**ОРИЕНТИРОВОЧНЫЙ СПИСОК ВОПРОСОВ К ЭКЗАМЕНУ**

**1.**Предмет изучения, структура курса. Общее понятие задачи оптимизации. Классификация задач оптимизации. Обзор методов решения оптимизационных задач.

**2**.Общая формулировка задачи линейной оптимизации. Формы записи задач линейной оптимизации.

**3**.Геометрический метод решения задачи линейной оптимизации.

**4.**Транспортная задача. Математическая модель транспортной задачи. Алгоритм решения транспортной задачи. Методы построения исходного опорного решения.

**5**.Метод потенциалов нахождения оптимального решения транспортной задачи.

**6**.Общие принципы решения задач оптимизации методом ветвей и границ.

**7**.Комбинаторные методы решения оптимизационных задач. Генерация подмножеств заданного множества.

**8**.Комбинаторные методы решения оптимизационных задач. Генерация сочетаний.

**9.**Комбинаторные методы решения оптимизационных задач. Генерация перестановок.

**10**.Комбинаторные методы решения оптимизационных задач. Генерация размещений.

**11**.Динамическое программирование. Вычислительная схема решения задачи динамического программирования (на примере решения задачи о рюкзаке).

**12**.Рекурсивные алгоритмы.

**13**.Математические основы сетевого планирования. Основные понятия теории графов.

**14**.Математические основы сетевого планирования. Кратчайшие пути между вершинами графа.

**15**.Математические основы сетевого планирования. Максимальные пути между вершинами графа.

**16**.Сетевые модели. Применение сетевых моделей. Сетевые графики.

**17**.Минимальные покрывающие деревья. Основные алгоритмы нахождения минимального остовного дерева.

**18**.Оптимизационные алгоритмы на графах. Алгоритм поиска в ширину.

**19**.Оптимизационные алгоритмы на графах. Алгоритм поиска в глубину.

**20**.Оптимизационные алгоритмы на графах. Топологическая сортировка.

**21**.Потоки в сетях. Задача о максимальном потоке. Теорема Форда-Фалкерсона.

**22**.Задачи нелинейного программирования. Основные алгоритмы решения.

**23**.Постановка задачи векторной оптимизации.

**24**.Методы решения задач векторной оптимизации.

Некоторые вопросы могут быть разбиты на 2. Например в динамическом программировании выделится дистанция Левенштейна, в комбинаторике - отдельные задачи.

**1. Предмет изучения, структура курса. Общее понятие задачи оптимизации. Классификация задач оптимизации. Обзор методов решения оптимизационных задач.**

**Математическое программирование** –раздел высшей математики, посвященный решению задач оптимизации, т.е. задач, связанных с нахождением экстремумов функций нескольких переменных, при наличии ограничений на переменные.

Методами мп решаются задачи о распределении ресурсов, планировании выпуска продукции, ценообразования, транспортные задачи и т.д.

Построение математической модели экономической задачи включает этапы:  
- Выбор переменных задачи

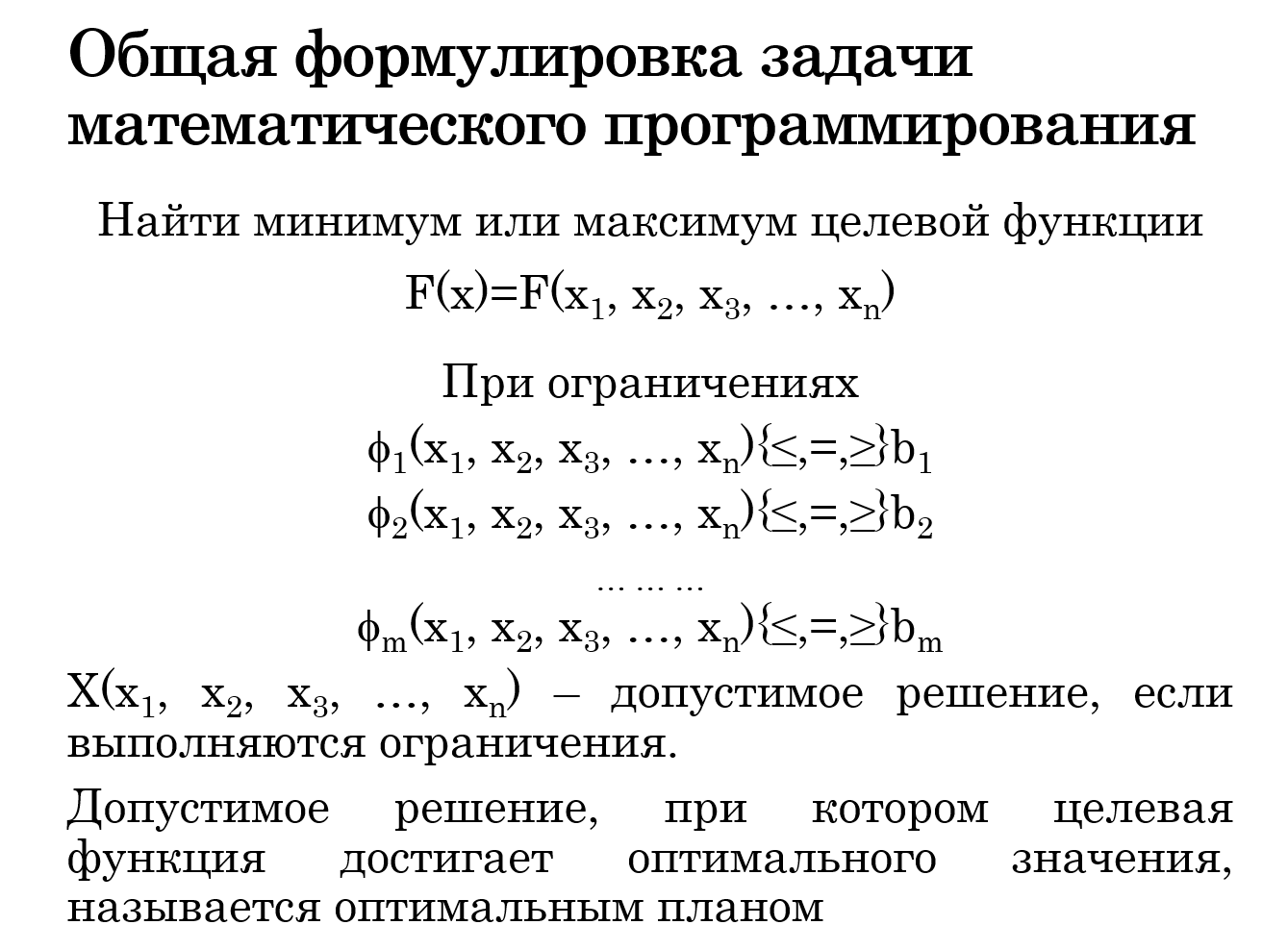
- Составление системных ограничений

- Выбор целевой функции

**Переменные задачи** – величины x1, x2, x3, …, xn которые полностью характеризуют экономический процесс. Их обычно записывают в виде вектора X = (x1, x2, x3, …, xn).

**Система ограничений** включает в себя систему уравнений и неравенств, которым удовлетворяют переменные задачи и которые следуют из ограниченности ресурсов или других экономических или физических условий, например, положительности переменных и т. п.

**Целевая функция** – функция переменных задачи, которая характеризует качество выполнения задачи, и экстремум которой требуется найти.



