Разработка рекурсивного алгоритма решения олимпиадной задачи по информатике на переливания

Ю. В. Кулаков, email: kulak@list.ru

Тамбовский государственный технический университет

Аннотация. *Разработан алгоритм решения олимпиадной задачи на переливания, который во всех рассмотренных вариантах исходной наполненности сосудов гарантированно находит кратчайший путь переливаний или, по крайней мере, не явно сократимый путь.*

Ключевые слова: *задача на переливания, рекурсивный алгоритм, исходная наполненность сосудов, явно сократимый путь переливаний, кратчайший путь переливаний.*

Введение

Задачи на переливание – один из видов старинных задач. Они возникли много веков назад, но до сих пор вызывают интерес у любителей математики в качестве логических задач.

Часто суть этих задач сводится к следующему. Имея несколько сосудов разного объёма, один из которых наполнен жидкостью, требуется разделить её в каком-либо отношении или отлить какую-либо ее часть при помощи других сосудов за наименьшее число переливаний.

В задачах на переливания требуется указать последовательность действий, при которой осуществляется требуемое переливание и выполнены все условия задачи.

Не обошли стороной задачи на переливания и разработчики олимпиадных задач по информатике и программированию. В частности, задачу с соответствующим названием «Переливания», решаемую в данной работе, можно найти в интернете в качестве одной из задач I Всероссийской заочной олимпиады школьников по информатике [1].

1. Постановка олимпиадной задачи

Есть три сосуда с водой. В одном из них A миллилитров воды, в другом – B миллилитров, в третьем – C. Разрешается следующая операция. Можно перелить воду из одного сосуда в другой так, чтобы в том сосуде, в который мы переливаем, количество воды после переливания было в два раза больше, чем до переливания. То есть, если до переливания в сосудах было A, B и C миллилитров соответственно, и мы переливаем, например, из второго сосуда в третий, то после

переливания в сосудах должно оказаться A, B – C, 2C миллилитров соответственно (такое переливание можно делать только при условии, когда B ≥ C).

Требуется написать программу, которая определит, можем ли мы в результате освободить один из сосудов.

Входными данными являются неотрицательные целые числа A, B, C – количество воды в каждом из сосудов изначально.

Если освободить один из сосудов можно, то необходимо вывести сначала количество операций, которое для этого понадобится, а дальше – сами операции. Каждая операция описывается двумя числами – номером сосуда, из которого мы переливаем, и номера сосуда, куда переливаем. Минимизировать количество операций переливания не требуется.

Если освободить сосуд невозможно, то необходимо вывести одно число: минус 1.

Например, если A = 1, B = 2 и C = 10, то необходимо сделать два переливания: сначала переливание из третьего сосуда в первый, а затем из второго в первый. Если A = 0, B = 1 и C = 0, то никаких переливаний не требуется.

Заметим, что для других начальных объёмов воды в сосудах последовательность переливаний может быть далеко не очевидной. В данной работе сконструируем рекурсивный алгоритм решения рассматриваемой олимпиадной задачи, поскольку рекурсивное построение алгоритма является наиболее естественным путём решения проблем [2, 3].

2. Разработка алгоритма решения задачи

Для исследования разрабатываемого рекурсивного алгоритма решения олимпиадной задачи «Переливания» созданы с использованием генератора псевдослучайных последовательностей семь вариантов изначальной целочисленной (различной) наполненности сосудов (когда объёмы воды в каких-либо двух сосудах совпадают решение очевидно):

Вариант 1) A = 8, B = 3, C = 11.

Вариант 2) A = 10, B = 2, C = 6.

Вариант 3) A = 11, B = 14, C = 3.

Вариант 4) A = 4, B = 7, C = 2.

Вариант 5) A = 14, B = 5, C = 8.

Вариант 6) A = 5, B = 4, C = 6.

Вариант 7) A = 7, B = 6, C = 12.

Заметим, что принципиально возможных переливаний с использованием трёх сосудов (без учёта их наполненности) всего шесть. Перечислим их в лексикографическом порядке: из первого сосуда во

второй (переливание 12), из первого в третий (переливание 13), из второго в первый (переливание 21), из второго в третий (переливание 23), из третьего в первый (переливание 31) и, наконец, из третьего во второй (переливание 32).

Основной алгоритм решения задачи будет использовать вспомогательный рекурсивный алгоритм, названный нами именем «transfusion»:

Шаг 1. Начало алгоритма.

