# Анализ алгоритмов Семинар 4

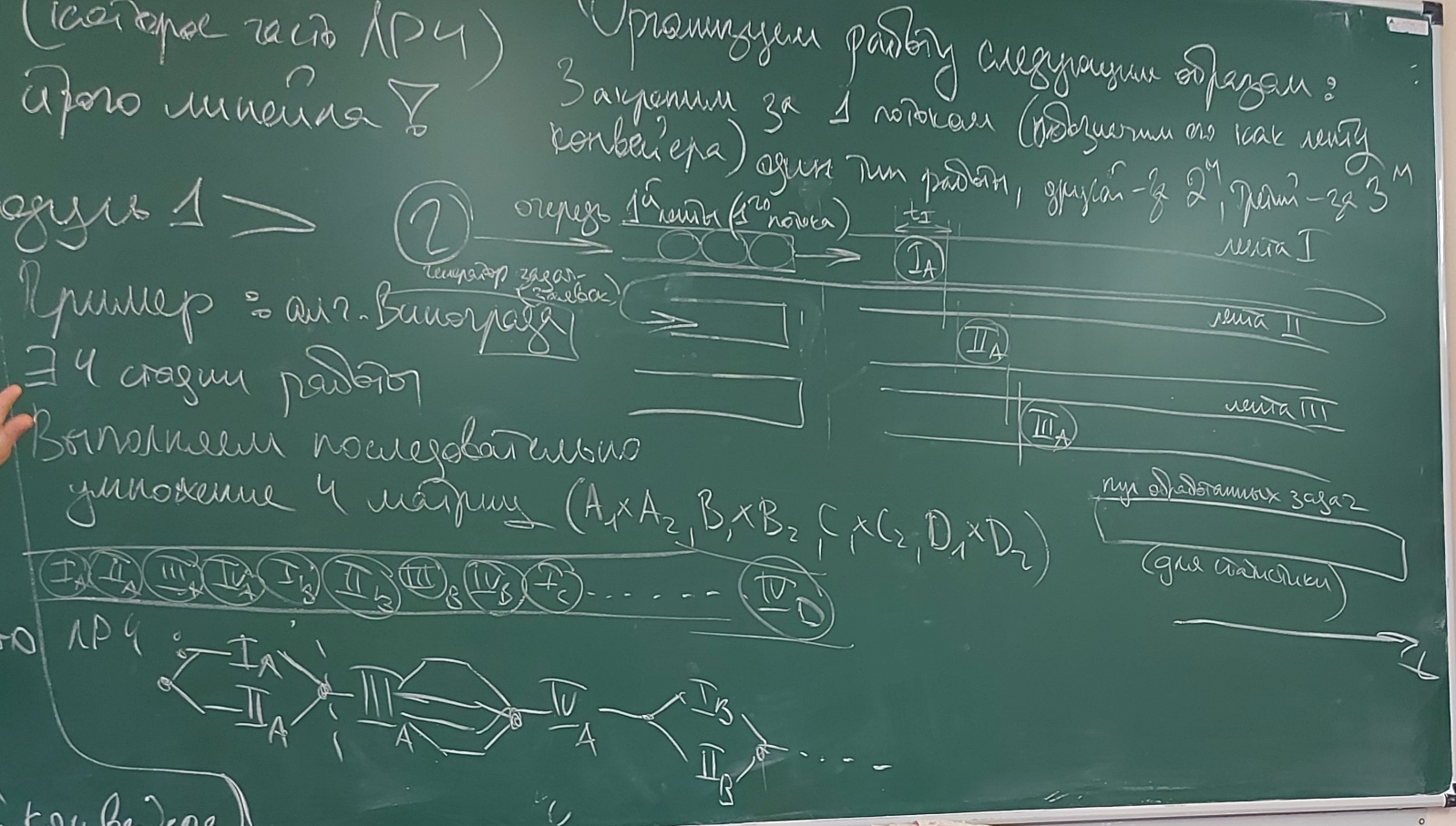
Выдана (с требованиями) презентация (умножение матриц нельзя) по ДЗ (которое часть ЛР4)  
**Важно:** операционная история строго линейна!  
Никак развилок! </модуль 1>

## Лабораторная работа 5 (Конвейер)

Производство бывает:

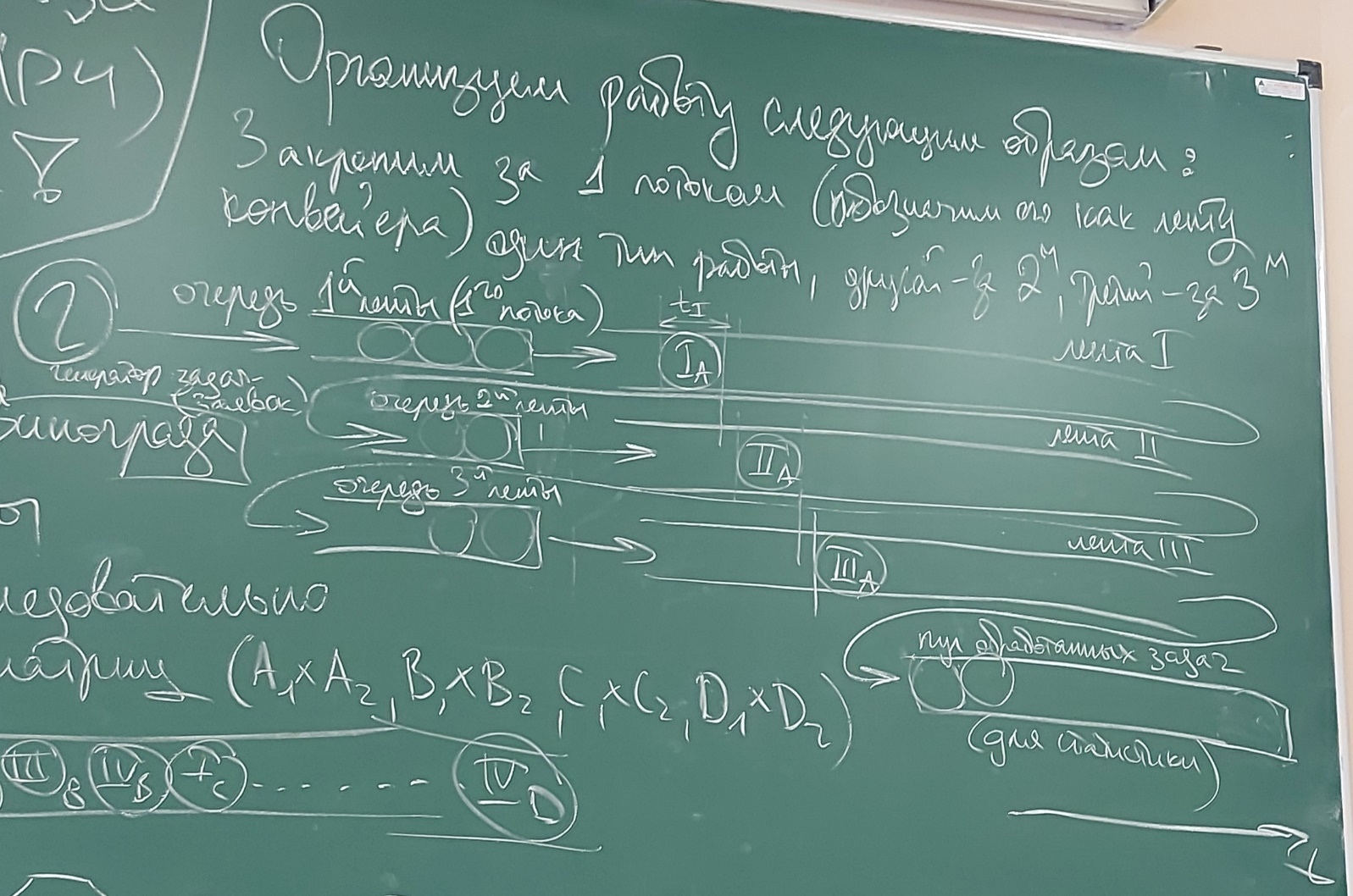
* Штучное
* Серийное

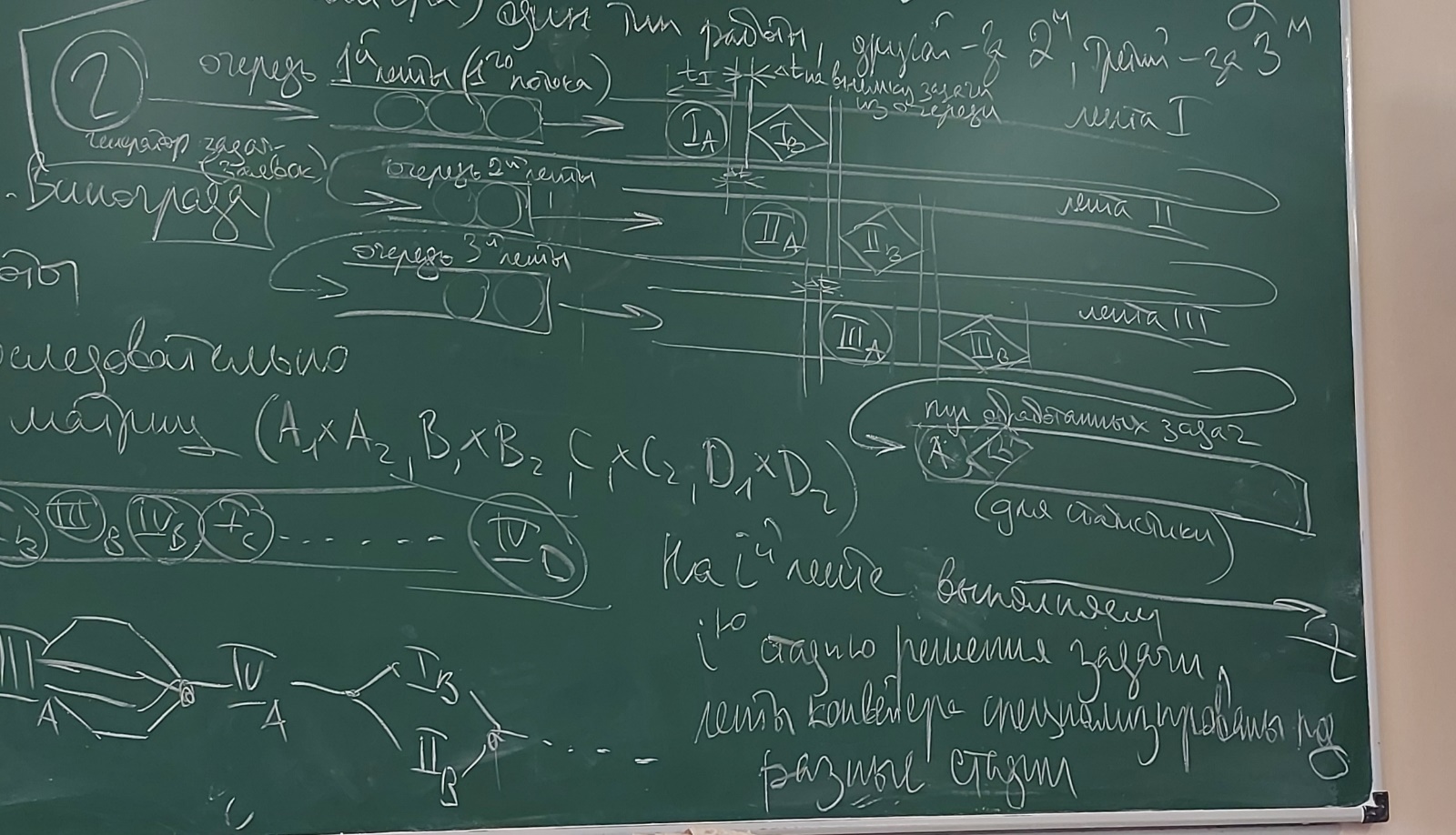
**Конвейер** (из истории) – механизм организации труда, когда производство изделия разбивается на стадии и конкретные работники закрепляются за разными изделиями. (а часто и за лентами конвейера)