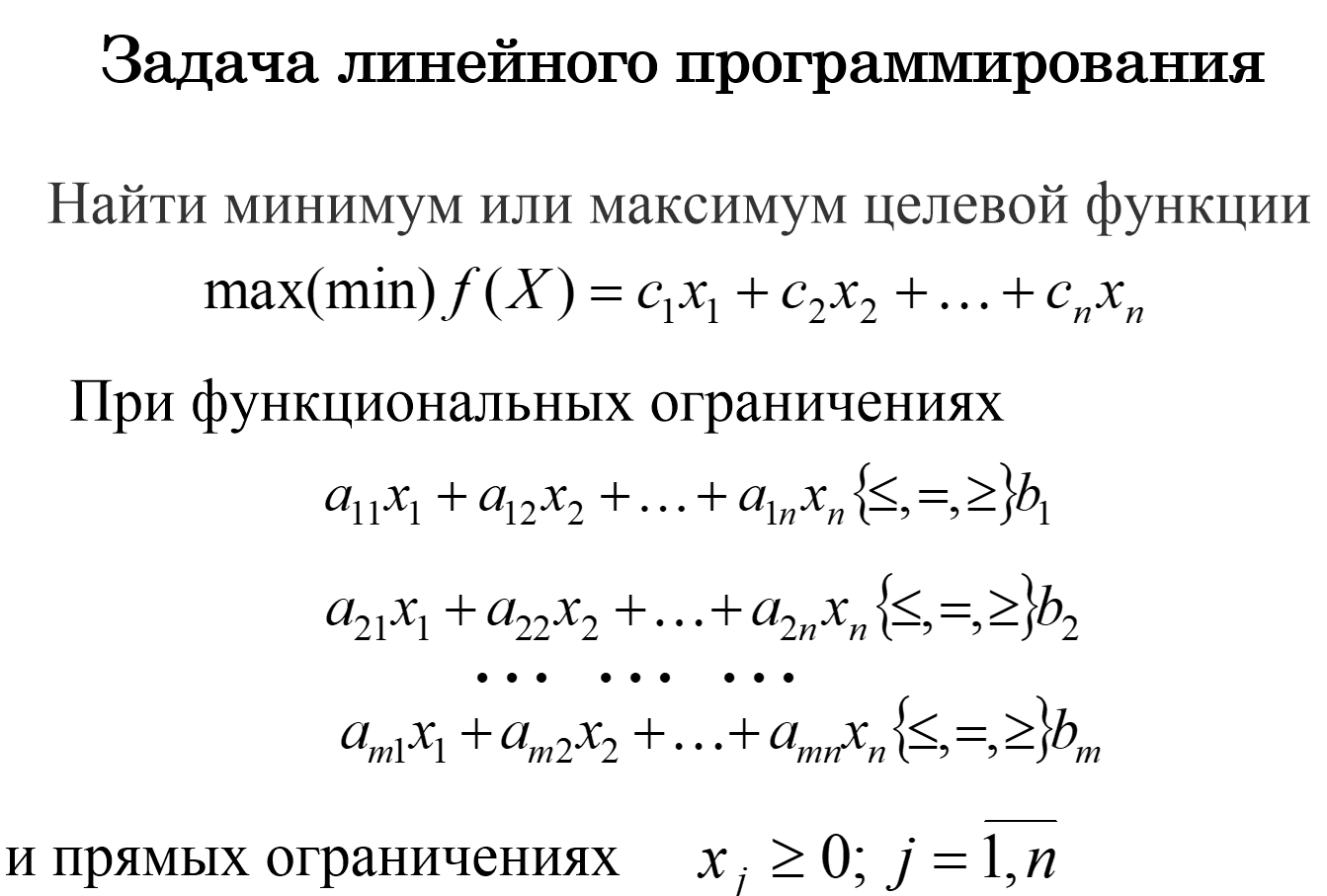
****

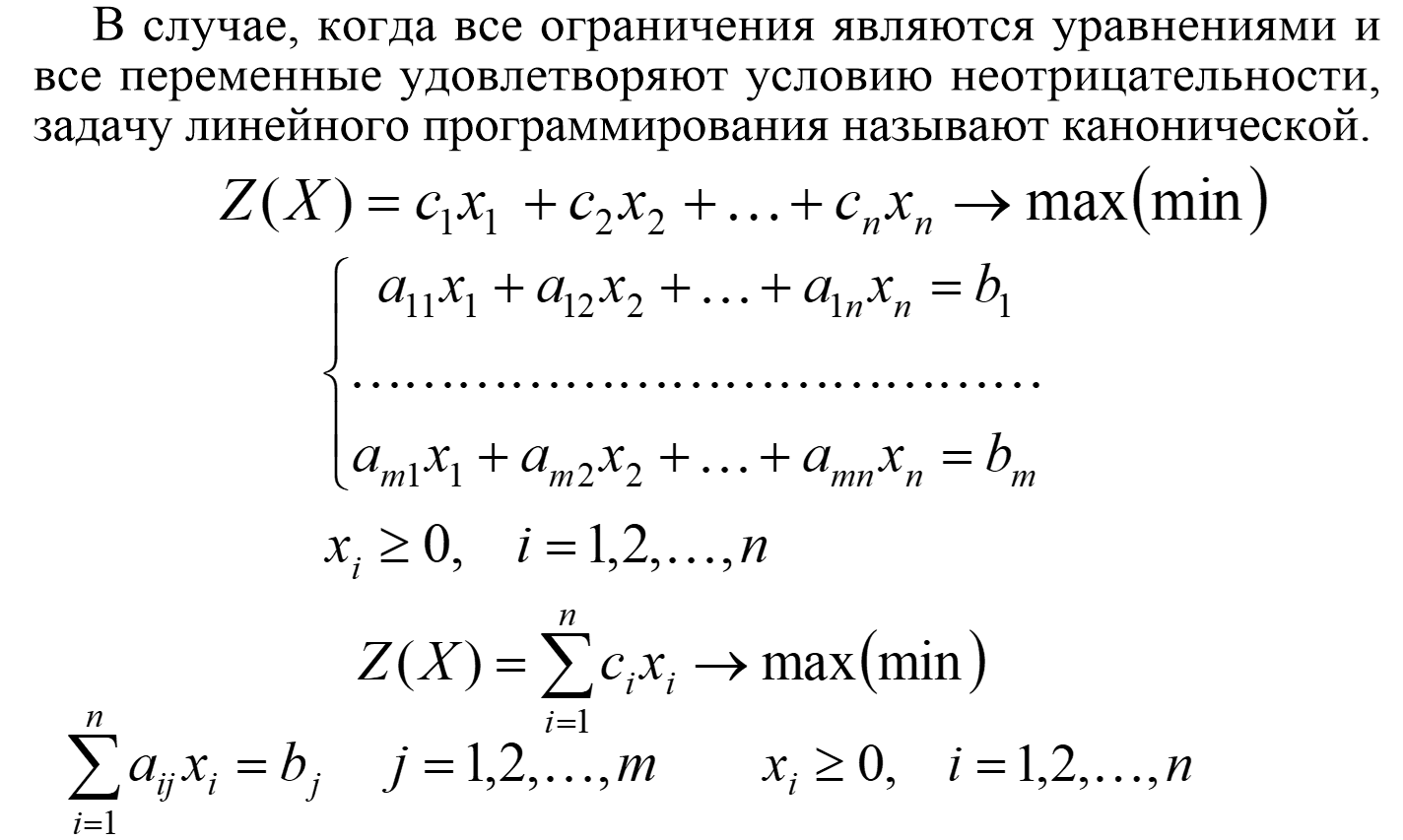
**2. Общая формулировка задачи линейной оптимизации. Формы записи задач линейной оптимизации.**

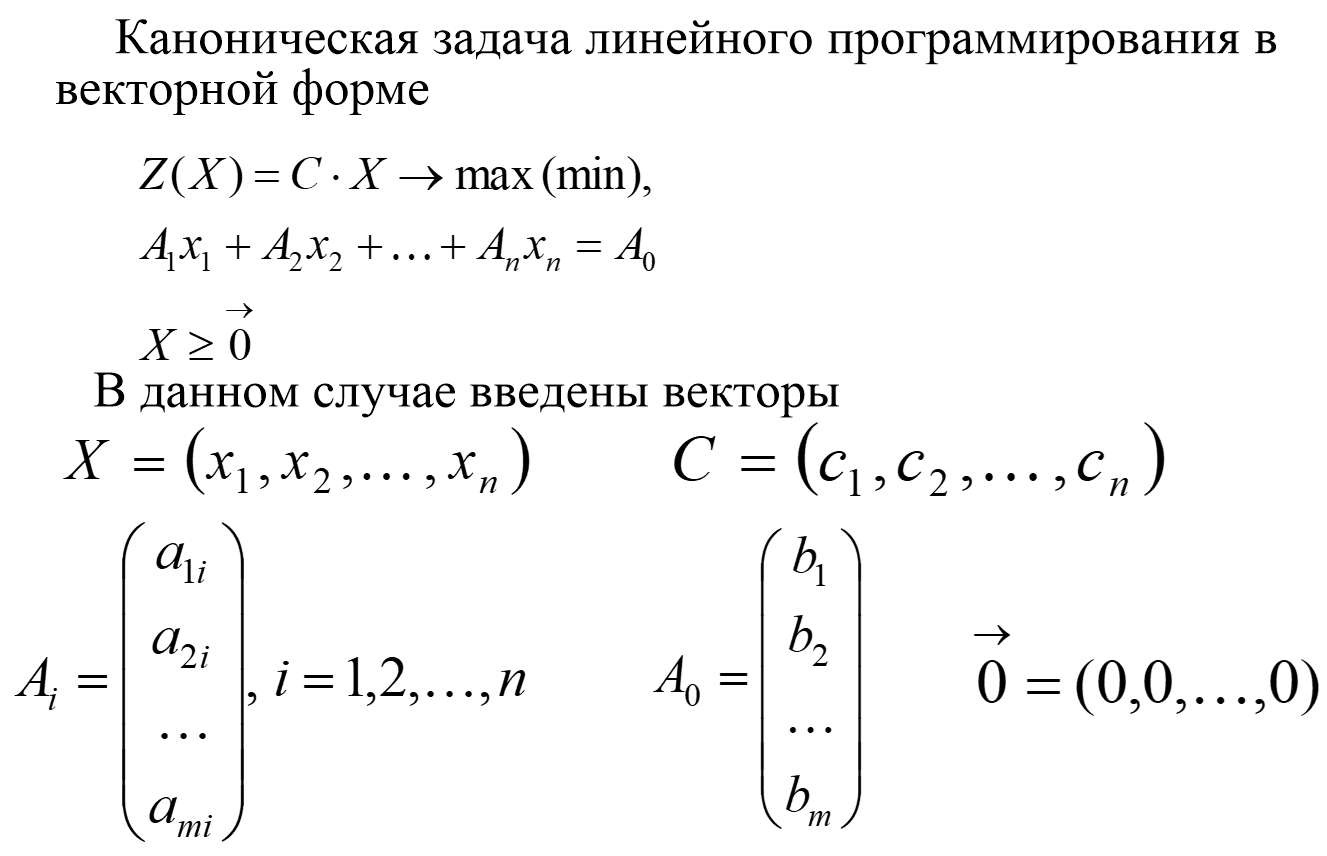
Задача линейного программирования (ЛП) состоит в нахождении минимума (или максимума) линейной функции при линейных ограничениях.

