题目描述

五张牌，每张牌由牌大小和花色组成，牌大小2~10、J、Q、K、A，牌花色为红桃、黑桃、梅花、方块四种花色之一。

判断牌型:

牌型1，同花顺：同一花色的顺子，如红桃2红桃3红桃4红桃5红桃6。
牌型2，四条：四张相同数字 + 单张，如红桃A黑桃A梅花A方块A + 黑桃K。
牌型3，葫芦：三张相同数字 + 一对，如红桃5黑桃5梅花5 + 方块9梅花9。
牌型4，同花：同一花色，如方块3方块7方块10方块J方块Q。
牌型5，顺子：花色不一样的顺子，如红桃2黑桃3红桃4红桃5方块6。
牌型6，三条：三张相同+两张单。

说明：

（1）五张牌里不会出现牌大小和花色完全相同的牌。
（2）编号小的牌型较大，如同花顺比四条大，依次类推。
（3）包含A的合法的顺子只有10 J Q K A和A 2 3 4 5;类似K A 2 3 4的序列不认为是顺子。

输入描述

输入由5行组成，每行为一张牌大小和花色，牌大小为2~10、J、Q、K、A，花色分别用字符H、S、C、D表示红桃、黑桃、梅花、方块。

输出描述

输出牌型序号，5张牌符合多种牌型时，取最大的牌型序号输出。

用例

输入	4 H 5 S 6 C 7 D 8 D
输出	5
说明	4 5 6 7 8构成顺子，输出5

输入	9 S 5 S 6 S 7 S 8 S
输出	1
说明	既是顺子又是同花，输出1，同花顺

题目解析

这道题应该是一道逻辑题，这道题目输入了五张牌，然后我们需要对这五张牌进行六种情况的分析，并且分析有优先级：

1. 是否为同花顺
2. 是否为四条
3. 是否为葫芦
4. 是否同花
5. 是否为顺子
6. 是否为三条

若满足了前面，则后面的就不需要再判断了。

另外，同花顺情况判断，其实就是 同花 + 顺子，因此在写代码时可以再拆下代码。

上面六种情况的判断，其实可以将五张牌的大小和花色分开来，分别检查。因此我将输入的五张牌的大小存入nums数组，花色存入colors数组。

其中顺子的判断，其实不关心花色，只关心牌大小，我们只需要为牌定义好大小后，进行大小升序排序，若后面一张牌的大小总是比其前面一张牌大1，则就是顺子。

另外还一个特殊情况，即'A2345'也是顺子，这里我们给A定义的大小是14，因此nums升序排序后，只需要判断nums.join('')是否和'2345A'相等，即可判定为顺子。

而四条的判断，也不关心花色，只关系牌大小，四条即 四个相同牌大小，一个不同牌大小。这个很好判断，我这里的方案是，new Set(nums)来去重，若去重后不是两张牌，那么就肯定不是四条，若有两张牌，则任取一张，看在nums中有几个，若为1个或4个，则可以判定为四条。

葫芦、三条的判定和四条类似。

同花的判定，即new Set(colors)，若去重后只有一个花色，则判定为同花。

另外，本题没有说如果输入的五张牌都不满足上面六种情况时，该输出啥，一个可能是，用例保证输入的五张牌肯定满足上面六种情况之一，因此不需要考虑这种异常场景。一个可能是，出题人遗漏了这种情况的输出说明，我这里把不满足上面六种情况的输出自定义为0.

Java算法源码

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] nums = new String[5];
    String[] colors = new String[5];

    for (int i = 0; i < 5; i++) {
      nums[i] = sc.next();
      colors[i] = sc.next();
    }

    System.out.println(getResult(nums, colors));
  }

```

```java
  public static int getResult(String[] nums, String[] colors) {
    Arrays.sort(nums, (a, b) -> cards(a) - cards(b));

    if (isShunzi(nums) && isTonghua(colors)) return 1;
    else if (isSitiao(nums)) return 2;
    else if (isHulu(nums)) return 3;
    else if (isTonghua(colors)) return 4;
    else if (isShunzi(nums)) return 5;
    else if (isSantiao(nums)) return 6;
    else return 0;
  }

  // 牌大小 映射为 数值
  public static int cards(String num) {
    switch (num) {
      case "J":
        return 11;
      case "Q":
        return 12;
      case "K":
        return 13;
      case "A":
        return 14;
      default:
        return Integer.parseInt(num);
    }
  }

  // 顺子
  public static boolean isShunzi(String[] nums) {
    if ("2345A".equals(String.join("", nums))) return true;

    for (int i = 1; i < nums.length; i++) {
      int num1 = cards(nums[i - 1]);
      int num2 = cards(nums[i]);
      if (num1 + 1 != num2) return false;
    }

    return true;
  }

  // 同花
  public static boolean isTonghua(String[] colors) {
    // 同花牌的所有花色都一样
    return new HashSet<String>(Arrays.asList(colors)).size() == 1;
  }

  // 四条
  public static boolean isSitiao(String[] nums) {
    // 四条由两部分组成，一个部分四张相同牌，一个部分一张牌
    return countNums(nums, 2, 4);
  }