Шаг 2. Ввести значения наполненности сосудов A, B, C.

Шаг 3. Вызвать вспомогательный алгоритм transfusion.

Шаг 4. Конец (основного) алгоритма.

Начальная версия рекурсивного алгоритма transfusion (версия 0) была основана на идее рекурсивного выполнения допустимых переливаний в лексикографическом порядке и ожидания освобождения одного из сосудов:

Шаг 1. Начало алгоритма.

Шаг 2. Вывести значения наполненности сосудов A, B, C.

Шаг 3. Если A = 0 или B = 0 или C = 0, то конец алгоритма; иначе перейти к следующему шагу.

Шаг 4. Если A ≥ B, то вывести «–12–>», A ← A – B, B ← 2B, вызвать вспомогательный алгоритм transfusion; иначе к следующему шагу.

Шаг 5. Если A ≥ C, то вывести «–13–>», A ← A – C, C ← 2C, вызвать вспомогательный алгоритм transfusion; иначе перейти к следующему шагу.

Шаг 6. Если B ≥ A, то вывести «–21–>», B ← B – A, A ← 2A, вызвать вспомогательный алгоритм transfusion; иначе к следующему шагу.

Шаг 7. Если B ≥ C, то вывести «–23–>», B ← B – C, C ← 2C, вызвать вспомогательный алгоритм transfusion; иначе перейти к следующему шагу.

Шаг 8. Если C ≥ A, то вывести «–31–>», C ← C – A, A ← 2A, вызвать вспомогательный алгоритм transfusion; иначе перейти к следующему шагу.

Шаг 9. Если C ≥ B, то вывести «–32–>», C ← C – B, B ← 2B, вызвать вспомогательный алгоритм transfusion; иначе перейти к следующему шагу.

Шаг 10. Конец (вспомогательного) алгоритма.

Рассмотрим процессы переливаний, которые определяет вспомогательный рекурсивный алгоритм transfusion версии 0 в семи сгенерированный вариантах изначальной наполненности сосудов. При

этом решение задачи будем называть успешным, если соответствующий процесс переливаний конечен и останавливается с освобождением одного из сосудов.

Вариант 1) (8, 3, 11) –12–> (5, 6, 11) –21–> (10, 1, 11) –12–> (9, 2, 11) –12–> (7, 4, 11) –12–> (3, 8, 11) –21–> (6, 5, 11) –12–> (1, 10, 11) –21–> (2, 9, 11) –21–> (4, 7, 11) –21–> (8, 3, 11) и так далее (рис. 1). Следовательно, количество переливаний n = ∞.

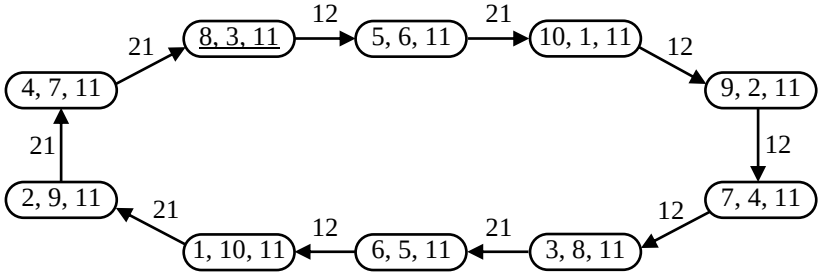

Рис. 1. Цикл бесконечных переливаний

Вариант 2) (10, 2, 6) –12–> (8, 4, 6) –12–> (4, 8, 6) –21–> (8, 4, 6) (Количество переливаний n = ∞).

Вариант 3) (11, 14, 3) –13–> (8, 14, 6) –13–> (2, 14, 12) –21–> (4, 12, 12) –21–> (8, 8, 12) –12–> (0, 16, 12) (успех, количество переливаний n = 5).

Вариант 4) (4, 7, 2) –13–> (2, 7, 4) –21–> (4, 5, 4) –13–> (0, 5, 8) (успех, n = 3).

Вариант 5) (14, 5, 8) –12–> (9, 10, 8) –13–> (1, 10, 16) –21–> (2, 9, 16) –21–> (4, 7, 16) –21–> (8, 3, 16) –12–> (5, 6, 16) –21–> (10, 1, 16) –12–> (9, 2, 16) –12–> (7, 4, 16) –12–> (3, 8, 16) –21–> (6, 5, 16) –12–> (1, 10, 16) (бесконечный цикл, n = ∞).

