Консенсус и сила атомарных операций

На лекции мы построили **wait-free иерархию** для атомарных операций, которые доступны в архитектурах современных процессоров:

Операция	**Число консенсуса**
Read + Write	1
Коммутирующие и затирающие операции (Test-And-Set, Fetch-And-Add)	2
Compare-And-Set	inf

Число консенсуса для операции (класса операций) определяется как максимальное число потоков, которые могут прийти к консенсусу (выбрать общее значение), используя только атомарные чтения/записи и данную операцию (операции из данного класса).

Попробуем представить мир, в котором нам доступны новые атомарные операции:

A. Атомарная запись в n ячеек - 2+3 баллов

Рассмотрим операцию Multi-Write(r_1, v_1, r_2, v_2, ..., r_n, v_n), которая атомарно записывает значения v_1, v_2, ..., v_n в ячейки r_1, r_2, ..., r_n соответственно.

Докажите, что такая операция вместе с обычными точечными чтениями / записями позволяет решить wait-free консенсус по крайней мере для n потоков.

Подсказка: Сначала придумайте протокол для атомарной записи в две ячейки памяти.

За решение для записи в две ячейки - 2 балла, за решение для записи в n ячеек - 5 баллов.

B. Атомарное чтение из n ячеек - 1 + 3 балла

A) Покажите, что атомарное чтение сразу из n ячеек (вместе с обычными операциями чтения/записи одной ячейки) имеет число консенсуса 1.

B) Покажите, как эмулировать атомарное чтение из двух ячеек памяти с помощью обычных чтений и записей над одной ячейкой. Можно считать, что ширина ячейки памяти не ограничена.

C. Консенсус с помощью пустых очередей - 3 балла

Придумайте протокол консенсуса для двух потоков с помощью двух **изначально пустых** wait-free очередей.

Очереди умеют только две атомарные операции: Q.Enqueue(x) - добавить элемент в конец очереди, и Q.Dequeue() - извлечь элемент из начала очереди.

Подсказка:

Нетрудно придумать протокол, с помощью которого выделенный заранее поток сможет понять, пришел он первым или вторым. Но нужно, чтобы то же самое понял и другой поток.

Потоки не обязаны действовать симметрично!

Мораль задачи:

Мы знаем, что на атомарных чтениях и записях нельзя решить задачу консенсуса даже для двух потоков. Если при этом мы научились решать эту задачу для двух потоков с помощью очередей, то тем самым доказали важный отрицательный результат:

Нельзя реализовать wait-free очередь, используя только лишь атомарные операции чтения и записи.

D. Memory-to-Memory Swap - 2 балла

Рассмотрим операцию Mem-Swap(x, y), которая атомарно меняет местами содержимое двух ячеек памяти x и y и ничего при этом не возвращает.

Найдите число консенсуса для этой операции.
Предложенный вами алгоритм должен быть эффективным.

Замечание:

Операция Mem-Swap на первый взгляд похожа на r.Exchange(v), но они отличаются: Exchange меняет только одну ячейку в разделяемой памяти, в то время как Mem-Swap затрагивает две ячейки.

E. Memory-to-Memory Copy - 3 балла

Рассмотрим операцию Mem-Copy(x, y), которая атомарно копирует значение из ячейки x в ячейку y, и ничего не возвращает (т.е. скопированное значение можно узнать только с помощью последующего чтения).

Найдите число консенсуса для этой операции.

Замечание:

Как правило, поток, который "пришел" первым (т.е. первым выполнил некоторую атомарную операцию записи в память), выигрывает консенсус, а остальные потоки выбирают предложенное им значение. В данном случае решение будет хитрее.

Во всех задачах выше алгоритму известно число потоков n.

F. Робастность wait-free иерархии - 3 балла

Мы определили число консенсуса для операции X как максимальное число потоков, которые могут выбрать общее значение, используя только атомарные регистры с операциями Read и Write и операцию X.

Возникает естественный вопрос: а какого результата можно добиться, если использовать сразу несколько разных операций? Можно ли комбинировать более слабые операции и строить из них более сильные объекты (с большим числом консенсуса)?

На лекции мы рассмотрели пример: оказалось, что комбинирование перезаписывающих и коммутирующих RMW-операций (Test-And-Set и Fetch-And-Add) с числом консенсуса 2 не помогает решать консенсус для трех потоков.

В этой задаче мы рассмотрим другой пример:

Рассмотрим класс операций, в который помимо Read и Write входят операции Increment и Multiply(v). Операция Increment увеличивает значение регистра на 1, операция Multiply(v) - умножает значение

регистра на аргумент v. Ни одна из операций не возвращает значения, для его извлечения нужно явно выполнить Read.

Несложно показать, что каждая из операций этого класса имеет число консенсуса 1.

Рассмотрим задачу **бинарного** консенсуса, в которой потоки могут предлагать только значения 0 и 1.

Покажите, что заданный класс операций позволяет решать бинарный консенсус для произвольного числа потоков.

Предложенный алгоритм не должен знать число потоков n.

Требование к решениям: В каждой из задач вы должны не просто предложить алгоритм достижения консенсуса, но и привести обоснование его корректности, т.е. показать, что выполняются свойства agreement, validity, wait-freedom.