原题链接

蓝桥杯-无聊的逗

问题描述

逗志芃在干了很多事情后终于闲下来了，然后就陷入了深深的无聊中。不过他想到了一个游戏来使他更无聊。他拿出 n 个木棍，然后选出其中一些粘成一根长的，然后再选一些粘成另一个长的，他想知道在两根一样长的情况下长度最长是多少。

输入格式

第一行一个数 n，表示 n 个棍子。第二行 n 个数，每个数表示一根棍子的长度。

输出格式

一个数，最大的长度。

样例输入

```
4
1 2 3 1
```

样例输出

```
3
```

数据规模与约定

$n \leq 15$

解题思路

范围是 $n \leq 15$, 因此我们可以采用状态压缩的方式来求解此题。

简单介绍一下状态压缩：用二进制来表示状态，即一堆 01 的二进制代码，通常用 1 表示该位置被使用，0 表示该位置未被使用。

举个例子：对于样例，我们选择 0号位置和 1 号位置，那么该状态就是 0011，如果选择 1 号位置和 2号位置，那么该状态就是 0110。

所以一共有 $0 \ldots (1 << n) - 1$ 种状态。

所以对于此题，我们只需要枚举全部状态，并计算每种状态的木棍长度，最后求出不同状态但长度相等，的最大长度即可。

解题代码

```
import java.io.*;
```

```java
import java.util.*;

public class Main{
    static final int N = 33000;
    static int n;
    static int[] state = new int[N];
    static int[] a = new int[20];

    public static void main(String[] args) throws Exception {
        BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
        PrintWriter pw = new PrintWriter(System.out);
        n = Integer.parseInt(bf.readLine().split(" ")[0]);
        String[] s = bf.readLine().split(" ");
        for (int i = 0; i < n; i ++ ) {
            a[i] = Integer.parseInt(s[i]);
        }

        for (int i = 0; i < (1<<n); i ++ ) {//枚举所有状态
            for (int j = 0; j < n; j ++ ) {//枚举状态可能包含的所有数
                if ((i & (1<<j)) != 0) { //如果该状态下的第j个数被选上
                    state[i] += a[j];//该状态就加上其长度
                }
            }
        }
        int res = 0;
        for (int i = 0; i < (1<<n); i ++ ) {
            for (int j = 0; j < (1<<n); j ++ ) {
                if ((i&j)==0 && state[i] == state[j]) {//如果两种状态不相等，但是长度相等
                    res = Math.max(res, state[i]);//更新最大相等长度
                }
            }
        }
        pw.println(res);
        pw.close();
    }
}
```