Вариант 6) (5, 4, 6) –12–> (1, 8, 6) –21–> (2, 7, 6) –21–> (4, 5, 6) –21–> (8, 1, 6) –12–> (7, 2, 6) –12–> (5, 4, 6) (бесконечный цикл, n = ∞).

Вариант 7) (7, 6, 12) –12–> (1, 12, 12) –21–> (2, 11, 12) –21–> (4, 9, 12) –21–> (8, 5, 12) –12–> (3, 10, 12) –21–> (6, 7, 12) –21–> (12, 1, 12) –12–> (11, 2, 12) –12–> (9, 4, 12) –12–> (5, 8, 12) –21–> (10, 3, 12) –12–> (7, 6, 12) (бесконечный цикл, n = ∞).

Анализ процессов переливаний говорит о том, что только в вариантах 3 и 4 задача решена успешно, т.е. освобождён один из сосудов за пять и три переливания соответственно. Однако, решение в варианте 3 могло быть получено за четыре переливания вместо пяти, если бы после наступления состояния наполненности сосудов (4, 12, 12) с одинаковым объёмом воды во втором и третьем сосудах было

выполнено переливание 23 или 32 с получением состояний (4, 0, 24) и (4, 24, 0) соответственно.

В вариантах 1, 2, 5, 6 и 7 процессы переливаний попадают в некоторые бесконечные циклы.

Версия 1 рекурсивного алгоритма transfusion, по сравнению с его начальной версией – рекурсивного выполнения допустимых переливаний в лексикографическом порядке и ожидания освобождения одного из сосудов, предусматривает в процессе переливаний контроль выравнивания объёмов воды в каких-либо двух сосудах.

Приведём процессы переливаний, определяемые вспомогательным рекурсивным алгоритмом transfusion версии 1 в вариантах 3 и 7, поскольку только они отличаются от соответствующих процессов по алгоритму версии 0.

Вариант 3) (11, 14, 3) –13–> (8, 14, 6) –13–> (2, 14, 12) –21–> (4, 12, 12) –23–> (4, 0, 24) (успех, количество переливаний n = 4).

Вариант 7) (7, 6, 12) –12–> (1, 12, 12) –23–> (1, 0, 24) (успех, количество переливаний n = 2).

Анализ этих процессов говорит о том, что в варианте 3, как и ожидалось при контроле выравнивания объёма воды в каких-либо двух сосудах, решение получено за четыре переливания вместо пяти: после наступления состояния наполненности сосудов (4, 12, 12) было выполнено переливание 23 с получением состояния (4, 0, 24). В варианте 7 вместо зацикливания алгоритма transfusion версии 0 решение получено всего за два переливания. Однако в вариантах 1, 2, 5 и 6 процессы переливаний по-прежнему попадают в некоторые бесконечные циклы.

Во вспомогательном рекурсивном алгоритме transfusion версии 2 мы избавляемся от попадания процессов переливаний в бесконечные циклы за счёт использования внешней, по отношению к вспомогательному рекурсивному алгоритму, памяти. В ней хранятся состояния наполненности сосудов (A, B, C), начиная с исходного, по мере их наступления. При этом в процессе выполнения переливаний, если предполагаемое алгоритмом переливание приводит к состоянию наполненности, которое отсутствует в памяти состояний, то оно записывается в эту память и переливание реализуется; в противном случае предпринимается аналогичная попытка выполнения альтернативного переливания. Для экономии памяти ЭВМ и машинного времени при программной реализации алгоритма упомянутая выше внешняя память организована в виде бинарного дерева и использован алгоритм поиска по дереву со вставкой [4], названный нами именем «search_insert»:

Рассмотрим только те процессы переливаний по алгоритму transfusion версии 2, которые по алгоритму предыдущей версии были бесконечными.

Вариант 1) (8, 3, 11) –12–> (5, 6, 11) –21–> (10, 1, 11) –12–> (9, 2, 11) –12–> (7, 4, 11) –12–> (3, 8, 11) –21–> (6, 5, 11) –12–> (1, 10, 11) –21–> (2, 9, 11) –21–> (4, 7, 11) –31–> (8, 7, 7) –23–> (8, 0, 14) (успех, количество переливаний n = 11).

Вариант 2) (10, 2, 6) –12–> (8, 4, 6) –12–> (4, 8, 6) –23–> (4, 2, 12) –12–> (2, 4, 12) –31–> (4, 4, 10) –12–> (0, 8, 10) (успех, количество переливаний n = 6).

