

Последняя задача

Решение .

Давайте попробуем изобразить процесс получения результата как-нибудь понагляднее. Последовательно заменяем два числа на одно? Естественно изобразить это в виде дерева. Я не буду подробно и строго выписывать, что это за дерево, и как оно получается – просто приведу несколько примеров, и всё станет понятно:

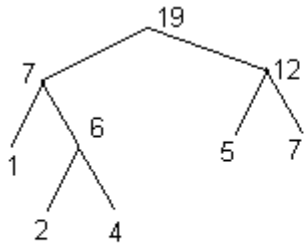


рисунок 1

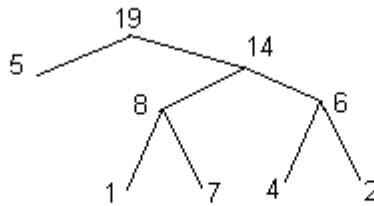


рисунок 2

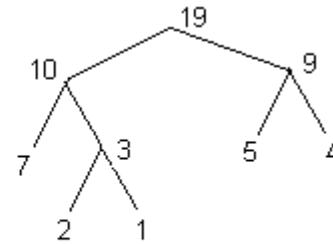


рисунок 3

Рисунок 1 соответствует такому процессу: $5+7 \rightarrow 12$, $2+4 \rightarrow 6$, $1+6 \rightarrow 7$, $12+7 \rightarrow 19$,
рисунок 2 соответствует процессу $1+7 \rightarrow 8$, $2+4 \rightarrow 6$, $8+6 \rightarrow 14$, $5+14 \rightarrow 19$,
а рисунок 3 отображает вот такое сложение: $1+2 \rightarrow 3$, $4+5 \rightarrow 9$, $3+7 \rightarrow 10$, $10+9 \rightarrow 19$.

Получается бинарное дерево, да не простое, а, скажем так, *строго бинарное дерево*. Слово “строго” здесь употребляется в том смысле, что у каждой вершины либо ровно 2 потомка, либо потомков нет вовсе (и тогда эта вершина – лист). Числовые метки листьев – это заданные штрафные очки, метки *внутренних вершин* (будем здесь считать все не листовые вершины, включая корень, внутренними) равны сумме двух меток в потомках вершины. Понятно, что окончательный результат – это сумма меток во всех внутренних вершинах.

Легко видеть, что если имеется некоторое строго бинарное дерево с N листьями и листья помечены заданными числами, то окончательный результат не зависит от порядка выполнения операций сложения – лишь бы только к моменту вычисления метки внутреннего узла были вычислены метки обоих его потомков. Так, дерево на рисунке 1 соответствует не только вышеописанному процессу сложения, но и, например, такому: $2+4 \rightarrow 6$, $5+7 \rightarrow 12$, $1+6 \rightarrow 7$, $12+7 \rightarrow 19$. Возможны, разумеется, и другие варианты.

Итак, дерево и разметка его листьев однозначно определяют окончательный результат.

Давайте посмотрим, как можно иначе вычислить окончательный результат для заданного дерева и разметки его листьев. Рассмотрим для начала всё тот же рисунок 1. Окончательный результат равен $6+7+12+19$. Откуда взялись эти числа? Понятно, что это суммы каких-то заданных чисел. Заданные числа в примере: 1, 2, 4, 5, 7. Мы видим, что $6 = 2+4$, $12 = 5+7$, $7 = 1+6 = 1+2+4$, $19 = 7+12 = 1+2+4+5+7$. Мы видим, что 2 входит как слагаемое в 6, 7 и 19, а 5 входит в 12 и 19 и т.д. Легко видеть, что каждая листовая метка входит как слагаемое в метку каждой внутренней вершины (включая корень), лежащей на пути между соответствующим листом и корнем дерева, как это изображено на рисунке 1а.

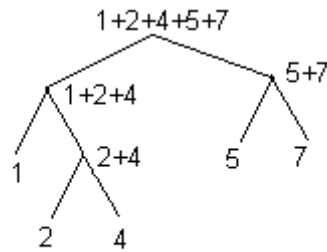


рисунок 1а

Вспомним, что окончательный результат равен сумме меток всех внутренних вершин. Взглянув на содержание этих меток, на то, как они собираются из листовых меток, видим, что окончательный результат равен

,

где n_i – это i -ое заданное число, а h_i – высота листа, меткой которого это число является. *Высота листа* – это, как обычно, расстояние (количество рёбер) на пути от листа до корня дерева.

Повторюсь, дерево и разметка его листьев однозначно определяют окончательный результат. Мы даже знаем, как этот результат вычислить: $\sum_i n_i \cdot h_i$

А как получить минимальный возможный результат. Давайте начнём с чего попроще... (Авось, до трудного дело и не дойдёт, авось пока дойдёт до трудного, начальство всё отменит ☺. Забегая вперёд, скажу, что в данном случае так оно и случится ☺). Допустим, что дерево нам уже дано. Дерево строго бинарное (напомню, это означает, что каждая из его нелистовых вершин имеет ровно двух потомков) с N листьями. Как нам разметить листья, чтобы сумма $\sum_i n_i \cdot h_i$ была как можно меньше? Ответ очевиден: листья с большей высотой

надо отмечать меньшими числами. Иначе говоря, можно действовать так: заданные числа сортируем по возрастанию, листья сортируем по убыванию высоты, и первое число ставим возле первого листа, второе – возле второго и т.д. ...

