

История

Автор задачи: Илья Бурков, разработчик: Андрей Павлов

Формально в условии требуется построить граф на n вершинах, в котором присутствуют все ребра (i, j) т.ч. $a_i + a_j = S$ или $a_i \oplus a_j = X$. Ответом на задачу было совершенное паросочетание в данном графе, причем граф необязательно двудольный.

Для решения подгруппы $n \leq 20$ требовалось перебрать всевозможные разбиения на пары и проверить, что они подходят под условия.

В подгруппе 3 в графе не присутствовали ребра, выполняющие условия $a_i + a_j = S$. Значит подходили только пары чисел $(a, a \oplus X)$. Тогда достаточно посчитать cnt_a — количество вхождений каждого числа, после чего достаточно проверить, что $cnt_a = cnt_{a \oplus X}$ при условии, что $X > 0$.

Если же $X = 0$, то все корректные пары, в которых выполняется XOR условия это $a_i = a_j$, поэтому для подгруппы 3 достаточно проверить что cnt_a делится на 2 для каждого a . Для подгруппы 4 у нас теперь снова есть пары вида $a_i + a_j = S$, а еще мы умеем ставить в пару два одинаковых значения a , поэтому достаточно проверить, что $cnt_a \equiv cnt_{S-a} \pmod{2}$, и не забыть обработать отдельно случай $2a = S$.

Далее будем считать, что случай $X = 0$ обработан отдельно и $X > 0$.

Рассмотрим подгруппу 6, в ней все a_i различны. Но тогда попробуем понять, с кем связана вершина i : из решения двух равенств можно понять, что $a_j = S - a_i$ или $a_j = X \oplus a_i$, так как все числа различны, то такое j единственно в обоих случаях. А значит в нашем графе из вершины i исходит не более 2 ребер, иногда может исходить меньше, т.к. подобного j может не найтись в массиве. Значит наш граф выглядит как компоненты из простых циклов и путей. Для такого графа можно конструктивно найти паросочетание, используя любой из алгоритмов обхода графов за $O(n)$.

Теперь перейдем к полному решению: в нем у нас каждое число может встречаться сколько угодно раз, поэтому построим наш граф немного иначе: запомним тот же массив cnt и оставим в массиве только различные числа, построим тот же граф и попытаемся применить решение для подгруппы 6. Главное отличие в том, что теперь у этих путей и циклов есть еще одно число на вершине cnt . По сути у нас есть операция для ребра уменьшить число на обоих его концах, такими операциями требуется сделать на всех концах значение 0.

В случае пути достаточно понять, что для конца этого пути есть число на нем x , тогда нужно взять единственное ребро с его концом ровно x раз, тогда значение в этой вершине станет 0, а в другой $y - x$ и можно будет мысленно удалить данную вершину, после чего путь уменьшится на 1 и можно будет итеративно обработать весь путь. В случае если $y - x < 0$ или осталась одна вершина с ненулевым значением, то ответа не существует.

Теперь если мы рассматриваем цикл, то в нем можно зафиксировать любую вершину и выписать от нее в какую-либо сторону цикл. После чего можно перебрать сколько раз, первое ребро на цикле будет взято, после чего можно будет мысленно забыть про это ребро и мы вернемся к случаю с путем, но это работает долго, попробуем оптимизировать. Формально на цикле записаны по порядку числа c_1, c_2, \dots, c_k , и мы перебираем такое $x \leq \min(c_1, c_k)$, после чего рассматриваем уже путь с записанными на нем числами $c_1 - x, c_2, \dots, c_{k-1}, c_k - x$. Тогда давайте посмотрим какие условия должны выполняться для пути, чтобы он был хорошим. $c_1 \leq c_2, c_2 - c_1 \leq c_3, c_3 - (c_2 - c_1) \leq c_4, c_4 - (c_3 - (c_2 - c_1)) \leq c_5 \dots$ и $c_k - (c_{k-1} - (\dots - (c_2 - c_1))) = 0$. Тогда если мы вставим сюда параметр x , получим, что для каждого четного i будет какое-то уже константное выражение $-x$, а в случае нечетных наоборот $+x$, следовательно у нас накладываются $O(k)$ ограничений на x сверху и снизу, обозначим их за $low, high$. Тогда достаточно взять любое x из отрезка $[low, high]$, подходящее под последнее условие, которое также решается достаточно легко. Из проблем в реализации существует случай $2 \cdot a_i = S$, но в данном решении оно обрабатывается достаточно просто.

Итоговую реализацию можно написать с помощью хеш-мап за $O(l_1 + l_2 + \dots + l_m) = O(n)$ где l_i размер i -й компоненты, что заходит на 100 баллов.