

22、斗地主顺子，考点 or 实现——数据结构/栈

题目描述

在斗地主扑克牌游戏中，扑克牌由小到大的顺序为：3,4,5,6,7,8,9,10,J,Q,K,A,2，玩家可以出的扑克牌阵型有：单张、对子、顺子、飞机、炸弹等。

其中顺子的出牌规则为：由至少5张由小到大连续递增的扑克牌组成，且不能包含2。

例如：{3,4,5,6,7}、{3,4,5,6,7,8,9,10,J,Q,K,A}都是有效的顺子；而{J,Q,K,A,2}、{2,3,4,5,6}、{3,4,5,6}、{3,4,5,6,8}等都不是顺子。

给定一个包含13张牌的数组，如果有满足出牌规则的顺子，请输出顺子。

如果存在多个顺子，请每行输出一个顺子，且需要按顺子的第一张牌的大小（必须从小到大）依次输出。

如果没有满足出牌规则的顺子，请输出No。

输入描述

13张任意顺序的扑克牌，每张扑克牌数字用空格隔开，每张扑克牌的数字都是合法的，并且不包括大小王：2 9 J 2 3 4 K A 7 9 A 5 6

不需要考虑输入为异常字符的情况

输出描述

组成的顺子，每张扑克牌数字用空格隔开：3 4 5 6 7

用例

输入	2 9 J 2 3 4 K A 7 9 A 5 6
输出	3 4 5 6 7
说明	13张牌中，可以组成的顺子只有1组：3 4 5 6 7。

输入	2 9 J 10 3 4 K A 7 Q A 5 6
输出	3 4 5 6 7 9 10 J Q K A
说明	13张牌中，可以组成2组顺子，从小到大分别为：3 4 5 6 7 和 9 10 J Q K A

输入	2 9 9 9 3 4 K A 10 Q A 5 6
输出	No
说明	13张牌中，无法组成顺子。

题目解析

先看一个例子：

3 4 5 6 6 7 7 8 9 10

这个例子该输出什么呢？

如果是优先最长顺子的话，那么应该输出：

```
3 4 5 6 7 8 9 10
```

如果是优先最多顺子的话，那么应该输出：

```
3 4 5 6 7  
6 7 8 9 10
```

经过考友实际机试验证，本题应该优先输出最多的顺子。

我的解题策略如下：

首先定义一个长度为15的数组count，分别记录2~A扑克牌的数量，其中2~A扑克牌分别对应索引2~14。

再定义一个total变量，记录总牌数。

然后，开始找五张顺子，查找索引i范围依次是：3~7，4~8，5~9，6~10，...，10~14

如果对应范围的每一个count[i]都大于0，则对应范围可以形成五张顺子，只要形成五张顺子，则

- total -= 5
- 范围内每一个 count[i] -= 1

当把五张顺子找完后，继续检查total是否大于0，若total > 0，则说明还有剩余牌，此时我们外层遍历每一个顺子，内层遍历每一个剩余牌，如果剩余牌可以追加到顺子尾巴，形成新顺子，则更新顺子尾巴。

