DESIGN AND ANALYSIS OF ALOGORITHM

TOPIC:

CAPSASUSN USING GREEDY ALORITHM

TEAM MEMBERS:

- RA2011003010904
- RA2011003010905
- RA2011003010913

Capsa Susun using Greedy Algorithm

Abstract.

The Capsa Susun game is one of the most popular types of games in parts of the Asia Pacific region, especially Hong Kong. In this century also the game of Capsa Susun began to be popular with the people of Indonesia, especially the people who enjoyed gambling games. Overall Capsa Susun means a game of thirteen cards arranged. This game is played by four people. How to play Capsa Susun is by arranging cards that have been arranged in three different levels where the top card consists of three cards and the other two levels consist of five cards. And the way to determine the winner in this game is the player who wins the winnings of each level. Each level has a different count. With the risk, victory becomes an absolute thing that must be achieved. One way is by using the greedy algorithm. This algorithm develops based on the experience and logic of the players who have played it.

1. Introduction

The greedy algorithm is based on the word “greedy” which means greedy. The essence of this algorithm is to take the most part that can be taken at this time, without taking into account the consequences that will be faced later. In other words, the greedy algorithm takes choices that will provide the best local solutions in the future that it will also provide global solutions or the best solutions overall. So, the greedy algorithm assumes that the local optimum solution is part of the global optimum solution.

When playing a common capsa susun which is usually a dilemma is when choosing between strengthening the middle or the upper part, for example: level one = JJQQQ with a combination of interest is then a level two = 9-9-8-8-7 with a combination of interest is then level three = AS-K-3. The above arrangement can be changed to level two = 8-8-AS-K-7 with a combination of interest up to then level three = 9-9-3. With the alternative arrangement, the third level has a higher potential for winning.

And the problem is can we apply greedy algorithm method in Capsa Susun for getting us a guaranteed win situation ?

2. Analysis of Algorithm

2.1 Elaboration of algorithm

There are 3 main algorithm process that are used in this program. The first algorithm is getCard(), in this algorithm the program will process the cards that are given to the program by the user and contain it into an array. The proper syntax for a card to be read by the program is the number of the card or the first letter of the card i.e. jack ‘j’, queen ‘q’, king ’K’; then will be followed by the card’s suits (s for spades, h for hearts, c for clubs, d, for diamonds). For example, if we have a king of spades then the formal way of representing the card in the program is by “Ks”. The next step is to sort the order of the cards by points, then separate the cards representative by comma. Therefore, the proper example is “2c,2d,3h,4d,4h,5d,6c,6h,8c,8h,9h,10c,Kh”.

The second algorithm is checkCard(), in this function the program will calculate the points for each one of the card and place the points into an array. The points are represented by a number from 2 to 14. 2 of any suits are assumed to be the lowest point of a card which is 2 points. And aces of any suits are assumed to be the highest point of a card which is 14.

The third algorithm is Calculation(), the most important algorithm is the Calculation() because in this algorithm the program will determine the output by using greedy algorithm. First the function will determine if there is a straight in the set of cards, if there is such subset in the set it will be taken as the output, the maximum possibility of having a straight is 2 straights because in 13 cards it can only have 2 set of straight. Therefore, if the function found a straight it will be taken to line 1, and if there are 2 it will be taken to line 1 and line 2. Next the function will search for pairs, if there are more than one pair then it can be taken to line 1 or 2 as two-pairs if the lines are not taken by straights. A pair can also be taken to line 3. Next the function will try to find the biggest high card in the set if a line is still empty.

2.2 Analyze the time complexity

The time complexity of this algorithm is constant (n). Because the input are constant, never change the value that is 13.

3. Experimental Results

Figure 1: This is the ilustration of capsa susun game.

