Sample 3.

Input. 1, 2, 3, 4, 5, 5, 7, 7, 8, 10, 12, 19, 25.

Output. 1.

Explanation. $1+3+7+25 = 2+4+5+7+8+10 = 5+12+19$

Solution

Algorithm. This is a generalization of the knapsack without repetitions problem. Let $\mathit{partitionable}(i, S_1, S_2) = \mathit{true}$ if it possible to partition the first i souvenirs into three disjoint subsets such that the sum of the first part is S_1 and the sum of the second part is S_2 (hence, the sum of the third part is $V - S_1 - S_2$), and *false* otherwise. Recurrence relation:

$$\begin{aligned}\mathit{partitionable}(i, S_1, S_2) = &\mathit{partitionable}(i-1, S_1, S_2) \vee \\ &\mathit{partitionable}(i-1, S_1 - \mathit{value}_i, S_2) \vee \\ &\mathit{partitionable}(i-1, S_1, S_2 - \mathit{value}_i)\end{aligned}$$

(the $\vee$ sign here is the logical or operation). The answer for the initial problem is $\mathit{partitionable}(n, V/3, V/3)$. Since i ranges from 0 to n and S_1, S_2 range from 0 to V we have $O(nV^2)$ subproblems. We solve all of them by first ranging i from 1 to n and then ranging S_1, S_2 from 0 to V. The base cases: $\mathit{partitionable}(i, 0, 0) = \mathit{true}$ for all i, and $\mathit{partitionable}(0, S_1, S_2) = \mathit{false}$ for all S_1, S_2 with $S_1 + S_2 > 0$.

Correctness. The recurrence relation is correct since item i must be either in the third part, or in the first part, or in the second part. Another way of looking at this relation is the following: to get a partition of the first i souvenirs into three parts we take a partition of the first $i-1$ souvenirs into three parts and add the i-th souvenir to one of the parts.

Running time. The running time is $O(nV^2)$ since there are so many subproblems and the solution to each of them is computed (through solutions to smaller subproblems) in constant time.

5 Basic Data Structures

5.1 Computing tree height

Description. The goal of this problem is to compute the height of the given tree. The tree is given in the following format. Assume that the tree has n nodes and that the nodes are numbered from 0 to $n-1$ and that the vertex 0 is the root. Then the tree is given by specifying the parents of nodes $1, 2, \ldots, n-1$.

Input. A sequence $\mathit{parent}_1, \mathit{parent}_2, \ldots, \mathit{parent}_{n-1}$. (It is guaranteed that the input specifies a correct tree, you do not need to additionally check this.)

Output. The height of the tree (i.e., the number of nodes on a longest path from the root to a leaf).

Running time. $O(n)$.

Sample.

Input. $0, 4, 0, 3$.

Output. 4.

Explanation. The corresponding tree is shown below. The longest path is $0 \to 3 \to 4 \to 2$.