最后，打印每一个顺子即可。

注意，由于我们找五张顺子时，是按照从左到右查找的，因此顺子找出来后就是符合要求顺序的，因此不需要额外排序。

## 优先最多顺子解法

### Java算法源码

```
1 import java.util.ArrayList;
2 import java.util.Scanner;
3 import java.util.StringJoiner;
4
5 public class Main {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         String[] cards = sc.nextLine().split(" ");
9         getResult(cards);
10    }
11
12    public static void getResult(String[] cards) {
13        // 总牌数
14        int total = cards.length;
15
16        // 记录每张牌的数量
17        int[] count = new int[15];
18        for (String card : cards) {
19            int num = mapToNum(card);
20            count[num]++;
21        }
22
23        // 记录顺子，顺子表示为数组，数组含义：[起始牌，结束牌]
24        ArrayList<int[]> straights = new ArrayList<>();
25
26        for (int i = 3; i <= 10; i++) {
```

```

26     for (int i = 3; i <= 10; i++) {
27         // 先求五张的顺子
28         if (isStraight(count, i, i + 4)) {
29             // 如果可以组成五张顺子, 则记录该顺子
30             straights.add(new int[] {i, i + 4});
31             // 总牌数减5
32             total -= 5;
33             // 对应的牌的数量减1
34             for (int j = i; j <= i + 4; j++) count[j]--;
35         }
36     }
37
38     // 如果五张的顺子求解完后, 还有剩余牌
39     if (total > 0) {
40         // 则尝试将剩余牌追加到五张顺子后面
41         for (int[] straight : straights) {
42             for (int i = 3; i <= 14; i++) {
43                 if (count[i] > 0 && i - straight[1] == 1) {
44                     straight[1] = i;
45                     count[i]--;
46                 }
47             }
48         }
49     }
50
51     // 如果没有顺子, 则返回No
52     if (straights.size() == 0) {
53         System.out.println("No");
54         return;
55     }

```

```

55     }
56
57     // 如果有顺子, 则打印顺子
58     for (int[] straight : straights) {
59         int start = straight[0];
60         int end = straight[1];
61
62         StringJoiner sj = new StringJoiner(" ");
63         for (int i = start; i <= end; i++) sj.add(mapToCard(i));
64         System.out.println(sj);
65     }
66 }
67
68 public static boolean isStraight(int[] count, int start, int end) {
69     int i = start;
70     for (; i <= end; i++) {
71         if (count[i] == 0) break;
72     }
73     return i > end;
74 }
75
76 public static int mapToNum(String card) {
77     switch (card) {
78         case "J":
79             return 11;
80         case "Q":
81             return 12;
82         case "K":
83             return 13;
84         case "A":

```

```

84     case "A":
85         return 14;
86     default:
87         return Integer.parseInt(card);
88     }
89 }
90
91 public static String mapToCard(int num) {
92     switch (num) {
93         case 11:
94             return "J";
95         case 12:
96             return "Q";
97         case 13:
98             return "K";
99         case 14:
100             return "A";
101         default:
102             return num + "";
103     }
104 }
105 }

```

## JS算法源码

```

1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5      input: process.stdin,
6      output: process.stdout,
7  });
8
9  rl.on("line", (line) => {
10     const cards = line.split(" ");
11     getResult(cards);
12 });
13
14 function getResult(cards) {
15     // 总牌数
16     let total = cards.length;
17
18     // 记录每张牌的数量
19     const count = new Array(15).fill(0);
20     for (let card of cards) {
21         const num = mapToNum(card);
22         count[num]++;
23     }
24
25     // straights记录顺子，其元素是顺子，顺子表示为数组：[起始牌，结束牌]
26     const straights = [];
27
28     // i 表示顺子的起始牌取值

```

```

28 // i 表示顺子的起始牌取值
29 for (let i = 3; i <= 10; i++) {
30     // 先求五张的顺子
31     if (isStraights(count, i, i + 4)) {
32         // 如果可以组成五张顺子, 则记录该顺子
33         straights.push([i, i + 4]);
34         // 总牌数减5
35         total -= 5;
36         // 对应的牌的数量减1
37         for (let j = i; j <= i + 4; j++) count[j]--;
38     }
39 }
40
41 // 如果五张的顺子求解完后, 还有剩余牌
42 if (total > 0) {
43     // 则尝试将剩余牌追加到五张顺子后面
44     for (let straight of straights) {
45         for (let i = 3; i <= 14; i++) {
46             if (count[i] > 0 && i - straight[1] == 1) {
47                 straight[1] = i;
48                 count[i]--;
49             }
50         }
51     }
52 }
53
54 // 如果没有顺子, 则返回No
55 if (straights.length == 0) {
56     console.log("No");