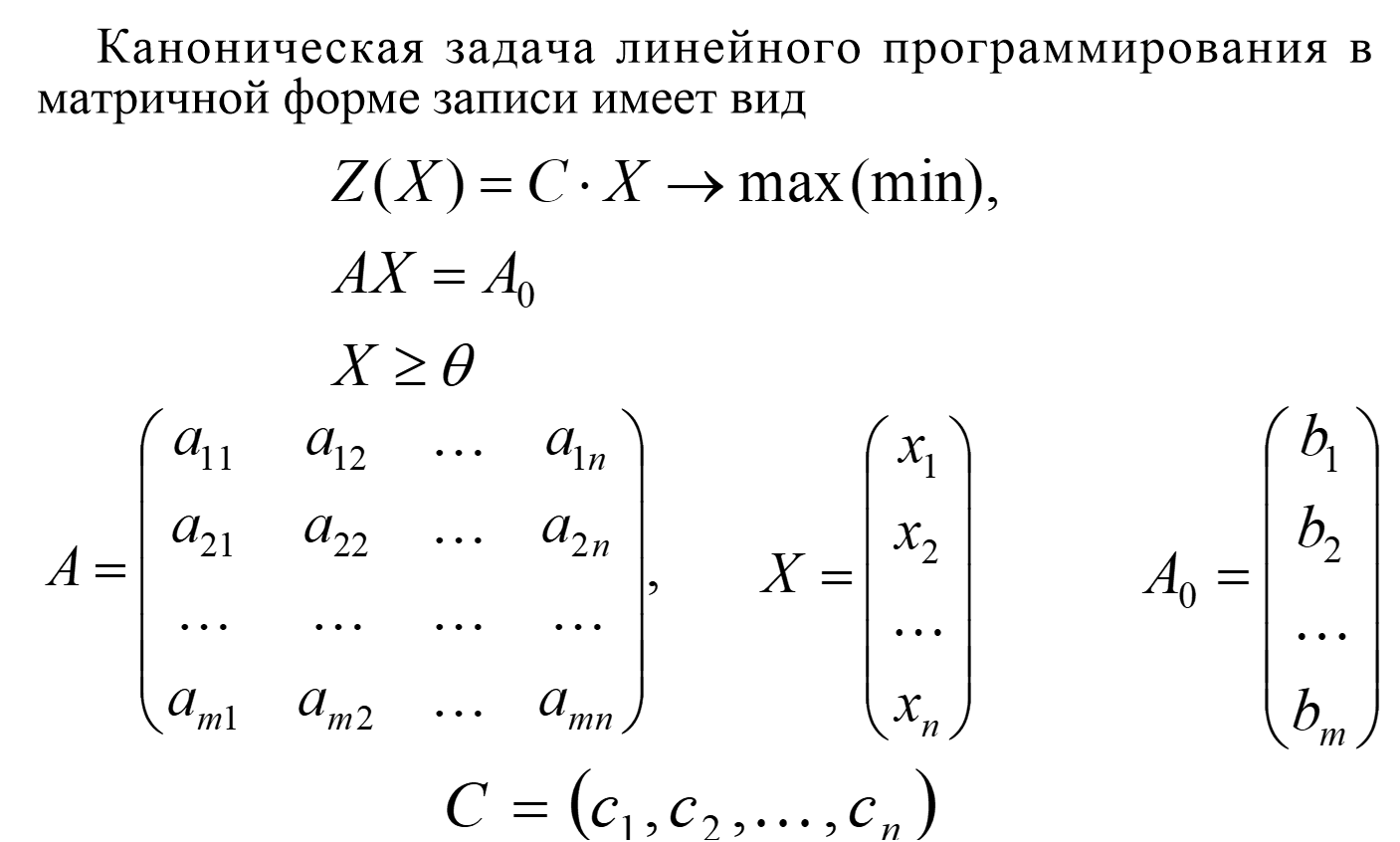
Существует общая, каноническая и стандартная формы записи.

Общая форма:

****

****

****

****

**3. Геометрический метод решения задачи линейной оптимизации.**

Решение системы линейных уравнений можно осуществить методом Жордана-Гаусса и геометрическим.

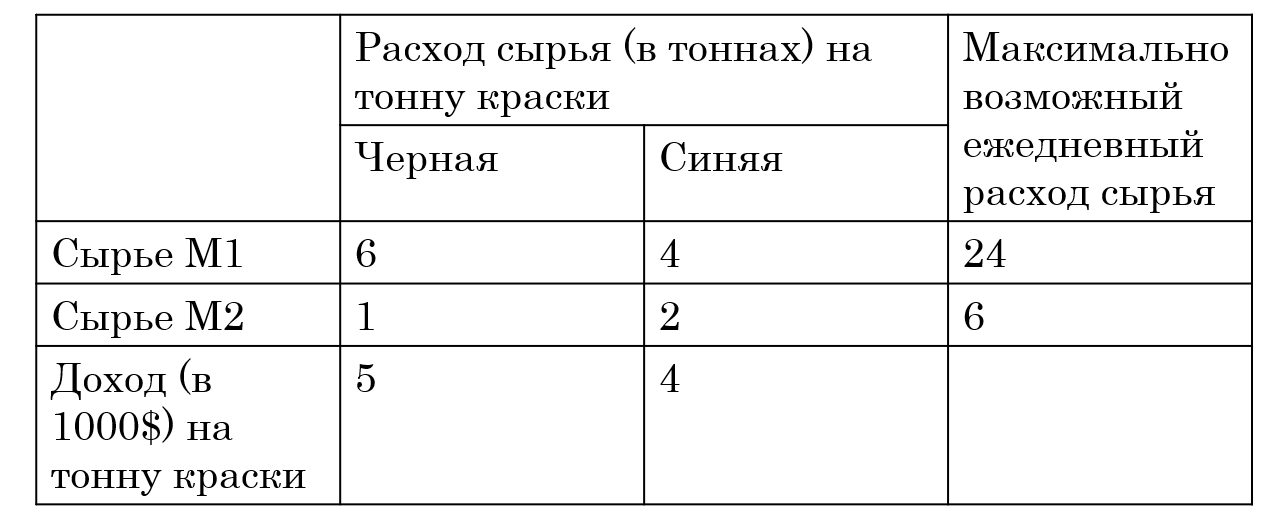
Графический или геометрический метод решения задачи ЛП состоит из 2 этапов:

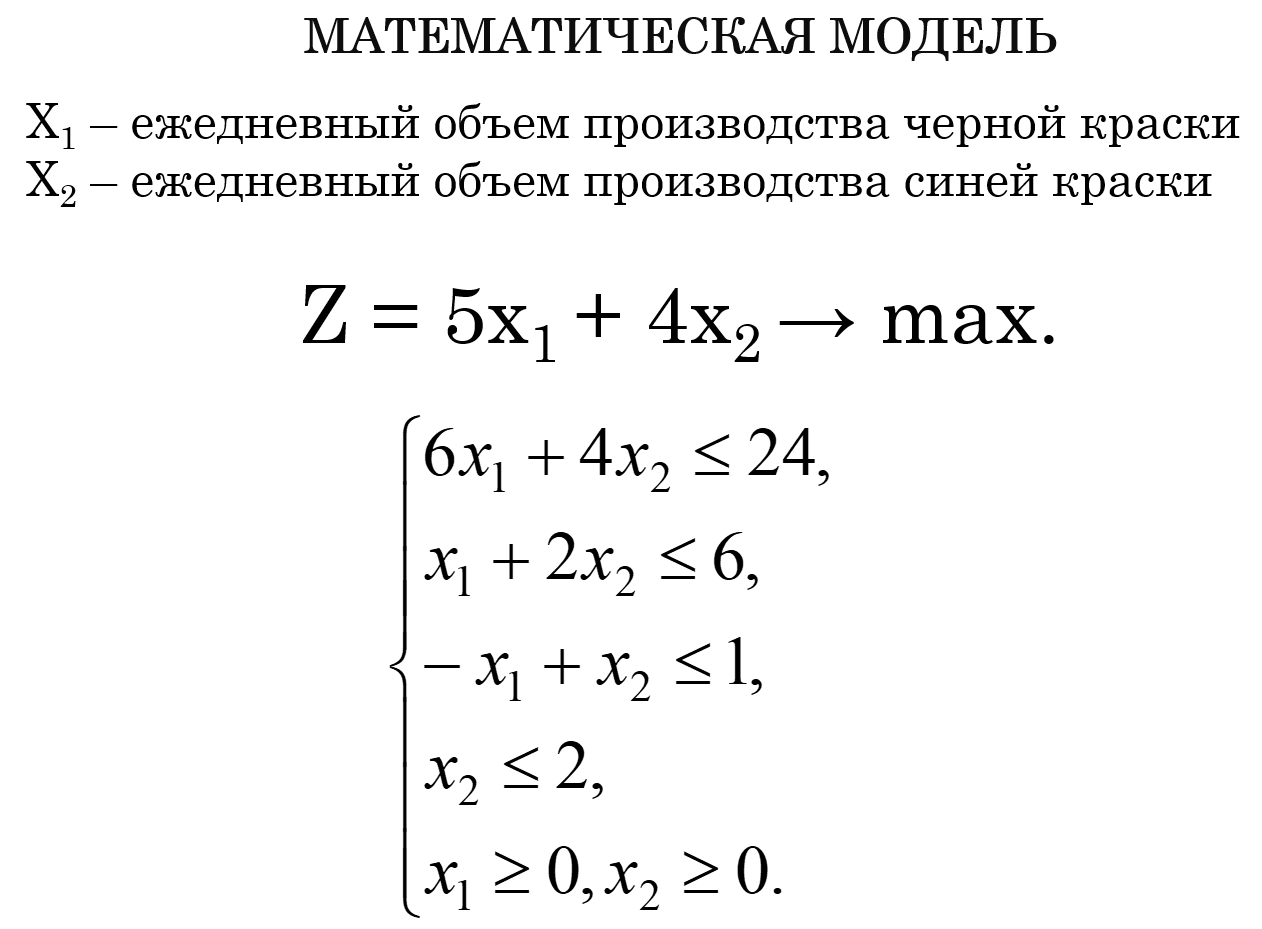
1. Построение пространства допустимых решений, удовлетворяющих всем ограничениям модели.
2. Нахождение оптимального решения среди всех точек пространства допустимых решений.

