Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по «Алгоритмам и структурам данных»

Выполнил:

Студент группы P3200

Шишкин Н.Д.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2019

Задача №2025 «Стенка на стенку»

Пояснение к примененному алгоритму:

Пусть существует оптимальное распределение n человек по k командам, (a[i] - количество людей в команде с номером i).

Добавим ещё одного человека, необходимо понять в какую команду его нужно добавить, чтобы новое распределение также было оптимальным.

Число боев - сумма попарных произведений человек в командах для всех возможных пар команд, т. е.:

$\mathrm{Old\_max} = (\sum_{i=1}^{k\ (i\ != 1)} a[1] * a[i] + \sum_{i=1}^{k\ (i\ != 2)} a[2] * a[i] + \ldots + \sum_{i=1}^{k\ (i\ != k)} a[k] * a[i]\ )\ /2$

(/2, потому что дважды считаем каждый бой)

Добавим к некой команде с номером x одного человека, тогда:

$$\mathrm{New} = \mathrm{Old\_max} + (\sum_{i=1}^{k\ (i\ != x)} a[i] + \sum_{i=1}^{k\ (i\ != x)} a[i])\ /2 = \mathrm{Old\_max} + \sum_{i=1}^{k\ (i\ != x)} a[i]$$

В таком случае очевидно, что New максимален в том случае, когда максимально суммарное количество человек в остальных командах кроме x, значит x - команда с минимальным числом людей.

Рассмотрим теперь увеличение числа команд, добавим (k+1)ую команду, и добавим туда одного человека из команды x, тогда:

$$\mathrm{New} = \mathrm{Old\_max} + \sum_{i=1}^{k} a[i] - \sum_{i=1}^{k\ (i\ != x)} a[i]$$

Очевидно, что новое значение больше, чем старое, значит увеличение числа команд увеличивает (в крайнем случае если в команде никого нет - не уменьшает) число боёв.

Отталкиваясь от этих фактов, рассмотрим произвольную ситуацию с n людьми и k командами (берем максимально возможное число команд) $n \geq k$. Возьмем k людей, тогда единственное распределение - по одному человеку в каждую команду. Добавим одного человека - в данном случае в любую команду. Тогда после добавления k людей в каждой команде будет по 2 человека. Продолжаем по аналогии, тогда k людей можно будет добавить ещё $(n-k)\ div\ k$ раз. Останется $(n-k)\ mod\ k$ людей, которых мы добавим по одному человеку в какие-то (не одни и те же) команды. Итого получим: В $k - n\ mod\ k$ командах по $n\ div\ k$ людей, в $n\ mod\ k$ - по $(n\ div\ k) + 1$. Согласно предыдущим фактам, такое распределение даст максимальное число боев.

Найдем число боев внутри групп с одинаковым количеством людей и между ними (путем простого подсчёта числа сочетаний по 2) - это ответ.

Сложность алгоритма - O(1)

Задача №1005 «Куча камней»

Пояснение к примененному алгоритму:

Заведем битовый массив от 0 до Суммы весов камней / 2, 1 - означает что мы можем получить такую сумму складывая камни, 0 - нет. Установим нулевой бит в 1 (сумму весом 0 мы можем получить и без камней)

Дальше в цикле перебираем веса камней, при этом битово перемножая наш массив с самим собой, смещенным вправо на вес текущего камня (т.к. если i-ый бит == 1, то добавив к этой сумме текущий камень мы получим сумму i + w, а не добавив оставим сумму i, что и соответствует перемножению со смещенной версией массива).

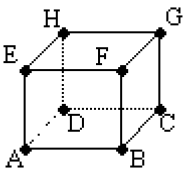
Нас интересует минимальная разница сумм, поэтому найдем с конца массива первый ненулевой бит - его номер и будет являться суммой для достижения минимальной разницы.

Время работы - O(n*S), где S = $(\sum_{i=1}^{n} \mathrm{w_i} \ )/2$

(в худшем случае S = 1 000 000)

Задача №1155 «Дуоны»

Пояснение к примененному алгоритму:



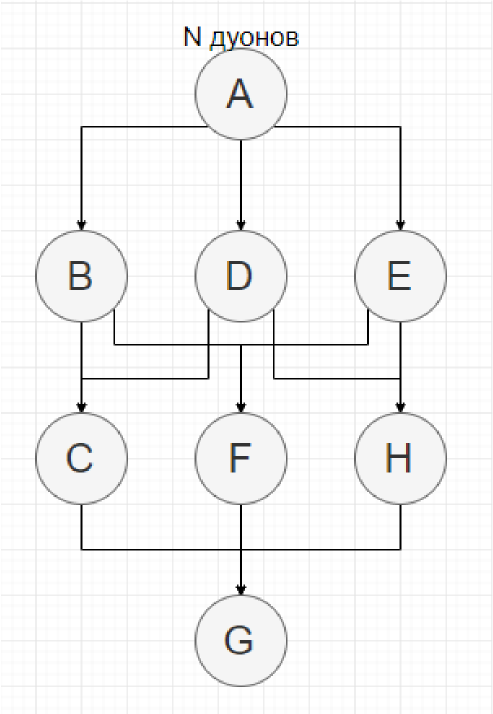
Рассмотрим ситуацию, когда дуоны есть только в одной вершине (Например в A).