Вариант 5) (14, 5, 8) –12–> (9, 10, 8) –13–> (1, 10, 16) –21–> (2, 9, 16) –21–> (4, 7, 16) –21–> (8, 3, 16) –12–> (5, 6, 16) –21–> (10, 1, 16) –12–> (9, 2, 16) –12–> (7, 4, 16) –12–> (3, 8, 16) –21–> (6, 5, 16) –31–> (12, 5, 10) –12–> (7, 10, 10) –23–> (7, 0, 20) (успех, n = 14).

Вариант 6) (5, 4, 6) –12–> (1, 8, 6) –21–> (2, 7, 6) –21–> (4, 5, 6) –21–> (8, 1, 6) –12–> (7, 2, 6) –13–> (1, 2, 12) –21–> (2, 1, 12) –31–> (4, 1, 10) –12–> (3, 2, 10) –12–> (1, 4, 10) –21–> (2, 3, 10) –31–> (4, 3, 8) –12–> (1, 6, 8) –21–> (2, 5, 8) –32–> (2, 10, 3) –21–> (4, 8, 3) –21–> (8, 4, 3) (неудача, n = 17).

Сравнение данных решений с результатами работы алгоритма transfusion версии 1 позволяет сделать следующие выводы.

В вариантах 1, 2 и 5 достигнут прорыв – вместо бесконечного числа переливаний получаем успешное решение задачи за 11, 6 и 14 шагов соответственно.

Например, процесс переливаний в варианте 1 протекает таким образом.

Начальное состояние (8, 3, 11) помещается в корень пустого бинарного дерева.

Осуществление предполагаемых алгоритмом переливаний 12, 21, 12, 12, 12, 21, 12, 21, 21 последовательно приводит к состояниям наполненности (5, 6, 11), (10, 1, 11), (9, 2, 11), (7, 4, 11), (3, 8, 11), (6, 5, 11), (1, 10, 11), (2, 9, 11) и (4, 7, 11), которые вставляются в дерево.

Очередное предполагаемое алгоритмом переливание 21 приведёт к состоянию наполненности (8, 3, 11), которое присутствует в дереве и поэтому в дерево не вставляется и переливание не осуществляется.

Следующее, предполагаемое алгоритмом, переливание 31 приведёт к состоянию наполненности (8, 7, 7), которое отсутствует в дереве и поэтому вставляется в дерево, которое приобретает вид, изображённый на рис. 2. Переливание (4, 7, 11) –31–> (8, 7, 7) осуществляется.

Наконец в состоянии наполненности (8, 7, 7) алгоритм обнаруживает равенство объёмов воды во втором и третьем сосудах,

выполняется последнее переливание (8, 7, 7) –23–> (8, 0, 14), что приводит к успешному решению задачи.

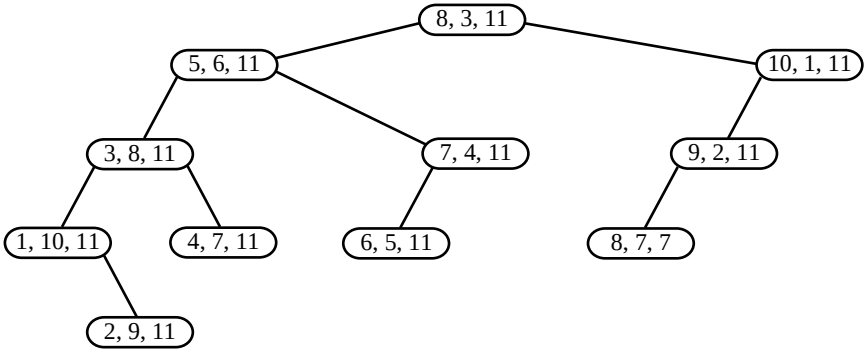

Рис. 2. Дерево наступивших состояний наполненности сосудов

Заметим, что вариант 6 заслуживает особенного внимания, в нём вместо бесконечного числа переливаний по алгоритму версии 1 получаем некоторое конечное состояние наполненности сосудов за 17 шагов, которое, однако, не является решением задачи, поскольку ни один сосудов не освобождён от жидкости.

Анализ данной ситуации говорит о том, что полученное в процессе переливаний конечное состояние наполненности сосудов (8, 4, 3) является тупиковым состоянием поиска. Эта ситуация характерна тем, что любое дальнейшее допустимое переливание из 12, 13 или 23 приводит к уже встречавшемуся в процессе данных переливаний состоянию наполненности сосудов (4, 8, 3), (5, 4, 6) и (8, 1, 6) соответственно. И поэтому алгоритм завершает свою работу, так и не дождавшись освобождения ни одного из сосудов.