  // 葫芦
```

```java
  // 葫芦
  public static boolean isHulu(String[] nums) {
    // 葫芦由两部分组成，一个部分三张牌相同，一个部分两张牌相同
    return countNums(nums, 2, 3);
  }

  // 三条
  public static boolean isSantiao(String[] nums) {
    // 三条由三部分组成，第一个部分由三张相同牌组成，第二个、第三个部分分别是两种不同的牌
    return countNums(nums, 3, 3);
  }

  private static boolean countNums(String[] nums, int partCount, int maxSameNumCount) {
    HashMap<String, Integer> count = new HashMap<>();

    for (String num : nums) {
      count.put(num, count.getOrDefault(num, 0) + 1);
    }

    if (count.keySet().size() != partCount) return false;

    return count.containsValue(maxSameNumCount);
  }
}
```

JS算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 5) {
    const arr = lines.map((line) => line.split(" "));
    console.log(getResult(arr));
    lines.length = 0;
  }
});

function getResult(arr) {
  const nums = [];
  const colors = [];

  for (let [num, color] of arr) {
    nums.push(num);
    colors.push(color);
  }

  nums.sort((a, b) => cards(a) - cards(b));

```

```
  if (isShunzi(nums) && isTonghua(colors)) return 1;
  else if (isSitiao(nums)) return 2;
  else if (isHulu(nums)) return 3;
  else if (isTonghua(colors)) return 4;
  else if (isShunzi(nums)) return 5;
  else if (isSantiao(nums)) return 6;
  else return 0;
}

function cards(num) {
  switch (num) {
    case "J":
      return 11;
    case "Q":
      return 12;
    case "K":
      return 13;
    case "A":
      return 14;
    default:
      return parseInt(num);
  }
}

```

```
// 顺子
function isShunzi(nums) {
  if (nums.join("") === "2345A") return true;

  for (let i = 1; i < nums.length; i++) {
    const num1 = cards(nums[i - 1]);
    const num2 = cards(nums[i]);

    if (num1 + 1 !== num2) return false;
  }
  return true;
}

// 同花
function isTonghua(colors) {
  // 同花牌的所有花色都一样
  return new Set(colors).size === 1;
}

// 四条
function isSitiao(nums) {
  // 四条由两部分组成，一个部分四张相同牌，一个部分一张牌
  return countNums(nums, 2, 4);
}

```

```javascript
// 葫芦
function isHulu(nums) {
  // 葫芦由两部分组成，一个部分三张牌相同，一个部分两张牌相同
  return countNums(nums, 2, 3);
}

// 三条
function isSantiao(nums) {
  // 三条由三部分组成，第一个部分由三张相同牌组成，第二个、第三个部分分别是两种不同的牌
  return countNums(nums, 3, 3);
}

function countNums(nums, partCount, maxSameNumCount) {
  const count = {};

  for (let num of nums) {
    count[num] = (count[num] ?? 0) + 1;
  }

  if (Object.keys(count).size != partCount) return false;

  return Object.values(count).includes(maxSameNumCount);
}
```

Pyhon算法源码

```python
# 输入获取
arr = [input().split() for _ in range(5)]


# 牌大小 映射为 数值
def cards(num):
    if num == "J":
        return 11
    elif num == "Q":
        return 12
    elif num == "K":
        return 13
    elif num == "A":
        return 14
    else:
        return int(num)


```

```python
def countNums(nums, partCount, maxSameNumCount):
    count = {}

    for num in nums:
        if count.get(num) is None:
            count[num] = 0
        count[num] += 1

    if len(count.keys()) != partCount:
        return False

    return maxSameNumCount in count.values()


# 三条
def isSantiao(nums):
    # 三条由三部分组成，第一个部分由三张相同牌组成，第二个，第三个部分分别是两种不同的牌
    return countNums(nums, 3, 3)


# 葫芦
def isHulu(nums):
    # 葫芦由两部分组成，一个部分三张牌相同，一个部分两张牌相同
    return countNums(nums, 2, 3)


```

```python
# 四条
def isSitiao(nums):
    # 四条由两部分组成，一个部分四张相同牌，一个部分一张牌
    return countNums(nums, 2, 4)


# 同花
def isTonghua(colors):
    # 同花牌的所有花色都一样
    return len(set(colors)) == 1


# 顺子
def isShunzi(nums):
    if "".join(nums) == "2345A":
        return True

    for i in range(1, len(nums)):
        if cards(nums[i - 1]) + 1 != cards(nums[i]):
            return False
        return True


```

```python
# 算法入口
def getResult():
    nums = []
    colors = []

    for num, color in arr:
        nums.append(num)
        colors.append(color)

    nums.sort(key=lambda x: cards(x))

    if isShunzi(nums) and isTonghua(colors):
        return 1
    elif isSitiao(nums):
        return 2
    elif isHulu(nums):
        return 3
    elif isTonghua(colors):
        return 4
    elif isShunzi(nums):
        return 5
    elif isSantiao(nums):
        return 6
    else:
        return 0


# 算法调用
print(getResult())
```