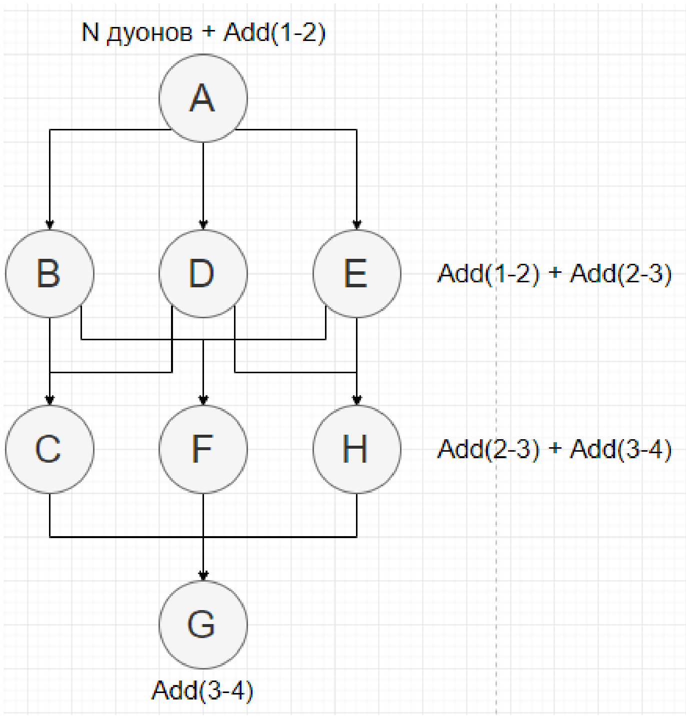
Имеем четыре «уровня» вершин относительно A. Добавление или удаление пары дуонов может затронуть только два соседних уровня.

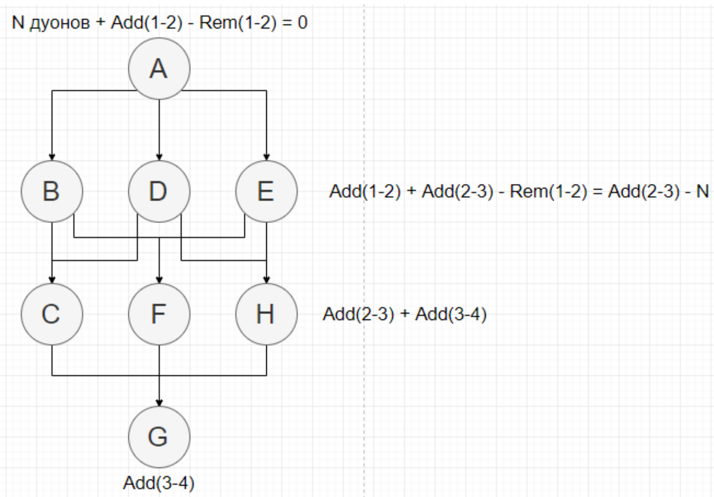
Допустим каким-то образом систему очистили, пусть тогда Add(n - n+1) и Rem(n - n+1) отражают операции добавления/удаления пары дуонов, где n - один из уровней от 1 до 3.

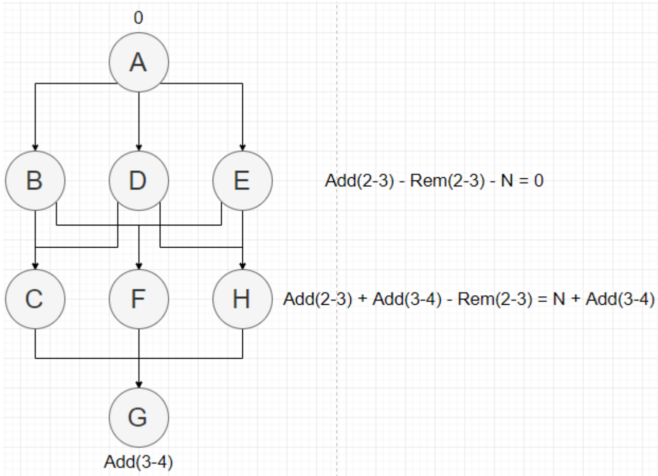
Последовательной рассмотрим операции, следя за суммарным числом дуонов на каждом уровне.