Кроме того, пути переливаний, ведущие от изначальной наполненности сосудов (A, B, C) к освобождению одного из них, в полученных успешных решениях задачи, как правило, не являются кратчайшими.

Упомянутые в вариантах 1, 2, 4, 5 и 6 не кратчайшие пути переливаний назовём явно сократимыми путями.

В рекурсивном алгоритме transfusion версии 3 откажемся от использования внешней по отношению к алгоритму памяти по следующим причинам. Во-первых, при этом мы не всегда получаем успешное решение задачи, а, во-вторых, получаемые пути к освобождению одного из сосудов, как выяснилось, далеко не всегда являются кратчайшими путями.

Теперь переливания будем выполнять сериями, а в каждой серии они будут направлены на отливание из сосуда средней наполненности максимального объёма воды, кратного объёму жидкости в сосуде с наименьшей наполненностью. При этом используется третий сосуд и по-прежнему осуществляется контроль выравнивания объёмов воды в каких-либо двух сосудах:

Шаг 1. Начало алгоритма.

Шаг 2. Если A = 0 или B = 0 или C = 0, то конец алгоритма; иначе перейти к следующему шагу.

Шаг 3. Если A = B, то вывести «–12–>», A ← A – B, B ← 2B, вызвать transfusion; иначе перейти к следующему шагу.

Шаг 4. Если A = C, то вывести «–13–>», A ← A – C, C ← 2C, вызвать transfusion; иначе перейти к следующему шагу.

Шаг 5. Если B = C, то вывести «–23–>», B ← B – C, C ← 2C, вызвать transfusion; иначе перейти к следующему шагу.

Шаг 6. Если A < B и B < C, то выполнить шаги 7 и 8; иначе перейти к шагу 15.

Шаг 7. Переменной q присвоить результат целочисленного деления B на A (q ← B div A).

Шаг 8. Пока q ≠ 0 выполнить шаги 9, 10, …, 13.

Шаг 9. Переменной p присвоить значение остатка от целочисленного деления q на 2 (p ← q mod 2).

Шаг 10. Если p = 1, то вывести «–21–>», B ← B – A; иначе вывести «–31–>», C ← C – A. A ← 2A.

Шаг 11. Если A = C, то вывести «–13–>», A ← A – C, C ← 2C, вызвать алгоритм transfusion, q ← 0.

Шаг 12. Если B = C, то вывести «–23–>», B ← B – C, C ← 2C, вызвать алгоритм transfusion, q ← 0.

Шаг 13. q ← q div 2.

Шаг 14. Если A ≠ 0 и B ≠ 0 и C ≠ 0, то вызвать алгоритм transfusion.

Заметим, что шаги 15, 16, …, 23 аналогичны шагам 6, 7, …, 14.

Шаг 15. Если A < C и C < B, то выполнить шаги 16 и 17; иначе перейти к шагу 24.

Шаг 16. q ← C div A.

Шаг 17. Пока q ≠ 0 выполнить шаги 18, 10, …, 22.

Шаг 18. p ← q mod 2.

Шаг 19. Если p = 1, то вывести «–31–>», C ← C – A; иначе вывести «–21–>», B ← B – A. A ← 2A.

Шаг 20. Если A = B, то вывести «–12–>», A ← A – B, B ← 2B, вызвать алгоритм transfusion, q ← 0.

Шаг 21. Если C = B, то вывести «–32–>», C ← C – B, B ← 2B, вызвать алгоритм transfusion, q ← 0.

Шаг 22. q ← q div 2.

Шаг 23. Если A ≠ 0 и B ≠ 0 и C ≠ 0, то вызвать алгоритм transfusion.

Шаги 24, 25, …, 32 аналогичны шагам 15, 16, …, 23.

Шаги 33, 34, …, 41 аналогичны шагам 24, 25, …, 32.

Шаги 42, 43, …, 50 аналогичны шагам 33, 34, …, 41.

Шаги 51, 52, …, 59 аналогичны шагам 42, 43, …, 50.

Шаг 60. Конец (вспомогательного) алгоритма.

Рассмотрим процессы переливаний по вспомогательному алгоритму transfusion версии 3 для всех семи вариантов изначальной наполненности сосудов.

