**Poker Ruby**

cd /Users/local-kieu/programming\_skills/four\_languages/poker/ruby

poker\_ruby.txt

require "pp"

def **compare\_ary**(ary1, ary2)

ary1.each\_with\_index do |e, idx|

return -1 if e < ary2[idx]

return 1 if e > ary2[idx]

end

0

end

def **best\_hands(hands)**

besthand = hands.shift

pk1 = Poker.new(besthand)

best\_score, best\_ranks = pk1.score\_hand # score\_hand => integer, hash

hands.each\_with\_index do |hand, idx |

pk = Poker.new(hand)

score, cur\_ranks = pk.score\_hand

if score > best\_score

best\_score = score

besthand = hand

**elsif score == best\_score**

# compare ranks(keys) between best\_ranks and cur\_ranks

best\_keys = best\_ranks.keys

ranks\_keys = cur\_ranks.keys

result = compare\_ary(best\_keys, ranks\_keys)

if result < 0

best\_score = score

besthand = hand

elsif result == 0

besthand = [besthand] # change besthad from String to Array

besthand << hand

end

end

end

besthand.kind\_of?(String) ? [besthand] : besthand

end

**class Poker**

RANKS\_INDEX = '--23456789TJQKA'

attr\_accessor :suits, # array of suit chars

:ranks\_hsh, # hash: key = rank, value = rank count, is sorted with ranks value DESC

:hand # current hand

def initialize(hand)

@hand = hand

# @suits = array of suit

# @ranKs\_hsh = sorted ranks hash: value is count, ranks to RANKS\_INDEX

**@suits, @ranks\_hsh = get\_suit\_and\_rank**s(@hand)

end

### 1 convert hand to suit and ranks hash with count

def **get\_suit\_and\_ranks(**hand)

# change 10 to T

hand1 = hand.gsub('10', 'T')

hand2 = hand1.gsub(' ','')

suits = []

hand2.each\_char.with\_index { |e, idx| suits << e unless idx.even? } #=> ["S", "H", "S", "D", "H"]

ranks = []

hand2.each\_char.with\_index { |e, idx| ranks << e unless idx.odd? } #=> ["2", "4", "6", "4", "J"]

ranks2 = []

ranks.each { |e| ranks2 << '--23456789TJQKA'.index(e) } # => [2, 4, 6, 4, 11]

rsorted\_ranks = ranks2.sort.reverse # => [11,6,4,2,2]

ranks\_hsh = {} # => {11=>1, 6=>1, 4=>1, 2=>2}

rsorted\_ranks.each {|e| ranks\_hsh.key?(e) ? ranks\_hsh[e] += 1 : ranks\_hsh[e] = 1 }

**return suits, ranks\_hsh**

end

**def score\_hand()**

kinds = @ranks\_hsh.values.sort.reverse # [2, 1, 1, 1]

ranks\_keys = @ranks\_hsh.keys.sort # [2, 4, 6, 11] sorted, but no reverse

# **check for flush**

flush = @suits.all? { |e| suits[0] == e }

# check **special straight** case [2,3,4,5,14]: then modify ranks\_hsh

if ranks\_keys.eql?([2, 3, 4, 5, 14])

@ranks\_hsh = {1=> 1, 2=> 1, 3=> 1, 4=> 1, 5=> 1}

ranks\_keys = @ranks\_hsh.keys

end

# **check for straight**

straight = ranks\_keys.each\_cons(2).all? { |a,b| b == a + 1 }

**# calculate score**

score = 0

case

when straight && flush

score = 8

when kinds.eql?([4,1])

score = 7

when kinds.eql?([3,2])

score = 6

when flush

score = 5

when straight

score = 4

when kinds.eql?([3,1,1])

score = 3

when kinds.eql?([2,2,1])

score = 2

when kinds.eql?([2,1,1,1])

score = 1

else

score = 0

end

# ranks\_hsh = {11=>1, 6=>1, 4=>1, 2=>2} = sort reverse key

# create ranks\_values hash with ranks key is sorted DESC, then values is sorted DESC

ranks\_values = ranks\_hsh.sort {|a1, a2| a2[1] <=> a1[1] }.to\_h

# => {2=>2, 11=>1, 6=>1, 4=>1}

[score, ranks\_values]

end

end

hands = ['2H 3H 2C 3S 5D','AH 2C 3S 4S 5S'] # ace low straight

best\_hands(hands)