**Задание 1**

*Дано*

Данные хранятся в блоках памяти. Пользователь отправляет команды чтения данных из блоков в некотором порядке. Некоторые из этих команд можно объединить в кластеры, т.е. в группы команд в которых блоки часто читаются последовательно (друг за другом). Пусть уже известны 2 алгоритма кластеризации команд, но неизвестно как сравнить какой из алгоритмов для множества последовательностей команд.

*Пример*

Последовательность команд: 2 5 7 2 5 3 8 7 2 5 7 2 3 5 7 2 5 7

Алгоритм А1 может выделить кластер 2 5 7

Алгоритм А2 может выделить кластер 5 7 2 5

*Найти*

предложить метрику для оценки качества алгоритмов кластеризации в двух случаях:

a) ограничения на память и вычислительные ресурсы отсутствуют,

b) есть ограничения на память и вычислительные ресурсы.

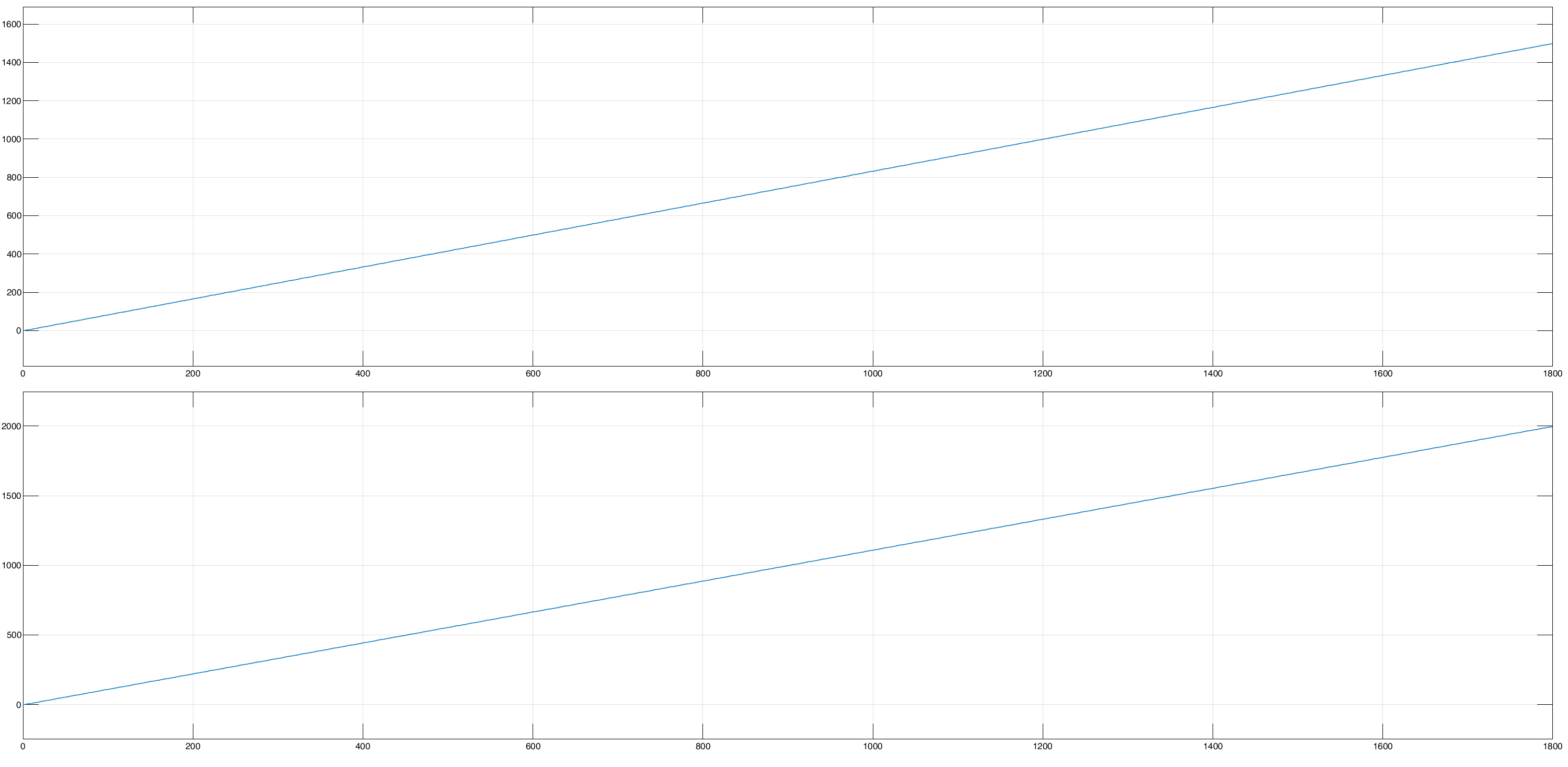
(написать явно формулу и/или алгоритм вычисления метрики)