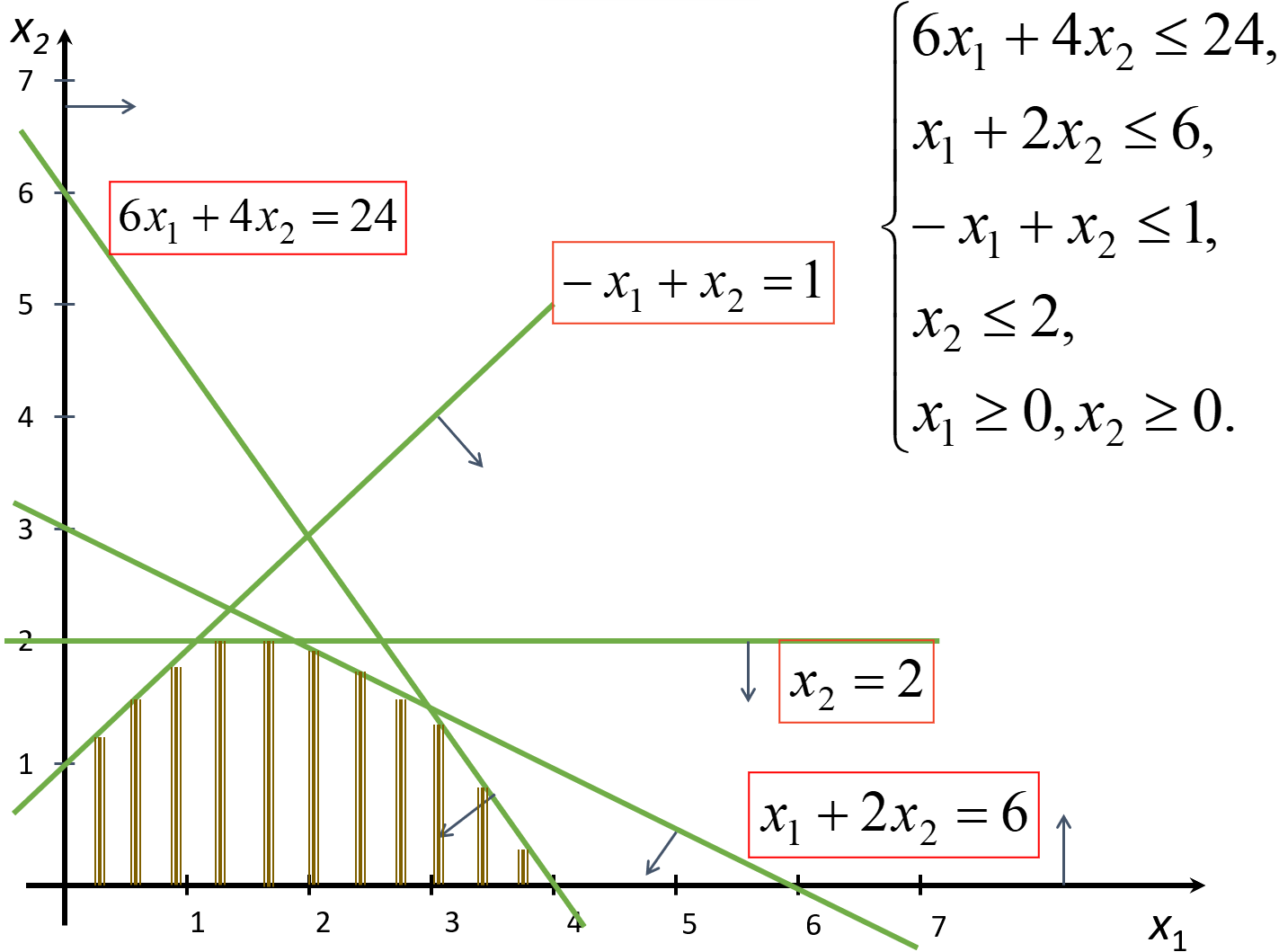
Алгоритм решения:

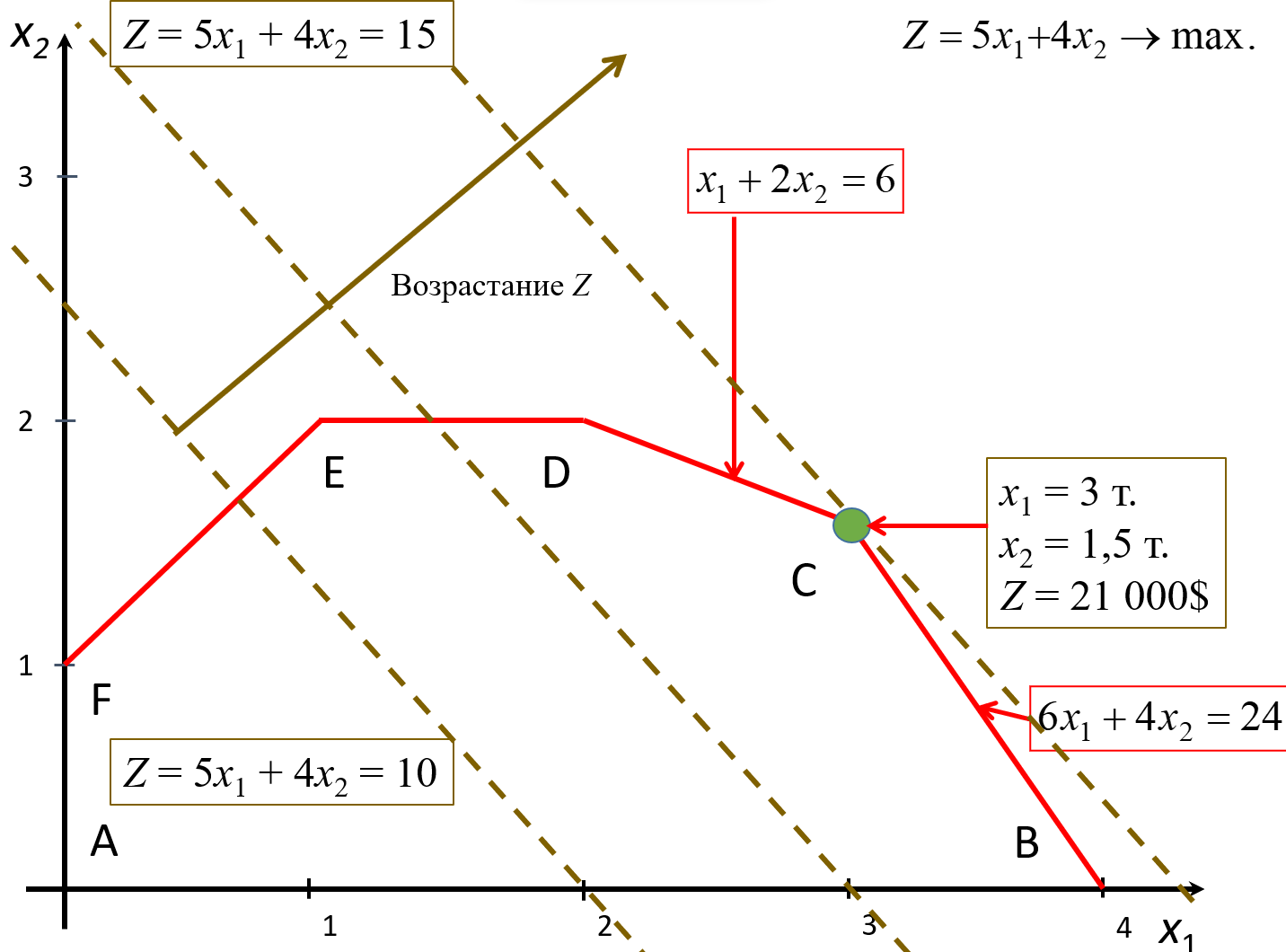
1. Строится область допустимых решений
2. Строится вектор нормали к линии уровня с точкой приложения в начале координат
3. Перпендикулярно вектору нормали проводится одна из линий уровня, проходящая через начало координат
4. Линии уровня перемещаются до положения опорной прямой. На этой прямой и будут находится мин или макс функции

В зависимости от вида области допустимых решений и целевой функции задача может иметь единственное решение, бесконечное множество решений или не иметь ни одного оптимального решения.