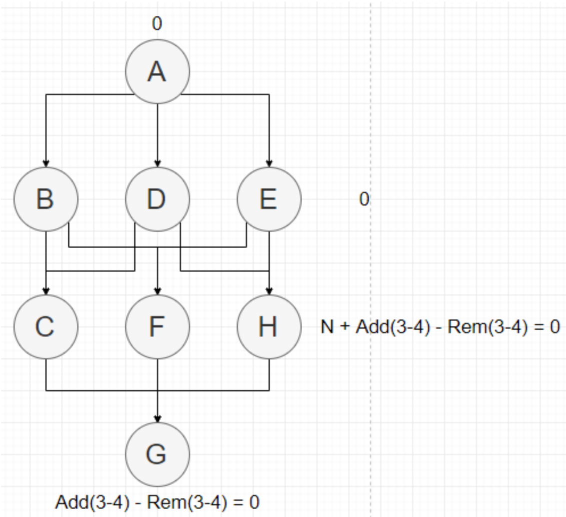
1.) Произведем все добавления
2.) Произведем удаление дуонов между 1 и 2 уровнем
3.) Между 2 и 3.
4.) Между 3 и 4.











В итоге получаем равенство N = 0, что противоречит условию. Допустим, что на 3ем уровне содержалось бы K дуонов, тогда в итоге бы мы получили N + K = 0, что также невозможно. Однако если бы на 4ом уровне было M дуонов, мы бы получили N - M = 0, что вполне достижимо. То есть, очистка дуонов возможна при одинаковом количестве дуонов в диагонально противоположных вершинах куба.

В общем условие очистки будет звучать так: Дуоны на первом уровне + Дуоны на третьем уровне - Дуоны на втором уровне - Дуоны на третьем уровне = 0.

Отталкиваясь от этого, составим алгоритм:

Проверим выполнимость условия очистки. Если оно выполнилось, тогда:

Сначала пройдемся по всем ребрам, и удалим все возможные пары дуонов для каждого ребра. Затем возьмем грань и удалим все общие дуоны между вершинами грани и диагонально противоположными им вершинами, при необходимости добавляя недостающие ребра.

Сложность алгоритма - O(n), где n - общее количество дуонов.

Задача №1296 «Гиперпереход»

Пояснение к примененному алгоритму:

Нас интересует максимально возможная сумма на части массива, воспользуемся частичными суммами (т.е. храним сумму всех чисел от начала до текущей ячейки), в таком случае сумма на подотрезке - это разность частичных сумм правого и левого конца отрезка. Эта разность тем больше, чем больше правый конец и меньше левый. Значит идем по массиву, и поддерживаем три числа - минимальную и максимальную частичную сумму, а также их максимальную разность. Вычислив i-ую частичную сумму, проверим два условия:

- Если она меньше минимальной сохраненной, обновляем последнюю и сбрасываем максимальную сохраненную (ибо она была до новой минимальной, т. е. левее, но мы идем по отрезку только слева направо).
- Если она больше максимальной сохраненной, обновляем последнюю и при надобности обновляем минимальную разность.

В ответ выводим минимальную разность или 0, если он больше.

Алгоритм корректен, потому что на каждом шаге мы поддерживаем корректное значение максимальной разности, и обрабатываем все варианты её изменения при добавлении нового элемента.

Сложность алгоритма - O(n)

Задача №1401 «Игроки»

Пояснение к примененному алгоритму:

Разбиваем рекурсивно данное поле на 4 поля по четвертям, в зависимости от того, в какой четверти находится выколотая клетка, заполняем углы близкие к центру поля остальных четвертей числом, получаем одну из данный фигур, для самих четвертей вызываем разбиение с этими углами как выколотыми клетками. Если получили поле 2 на 2 - заполняем его. Всё это обернуто в виде функции, которая принимает координаты 2 углов поля, которое надо заполнить и выколотой клетки внутри него.

Иллюстрация:

x\y	1	-	-	n/2	n/2+1	-	-	n
1	*	*	*	*	*	*	*	*
-	*	0	*	*	*	*	*	*
-	*	*	*	*	*	*	*	*
n/2	*	*	*	*	1	*	*	*
n/2+1	*	*	*	1	1	*	*	*
-	*	*	*	*	*	*	*	*
-	*	*	*	*	*	*	*	*
n	*	*	*	*	*	*	*	*

Вызываем функцию рекурсивно от этих частей

Сложность алгоритма $T(n) = 4*T(\frac{n}{2}) + O(1) = O(n^2)$