最佳的出牌方法题目描述：

手上有一副扑克牌，每张牌按牌面数字记分（J=11,Q=12,K=13，没有大小王），出牌时按照以下规则记分：

■ 出单张，记牌面分数，例如出一张 2，得分为 2

■ 出对或 3 张，记牌面分数总和再 x2，例如出 3 张 3，得分为(3+3+3)x2=18

■ 出 5 张顺，记牌面分数总和再 x2，例如出 34567 顺，得分为(3+4+5+6+7)x2=50

■ 出 4 张炸弹，记牌面分数总和再 x3，例如出 4 张 4，得分为 4x4x3=48

求出一副牌最高的得分数

输入描述：

按顺序排好的一副牌，最少 1 张，最多 15 张。

1-9 输入为数字 1-9，10 输入为数字 0，JQK 输入为大写字母 JQK。

无需考虑输入非法的情况，例如输入字符不在[0-9JQK]范围或某一张牌超过 4 张。

输出描述：

最高的得分数

补充说明：

积分规则中没有的出牌方式不支持，例如不支持 3 带 1、4 带 2，不支持 5 张以上的顺，且 10JQKA（0JQK1）不算顺。

示例 1

输入：

```
33445677
```

输出：

```
67
```

说明：

出对 3、对 4、对 7，单张 5、6，得分为 67；出 34567 顺，再出单张 3、4、7，得分为 64

因此最高得分是按对出，可得到最高分 67，输出结果 67

代码报告

```
import java.util.*;


// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    private static Map<Integer, Integer> map = new HashMap<Integer, Integer>();
    private static Map<Integer, Integer> lastNumMap = new HashMap<Integer, Integer>();
    private static Map<Character, Integer> valueMap = new
    HashMap<Character, Integer>();
    private static int finalState = 0, ans = 0;

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        for (int i = 1; i <= 9; i++) {
            valueMap.put((char)('0' + i), i);
        }
```

```
valueMap.put('0', 10);
valueMap.put('J', 11);
valueMap.put('Q', 12);
valueMap.put('K', 13);
// 注意 hasNext 和 hasNextLine 的区别
while (in.hasNextLine()) { // 注意 while 处理多个 case
    ans = 0;
    finalState = 0;
    map = new HashMap<Integer, Integer>();
    lastNumMap = new HashMap<Integer, Integer>();
    String s = in.nextLine();
    char[] arr = new char[s.length()];
    for (int i = 0; i < arr.length; i++) {
        arr[i] = s.charAt(i);
        finalState |= (1 << i);
    }
    int[] arrInt = new int[arr.length];
    for (int i = 0; i < arr.length; i++) {
        arrInt[i] = valueMap.get(arr[i]);
    }
    traceback(0, arrInt, 0);
    System.out.println(ans);
```

```
        }
    }
    private static void traceback(int state, int[] arr, int total) {
        if (state == finalState) {
            ans = Math.max(ans, total);
        }
        int value = map.getOrDefault(state, 0);
        if (value != 0 && total > value) {
            map.put(state, total);
        } else if (value != 0) {
            return;
        }else {
            map.put(state, total);
        }
        int used = 0;
        // 五张
        String s = "";
        int next = 1;
        for (int i = 0; i < arr.length - 4; i++) {
            used = 1 << i;
            int cnt = 1, last = arr[i], index = i + 1;
            StringBuffer sb = new StringBuffer().append(last + " ");
```

```
        while (cnt < 5 && index < arr.length) {
            if (arr[index] - 1 == last) {
                cnt++;
                last = arr[index];
                used |= 1 << index;
                sb.append(last + " ");
                if (cnt == 2) {
                    next = index;
                }
            }
            index++;
        }
        if (cnt == 5 && (state & used) == 0) {
            if (s.equals(sb.toString())) {
                map.put(state | used, total + arr[i] * 10 + 20);
            } else {
                s = sb.toString();
                traceback(state | used, arr, total + arr[i] * 10 + 20);
            }
            i = next;
        }
    }
```

```
// 炸弹
for (int i = 0; i < arr.length - 3; i++) {
    used = 1 << i;
    used |= 1 << (i + 1);
    used |= 1 << (i + 2);
    used |= 1 << (i + 3);
    if ((state & used) == 0 &&
            arr[i] == arr[i + 1] && arr[i] == arr[i + 2] &&
            arr[i] == arr[i + 3]
       ) {
        traceback(state | used, arr, total + arr[i] * 12);
    }
}
// 三张
int last = -1;
for (int i = 0; i < arr.length - 2; i++) {
    used = 1 << i;
    used |= 1 << (i + 1);
    used |= 1 << (i + 2);
    if ((state & used) == 0 &&
            arr[i] == arr[i + 1] && arr[i] == arr[i + 2]) {
        if (arr[i] == last){
```

```
                map.put(state | used, total + arr[i] * 6);
            }else{
                last = arr[i];
                traceback(state | used, arr, total + arr[i] * 6);
            }
        }
    }
    // 对子
    last = -1;
    for (int i = arr.length-1; i > 0; i--)  {
        used = 1 << i;
        used |= 1 << (i - 1);
        if ((state & used) == 0 && arr[i] == arr[i - 1]) {
            if (arr[i] == last){
                map.put(state | used, total + arr[i] * 4);
            }else{
                last = arr[i];
                traceback(state | used, arr, total + arr[i] * 4);
            }
        }
    }
    // 单张
```

```
        for (int i = arr.length-1; i > -1; i--) {
            used = 1 << i;
            if ((state & used) == 0) {
                traceback(state | used, arr, total + arr[i]);
            }
        }
    }
}
```