合并水果 ☆☆

题目描述

在一个果园里，张力恒同学已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。张力恒决定把所有的果子合成一堆。

每一次合并，张力恒可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过 $n-1$ 次合并之后，就只剩下一堆了。张力恒在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以张力恒在合并果子时要尽可能地节省体力。假定每个果子重量都为 1，并且已知果子的种类数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

输入输出格式

输入：第一行，一个整数 $n(1 \leq n \leq 10000)$，表示果子的种类数；第二行，包含 n 个整数，用空格分隔，第 i 个整数 $a_i(1 \leq a_i \leq 20000)$，表示第 i 种果子的数目。

输出：一个整数，也就是最小的体力耗费值。输入数据保证这个值小于 2^{31}。

输入示例	输出示例
3 1 2 9	15

样例解释

步骤	当前果堆状态	合并的果堆	本次消耗体力	累计消耗体力
初始状态	1, 2, 9	-	0	0
第一步	1, 2, 9	$1+2=3$	3	3
第二步	3, 9	$3+9=12$	12	15

算法分析

1. **问题分析**

   要使总体力消耗最小，每次需要合并重量最小的两堆果子，重量小的果子被合并次数多，重量大的果子被合并次数少，总体力消耗最小。

2. **算法选择**

   A. **朴素思想：** 每次从果堆中选出最小的两堆进行合并，再将合并后的重量插入数组并重新排序，重复操作直到只剩一堆。每次排序花费时间为 $O(n\log n)$，一共进行 $n-1$ 次合并，所以总体时间复杂度为 $O(n^2\log n)$，无法在 $1s$ 内执行完。

   B. **使用优先队列（priority_queue）优化：** 使用最小堆维护最小元素，每次取出和插入元素均 $O(\log n)$，一共进行 $n-1$ 次取出和插入，因此总体复杂度 $O(n\log n)$，可以在 $1s$ 内执行完毕。

C. 实现思路

- 将所有果堆的重量放入最小堆中
- 每次取出两个最小元素，合并并累加体力消耗
- 将合并后的新堆重量放回堆中，重复直到堆只剩一个元素

背景知识

- **优先队列（堆）**：一种特殊的数据结构，可以快速获取和删除最小（或最大）元素。

```
priority_queue<int, vector<int>, greater<int>> pq;
```

  - **第一个参数 int**：指定队列中存储的元素类型
  - **第二个参数 vector<int>**：指定底层使用的容器，通常是vector
  - **第三个参数 greater<int>**：比较函数，决定元素的排列顺序
    - less<int>：最大堆，最大的元素在顶部（默认值）
    - greater<int>：最小堆，最小的元素在顶部

常用操作：

操作	功能说明	时间复杂度
pq.push(x)	将元素 x 插入堆	$O(\log n)$
pq.pop()	删除堆顶部（最小或最大）元素	$O(\log n)$
pq.top()	获取堆顶部元素	$O(1)$
pq.empty()	判断堆是否为空	$O(1)$
pq.size()	获取堆中元素个数	$O(1)$

核心代码

```
while (pq.size() >= 2) { // 重复操作直到堆中只剩一个元素
    int u = pq.top();
    pq.pop();
    int v = pq.top();
    pq.pop();
    cnt += u + v; // 取出两个最小的元素，合并它们，并累加体力消耗
    pq.push(u + v); // 合并结果放回堆中
}
```

拓展思考

- 如果每次合并三堆、四堆，甚至 k($k \le 10^5$) 堆果子，算法该如何修改？
- 如果要求改成求消耗体力的最大值，该如何实现？