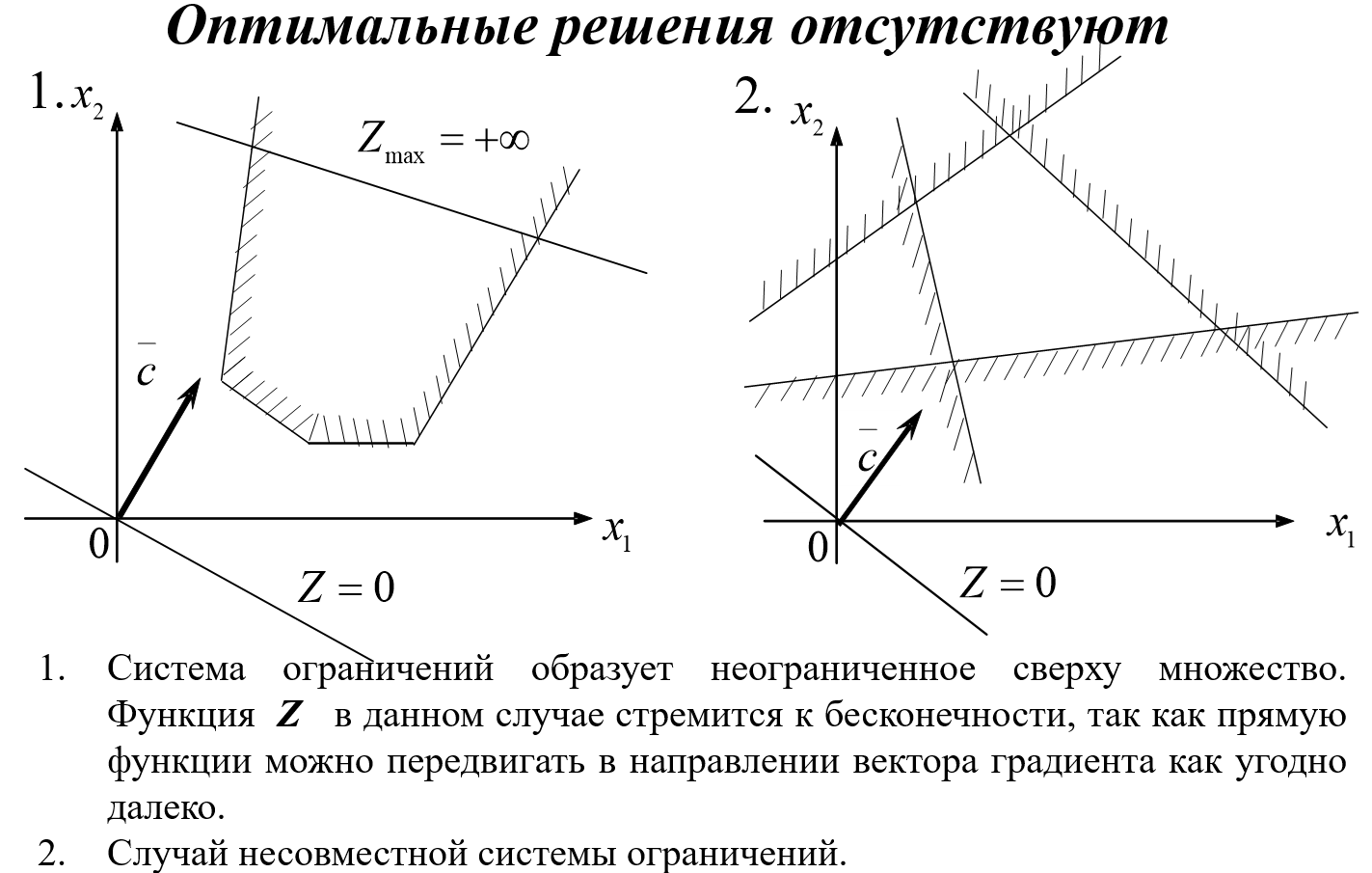
Пример:  






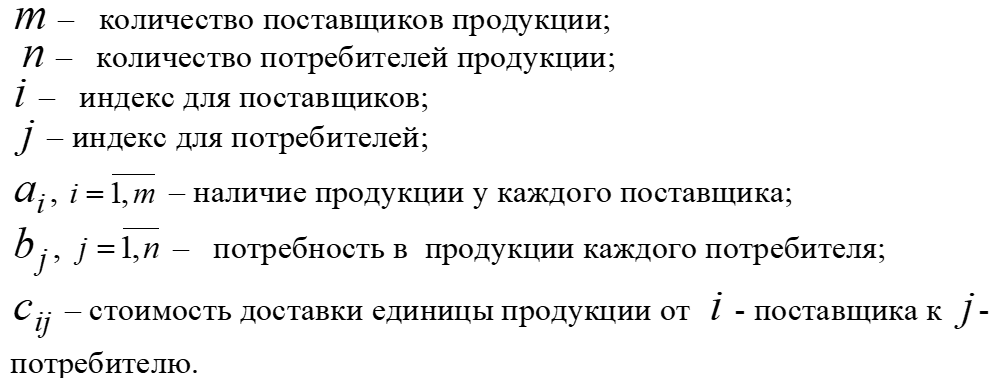


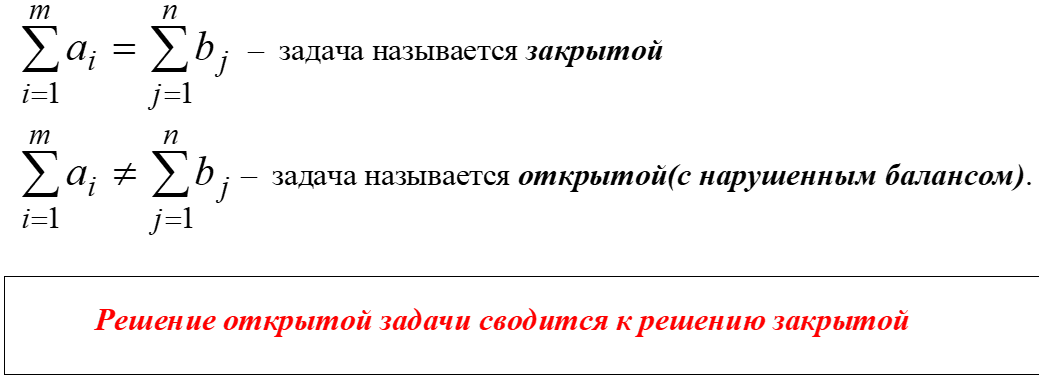


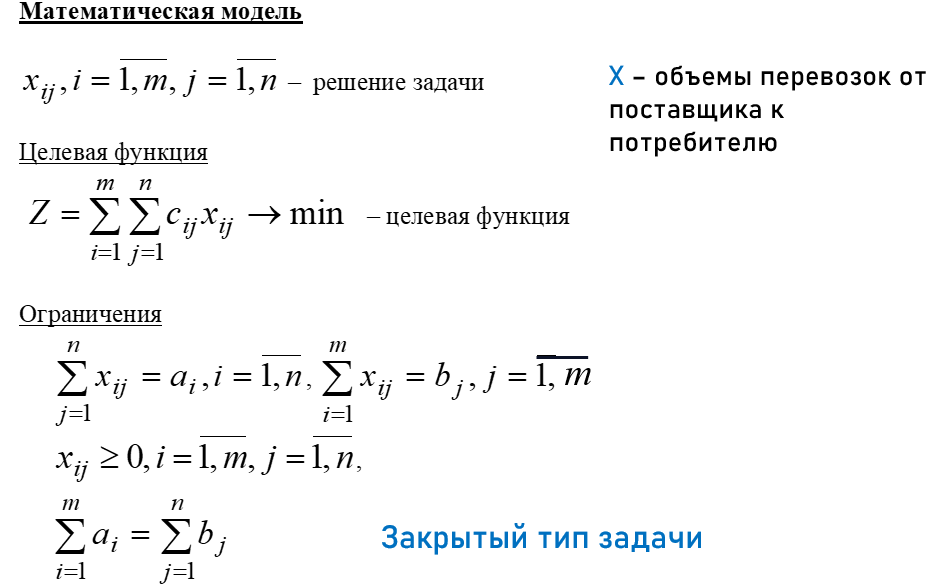


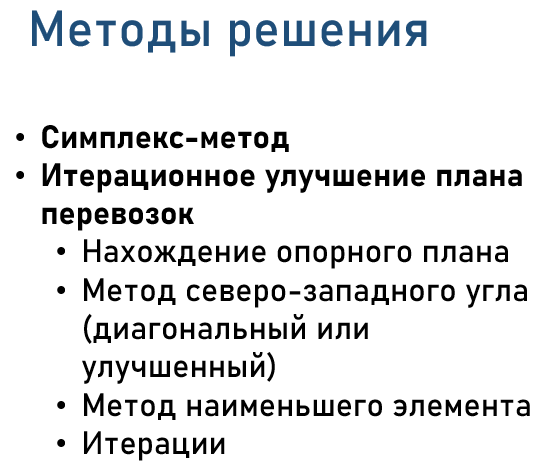
**4. Транспортная задача. Математическая модель транспортной задачи. Алгоритм решения транспортной задачи. Методы построения исходного опорного решения.**

Транспортные задачи – специальный класс задач линейного программирования, в которых необходимо найти условный экстремум. Эти задачи описывают перемещение (перевозку) какого-либо товара из пункта направления (например, места производства) в пункт назначения (склад или магазин). Назначение транспортной задачи – определение объемов перевозок из пунктов отправления в пункты назначения с минимальной суммарной стоимостью перевозок.









Общая транспортная задача с m пунктами отправления и n пунктами назначения имеет m+n ограничений в виде равенств, по одному на каждый пункт отправления и назначения. Т. к. транспортная задача д.б. сбалансированной, то одно из этих равенств избыточно. Т.о. транспортная задача имеет m+n+1 независимых ограничений, отсюда вытекает, что начальное базисное решение состоит из m+n+1 базисных переменных.

**Пример:**

1. **Метод наименьшей стоимости.**

Пусть  поставщиков продукции,  потребителей продукции

Запасы , Потребность 

Затраты на перевозку продукции 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | 2 | 20 | 11 | **15** |
| **2** | 12 | 7 | 9 | 20 | **25** |
| **3** | 4 | 14 | 16 | 18 | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

Сначала в таблице находим ячейку с наименьшей стоимостью. Затем переменной в этой ячейке присваивается наибольшее значение, допускаемое ограничениями по спросу и предложению (если таких несколько, то выбор произволен). Далее вычеркивается соответствующий столбец или строка и корректируется спрос и предложение. Затем просматриваются не вычеркнутые ячейки, и выбирается новая ячейка с минимальной стоимостью и т.д.