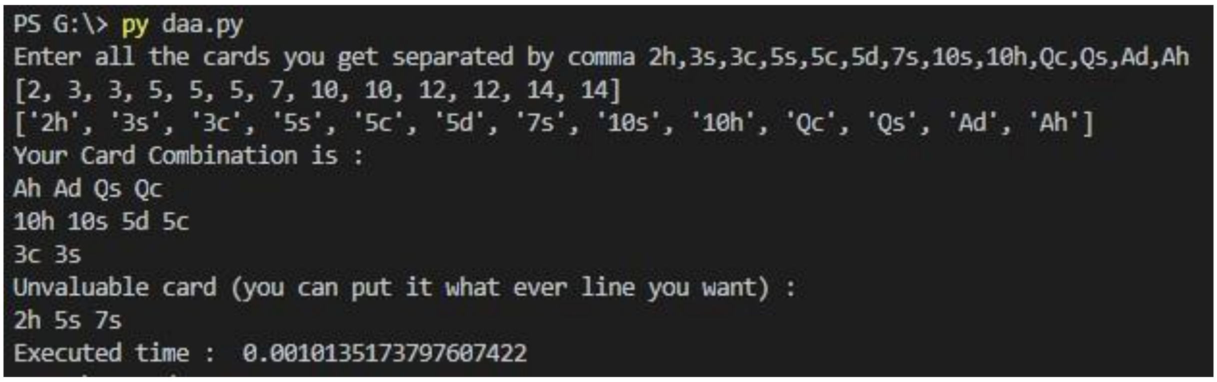
The sequence of numbers in the game Capsa Susun is As / 1 (one), 2, 3, 4, 5 ... etc. And for the sequence of images in this game are from the lowest Diamond (Diamond), curly (Spade), heart (Heart), waru (Club). And the cards that have been distributed have a variety of arrangements with different levels of victory.

Below is a card arrangement in the game Capsa Susun from the highest to the lowest:

- Royal Flush (10, J, Q, K, A with the same picture)
- Straight Flush (five consecutive cards below the number 10 with the same picture).
- Four of a kind (four cards that are equal to / A, A, A, A / 2,2,2,2 / 3,3,3,3 ... etc.).
- Full House (a combination of three cards of the same value and two cards worth the same / A, A, A and 2.2 / 3,3,3 and 4,4 ... etc.).
- Flush (five cards with the same picture)
- Straight (five consecutive cards)
- Three of a kind (three cards of the same value)
- Double Pair (a combination of two cards worth the same as two cards of equal value).
- Pair (two cards of equal value)
- High Card (the highest value card from the arrangement of three or five existing cards).

In our program, we only use Straight, Double pair, and a high card as the card arrangement.

No. 1



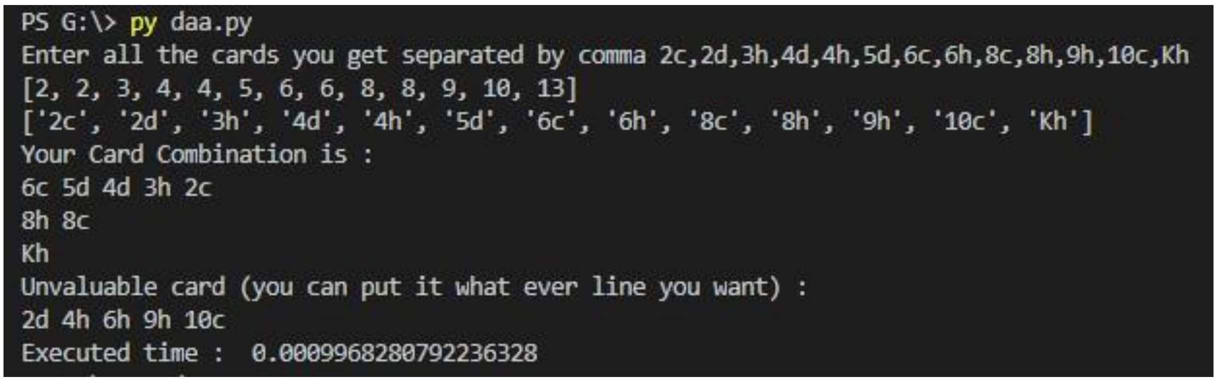
```
PS G:\> py daa.py
Enter all the cards you get separated by comma 2h,3s,3c,5s,5c,5d,7s,10s,10h,Qc,Qs,Ad,Ah
[2, 3, 3, 5, 5, 5, 7, 10, 10, 12, 12, 14, 14]
['2h', '3s', '3c', '5s', '5c', '5d', '7s', '10s', '10h', 'Qc', 'Qs', 'Ad', 'Ah']
Your Card Combination is :
Ah Ad Qs Qc
10h 10s 5d 5c
3c 3s
Unvaluable card (you can put it what ever line you want) :
2h 5s 7s
Executed time :  0.0010135173797607422
```

Figure 2: This is the result when we run the program.