```

```

56     console.log("No");
57     return;
58 }
59
60 // 如果有顺子, 则打印顺子
61 for (let [start, end] of straights) {
62     const ans = [];
63     for (let i = start; i <= end; i++) ans.push(mapToCard(i));
64     console.log(ans.join(" "));
65 }
66 }
67
68 function isStraights(count, start, end) {
69     let i = start;
70     for (; i <= end; i++) {
71         if (count[i] == 0) break;
72     }
73     return i > end;
74 }
75
76 function mapToNum(card) {
77     switch (card) {
78         case "J":
79             return 11;
80         case "Q":
81             return 12;
82         case "K":
83             return 13;
84         case "A":
85             return 14;

```

```

85     return 14;
86     default:
87         return parseInt(card);
88     }
89 }
90
91 function mapToCard(num) {
92     switch (num) {
93         case 11:
94             return "J";
95         case 12:
96             return "Q";
97         case 13:
98             return "K";
99         case 14:
100             return "A";
101         default:
102             return num + "";
103     }
104 }

```

## Python算法源码

```

1  # 输入获取
2  cards = input().split()
3
4
5  def mapToCard(num):
6      if num == 11:
7          return "J"
8      elif num == 12:
9          return "Q"
10     elif num == 13:
11         return "K"
12     elif num == 14:
13         return "A"
14     else:
15         return str(num)
16
17
18  def mapToNum(card):
19     if card == "J":
20         return 11
21     elif card == "Q":
22         return 12
23     elif card == "K":
24         return 13
25     elif card == "A":
26         return 14
27     else:
28         return int(card)

```

```

28         return int(card)
29
30
31 def isStraights(count, start, end):
32     i = start
33     while i <= end:
34         if count[i] == 0:
35             break
36         else:
37             i += 1
38     return i > end
39
40
41 # 算法入口
42 def getResult():
43     # 总牌数
44     total = len(cards)
45
46     # 记录每张牌的数量
47     count = [0] * 15
48     for card in cards:
49         num = mapToNum(card)
50         count[num] += 1
51
52     # 记录顺子, 顺子表示为数组, 数组含义: [起始牌, 结束牌]
53     straights = []
54
55     for i in range(3, 11):
56         # 先求五张的顺子
57         if isStraights(count, i, i + 4):

```

```

58         # 如果可以组成五张顺子, 则记录该顺子
59         straights.append([i, i + 4])
60         # 总牌数减5
61         total -= 5
62         # 对应的牌的数量减1
63         for j in range(i, i + 5):
64             count[j] -= 1
65
66     # 如果五张的顺子求解完后, 还有剩余牌
67     if total > 0:
68         # 则尝试将剩余牌追加到五张顺子后面
69         for straight in straights:
70             for i in range(3, 15):
71                 if count[i] > 0 and i - straight[1] == 1:
72                     straight[1] = i
73                     count[i] -= 1
74
75     # 如果没有顺子, 则返回No
76     if len(straights) == 0:
77         print("No")
78         return
79
80     # 如果有顺子, 则打印顺子
81     for start, end in straights:
82         ans = [mapToCard(num) for num in range(start, end + 1)]
83         print(" ".join(ans))
84
85
86 # 调用算法
87 getResult()

```

## 优先最长顺子解法 (80%通过率)

### Java算法源码

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.LinkedList;
4 import java.util.Scanner;
5
6 public class Main {
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9         LinkedList<String> cards = new LinkedList<>(Arrays.asList(sc.nextLine().split(" ")));
10        getResult(cards);
11    }
12
13    public static void getResult(LinkedList<String> cards) {
14        cards.sort((a, b) -> map(a) - map(b));
15
16        LinkedList<String> stack = new LinkedList<>();
17        ArrayList<String> ans = new ArrayList<>();
18
19        while (cards.size() > 0) {
20            String bot = cards.removeFirst();
21
22            if (stack.size() == 0) {
23                stack.add(bot);
24                continue;
25            }
```