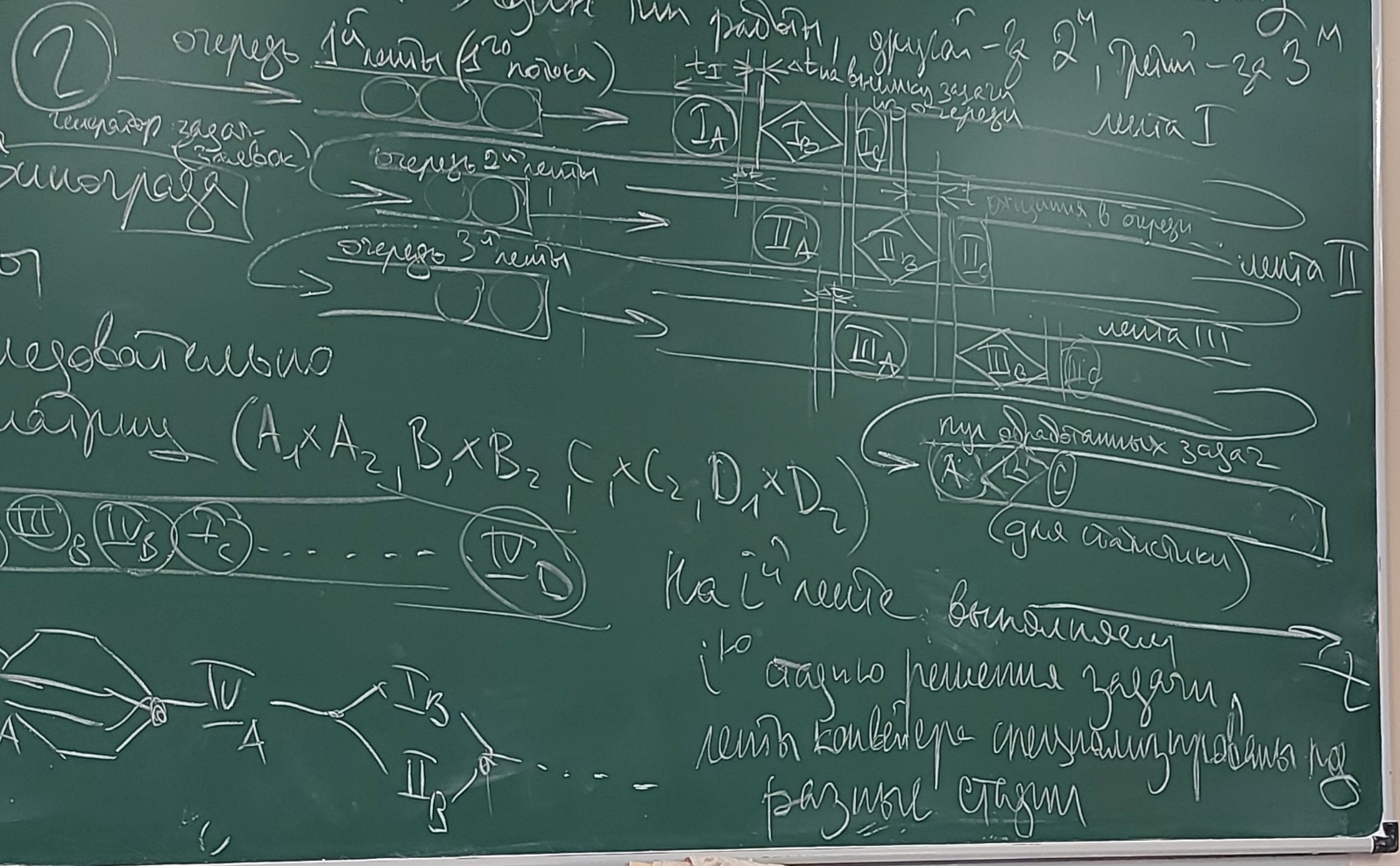
Пример: алг. Винограда  
существует 4 стадии работы  
Выполняем последовательно умножение 4 матриц

ЛР4:

Оргонизуем работу следующим образом: закрепим за 1 потоком (обозначим его как ленту конвейера) один тип работы, другой - за 2, третий – за 3

1 заявка (заявка А):  
  
На i-ой ленте выполняем i-ю стадию решения задачи, ленты конвейера специализированы под разные стадии

2ая заявка (B - задача, помеченная ромбом):  
  
На i-ой ленте выполняем i-ю стадию решения задачи, ленты конвейера специализированы под разные стадии

