## 1405. Палочки

*Закон Джунглей гласит совершенно ясно: каждый волк, заведя семью, может покинуть свою Стаю. Но как только его волчата подрастут и окрепнут, он обязан привести их на Совет Стаи, который обычно собирается раз в месяц, в полнолуние, и представить всем остальным волкам.*

Отец Волк дождался, пока его волчата немного подросли и начали понемногу бегать, и в одну из тех ночей, когда собиралась Стая, повёл их – вместе с Маугли и Матерью Волчицей – на Скалу Совета. Это была вершина холма, усеянная крупными валунами, за которыми могла укрыться целая сотня волков. Акела, большой серый волк-одиночка, избранный вожаком всей Стаи за силу и ловкость, взывал со своей скалы:

— Закон вам известен, закон вам известен! Смотрите же, волки!

Отец Волк вытолкнул на середину круга Лягушонка Маугли. Тот сел на землю, засмеялся и стал играть палочками.

Задачу он придумал себе простую: нужно было из этих палочек сложить прямоугольник максимальной площади – при этом использовать все палочки было не обязательно.

**Вход.** Первая строка содержит количество палочек *n* (1 ≤ *n* ≤ 16). Во второй строке заданы их длины – натуральные числа в диапазоне от 1 до 108.

**Выход.** Выведите максимальную площадь прямоугольника, который можно сложить из данного набора палочек, или число , если сложить прямоугольник невозможно.

|  |  |
| --- | --- |
| **Пример входа** | **Пример выхода** |
| 8  7 1 5 2 3 2 4 5 | 49 |

**РЕШЕНИЕ**

**динамическое программирование - маски подмножеств**

# Анализ алгоритма

Пусть S – данное множество палочек. Необходимо найти два непересекающихся подмножества A и B из S, такие, что палочки каждого из них можно разбить на два подмножества с равной суммой длин.

Для каждого подмножества *mask* ⊆ S вычислим сумму длин его палочек и запишем в sum[*mask*].

Переберём все подмножества I  S. Для каждого подмножества I переберём все его подмножества J  I. Если существует такое J, что 2 \* sum[J] = sum[I] (то есть суммарная длина палочек в J вдвое меньше суммы длин всех палочек из I), то подмножество I можно использовать в качестве противоположных сторон прямоугольника. В этом случае установим can[I] = ***true***.

Вся описанная процедура выполняется за O(3*n*).

Переберём все подмножества множества палочек. Если для некоторого множества I существует подмножество J  I такое, что can[J] = ***true*** и can[I xor J] = ***true*** (то есть множества J и I xor J не пересекаются), то тем самым получаем одно из возможных распределений палочек между горизонтальными и вертикальными сторонами прямоугольника.

Длины сторон такого прямоугольника равны sum[J] / 2 и sum[I xor J] / 2. Среди всех найденных вариантов выбираем прямоугольник с максимальной площадью.

# Реализация алгоритма

Длины палочек хранятся в массиве *a*. Для каждого подмножества палочек *mask* сохраняем их суммарную длину в массиве sum[*mask*]. Значение can[*mask*] устанавливаем в ***true***, если множество *mask* можно разделить на два подмножества с равной суммой длин – им будут соответствовать противоположные стороны прямоугольника.

#define MAX 16

int a[MAX];

int sum[1 << MAX];

bool can[1 << MAX];

Основная часть программы. Читаем входные данные.

scanf("%d",&n);

for (i = 0; i < n; i++)

scanf("%d", &a[i]);

Переберём все подмножества палочек и вычислим суммарную длину каждого из них.

for (i = 0; i < 1 << n; i++)

for (j = 0; j < n; j++)

if (i & (1 << j)) sum[i] += a[j];

Затем переберём все возможные подмножества множества палочек. Если для некоторого множества I существует подмножество J  I такое, что суммарная длина палочек в J вдвое меньше суммарной длины палочек в I, то установим can[I] = ***true***. В этом случае палочки из множества I можно использовать как противоположные стороны прямоугольника.

for (i = 1; i < 1 << n; i++)

{

int s = sum[i];

for (j = i; j > 0; j = (j - 1) & i)

if (sum[j] \* 2 == s)

{

can[i] = true;

break;

}

}

Переберём все подмножества множества палочек. Если для некоторого множества I существует подмножество J  I такое, что can[J] = ***true*** и can[I xor J] = ***true***, то получаем одно из возможных распределений палочек между горизонтальными и вертикальными сторонами прямоугольника. Длины сторон такого прямоугольника равны sum[J] / 2 и sum[I xor J] / 2.

res = 0;

for (i = 1; i < 1 << n; i++)

for (j = i; j > 0; j = (j - 1) & i)

if (can[j] && can[i ^ j])

res = max(res, sum[j] / 2 \* 1LL \* sum[i ^ j] / 2);

Выводим площадь найденного прямоугольника максимального размера.

printf("%lld\n",res);