*Решение*

Среди метрик используемых в кластерном анализе можно выделить «вариации на тему» евклибова расстояния, т.е. метрики которые так или иначе используют разность между координатами многомерного вектора кластера и соответствующего текущего вектора из потока комманд. Однако данные метрики не корректны для данного случая – поскольку важно полное совпадение координат (речь идет о координатах в отдельности, а не векторе целиком). Если координаты не совпадают то неважно на сколько – это уже просто другая комманда. Очевидно, что более эффективный кластер тот, который имеет наибольшую плотность совпадений в реальной последовательности команд. Выявить частоту таких совпадений можно используя метрику Хемминга, однако в кластерном анализе используют «Процент несогласия» - меру инвариантную к длине кластера (как раз наш случай – длины кластеров А1 и А2 разные). Теоретически длина последовательности команд ***x*** бесконечна, а кластер ***c*** имеет конечную длину *L* представляющего его числового вектора – значит измерение расстояния является непрерывным процессом на каждом шагу поступления новой команды. Блок измерения расстояний должен быть инвариантен ко времени и по сути является структурой с конечным импульсным откликом каждый *n*-й выход которого определяется выражением:

Также стоит учитывать, что частичные не совпадения (т.е. не нулевые расстояния не равные максимальному значению 1) также имеют значения, а значит более эффективный кластер тот, который имеет меньшую площадь под графиком процесса на выходе блока измерения расстояний. Если нет ограничения на память и вычислительные ресурсы, то можно интегрировать (накапливать) выход блока измерения расстояний бесконечно и иметь таким образом оценку эффективности того или иного кластера:

Так для примера в задании можем установить, что кластер А1 более эффективен:

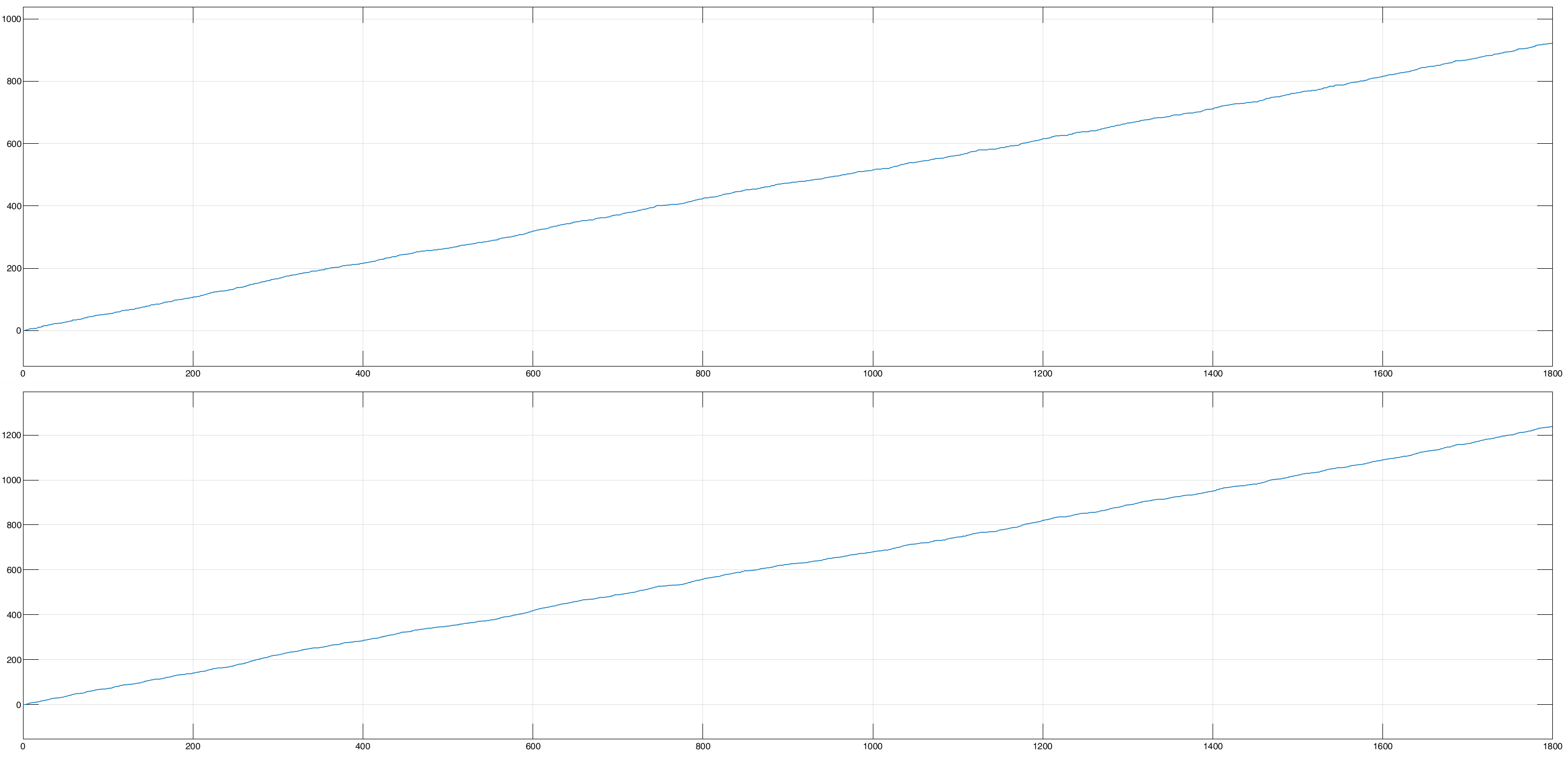


А2

А1

Рисунок 1 – Эксперимент для примера в задании для 1800 шагов (последовательность команд циклически повторяется)

Вообще алгоритм А1 оказывается более эффективным и при случайном поступлении команд с равномерным распределением между 2 и 8:



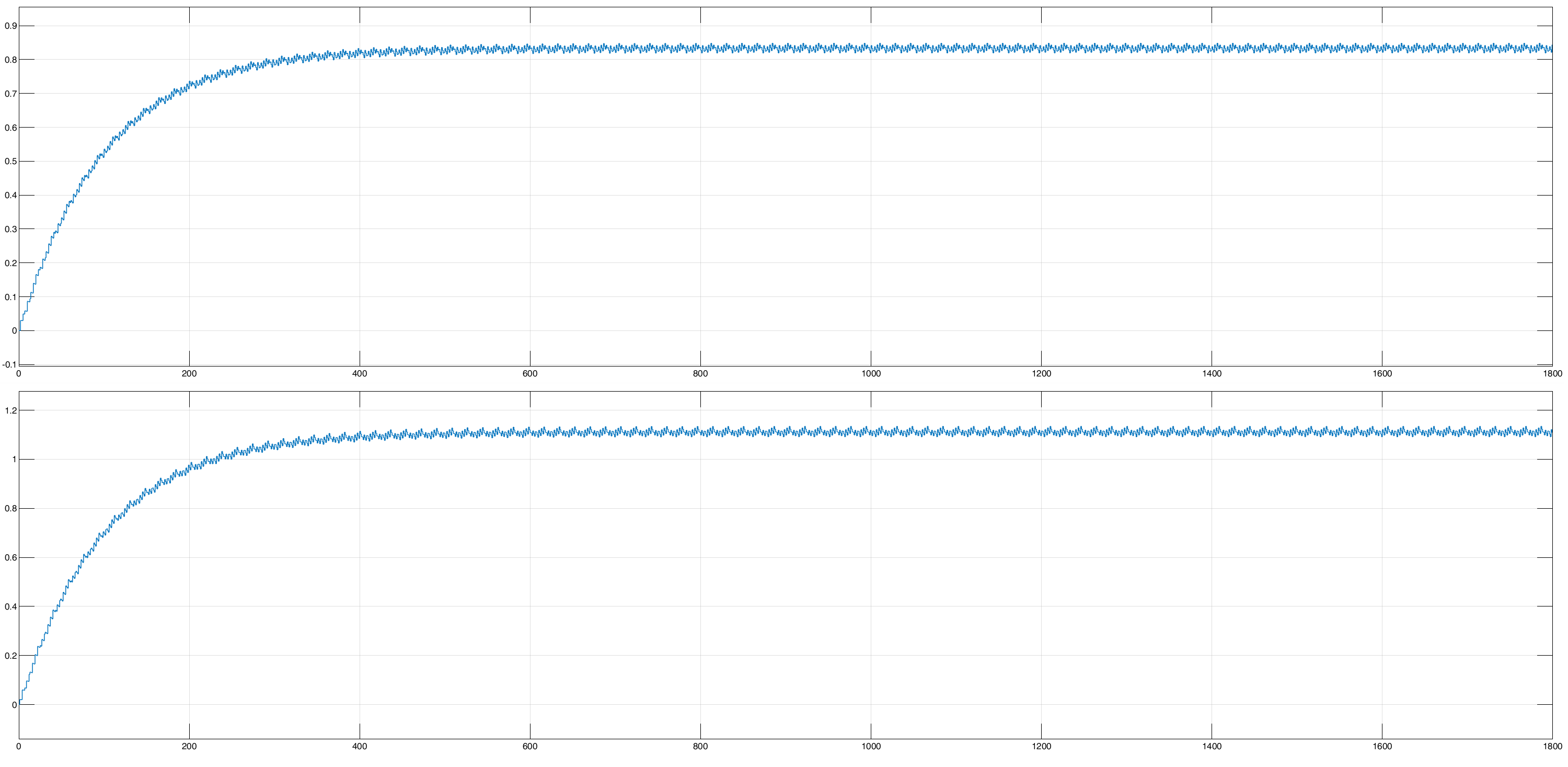
А2

А1

Рисунок 2 – Эксперимент для примера в задании для 1800 шагов (последовательность команд случайна между 2 и 8)

Когда имеет место ограничение на память и вычислительные ресурсы мы не можем накапливать бесконечно, тогда необходимо либо периодически сбрасывать аккумулятор, либо можно применить усреднение, например, вычисление экспоненциально среднего значения:

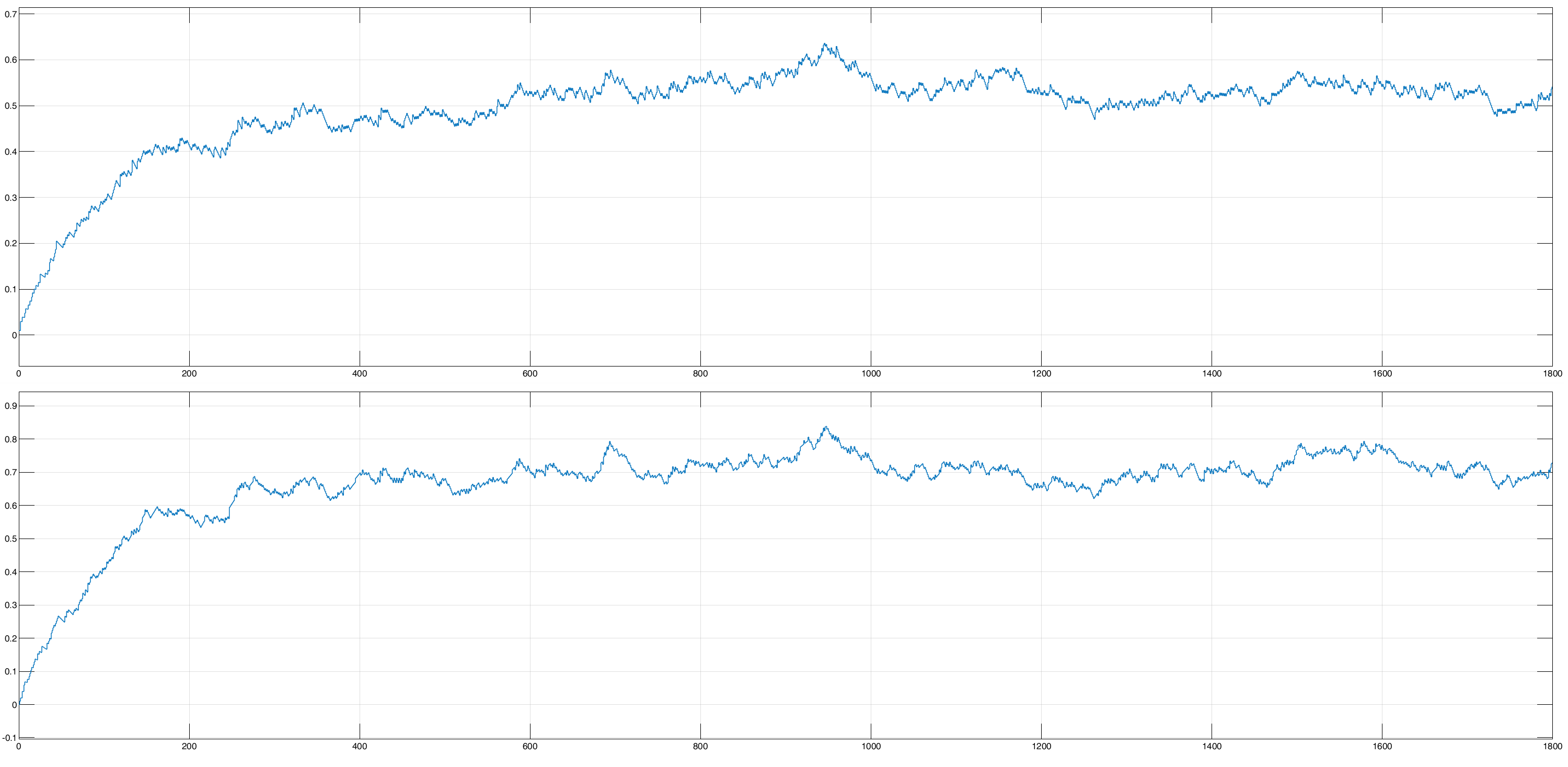
В таком варианте видно, что более эффективным оказался кластер А1 – как для последовательности из примера, так и для случайной последовательности с коэффициентом экспоненциального фильтра 0,01:



А2

А1

Рисунок 1 – Эксперимент для примера в задании для 1800 шагов (последовательность команд циклически повторяется)



А2

А1

Рисунок 2 – Эксперимент для примера в задании для 1800 шагов (последовательность команд случайна между 2 и 8)

**Задание 2**

*Дано*

Данные хранятся в блоках памяти. Операция полного удаления данных из блока довольно длительная, поэтому эта операция выполняется частями.

Длительность операции полного удаления является случайной величиной с известным распределением вероятностей.

Длительности всех операций частичного удаления данных (кроме последней операции), являются независимыми одинаково распределенными случайными величинами с известным распределением. Мат. ожидание последней операции меньше мат. ожидания остальных операций.

*Найти*

среднее количество операций частичного удаления за время полного удаления данных, указать условие применимости полученного решения, если это необходимо.

*Решение*

Поскольку длительность полного удаления является случайной величиной с известным распределнием вероятности, то мы можем считать что математической ожидание этой величины изветно. Поскольку длительности операций частичного удаления являются независмыми величинами с одинаковым распределением, то их математические ожидания можно считать равными. Также пусть математическое ожидание последней операции . Тогда среднее число операций частичного удаления (без учета последней):

Заметим что можно считать средним числом операций при равномерном распределении случайных величин.