This is our first experiment wich have given cards : 2h,3s,3c,5s,5c,5d,7s,10s,10h,Qc,Qs,Ad,Ah. And the result of combination from our program is :
1st line : Ah,Ad,Qs,Qc (Two Pair)
2nd line : 10h,10s,5d,5c (Two Pair)
3rd line : 3c,3s (One Pair)

And the card that didn't see the value wich can be put in any line are : 2h,5s,7s

No. 2



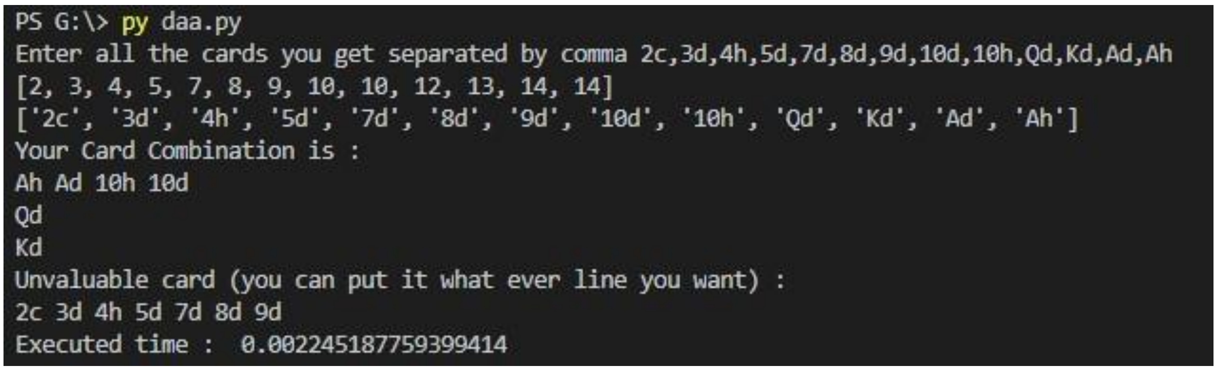
```
PS G:\> py daa.py
Enter all the cards you get separated by comma 2c,2d,3h,4d,4h,5d,6c,6h,8c,8h,9h,10c,Kh
[2, 2, 3, 4, 4, 5, 6, 6, 8, 8, 9, 10, 13]
['2c', '2d', '3h', '4d', '4h', '5d', '6c', '6h', '8c', '8h', '9h', '10c', 'Kh']
Your Card Combination is :
6c 5d 4d 3h 2c
8h 8c
Kh
Unvaluable card (you can put it what ever line you want) :
2d 4h 6h 9h 10c
Executed time :  0.0009968280792236328
```

Figure 3: This is the result when we run the program.

This is our first experiment wich have given cards : 2c,2d,3h,4d,4h,5d,6c,6h,8c,8h,9h,10c,Kh. And the result of combination from our program is :
1st line : 6c,5d,4d,3h,2c (Straight)
2nd line : 8h,8c (One Pair)
3rd line : Kh (Hight Card)

And the card that didn't see the value wich can be put in any line are : 2d,4h,6h,9h,10c