1)Выбор ячейки с наименьшим значением 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2** | 20 | 11 | **15** |
| **2** | 12 | 7 | 9 | 20 | **25** |
| **3** | 4 | 14 | 16 | 18 | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

2)

Выбор ячейки с наименьшим значением 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | 11 | **0** |
| **2** | 12 | 7 | 9 | 20 | **25** |
| **3** | **4** | 14 | 16 | 18 | **10** |
| **Потребность** | **5** | **0** | **15** | **15** | **35** |

3)

Выбор ячейки с наименьшим значением 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | 11 | **0** |
| **2** | 12 | 7 | **9** | 20 | **25** |
| **3** | **4|5** | 14 | 16 | 18 | **5** |
| **Потребность** | **0** | **0** | **15** | **15** | **30** |

4)

Выбор ячейки с наименьшим значением 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | 11 | **0** |
| **2** | 12 | 7 | **9 | 15** | 20 | **10** |
| **3** | **4 |5** | 14 | 16 | 18 | **5** |
| **Потребность** | **0** | **0** | **0** | **15** | **15** |

5)

Выбор ячейки с наименьшим значением 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | 11 | **0** |
| **2** | 12 | 7 | **9 | 15** | 20 | **10** |
| **3** | **4 |5** | 14 | 16 | 18 | **5** | **0** |
| **Потребность** | **0** | **0** | **0** | **10** | **10** |

6)

Должно быть базовых  переменных

Есть только 5, поэтому выбираем любую 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | **11 | 0** | **0** |
| **2** | 12 | 7 | **9 | 15** | **20** | **10** | **0** |
| **3** | **4 |5** | 14 | 16 | **18 | 5** | **0** |
| **Потребность** | **0** | **0** | **0** | **0** | **0** |

Первое допустимое решение

, , , ,  , 

Значение функции цели



1. 135 200 20 90

**5. Метод потенциалов нахождения оптимального решения транспортной задачи.**

В методе потенциалов каждой строке i и каждому столбцу j транспортной таблицы ставятся в соответствие числа (потенциалы) *ui* (поставщики)и *vj* (потребители). Для каждой базисной переменной xij потенциалы *ui* и *vj*удовлетворяют уравнению

*ui* + *vj* = *сij*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | **11 | 0** | **15** |
| **2** | 12 | 7 | **9 | 15** | **20** | **10** | **25** |
| **3** | **4 |5** | 14 | 16 | **18 | 5** | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

Потенциалы: , .

Определяем потенциалы для всех базисных переменных













Уравнений 6 неизвестных 7:

Присваиваем одному из них произвольное значение (обычно  ) ,

,





, , 

, ,,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | **11 | 0** | **15** |
| **2** | 12 | 7 | **9 | 15** | **20** | **10** | **25** |
| **3** | **4 |5** | 14 | 16 | **18 | 5** | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

Для свободных клеток

|  |  |
| --- | --- |
| Небазисная переменная |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Вводимой в базис будет переменная, имеющая наибольшее положительное значение – х22.

Определив вводимую в базис переменную, следует определить исключаемую из базиса переменную. Обозначим через θ количество груза, перевозимого по маршруту (2,2). Максимально возможное значение θ определяем из следующих условий:

1. Должны выполняться ограничения на спрос и предложение.
2. Ни по какому маршруту не должны выполняться перевозки с отрицательным объемом грузов.

Сначала строим замкнутый цикл, который начинается и заканчивается в искомой ячейке. Цикл состоит из последовательности горизонтальных и вертикальных отрезков( но не диагональных), соединяющих ячейки, соответствующие текущим базисным переменным, и ячейку, соответствующую вводимой переменной. Для того, чтобы удовлетворять ограничениям по спросу и предложению, надо поочередно отнимать и прибавлять θ к значениям базисных переменных, расположенных в угловых ячейках цикла. Направление обхода цикла (по часовой стрелке или против не имеет значения).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 15** | 20 | **11 | 0** | **15** |
| **2** | 12 | 7 | **9 | 15** | **20** | **10** | **25** |
| **3** | **4 |5** | 14 | 16 | **18 | 5** | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

Перемещаем 10 единиц товара по циклу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 5** | 20 | **11 | 10** | **15** |
| **2** | 12 | **7|10** | **9 | 15** | **20** | **0** | **25** |
| **3** | **4|5** | 14 | 16 | **18 | 5** | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

Допустимое решение

, , , ,  , 

Значение функции цели



10 110 70 135 20 90



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители**  **Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 5** | 20 | **11 | 10** | **15** |
| **2** | 12 | **7|10** | **9 | 15** | 20 | 0 | **25** |
| **3** | **4 |5** | 14 | 16 | **18 | 5** | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

Определяем потенциалы для всех базисных переменных ((2,4) уже не базисная)













Уравнений 6 неизвестных 7:

,

,







, , 

, ,,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Потребители Поставщики** | **1** | **2** | **3** | **4** | **Запасы** |
| **1** | 10 | **2| 5** | 20 | **11 | 10** | **15** |
| **2** | 12 | **7|10** | **9 | 15** | **20** | **25** |
| **3** | **4 |5** | 14 | 16 | **18 | 5** | **10** |
| **Потребность** | **5** | **15** | **15** | **15** | **50** |

|  |  |
| --- | --- |
| Небазисная переменная |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Остановка, когда нет положительных чисел или нет цикла.

**6. Общие принципы решения задач оптимизации методом ветвей и границ.**

*Метод ветвей и границ* – это общий алгоритмический метод решения задач комбинаторной оптимизации.

Метод ветвей и границ был предложен для решения общей задачи целочисленного линейного программирования.

Метод является вариацией полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений.

Область применения метода ветвей и границ:

* Задача о коммивояжере
* Минимаксные задачи о назначениях
* Задачи календарного планирования
* Задача о трех станках(вероятностные задачи)

***В основе метода лежат две процедуры:***

* ***процедура ветвления (BR), позволяющая разбивать множество допустимых решений на непересекающиеся подмножества,***
* ***процедура вычисления нижней или верхней границы (EV).***

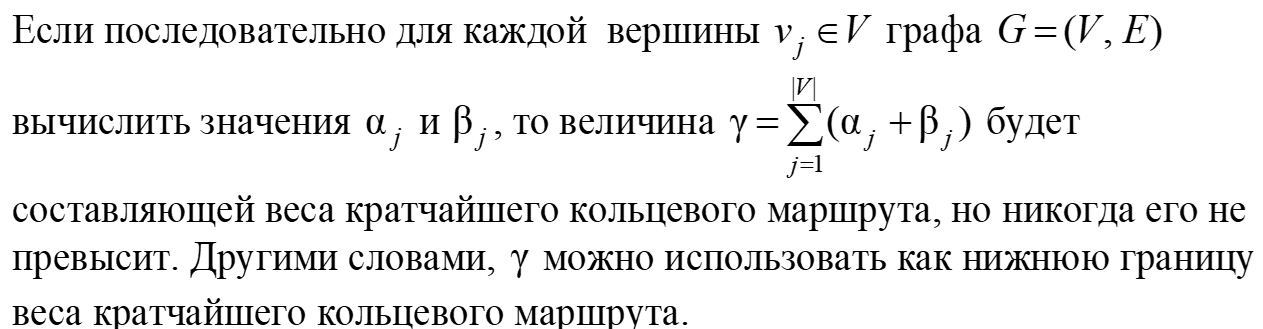
Пусть полный взвешенный ориентированный граф G = (V, E) с весовой функцией w моделирует города (множество вершин V) и расстояния между ними (взвешенные дуги E) в задаче коммивояжера.

Решение этой задачи сводится к отысканию кольцевого маршрута r проходящего через все вершины графа и имеющего минимальную сумму весов дуг, составляющих кольцевой маршрут (кратчайший кольцевой маршрут). Гамильтоновым циклом является такой цикл (замкнутый путь), который проходит через каждую вершину данного графа ровно по одному разу; то есть простой цикл, в который входят все вершины графа.

***Процедура EV (вычисления нижней или верхней границы ) Основывается на двух утверждениях:***

***Утверждение 1. Изменение всех элементов строки матрицы расстояний на одно и то же число не влияет на выбор оптимального маршрута коммивояжера.***

***Утверждение 2. Изменение всех элементов столбца матрицы расстояний на одно и то же число не влияет на выбор оптимального маршрута коммивояжера.***