```
26        }
27        String top = stack.getLast();
28        if (map(bot) - map(top) == 1) {
29            stack.add(bot);
30        } else if (map(bot) == map(top)) {
31            cards.add(bot);
32        } else {
33            if (stack.size() >= 5) ans.add(String.join(" ", stack));
34            stack.clear();
35            stack.add(bot);
36        }
37    }
38
39    if (stack.size() >= 5) ans.add(String.join(" ", stack));
40
41    if (ans.size() == 0) {
42        System.out.println("No");
43    } else {
44        ans.sort((a, b) -> map(a.charAt(0) + "") - map(b.charAt(0) + ""));
45        for (String an : ans) {
46            System.out.println(an);
47        }
48    }
49 }
50
51 public static int map(String card) {
52     switch (card) {
53         case "J":
54             return 11;
```



```

55     case "Q":
56         return 12;
57     case "K":
58         return 13;
59     case "A":
60         return 14;
61     case "2":
62         return 16;
63     default:
64         return Integer.parseInt(card);
65     }
66 }
67 }

```

## JS算法源码

```

1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5      input: process.stdin,
6      output: process.stdout,
7  });
8
9  rl.on("line", (line) => {
10     const cards = line.split(" ");
11     getResult(cards);
12 });
13
14 function getResult(cards) {
15     cards.sort((a, b) => map(a) - map(b));
16
17     const stack = [];
18     const ans = [];
19
20     while (cards.length > 0) {
21         const bot = cards.shift();
22
23         if (stack.length == 0) {
24             stack.push(bot);
25             continue;
26         }
27
28         const top = stack.at(-1);

```

```

28     const top = stack.at(-1);
29     if (map(bot) - map(top) == 1) {
30         stack.push(bot);
31     } else if (map(bot) == map(top)) {
32         cards.push(bot);
33     } else {
34         if (stack.length >= 5) ans.push(stack.join(" "));
35         stack.length = 0;
36         stack.push(bot);
37     }
38 }
39
40 if (stack.length >= 5) ans.push(stack.join(" "));
41
42 if (ans.length == 0) {
43     console.log("No");
44 } else {
45     ans.sort((a, b) => map(a[0]) - map(b[0])).forEach((s) => console.log(s));
46 }
47 }
48
49 function map(card) {
50     switch (card) {
51         case "J":
52             return 11;
53         case "Q":
54             return 12;
55         case "K":
56             return 13;
57         case "A":

```

```

57         case "A":
58             return 14;
59         case "2":
60             return 16;
61         default:
62             return parseInt(card);
63     }
64 }

```

## Python算法源码

```
1  # 输入获取
2  cards = input().split()
3
4
5  def map(card):
6      if card == "J":
7          return 11
8      elif card == "Q":
9          return 12
10     elif card == "K":
11         return 13
12     elif card == "A":
13         return 14
14     elif card == "2":
15         return 16
16     else:
17         return int(card)
18
19
20 # 算法入口
21 def getResult():
22     cards.sort(key=lambda x: map(x))
23
24     stack = []
25     ans = []
26
27     while len(cards) > 0:
28         bot = cards.pop(0)
```

```
28     bot = cards.pop(0)
29
30     if len(stack) == 0:
31         stack.append(bot)
32         continue
33
34     top = stack[-1]
35     if map(bot) - map(top) == 1:
36         stack.append(bot)
37     elif map(bot) == map(top):
38         cards.append(bot)
39     else:
40         if len(stack) >= 5:
41             ans.append(" ".join(stack))
42             stack.clear()
43             stack.append(bot)
44
45     if len(stack) >= 5:
46         ans.append(" ".join(stack))
47
48     if len(ans) == 0:
49         print("No")
50     else:
51         ans.sort(key=lambda x: map(x[0]))
52         for an in ans:
53             print(an)
54
55
```

```
51         ans.sort(key=lambda x: map(x[0]))
52         for an in ans:
53             print(an)
54
55
56 # 调用算法
57 getResult()
```