**Задание 3**

*Дано*

Данные хранятся в блоках памяти. Для каждого блока памяти, есть счетчик количества чтений данных из этого блока. При достижении счетчиком порога T, данные из данного блока должны быть скопированы в другой блок. Поскольку блоков много и максимальные значения счетчика для каждого блока очень большие, то требуются значительные ресурсы. Для уменьшения ресурсов может использоваться вероятностный счетчик. Инкремент счетчика происходит не каждый раз при чтении данных из блока, а с некоторой постоянной вероятностью P.

*Найти*

Найти порог для вероятностного счетчика, при достижении которого фактическое количество чтений блока не превысит T с вероятностью 99%.

*Решение*

Очевидно что значение вероятностного порога связано с порогом вероятностью линейно, однако с учетом условной вероятности , решение имеет вид:

**Задание 4**

*Дано*

Данные хранятся в блоках памяти. Задано количество блоков N. Пользователь читает блоки в произвольном порядке (равномерное распределение).

*Найти*

a) вероятность того, что каждый блок был прочитан не более k раз, при условии что всего было m чтений,

b) предложить способ вычисления при больших N, m, k.

*Решение*

Поскольку чтение каждого блока равновероятно, то при больших можно считать что среднее число чтений каждого блока , тогда:

При больших N, m, k можно предложить использовать число блоков равным степени двойки, а вычисления производить, также при пересечении m порогов равных также степени двойки.

**Задание 5**

*Дано*

Дан двумерный массив размером B \* P, каждая из B строк которого представляет собой блок памяти, а каждый из P элементов блока - страницу памяти. Возможны операции четырех типов:

1. Сделать страницу p (элемент массива) в блоке b недействительной (инвалидировать);

2. Удалить блок памяти b. В результате чего все страницы блока будут действительными (валидными). После каждой такой операции счетчик удалений блока b увеличивается на 1;

3. Получить номер блока и номер страницы с минимальным значением (EC + 1) \* (NVP + 1), EC - значение счетчика удалений, NVP - число действительных страниц;

4. Получить номер блока и номер страницы с максимальным значением (EC + 1) \* (NVP + 1);

Изначально все страницы во всех блоках действительные, а значения счетчиков удалений равны нулю.

*Найти*

Предложить структуру данных, которая позволяет выполнять указанные операции над двумерным массивом.

a) с минимальными затратами дополнительной памяти;

b) c минимальными затратами времени;

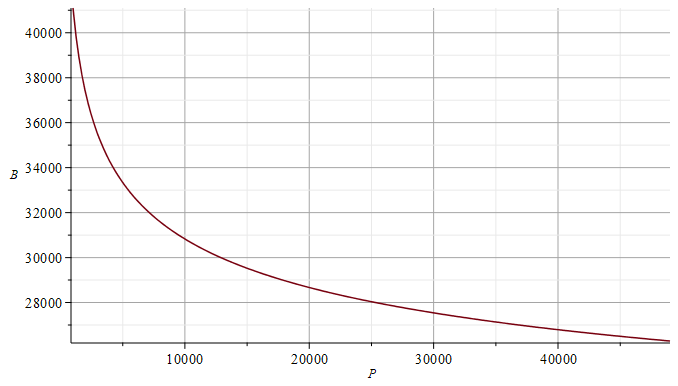
с) компромиссные варианты: какие максимальные значения B и P возможны, чтобы структура данных использовала не более Th Кб памяти (для определенности Th = 50).