No. 3



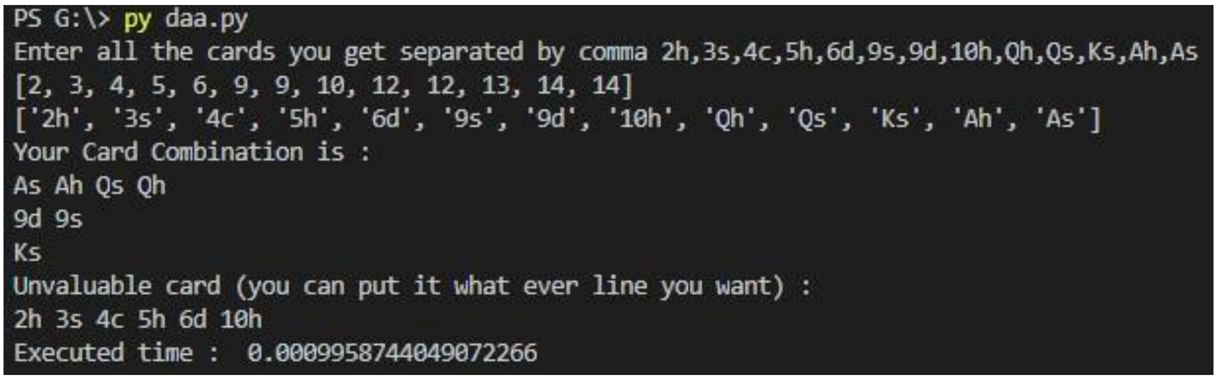
```
PS G:\> py daa.py
Enter all the cards you get separated by comma 2c,3d,4h,5d,7d,8d,9d,10d,10h,Qd,Kd,Ad,Ah
[2, 3, 4, 5, 7, 8, 9, 10, 10, 12, 13, 14, 14]
['2c', '3d', '4h', '5d', '7d', '8d', '9d', '10d', '10h', 'Qd', 'Kd', 'Ad', 'Ah']
Your Card Combination is :
Ah Ad 10h 10d
Qd
Kd
Unvaluable card (you can put it what ever line you want) :
2c 3d 4h 5d 7d 8d 9d
Executed time :  0.002245187759399414
```

Figure 4: This is the result when we run the program.

This is our first experiment wich have given cards : 2c,3d,4h,5d,7d,8d,9d,10d,10h,Qd,Kd,Ad,Ah. And the result of combination from our program is :
1st line : Ah,Ad,10h,10d (Two Pair)
2nd line : Qd (Hight Card)
3rd line : Kh (Hight Card)

And the card that didn’t see the value wich can be put in any line are : 2c,3d,4h,5d,7d,8d,9d

No. 4



```
PS G:\> py daa.py
Enter all the cards you get separated by comma 2h,3s,4c,5h,6d,9s,9d,10h,Qh,Qs,Ks,Ah,As
[2, 3, 4, 5, 6, 9, 9, 10, 12, 12, 13, 14, 14]
['2h', '3s', '4c', '5h', '6d', '9s', '9d', '10h', 'Qh', 'Qs', 'Ks', 'Ah', 'As']
Your Card Combination is :
As Ah Qs Qh
9d 9s
Ks
Unvaluable card (you can put it what ever line you want) :
2h 3s 4c 5h 6d 10h
Executed time :  0.0009958744049072266
```

Figure 5: This is the result when we run the program.

This is our first experiment wich have given cards : 2h,3s,4c,5h,6d,9s,9d,10h,Qh,Qs,Ks,Ah,As. And the result of combination from our program is :
1st line : As,Ah,Qs,Qh (Two Pair)
2nd line : 9d,9s (One Pair)
3rd line : Ks (Hight Card)

And the card that didn’t see the value wich can be put in any line are : 2h,3s,4c,5h,6d,10h

- Table Result

No	N	Times	Card
1	13	0.0010135173797607422	2h,3s,3c,5s,5c,5d,7s,10s,10h,Qc,Qs,Ad,Ah
2	13	0.0009968280792236328	2c,2d,3h,4d,4h,5d,6c,6h,8c,8h,9h,10c,Kh
3	13	0.002245187759399414	2c,3d,4h,5d,7d,8d,9d,10d,10h,Qd,Kd,Ad,Ah
4	13	0.0009958744049072266	2h,3s,4c,5h,6d,9s,9d,10h,Qh,Qs,Ks,Ah,As

Figure 6. This table is the result of the executed times and card input based on experiment.

