Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS

Create Team

Coding a logical sub pot system for texas holdem poker

Ask Question

2

Edit It seems like I am getting different responses on how the game actual works, and after reading the official rules, and talking to numerous poker buddies, I guess I don't know the rules myself. Any clarification would be appreciated.

I am working on a little poker game in MSVC++ 2010 Express, and have been stuck trying to come up with a way to code the sub pot system. For some reason I can not get my head around how it should work, and was wondering if SO could post some ways to go about it. Here is a particular situation that can, and more than likely will occur in a texas holdem poker game.

Situation:

Player A has first action with $50 chips and decides to go all in. Player B raises to $150. Player C has only $70s worth of chips and decides to go all in. Player D only has $20 and goes all in. Now, how can I devise a sub pot mechanism to track all these.

From what I understand, what would happen would be:

Player A creates the main pot with $50. You combine B and C's $50s to make the main pot $150. You then take Player B's leftover $100 and split it into $80 and $20. You then make a sub pot for Player B and C worth $40 (Player

into Player B, and Cs $40 sub pot, now worth $60. *(or does this not get added? Does it not get added to any bet since it cant cover the main pot of $50, if so then they dont get added to anything *

Now, when it goes down to evaluate. If Player A wins he wins the $150 from Player A, B and C. Next, Player B, C, and D go at it with their sub-pot worth $60.

If Player B wins, he wins everything.

If Player C wins, he wins the $150 from Player A, B, and C. He then challenges Player B, and D for the $60.

Player D can only win the $60, whereas, someone would have already won the Player A, B, and C pot when it goes down this far. *(depends if this gets added or not to B, and C's pot, since it doesn't cover the main 50$ bet)*

Is this how everything should work? I am having a hard time trying to figure out how I can track each bet, and sub-pot. Any ideas, or logical ways to implement it would help a lot. Thank you for your time. :-)

I was thinking about having each bet being a unique id, or maybe each round has an id, and add each bet to an array to be evaluated that also points to container with player information. I also have to take into consideration that some players might be in sub pots and also be in hand already and fold, which means, I have to track that too.

c++ containers poker

edited Sep 27 '12 at 22:29

364 1 6 25

Write the simplest case first, with just one player. Then add the second player. Then add many players. Then show us your code if you need to. – Peter Wood Sep 27 '12 at 7:48

This problem seems to have been answered already - stackoverflow.com/questions/5462583/… – Keldon Alleyne Sep 27 '12 at 8:14

3 Answers

2

In this example both main and side pots are calculated wrong.

RULE: The ruling principle is that each player matches as much of the opponents' bets as he has left in his stack.

Calculation:

1) First, we consider a player with the smallest stack (who went all in). In the current example this is a player D with $20.

2) Next we sum up $20 from each player (A,B,C,D) and the main pot is formed equal to $80, it is contested by all players.

3) Players' chips left A – $30, B – $130, C – $50, D – $0

4) Next we consider the second smallest stack, in the current example this is player A who has $30 left. The side pot1 is formed equal to $30(A) + $30(B) +$30(C)= $90. Player D can't win this side pot as he ran out of money.

6) The side pot2 is formed equal to $20(B) + $20(C)= $40. Player A can't win this side pot as he ran out of money.

7) Player B has $80 left, this amount is returned to him.

So we finally get:

Main pot = $80, contested by all players A,B,C,D

Side pot1 = $90, contested by A,B,C

Side pot2 = $40, contested by B,C

$80 returns to player B

answered Sep 27 '12 at 15:55

Enterra

54 2

Are you sure this is how it works? Can I see where you got that information. – User Sep 27 '12 at 22:28

See pokerjunkie.com/rules-governing-allin-situations-in-poker. Also it implemented in our software - enterra-poker.com – Enterra Sep 28 '12 at 2:24

Thank you. This clears up everything. :-) – User Sep 28 '12 at 2:38

0

Your example talks by itself. A subpot is created either on the first bet or each time the bet is different from the initial bet. There are some properties:

- One bet is a singleton subpot
- An existing pot should be able to be splitted if the new amount is different