Вариант 1) (8, 3, 11) –32–> (8, 6, 8) –13–> (0, 6, 16) (успех, количество переливаний n = 2).

Вариант 2) (10, 2, 6) –32–> (10, 4, 4) –32–> (10, 8, 0) (успех, количество переливаний n = 2).

Вариант 3) (11, 14, 3) –13–> (8, 14, 6) –13–> (2, 14, 12) –21–> (4, 12, 12) –32–> (4, 24, 0) (успех, количество переливаний n = 4).

Вариант 4) (4, 7, 2) –23–> (4, 5, 4) –13–> (0, 5, 8) (успех, n = 2).

Вариант 5) (14, 5, 8) –32–> (14, 10, 3) –23–> (14, 7, 6) –23–> (14, 1, 12) –12–> (13, 2, 12) –12–> (11, 4, 12) –32–> (11, 8, 8) –32–> (11, 16, 0) (успех, n = 7).

Вариант 6) (5, 4, 6) –12–> (1, 8, 6) –21–> (2, 7, 6) –31–> (4, 7, 4) –31–> (8, 7, 0) (успех, n = 4).

Вариант 7) (7, 6, 12) –12–> (1, 12, 12) –23–> (1, 0, 24) (успех, количество переливаний n = 2).

Сравнительный анализ данных решений с результатами работы алгоритма transfusion версии 2 позволяет сделать следующие выводы.

В вариантах 3 и 7 получаем решения задачи без уменьшения числа переливаний. В вариантах 1, 2, 4 и 5 получаем решения, в общем случае, за значительно меньшее число шагов. Наконец, и в варианте 6 изначальной наполненности сосудов получаем успешное решение задачи, причём всего за четыре переливания вместо приведших к тупиковой ситуации 17 переливаний по алгоритму transfusion версии 2.

Заметим, что ни один из семи путей к освобождению сосудов не содержит пар эквивалентных состояний их наполненности, следовательно, каждый из них является или кратчайшим путём к искомому состоянию, или, по крайней мере, не является явно сократимым путём.

Результативность разработанных версий 0, 1, 2 и 3 рекурсивного алгоритма transfusion сведём в таблицу.

Таблица

№ вари-анта	Изначальная наполненность сосудов			Результативность решения задачи и число выполненных переливаний для рекурсивного алгоритма transfusion			
	A	B	C	версия 0	версия 1	версия 2	версия 3
1	8	3	11	$n = \infty$	$n = \infty$	$n = 11$	$n = 2$
2	10	2	6	$n = \infty$	$n = \infty$	$n = 6$	$n = 2$
3	11	14	3	$n = 5$	$n = 4$	$n = 4$	$n = 4$
4	4	7	2	$n = 3$	$n = 3$	$n = 3$	$n = 2$
5	14	5	8	$n = \infty$	$n = \infty$	$n = 14$	$n = 7$
6	5	4	6	$n = \infty$	$n = \infty$	$n = 17$	$n = 4$
7	7	6	12	$n = \infty$	$n = 2$	$n = 2$	$n = 2$

Заметим, что каждое из решений олимпиадной задачи «Переливание» во всех вариантах изначальной наполненности сосудов (A, B, C), является более результативным по сравнению с предыдущим. При этом рекурсивный алгоритм версии 3 всегда гарантированно находит путь переливаний к освобождению одного из сосудов, который является или кратчайшим путём, или, по крайней мере, не является явно сократимым путём.

Заключение

Спроектированы четыре версии (0, 1, 2 и 3) рекурсивного алгоритма решения олимпиадной задачи «Переливания». При этом каждая его старшая версия повышает результативность решения олимпиадной задачи. Алгоритм версии 3 во всех рассмотренных вариантах исходной наполненности сосудов гарантированно находит кратчайший путь переливаний, который ведёт к освобождению одного из них, или, по крайней мере, не явно сократимый путь.

Список литературы

1. Олимпиады по программированию. I Всероссийская заочная олимпиада школьников по информатике [Электронный ресурс]: Задача L. Переливания. – Режим доступа: http://www.olympiads.ru/zaoch/2006/problems/l.shtml
2. Уайс, М.А. Организация структур данных и решение задач на C++ / М.А. Уайс. – М. : ЭКОМ Паблишерз, 2008. – 896 с.
3. Уинер, Р. Язык Турбо Си / Р. Уинер. – М.: Мир, 1991. – 384 с.
4. Кнут, Д. Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск / Д. Кнут. – М.: Мир, 1978. – 846 с.