This table is the result based on our experimental result. N is the number of card, Times is the executed times, and card is the given card for our program to execute.

- The Executed times G

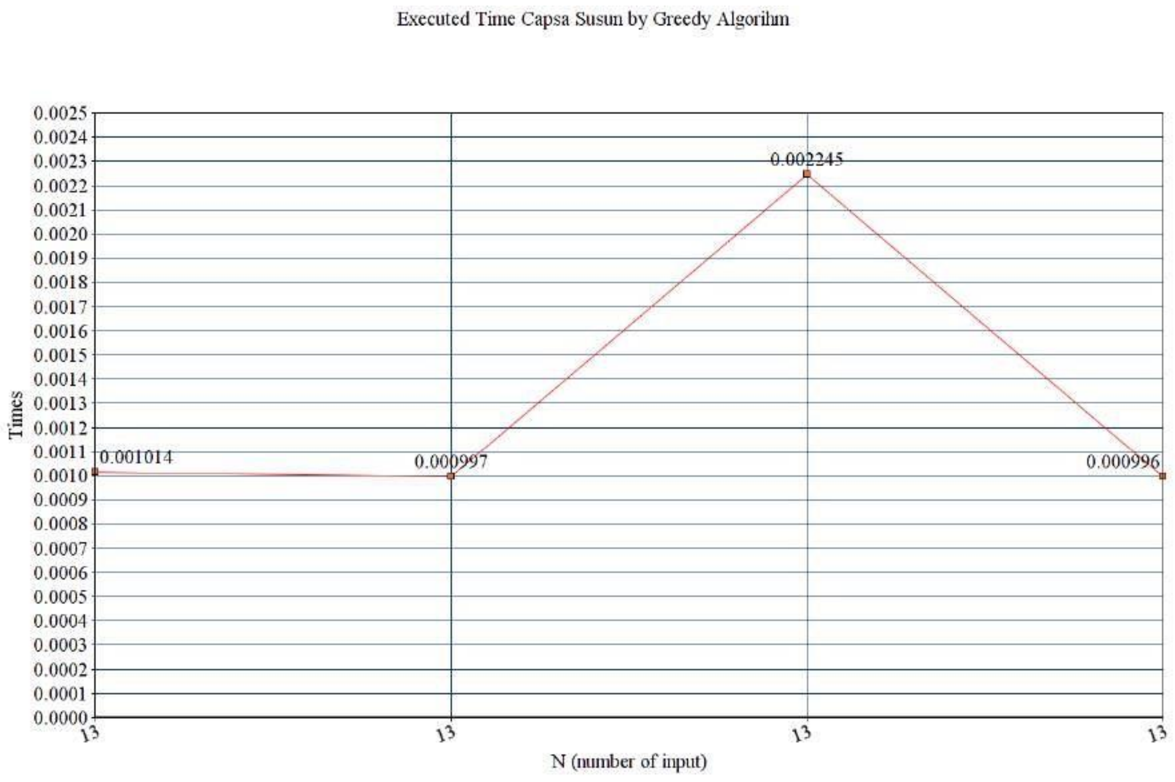


Figure 7: This is the graph of the result of the eksperiment table(on Figure 6).

4. Conclusion

In conclusion based on our 4 experiments, we can get the hight result in each line by using greedy algorihm. But it doesn't guaranteed that we will win that round. For example at experiment number 2, we have given cards are : 2c,2d,3h,4d,4h,5d,6c,6h,8c,8h,9h,10c,Kh.

And the result of combination from our program is :
1st line : 6c,5d,4d,3h,2c (Straight)
2nd line : 8h,8c (One Pair)
3rd line : Kh (Hight Card)

We can get the higher chance of winning by using this combination :
1st line : 8h,8c,6h,6c (Two Pair)
2nd line : 4d,4h,2c,2d (Two Pair)
3rd line : Kh (Hight Card)

Meaning that greedy algorihm can't guaranteed our winning situation at that round.