- a subpot consists of a unique amount of money and several players
- One bet is a singleton subpot
- Two subpots with the same amount (of money) can merged to form a new subpot (like sets)
- A subpot can be splitted in two supbot, with `amount = amount1 + amount2` ;
- Each time a new bet is added, first split on the difference, then merge pots of the same amount. ie

```
//laughable attempt
class Subpot{
  int amount; //oops i mean bet
  int pot; //actually function = amount x partici
  std::vector<Players*> participants;

  bool split(int amounta, int amountb, Subpot& a,
  static bool merge(Subpot& a, Subpot& b , Subpot
}
```

Now each time there is a new hand, you take the previous stable set of subpots and you create a the next generation of subpots

- create subpot hand
- add to the set of subpot (what set? see below)
- take the one with the smaller bet betSmall
- split all existing in pots of size betSmall, and bet - betSmall
- merge the ones of size betSmall
- if I am not mistaken now you have a stable set of subPot

answered Sep 27 '12 at 8:00

0

The way I'd do it is to make a separate container that holds wagers that are made, keeping track who wagered and how much.

```
std::list<std::map<PlayerID, Chips>> wagers;
unsigned n_raises;  // to keep track of number of rai
```

I assume you've a list/array of active players in a hand.

Now if a player makes a call, you simply insert his ID and chips amount in the map that's at the back of list.

If a player makes a raise, you insert an ID and matching chip amount in the map that's at the back of list and then *push back* another map. You insert the amount raised in this new back. Update `n_raises` and `to_call` accordingly.

When inserting a wager of a player, you need to iterate from the beginning of list and start the insertion when you first encounter a map where there's no chips from that player. There could be multiple raises in front and so you obviously can't always just insert in the last map.

The tricky case is when a player doesn't have enough chips to cover a full call. In this case you find the map where he rans out chips, *insert* a new map right after it, and transfer all the chips (of other players) the player couldn't cover in this new map. (We don't update `n_raises` in this case).

Here's how it would all look like for 4 players A,B,C,D:

```
Player A bet 100
```

```
Player B calls:

    map0: (A, 100) (B,100)

Player C raises to 300:

    map0: (A, 100) (B,100) (C,100)
    map1: (C, 200)

Player D calls:

    map0: (A, 100) (B,100) (C,100) (D,100)
    map1: (C, 200) (D,200)

Player A's turn, he folds.
Player B calls, but he's got only 50 left:

    map0: (A, 100) (B,100) (C,100) D(100):
    map1: (B, 50) (C, 50) (D,50)           <--- here w
    map2: (C, 150) (D, 150)

The betting round is over.
```

You can have a separate list for each betting round, or just one and keep a copy of iterator for a start of new round.

It's now easy to divide the pot. You determine the best hand and just start popping maps of the front of the list while the player with the best hand is involved in the map. If after that the list is empty, you're done. Otherwise, determine the second best hand and repeat.

In the example abouve, if player B ends up with the best hand, you push him map0 and map1. map2 would go to the winner of C and D.

Let's try your example:

> Player A has first action with $50 chips and decides to go all in. Player B raises to $150. Player C has only

```
map0: (A,50)
--------------------
map0: (A,50) (B,50)
map1: (B,100)
--------------------
map0: (A,50) (B,50) (C, 50)
map1: (B,20) (C,20)
map2: (B,80)
--------------------
map0: (A,20) (B,20) (C,20) (D,20)
map1: (A,30) (B,30) (C,30)
map2: (B,20) (C,20)
map3: (B,80)
```

I think this closely resembles the way we do it at an actual poker table and I'd probably aproach it this way if I were to write poker game. Hope it helps :)

edited Sep 27 '12 at 10:19

answered Sep 27 '12 at 10:11

jrok

45.9k 6 87 122

That seems like the easiest. But now I am bent on if I actually have the rules down--it would be depressing to code it all, and have to redo it because of the rules don't match official rules. – User Sep 27 '12 at 22:31

@Ohmages I didn't notice you got your calculation wrong in the question. But notice the last section in my answer, it's correct (just like explained in Enterra's answer): map0 is the main pot, contested by all players and it's worth $80. map1 is first side pot, worth $90 and contested by A, B and C. map2 is second side pot, worth $40, for B or C, and map3 is the leftover that gets returned to B. – jrok Sep 28 '12 at 11:09

Oh and, in case you want to clear any doubts about poker