*Решение*

Поскольку EC и NVP это характеристики блока, то все страницы в пределах одного блока имеют одинаковые параметры (EC + 1) \* (NVP + 1), тогда команды 3 и 4 касаются лишь блоков. Итак в структре В\*Р требуется два дополнительных столбца хранящих параметры EC и NVP.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| В0Р0 | … | В0Рm-1 | EC0 | NVP0 |
| В1Р0 | … | В1Рm-1 | EC1 | NVP1 |
| … | … | … | … | … |
| Вn-1Р0 | … | Вn-1Рm-1 | ECn-1 | NVPn-1 |

Поскольку ЕС счетчик удалений блока то его динамический диапазон не зависит от размерности таблицы, а максимально возможное число в NVP определяется числом страниц в блоке m, то общая оптимизация затрат памяти определяется затратами на хранение всех NVP и определяется выражением бит. Поскольку общий объем элементов , а множитель растет медленнее, то с точки зрения оптимизации памяти лучше таблицу представить в виде одной большой строки, но тогда каждая валидация будет сопровождаться удалением всех страниц, а это затраты времени и, если счетчик удалений ограничен, неправданная трата ресурса использования памяти. Очевидно, что оптимум с точки зрения времени удаления блока, доступа и ресурса является другой крайний случай – один столбец. Ниже представлен график зависисмости количества блоков от количества страниц в блоке при ограничении памяти выделяемой на NVP в 50 Кб.



Из него можно выбирать интересующее значение исходя изтого что рост P ведет к снижению памяти, а рост B к снижению времени доступа к страницам блока и экономии ресурса при удалении боков.

**Задание 6**

*Дано*

Дано 3 процессора, между которыми необходимо распределить N задач. Каждый из процессоров может выполнять задачи, суммарное время выполнения которых не превышает W. Каждая задача имеет время выполнения ai.

*Найти*

Предложить алгоритм, который распределяет задачи на все процессоры так, что:

1. Максимальное суммарное время выполнения всех задач на одном из процессоров (tmax) должно отличаться не более чем на δ от минимального суммарного времени выполнения на одном из процессоров (tmin), т.е. tmax - tmin ≤ δ и тем самым нагрузка должна быть распределена как можно более равномерно на процессоры;

2. Процессоры были максимально загружены в условиях ограничений, накладываемых пунктом 1.

Необязательно распределять все задачи.

*Решение*

Изначально происходит сортировка задач по времени выполнения отбольшего к меньшему. Далее просходит группировка задач для первого процессора от большего к меньшему с временем меньшим W, после для второго процессора тот же алгоритм, но с дополнительной проверкой, так чтобы его сумарное время не отличалось от первого более чем на δ/2. Если разница превышает δ/2 идет пропуск первой задачи в группе второго процессора и идет добор задач (по направлениюк меньшему). После проверка осуществляется занаво. Для третьего процессора повторяется алгоритм второго, но относительно суммарного времени второго и начиная с первой пропущенной задачи.

В случае второго условия можно отказаться от возвращения к пропущенным задачам.

**Задание 7**

*Дано*

Данные хранятся в блоках памяти. Дан массив, каждый элемент которого хранит информацию о командах чтения данных, а именно пару (адрес блока, количество читаемых блоков). В массиве возможно выделить подпоследовательности (кластеры) команд чтения, которые встречаются многократно. Подпоследовательности считаются кластерами, если в точности совпадают.

*Найти*

Предложить алгоритм нахождения подпоследовательностей команд чтения (кластеров):

1. Без ограничений по памяти (~Гигабайты) и времени (~часы);

2. С ограничением по памяти (~Килобайты) и времени (~секунды);

Критерии качества алгоритма кластеризации (нахождения кластеров) остаются на Ваше усмотрение.

Какие ограничения на кластеры необходимо накладывать?

Как оценить эффективность реализации алгоритмов с ограничением и без ограничения ресурсов?

Примером кластеризации является задача дефрагментации. В данном случае одинаковые подпоследовательности переносятся в одну область накопителя информации для лучшей производительности чтения. Соответственно, критериями качества кластеризации при такой постановке задачи является баланс между производительностью чтения и временем жизни диска после дефрагментации.

*Решение*

При остутсвтии ограничений естественным является накопление как можно большего числа команд и их перебор в поиске наиблее длинных кластеров с наибольшей плотностью. В случае ограничения времени и памяти необходим эффективный алгоритм кластеризации. На сегодняшний день перспективными являются алгоритмы кластеризации на основе машинного обучения нейронных сетей. Также интересным является подход не выявления наболее длинных кластеров с наибольшей плотностью и его хранение, а интерполяция команд.