1. Program code (font: Courier New)

```
# 13 cards
# red  , black  , red   , black
# diamonds, clubs, hearts, spades

# value of line 1 < line 2 < line 3

#get random input at https://www.random.org/playing-cards/
# diamonds :   "2d", "3d", "4d", "5d", "6d", "7d", "8d", "9d",
"10d", "Jd", "Qd", "Kd", "Ad"
# clubs : "2c", "3c", "4c", "5c", "6c", "7c", "8c", "9c", "10c",
"Jc", "Qc", "Kc", "Ac"
# hearts :  "2h", "3h", "4h", "5h", "6h", "7h", "8h", "9h",
"10h", "Jh", "Qh", "Kh", "Ah"
# spades :   "2s", "3s", "4s", "5s", "6s", "7s", "8s", "9s",
"10s", "Js", "Qs", "Ks", "As"
 import sys
import time

# function for getting the value of each card def checkCard(card):
diamonds = ["2d", "3d", "4d", "5d", "6d", "7d", "8d", "9d", "10d",
"Jd", "Qd", "Kd", "Ad"]
    clubs = ["2c", "3c", "4c", "5c", "6c", "7c", "8c", "9c",
"10c", "Jc", "Qc", "Kc", "Ac"]
    hearts = ["2h", "3h", "4h", "5h", "6h", "7h", "8h", "9h",
"10h", "Jh", "Qh", "Kh", "Ah"]
    spades = ["2s", "3s", "4s", "5s", "6s", "7s", "8s", "9s",
"10s", "Js", "Qs", "Ks", "As"]     point =
[None] * 13     for i in range(len(card)):
if (card[i] == diamonds[0]):
            point[i] = 2         elif
(card[i] == diamonds[1]):
            point[i] = 3         elif
(card[i] == diamonds[2]):
            point[i] = 4         elif
(card[i] == diamonds[3]):
            point[i] = 5         elif
(card[i] == diamonds[4]):
            point[i] = 6         elif
(card[i] == diamonds[5]):
            point[i] = 7         elif
(card[i] == diamonds[6]):
            point[i] = 8         elif
(card[i] == diamonds[7]):
            point[i] = 9         elif
(card[i] == diamonds[8]):
            point[i] = 10         elif
(card[i] == diamonds[9]):
            point[i] = 11         elif
(card[i] == diamonds[10]):
            point[i] = 12         elif
(card[i] == diamonds[11]):
            point[i] = 13         elif
(card[i] == diamonds[12]):
            point[i] = 14         elif
(card[i] == clubs[0]):
            point[i] = 2         elif
(card[i] == clubs[1]):
```

```
            point[i] = 3         elif
(card[i] == clubs[2]):
            point[i] = 4         elif
(card[i] == clubs[3]):
            point[i] = 5         elif
(card[i] == clubs[4]):
            point[i] = 6         elif
(card[i] == clubs[5]):
            point[i] = 7         elif
(card[i] == clubs[6]):
            point[i] = 8         elif
(card[i] == clubs[7]):
            point[i] = 9         elif
(card[i] == clubs[8]):
point[i] = 10
        elif (card[i] == clubs[9]):
            point[i] = 11         elif
(card[i] == clubs[10]):
            point[i] = 12         elif
(card[i] == clubs[11]):
            point[i] = 13         elif
(card[i] == clubs[12]):
            point[i] = 14         elif
(card[i] == hearts[0]):
            point[i] = 2         elif
(card[i] == hearts[1]):
            point[i] = 3         elif
(card[i] == hearts[2]):
            point[i] = 4         elif
(card[i] == hearts[3]):
            point[i] = 5         elif
(card[i] == hearts[4]):
            point[i] = 6         elif
(card[i] == hearts[5]):
            point[i] = 7         elif
(card[i] == hearts[6]):
            point[i] = 8         elif
(card[i] == hearts[7]):
            point[i] = 9         elif
(card[i] == hearts[8]):
            point[i] = 10         elif
(card[i] == hearts[9]):
            point[i] = 11         elif
(card[i] == hearts[10]):
            point[i] = 12         elif
(card[i] == hearts[11]):
            point[i] = 13         elif
(card[i] == hearts[12]):
            point[i] = 14         elif
(card[i] == spades[0]):
            point[i] = 2         elif
(card[i] == spades[1]):
            point[i] = 3         elif
(card[i] == spades[2]):
            point[i] = 4         elif
(card[i] == spades[3]):
            point[i] = 5         elif
(card[i] == spades[4]):
```

```
            point[i] = 6         elif
(card[i] == spades[5]):
            point[i] = 7         elif
(card[i] == spades[6]):
            point[i] = 8         elif