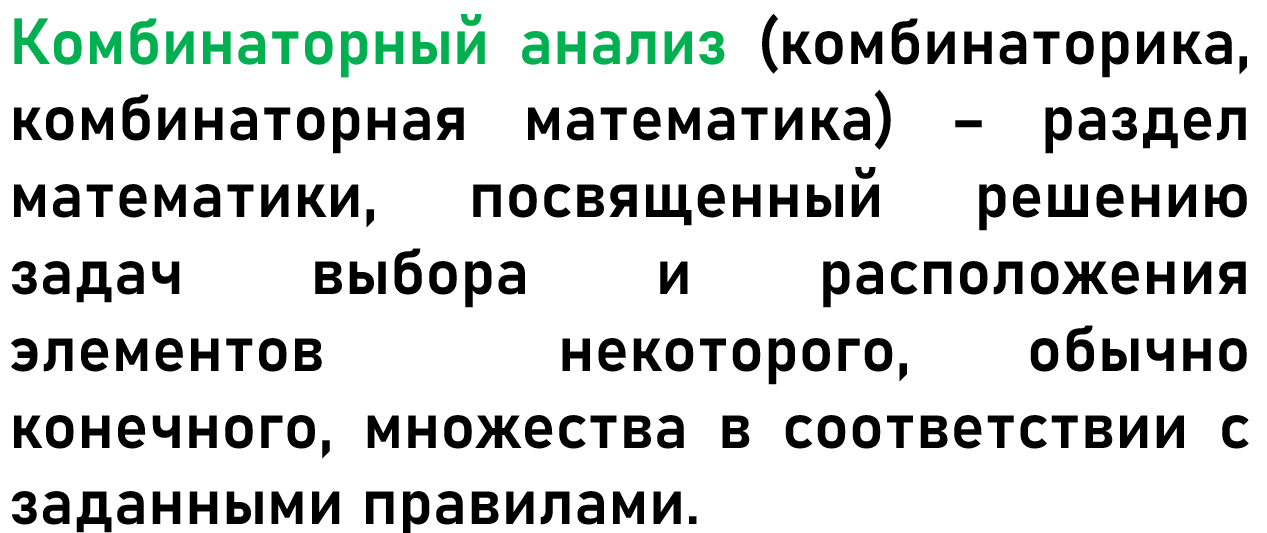


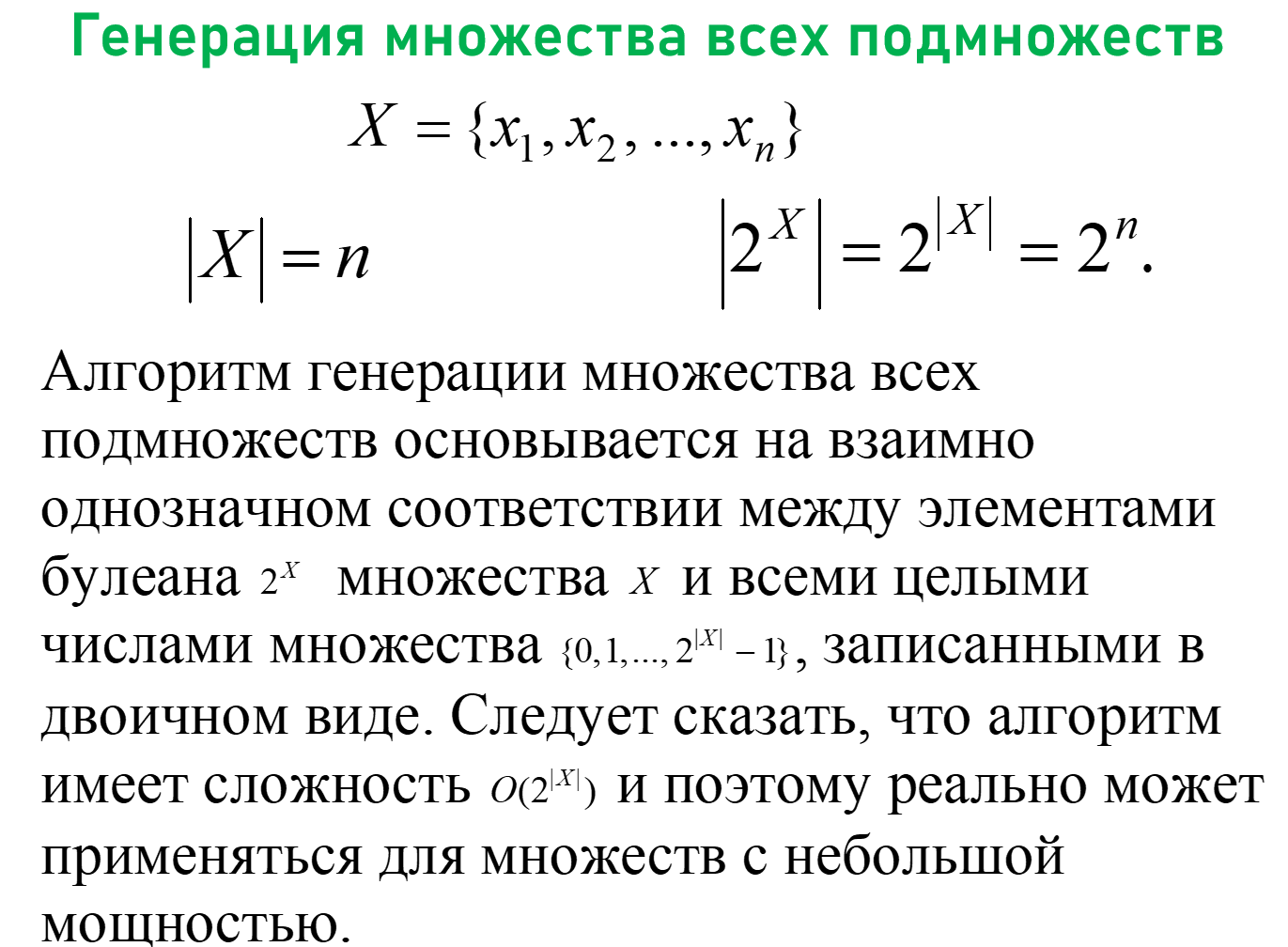
Алгоритм:

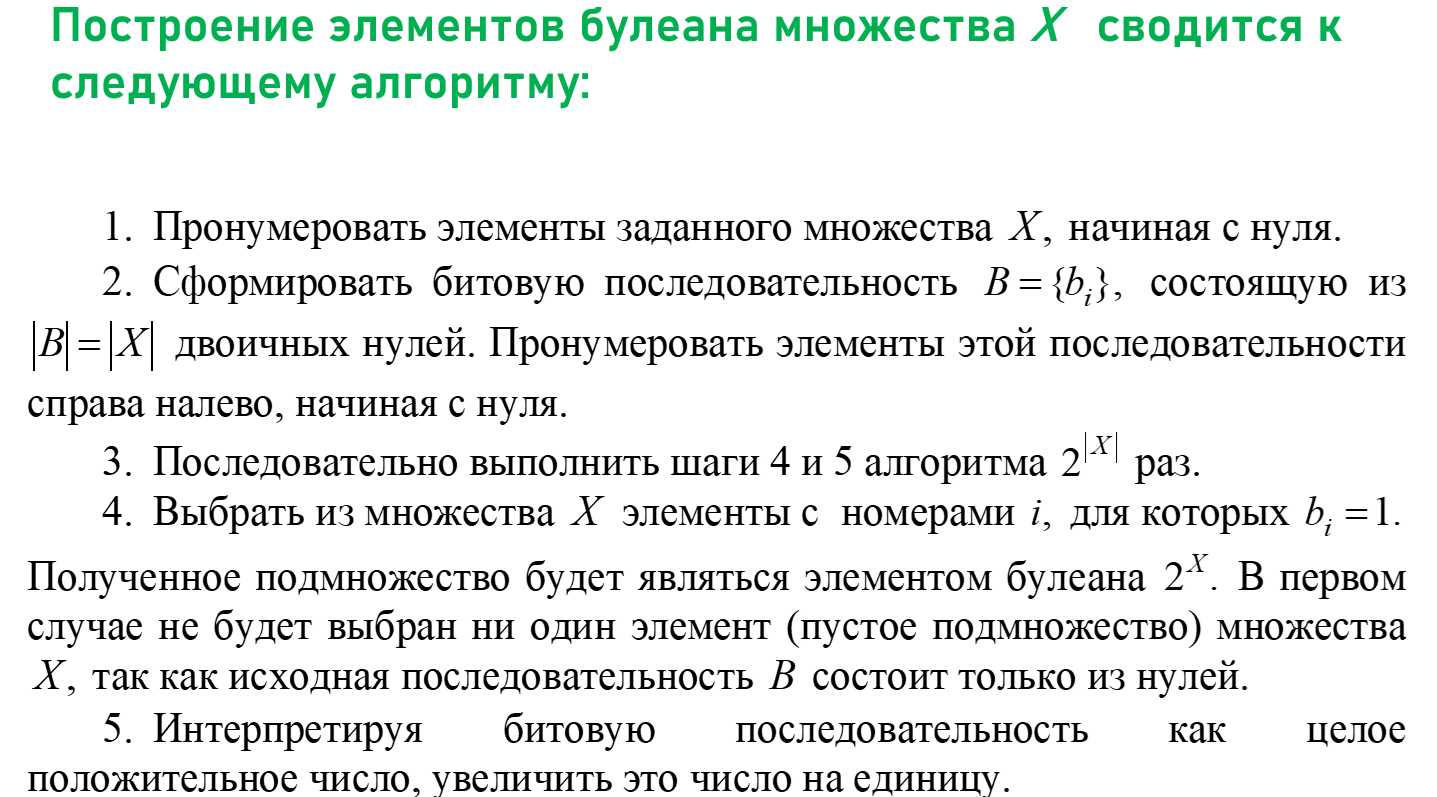
1. Находим мин значение в каждом столбце и строке
2. Производим редукцию строк и столбцов, считаем сумму мин значений - этой и есть корневая вершина дерева T
3. Находим оценку для каждого нуля, выбираем 0 с наибольшей оценкой и заменяем его на бесконечность. Удаление этой дуги позволяет получить самую большую константу приведения, т.е увеличить нижнюю границу. Ту строку и столбец в которой находится 2 бесконечности удаляем.
4. Повторяет до тех пор, пока не останется 1 дуга.
5. Расставить дуги в правильном порядке.