Прекрасно, расставлять метки, если дерево есть, мы уже умеем – это оказалось совсем легко. Вот только откуда взять то дерево, которое даст нам наименьший результат?

Давайте на время забудем об этом и поиграем ещё немного с расстановкой меток. Вопрос – как расставлять при сортировке листья с одинаковой высотой? Ответ – всё равно, их можно переставлять в произвольном порядке, на окончательный результат это не повлияет. А вот давайте-ка поставим на первые два места в этом ряду двух «братьев» – два листа с максимальной высотой, у которых родительская вершина – общая, ведь это не повлияет на окончательный результат.

А посмотрим теперь на процесс сложений, который приведёт к оптимальному дереву. Про него ничего не ясно. А нет, не совсем ничего... Понятно, что первое сложение – это сложение меток двух листьев-«братьев», причём неважно, каких братьев взять (мы ведь уже заметили, что одно и то же дерево с одной и той же разметкой может соответствовать различным способам получения результата). Так возьмём же ту пару, которая помечена двумя наименьшими числами. Хуже от этого не станет.

Что же получилось? А то, что независимо от того, какое именно дерево соответствует процессу вычисления конечного результата, мы можем начинать со сложения двух

наименьших чисел! И неважно, что мы не знаем, какое именно дерево приведёт к наименьшему результату – всё равно мы можем начинать именно со сложения двух наименьших чисел.

Вот и забудем про все на свете деревья и прочую флору: вот есть у нас набор чисел; вот мы заменили два наименьших числа на их сумму; вот у нас осталось $(N-1)$ чисел. И что с ними делать? Станный вопрос – конечно же, то же самое ☺

Как всё здорово получилось! Просто **на каждом шаге нужно два наименьших числа заменять на их сумму** (а эту сумму, конечно, добавлять к результату – нам же нужно выводить не описание процесса суммирования, а минимальный конечный результат). А при чём тут деревья? А уже и ни при чём. Конечно, мы можем построить дерево, соответствующее получаемому процессу, но нам это не нужно.

Наивная реализация полученного алгоритма – отсортировать заданные числа, а затем многократно удалять два наименьших числа и вставлять их сумму в оставшийся ряд чисел – имеет сложность $O(N^2)$. Конечно, можно поискать что-нибудь получше. И долго искать не приходится. У нас имеется изменяющееся множество чисел, нам надо (кроме операций убрать-добавить-заменить число) получать два наименьших числа множества. Вспомним, в предыдущей задаче была бинарная куча, которая позволяла легко и быстро найти наименьшее число (вернее в предыдущей задаче мы искали наибольшее число, но разницы никакой нет – только заменить $>$ на $<$ и наоборот). Нам нужны два наименьших числа, но это несложно – наименьший элемент множества лежит в корне бинарной кучи (в h_1), а второй наименьший элемент – это меньший из двух потомков корня, меньший из h_2 и h_3 . Паскаль-реализация данного алгоритма – короткая и ясная – приводится. Небольшой комментарий к ней всё-таки приведу. Будем говорить, что отрезок массива a от L до R обладает структурой бинарной кучи, если для всех k из интервала от L до R выполняются условия:

- если $2k \leq R$, то $a[k] \leq a[2k]$;
- если $2k+1 \leq R$, то $a[k] \leq a[2k+1]$.

Процедура `shift` получает на входе массив a , причём отрезок массива a от $(L+1)$ до R обладает структурой бинарной кучи, а в $a[L]$ лежит произвольное число. Процедура `shift` переставляет элементы отрезка массива $a[L..R]$ так, что в результате отрезок массива a от L до R приобретает структуру бинарной кучи. Процедура простая, ясная и, как легко видеть, имеет сложность $O(\log N)$. Отсюда сразу видно, что сложность алгоритма равна $O(N \cdot \log N)$.

И последнее замечание. Последнее по порядку, но, кажется, самое важное и интересное в этой задаче. Построенный алгоритм – есть в точности алгоритм Хаффмана (в русской транскрипции встречаются также варианты написания Хаффман, Хафмен и др.) построения *двоичного префиксного кода*. Метод был впервые описан в статье

D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102,
и назван по имени автора.

Я не буду здесь рассказывать, что такое двоичные префиксные коды, зачем они нужны и почему и в каком смысле код Хаффмана является оптимальным – среди множества изложений этой темы имеется немало очень хороших, красивых и интересных. Я просто некоторые из них выложу в учебных материалах (это не означает, что все остальные материалы о кодах Хаффмана я считаю нехорошими, либо некрасивыми, либо неинтересными). Очень советую почитать. Впрочем, я это уже сказал тремя строчками выше ☺.

В учебных материалах лежат фрагменты из книг

Кнут Д.Э. Искусство программирования, т.1. Основные алгоритмы, 3-е изд. 720 стр. М.: Вильямс, 2000.

Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-е изд. 1296 стр. М.: Вильямс, 2006:

Реньи А. Трилогия о математике, 376 стр. М.: Мир, 1980 (как я люблю эту книгу!)

Яглом А.М., Яглом И.М. Вероятность и информация, 512 стр., М.: Наука. 1973 (и эту тоже, но это всё-таки математика, а вот книга Реньи – совершенно блестящая книга, особенно тем, что математика там изложена удивительно содержательно, несмотря на практически полное отсутствие формул и выкладок).