have already played and therefore their latest information will be passed into the playCard function and similarly we can use that information to update the relevant counters that we are tracking.
The information of the play order of a game is extracted from the list of PlayerPoints passed in and processed through a function that divides the order into players that precede and succeed the current player, the current player is placed in the group the succeeds

1. Bidding round

The cards that are dealt to the other players that succeeds the current player in the order of play and this includes the dealer from last round is processed to update the counter for running count and cards left in the deck

2. Play turn of the current player

The cards that are dealt to the other players that precede the current player in the order of play are processed to update the counters as well.

In the process of tracking the counters which is essential in the bidding strategy and thus the strength of the player, parser combinators has enabled the custom data type to be easily converted and stored in a specific format as simple memory string and vice versa enable the conversion of them back to the required data type from string during retrieval by parsing the string through a sequence of parsers built from lower level parsers with specific constraint that tokenize only preset valid input patterns and converting them into their respective types so that they can be updated or used.

One particularly interesting obstacle that came up in tracking the running count was that if the card deck was shuffled in the play round before and during the next bidding round when we subtract all the cards that were dealt out and resulted in a negative card left count, we have to backtrack the play order of the players in reverse, take the negative count of the cards left in deck as those cards are the cards that are dealt from a new deck and track each of them backwards convert it to points using the hi-lo conversion method and use the sum of the points as the starting value of the running count again to get an accurate running count again. This is a reset and adjustment mechanism for the running count in order to ensure its accuracy is not compromised in the event of a deck shuffling.