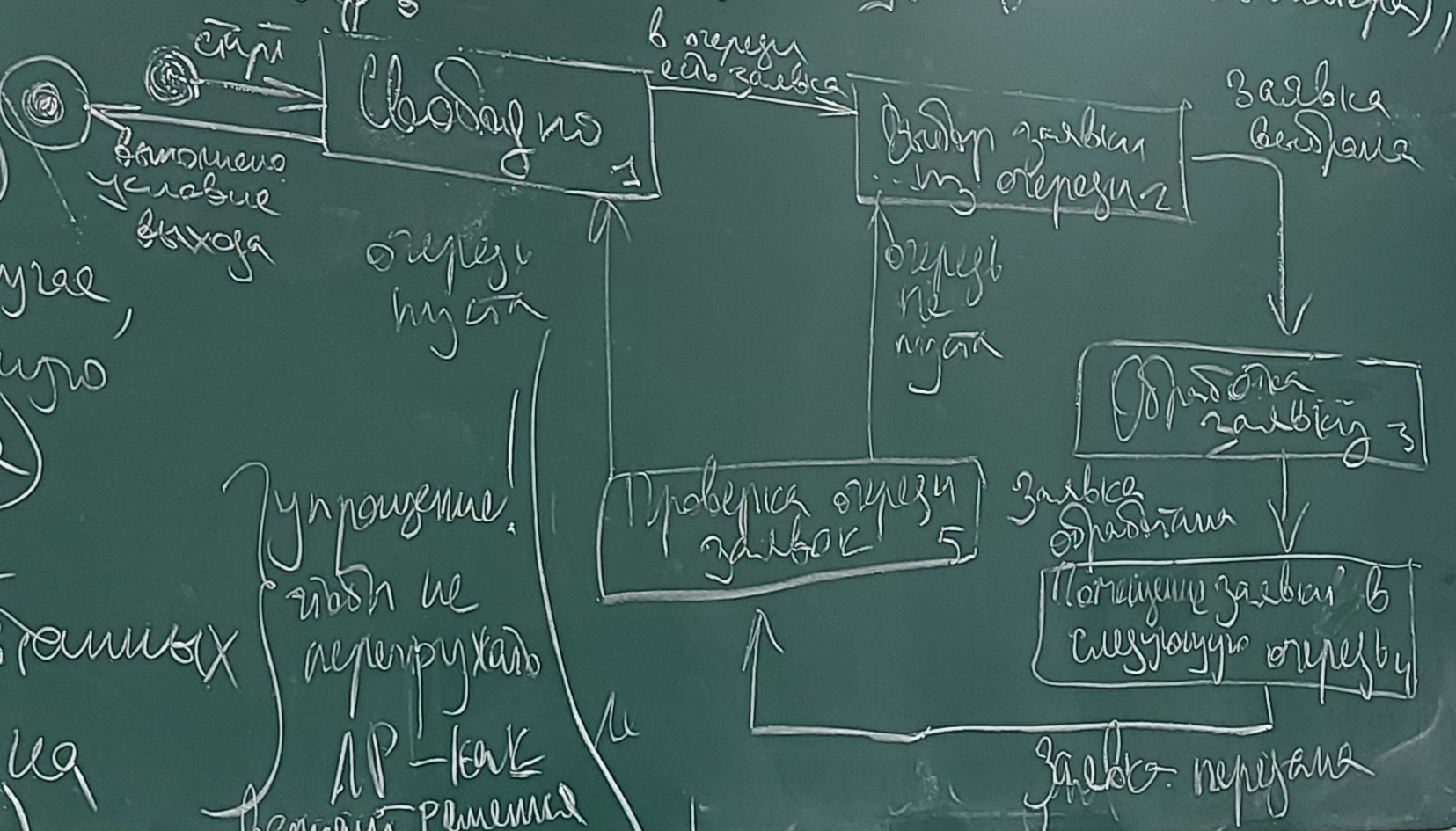
3-я задача (C):  


У разных потоков, реализующих работу разных лент конвейера разные делегаты (в примеру 1 лента выполняет 1ю стадию каждой задачи, 2-2ю, 3-3ю; буквой обозначена задача)

Необходимо обернуть решение **под**задачи: в отличии от ЛР4, поток не завершается по завершении своей одной задачи, а продолжает жить, при возможности изымая из своей очереди новую **заявку** для обработки (т.е. чтобы решить свою часть вычислений для этой заявки). Рабочий поток должен завершиться при наступлении специального события:

1. заявка специального вида (пакет с сигналом об окончании «рабочего дня»)
2. сообщение (аналогично в случае, если Вы используете событийную модель, см. event-события)
3. (разрешено в ЛР4) счётчик выполненных задач/обработанных заявок (в смысле плана на рабочий день)  
   *упрощение! Чтобы не перегружать ЛР – как вариант решения*

Если рабочий поток получает сигнал к завершению работы, надо сперва доделать начатую задачу, затем сообщить диспетчеру, что работа прекращена (выставить свой семафор!), и только тогда завершить работу (выйти из цикла)

Модель состояний, обрабатывающего устройства (которое мы реализуем в виде потока, ), один из возможных вариантов:  


