# Задача 1

Задачу можно представить, как граф переходов (рис. 1), где ребра - переходы(переливы), а вершины - состояния (уровни кваса в кувшинах). Тогда решением задачи будет найти такие не пересекающиеся пути, которые приводят к одинаковым уровням кваса в любых двух кувшинах и нулевому в третьем.

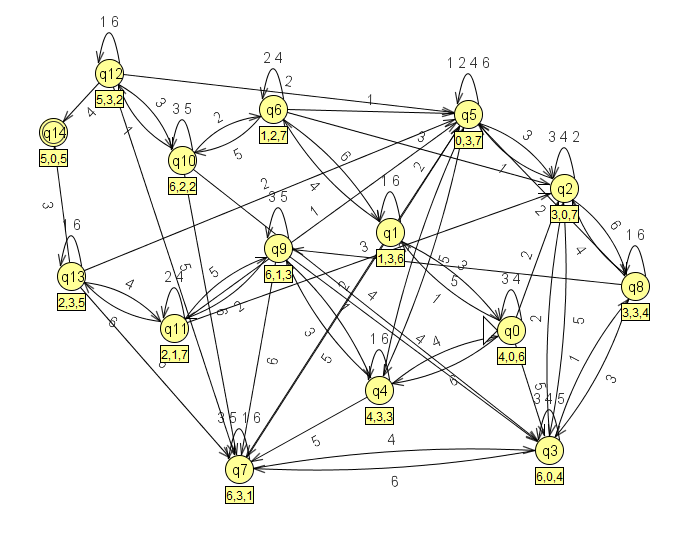


Рисунок 1  
qN в вершинах – номер состояния (в порядке обнаружения их алгоритмом), q0 – начальное, q14 - конечное  
a,b,c под вершинами – уровни кваса в кувшинах  
1,2,3,4,5,6 на ребрах – переливания 1->2, 1->3, 2->1, 2->3, 3->1, 3->2 соответственно

Решена при помощи некого подобия RAM-машины (рис. 2, машина с произвольным доступом к памяти, эквивалентна машине Тьюринга). В качестве памяти используются поля всех столбцов рекурсивного CTE (Обобщенного табличного выражения) на текущей итерации.

current\_state и prev\_state содержат текущее и предыдущее состояния машины соответственно.  
Состояния 1-6 – переливания между кувшинами, 0 и 7 – нужны для подготовки и синхронизации.

memory содержит граф переходов в виде иерархии в формате parent\_id@id@v11-v12-v13.

Например, 0@1@4-0-6 1@2@1-3-6 1@3@3-0-7 означает, что  
4-0-6 – начальное состояние, его id = 1, parent\_id = 0 - у него нет предыдущего состояния,   
1-3-6 – его id = 2, parent\_id = 1 (состояние 4-0-6), а получено оно может быть переливанием из 1-го кувшина в 2-ой  
3-0-7 – его id = 3, parent\_id = 1 (состояние 4-0-6), а получено оно может быть переливанием из 1-го кувшина в 3-ий  
mem\_ptr указывает из какого состояния в данный момент рассчитываются результаты переливаний.

