1 [CSP-J 2025] 多边形 / polygon

题目描述

小 R 有 n 根小木棍，第 i $(1 \le i \le n)$ 根小木棍的长度为 a_i。小 X 希望小 R 从这 n 根小木棍中选出若干根小木棍，将它们按任意顺序首尾相连拼成一个多边形。拼成多边形的条件：对于长度分别为 $l_1, l_2, \ldots, l_m$ 的 m 根小木棍，能拼成一个多边形当且仅当 $m \ge 3$ 且所有小木棍的长度之和大于所有小木棍的长度最大值的两倍，即 $\sum_{i=1}^{m} l_i > 2 \times \max_{i=1}^{m} l_i$。

小 X 提出的问题是：有多少种选择小木棍的方案，使得选出的小木棍能够拼成一个多边形？两种方案不同当且仅当选择的小木棍的下标集合不同。由于答案可能较大，你只需要求出答案对 $998,244,353$ 取模后的结果。

输入输出格式

输入：第一行，一个正整数 n，表示小木棍的数量。

第二行，n 个正整数 $a_1, a_2, \ldots, a_n$，表示小木棍的长度。

输出：一行一个非负整数，表示能拼成多边形的方案数对 $998,244,353$ 取模后的结果。

输入示例	输出示例
5 1 2 3 4 5	9
5 2 2 3 8 10	6

样例解释

样例1：共有9种选择方案能使选出的小木棍拼成多边形，具体方案如下：

1. 选择第 2,3,4 根小木棍，长度之和为 $2+3+4=9$，满足 $9 > 2 \times 4 = 8$；
2. 选择第 2,4,5 根小木棍，长度之和为 $2+4+5=11$，满足 $11 > 2 \times 5 = 10$；
3. 选择第 3,4,5 根小木棍，长度之和为 $3+4+5=12$，满足 $12 > 2 \times 5 = 10$；
4. 选择第 1,2,3,4 根小木棍，长度之和为 $1+2+3+4=10$，满足 $10 > 2 \times 4 = 8$；
5. 选择第 1,2,3,4,5 根小木棍，长度之和为 $1+2+3+4+5=15$，满足 $15 > 2 \times 5 = 10$。
6. ...

算法分析

1. 问题转化与补集思想

   直接计算合法方案较为困难，我们采用正难则反的策略。

- **总方案数推导**：对于 n 根木棍，每一根木棍都有“选”或“不选”两种状态。根据数学中的乘法原理，总的组合数为 $2 \times 2 \times \cdots \times 2 = 2^n$ 种。由于题目要求选出木棍（空集没有意义），我们扣除掉“一根都不选”的情况，因此总方案数：

$$Total = 2^n - 1$$

- **合法条件**：集合总和 $Sum > 2 \times Max$。
- **非法条件**：集合总和 $Sum \leq 2 \times Max$。
- 设集合中除最大值以外的元素和为 S_{rest}，则非法条件等价于：

$$S_{rest} + Max \leq 2 \times Max \implies S_{rest} \leq Max$$

- **结论**：若除最大边外的其余边之和不超过最大边，则该集合无法构成多边形。

2. 排序与枚举

（1） 首先将数组 a 从小到大**排序**。

（2） 枚举每一个 a_i 作为当前子集的**最大值**。

（3） 此时，我们只能从下标 $[0, i-1]$ 的木棍中选择若干根，使得它们的和 $\leq a_i$。这些组合即为以 a_i 为最大值的**非法方案**。

3. 动态规划（01背包）

- **定义**：$dp[j]$ 表示从之前的木棍中选出若干根，长度之和为 j 的方案数。
- **非法统计**：对于当前的 a_i，所有满足 $j \leq a_i$ 的 $dp[j]$ 之和，即为以 a_i 为最大值的非法方案数。

```
for (int j = 0; j <= a[i]; j++) {
    illegal = (illegal + dp[j]) % MOD;
}
```

- **转移**：统计完成后，将 a_i 加入背包供后续使用（01背包倒序更新）：$dp[j] = dp[j] + dp[j - a_i]$。

```
for (int j = Max; j >= a[i]; j--) {
    dp[j] = (dp[j] + dp[j - a[i]]) % MOD;
}
```

- **最终答案**：$Total - Illegal = (2^n - 1) - \sum$(针对每个 a_i 的非法方案)。

算法复杂度分析

- **时间复杂度**：$O(n \times V)$，其中 $V = \max(a_i)$。我们需要遍历 n 个元素，对于每个元素，都需要更新大小为 V 的背包数组。
- **空间复杂度**：$O(V)$，其中 $V = \max(a_i)$，用于存储动态规划数组 dp。

代码实现

```cpp
#include <bits/stdc++.h>
using namespace std;
using i64 = long long;
#define int long long
constexpr i64 inf = 1e18;
constexpr i64 MOD = 998244353;

int qpow(int a, int b, int m) {
    a %= m;
    int res = 1;
    while (b > 0) {
        if (b & 1) res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res % m;
}

void slu() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    sort(a.begin(), a.end());

    int Max = a.back(),illegal = 0;
    int res = qpow(2, n, MOD) - 1;
    vector<int> dp(Max + 1, 0);
    dp[0] = 1;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= a[i]; j++) {
            illegal = (illegal + dp[j]) % MOD;
        }
        for (int j = Max; j >= a[i]; j--) {
            dp[j] = (dp[j] + dp[j - a[i]]) % MOD;
        }
    }
    res = (res - illegal + MOD) % MOD;
    cout << res << endl;
}

signed main() {
    ios_base::sync_with_stdio(false);cin.tie(nullptr);
    slu();
    return 0;
}
```