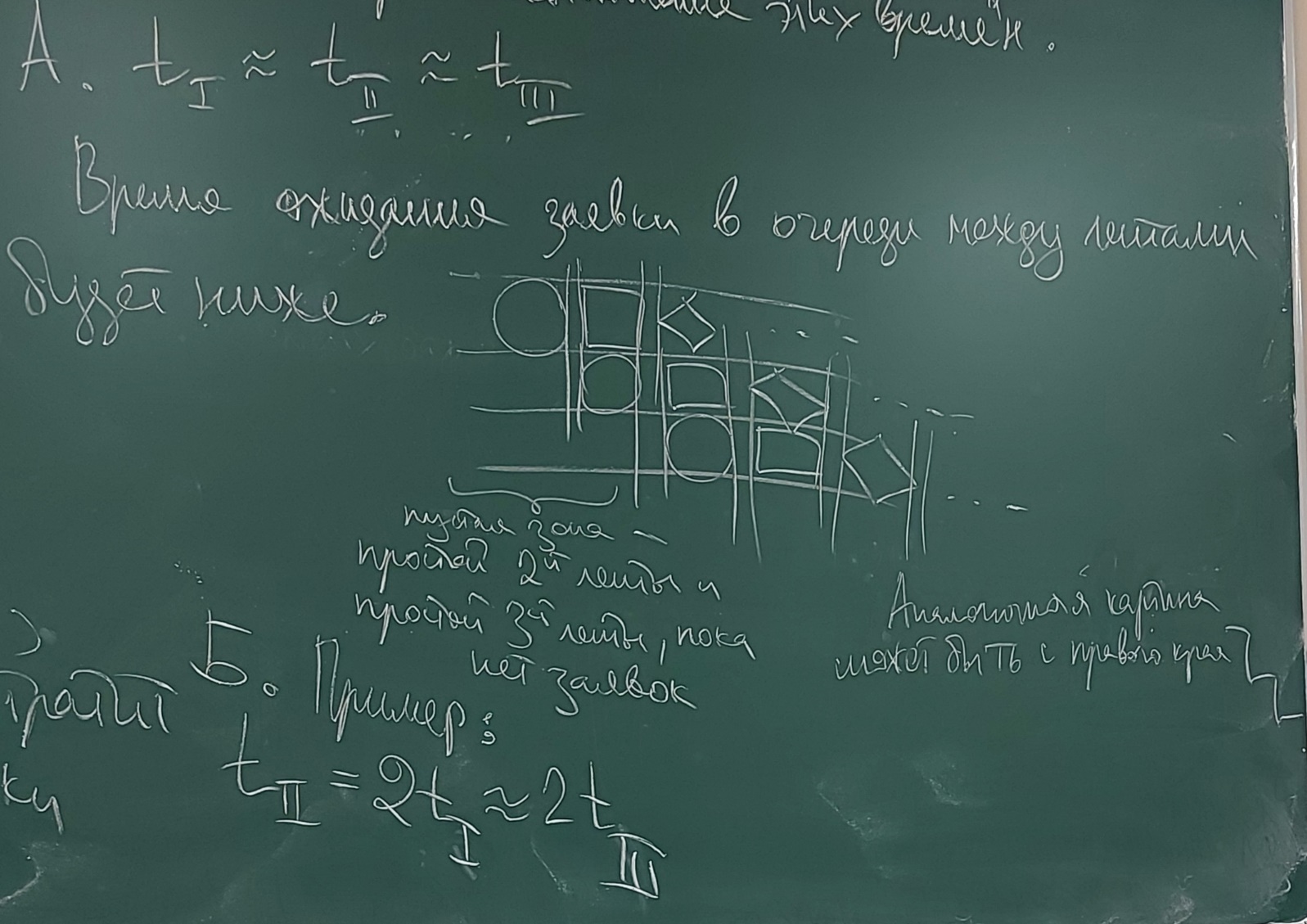
К очередям доступ следует обеспечить в монопольном режиме (как минимум из-за возможной ситуации «гонки», когда поток добавляющий заявку в очередь, будет инкрементировать счётчик заявок в очереди, а поток(и) извлекающий(ие) заявки из очереди будет(ут) декрементировать этот счётчик)