(card[i] == spades[7]):
            point[i] = 9         elif
(card[i] == spades[8]):
            point[i] = 10
        elif (card[i] == spades[9]):
            point[i] = 11         elif
(card[i] == spades[10]):
            point[i] = 12         elif
(card[i] == spades[11]):
            point[i] = 13         elif (card[i]
== spades[12]):             point[i] = 14
else:             print("error : invalid card")
sys.exit()     return point

# check Pair in card def
checkpair(point):     pair = [None]
i = len(point) - 1     while i > 0:
j = 0         a = [None] * 2         if
point[i] == point[i-1]:             a[j]
= i             a[j+1] = i-1
pair = pair + a             i = i - 2
j = j + 2         else:             i =
i - 1
         pair.pop(0)
return pair

#check Three Of A Kind (not used) def checkThreeOfaKind(point):     j
= 0     pair = [None] * 12     i = len(point) - 1     while i > 0:
if point[i] == point[i-1] and point[i-1] == point[i-2]:
pair[j] = i             pair[j+1] = i-1             pair[j+2] = i-2
i = i - 3             j = j + 3         else:             i = i - 1

    return pair

#Check Straight def
checkstraight(point):
    j = 0
    straight = [None]
    i = len(point) - 1     while i >= 0:         prec
= i - 5         if (prec >= 0):             tanda =
True             counter = [None] * 5             p =
0             while i >= prec and tanda:
if (point[i]-1 == point[i - 1]):
counter[p] = i                     p = p + 1
i = i - 1                 elif (point[i] == point[i -
1]):                     i = i - 1
prec = prec - 1                     if (prec <= 0):
counter[p] = i
                 else:
i = i - 1                     tanda =
False
```

```
             if (counter[4] != None):
straight = straight + counter
        else :
           i = i - 1

    straight.pop(0)     return
straight

# input card def getCard():     input_string = input("Enter all the
cards you get separated by comma ")
    card  = input_string.split(",")     while (len(card) != 13):
print("invalid   number   of   card   please   input   again")
print("")
        input_string = input("Enter all the cards you get separated by
comma ")
        card  = input_string.split(",")     return
card

# find the position the card in top three def
findBiggest3(point):     position = [0,0,0]
prevpos1 = 0     prevpos2 = 0     prevbiggest1
= point[0] prevbiggest2 = point[0]
    biggest1 = point[0]     biggest2 =
point[0]     biggest3 = point[0]
         for i in range(len(point)):
         if (biggest1 < point[i]):
             biggest1 = point[i]
position[0] = i
             if (biggest2 < biggest1):
                 prevbiggest1 = biggest2
prevpos1 = position[1]
                 biggest2 = biggest1
position[1] = position[0]

                biggest1 = prevbiggest1
position[0] = prevpos1
                 if (biggest3 < biggest2):
                     if (biggest3 < biggest1):
prevbiggest1 = biggest3
prevpos1 = biggest3

                        biggest3 = biggest1
position[2] = position[0]

                        biggest1 = prevbiggest1
position[0] = prevpos1

                     prevbiggest2 = biggest3
prevpos2 = position[2]

                     prevbiggest1 = biggest2
prevpos1 = position[1]

                     biggest3 = biggest2
position[2] = position[1]
                      biggest2 = prevbiggest2
position[1] = prevpos2
```

```
     return position

# find the position the card in top two def
findBiggest2(point):
    position = [0,0] prevpos1 = 0
    prevbiggest1 = point[0]
    biggest1 = point[0]
    biggest2 = point[0]
         for i in range(len(point)):
         if (biggest1 < point[i]):
             biggest1 = point[i]
position[0] = i
             if (biggest2 < biggest1):
                 prevbiggest1 = biggest2
prevpos1 = position[1]
                 biggest2 = biggest1
position[1] = position[0]

                biggest1 = prevbiggest1
position[0] = prevpos1

    return position