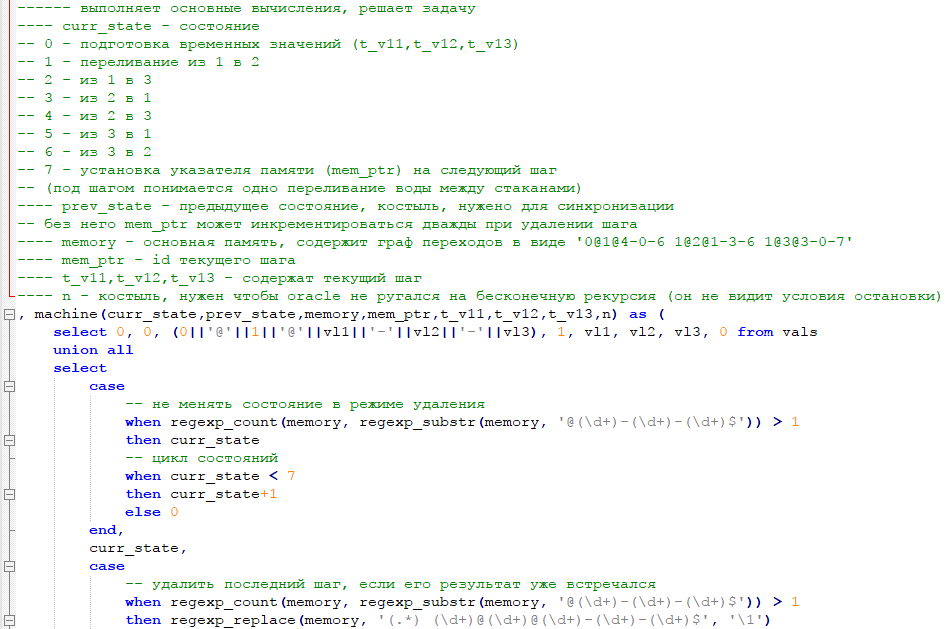
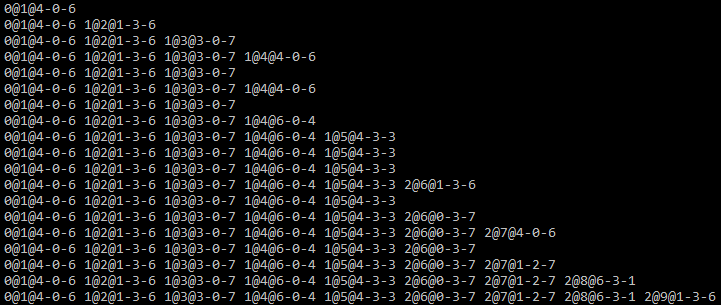


Рисунок 2

Работает всё это по следующему принципу:   
Начинаем с состояния 4-0-6, перельём из 1-го во 2-ой, получим 1-3-6, сохраняем его в память.  
На следующей итерации сначала проверим, не приходили ли мы к предыдущему состоянию ранее, если да – удаляем из памяти последнее состояние (откатываемся на один шаг назад).   
Далее делаем следующее переливание 1 –> 3 из того же состояния 4-0-6, опять сохраняем в память.  
На следующей итерации опять проверяем не было ли предыдущего состояния ранее.  
Повторяем эти действия пока не закончатся возможные варианты переливаний из состояния 4-0-6.  
Переходим к следующему состоянию 1-3-6 (если оно было удалено из памяти, то к 4-0-6 и т.д.) и так же пробуем 6 вариантов переливаний.  
Строка memory во время этих итераций будет выглядеть как на рис. 3.  


Рисунок

Всё это повторяем до тех пор пока не закончатся состояния. В результате работы получим граф как на рис. 4. В строке memory это будет выглядеть так: 0@1@4-0-6 1@2@1-3-6 1@3@3-0-7 1@4@6-0-4 1@5@4-3-3 2@6@0-3-7 2@7@1-2-7 2@8@6-3-1 3@9@3-3-4 5@10@6-1-3 7@11@6-2-2 10@12@2-1-7 11@13@5-3-2 12@14@2-3-5 13@15@5-0-5  
Два пути от q0 до q14 и будут ответами.

Далее CTE parse разбивает строку с состояниями (memory) на отдельные состояния, их id и parent\_id.  
А CTE parse2 преобразует эту иерархию в строки при помощи sys\_connect\_by\_path, и выводит лишь те, которые содержат в конце искомое конечное состояние.

Все подробности в комментариях в коде.

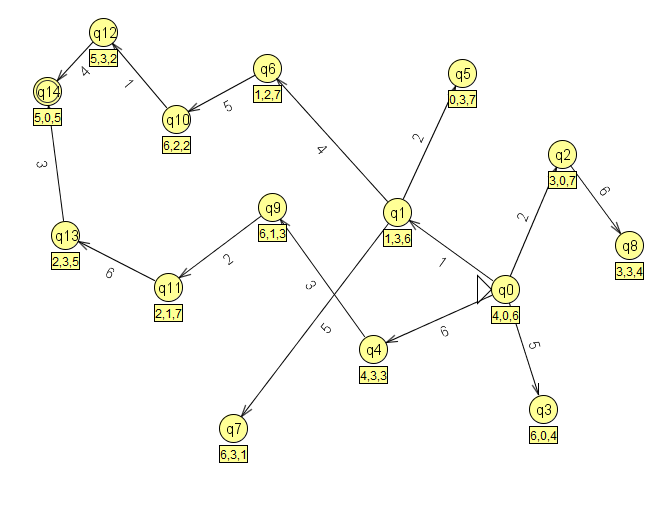


Рисунок 4