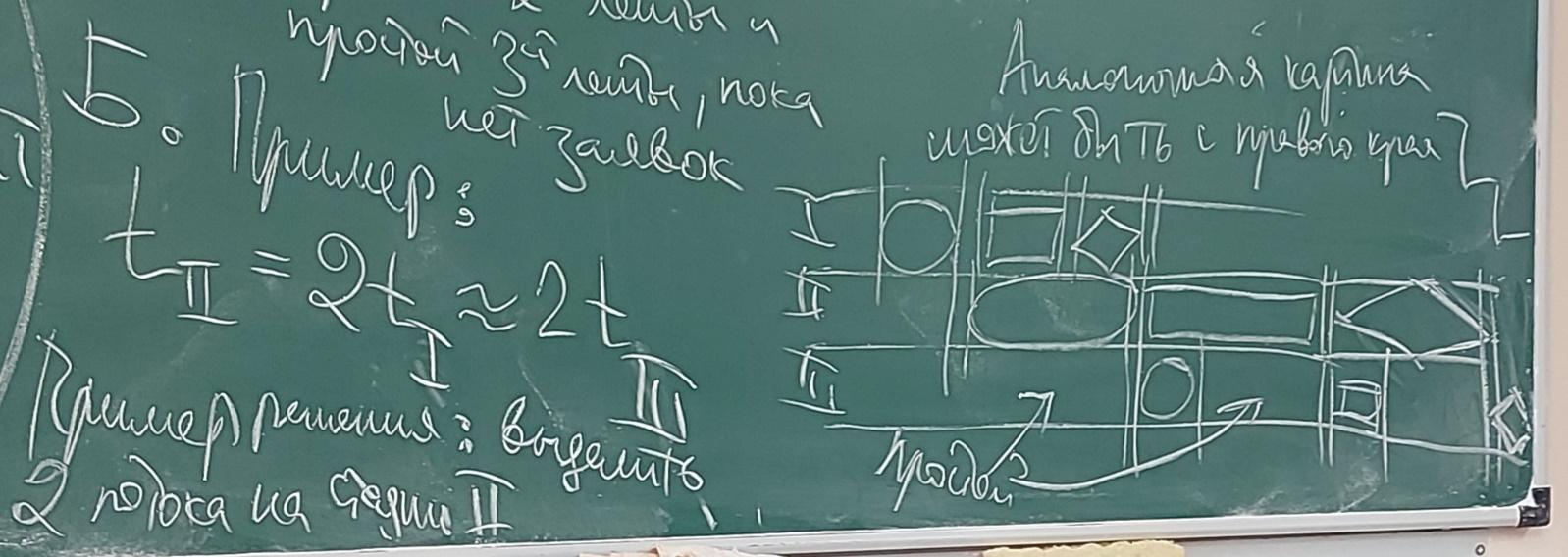
Ограничение по ЛР:

* Только нативные потоки либо язык go

Если обозначить среднее время (t) обработки i-й стадии заданы как , разброс времени , то i-я лента конвейера потратит на обработку одной заявки

Отдельный вопрос – соотношения этих времён.

Время ожидания заявки в очереди между лентами будет ниже.  
  
Аналогичная картина может быть с правого края (см. рисунок)

Пример решения: выделить 2 потока на стадии 2.

Требуется реализовать. Задание:

* Реализовать конвейер как минимум с 3 стадиями обработки одной заявки.
* Схемы алгоритмов – по аналогии с ЛР4 (на каждый тип работы своя схема, включая главный поток, он же поток - диспетчер).
* Генерация заявок (блок 2) – на ленту с некоторой задержкой между генерацией 2 заявок ЛИБО заранее заполняется входная очередь заявок (это упрощение)

Чтобы наглядно показать, что ленты конвейера работают параллельно, создадим обёртку класса заявки, добавив туда отметки времени поступления заявки в i-ю очередь и выемки заявки из i-й очереди (кроме, быть может, 1го времени, если генератор не динамический, а статический) и время окончания обработки заявки последней стадии.

На выходе, когда все заявки обработаны системой, собираем статистику:

1. На сколько/во сколько раз t обработки N заявок снижено по сравнению с последовательной обработкой одним потоком
2. Из пула обрабатываемых заявок достаём и аннотируем все метки t, сортируем, выводим лог
3. Min, max, и средние t:
   * Проведённое заявкой в системе
   * Проведённое заявкой в очереди

Постановку этой задачи, принцип вычисления и примеры статистики привести в отчёте, как и каждое принятое проектное решение.

*Тестирование должно быть*

*Тестирование не равно эксперименту (в 3 главе)*

*Эксперимент в 4ой главе*