# find the position the card in top one def
findBiggest1(point):     position = [0]
prevbiggest1 = point[0]     biggest1 =
point[0]
         for i in range(len(point)):
         if (biggest1 < point[i]):

            biggest1 = point[i]
position[0] = i
     return position

# Process of output Card def
Calculation(point, card):     start
= time.time()
     line1, line2, line3 = False, False, False
Straight = checkstraight(point)     check = 0
while (Straight != []):         i = 0
counter = 0         while counter < 5:
print(card[Straight[i]], end=' ')
card.pop(Straight[i])
point.pop(Straight[i])             Straight.pop(i)
counter = counter + 1
                 check = check
+ 1         print("")
     if (check == 1):
line3 = True     elif (check
== 2):         line3 = True
line2 = True
         if (line3 == False):         Pair =
checkpair(point)         if (Pair != []):
counter = 0             while (Pair != []):
i = 0                 j = 0                 while (j <
4 and Pair != []):
print(card[Pair[i]], end=' ')
```

```
card.pop(Pair[i])
point.pop(Pair[i])                       Pair.pop(i)
j = j + 1                     counter = counter + 1
print("")                          if (counter <= 4 and
counter > 0):
                line3 = True             elif (counter
<= 8 and counter > 0):                 line3 = True
line2 = True             elif (counter <= 10):
line1 = True                 line2 = True
line3 = True
                 if (line3 == False):
position = findBiggest3(point)
print(card[position[0]])
print(card[position[1]])
print(card[position[2]])
card.pop(position[0])
card.pop(position[1]-1)
card.pop(position[2]-2)         elif (line2 ==
False):             position =
findBiggest2(point)
print(card[position[0]])
print(card[position[1]])
card.pop(position[0])
card.pop(position[1]-1)         elif (line1 ==
False):             position =
findBiggest1(point)
print(card[position[0]])
card.pop(position[0])
         elif (line2 == False):         Pair =
checkpair(point)         if (Pair != []):
counter = 0             while (Pair != []):
i = 0                 j = 0                 while (j <
4 and Pair != []):
print(card[Pair[i]], end=' ')
card.pop(Pair[i])
point.pop(Pair[i])                       Pair.pop(i)
j = j + 1                     counter = counter + 1

print("")
             if (counter <= 4 and counter > 0):
                line2 = True             elif (counter
<= 8 and counter > 0):                 line2 = True
line1 = True
                 if (line2 == False):
position = findBiggest2(point)
print(card[position[0]])
print(card[position[1]])
card.pop(position[0])
card.pop(position[1]-1)         elif (line1 ==
False):             position =
findBiggest1(point)
print(card[position[0]])
card.pop(position[0])
         else:
        Pair = checkpair(point)         if (Pair !=
[]):             counter = 0             while (Pair !=
[]):                 i = 0                 j = 0
while (j < 4 and Pair != []):
```

```
print(card[Pair[i]], end=' ')
card.pop(Pair[i])
point.pop(Pair[i])
                    Pair.pop(i)
j = j + 1                     counter = counter
+ 1                 print("")
                         if (counter <= 4 and
counter > 0):                 line1 = True
                 if (line1 == False):
position = findBiggest1(point)
print(card[position[0]])
card.pop(position[0])

    end = time.time()

    print("Unvaluable card (you can put it what ever line you want)
:")     while (card != []):         print(card[0], end=' ')
card.pop(0)
     print("")
    print("Executed time : ", end - start)

# print the highest card that possible on line 1, 2, 3 (not used) def
printPosition(card, position):
    print("line 1 : ", card[position[0]])     print("line 2
: ", card[position[1]])     print("line 3 : ",
card[position[2]])
 def main():
    global card      global point
card = getCard()     point =
checkCard(card)     print(point)
print(card)
    print("Your Card Combination is : ")
    Calculation(point, card)
 main()
```

-