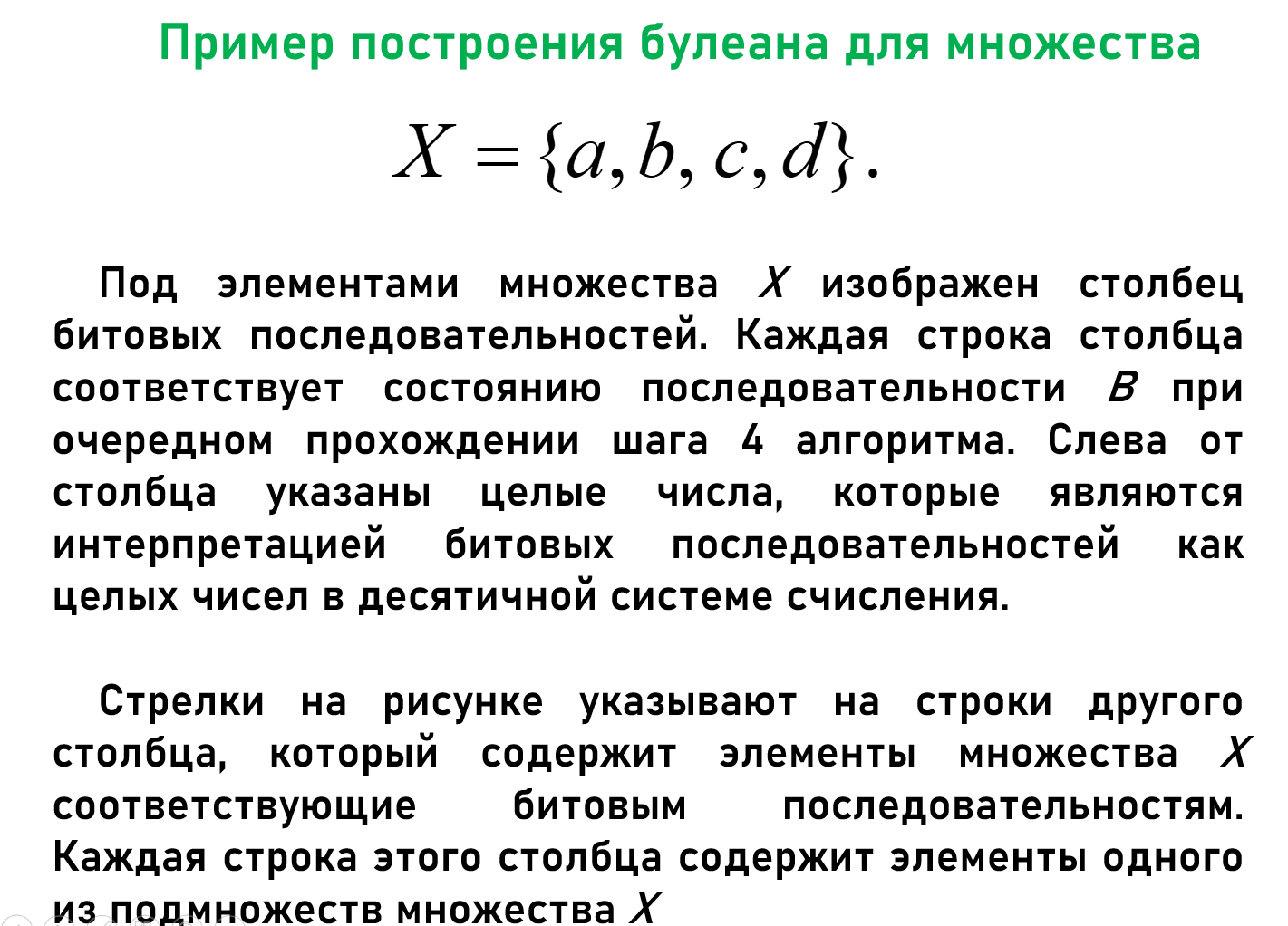
*!ЛК по этой теме супер большая (26 слайдов), в вопросе не вся инфа!*

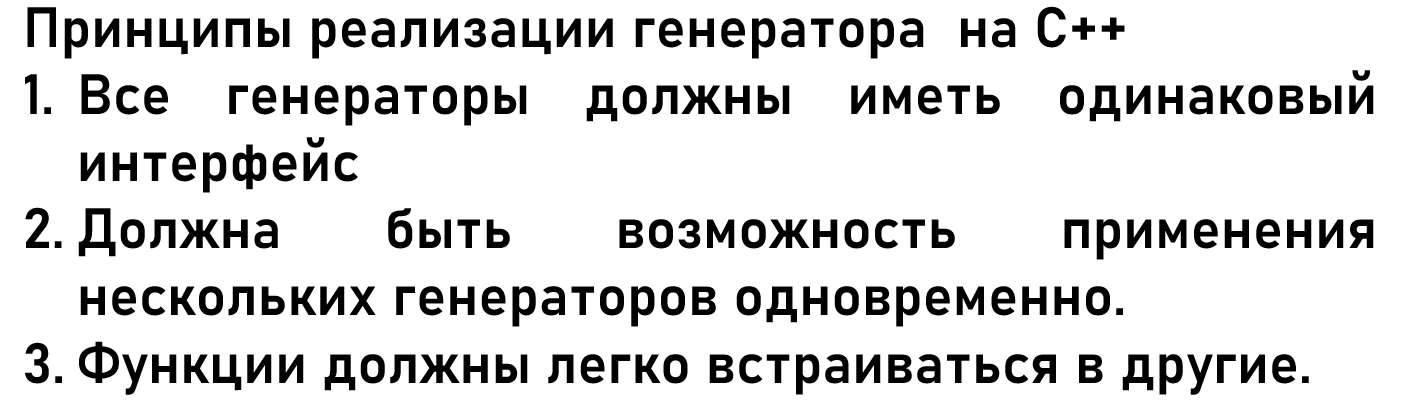
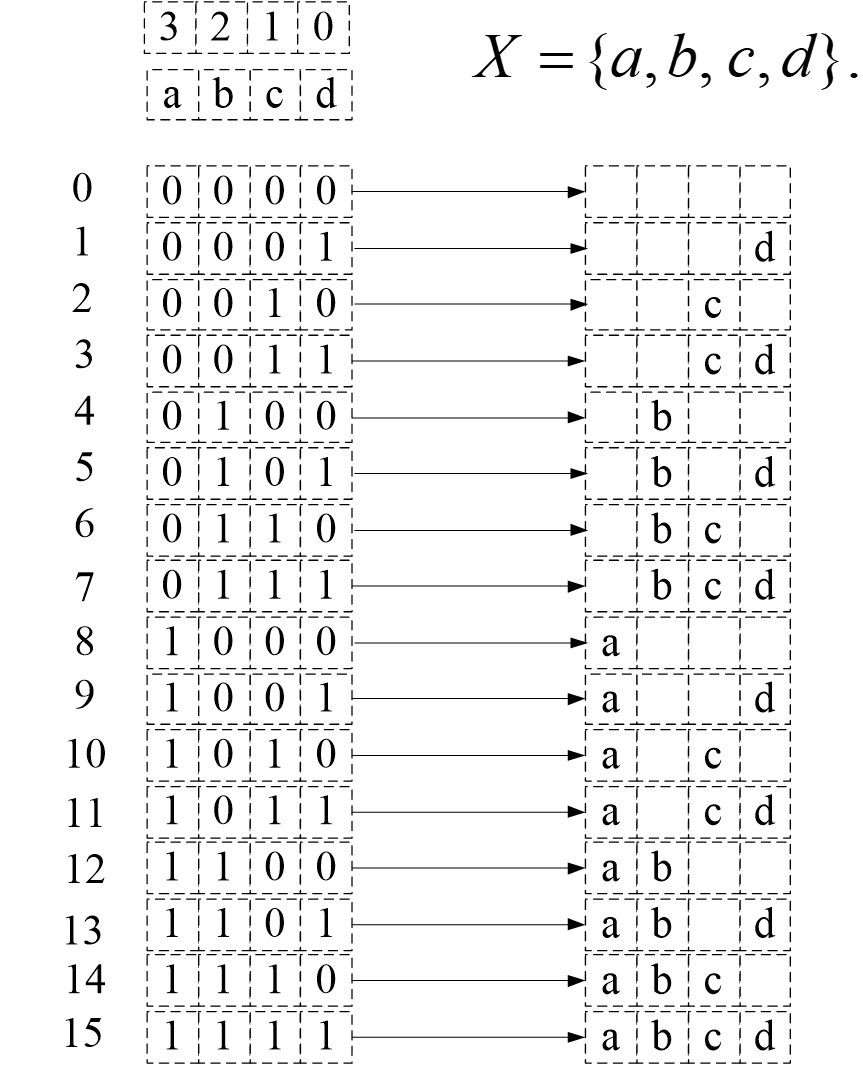
**7. Комбинаторные методы решения оптимизационных задач. Генерация подмножеств заданного множества.**



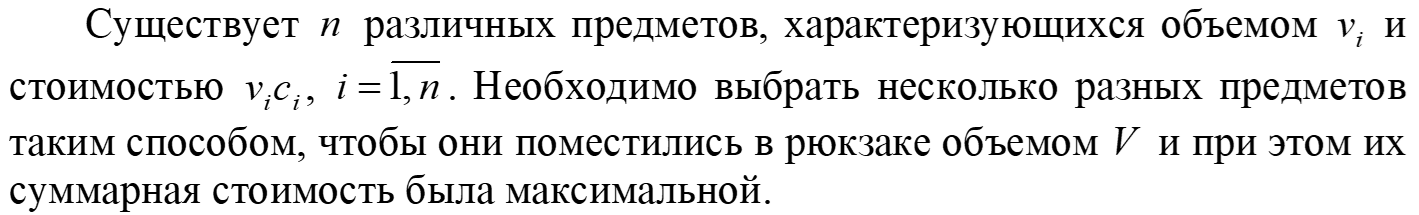




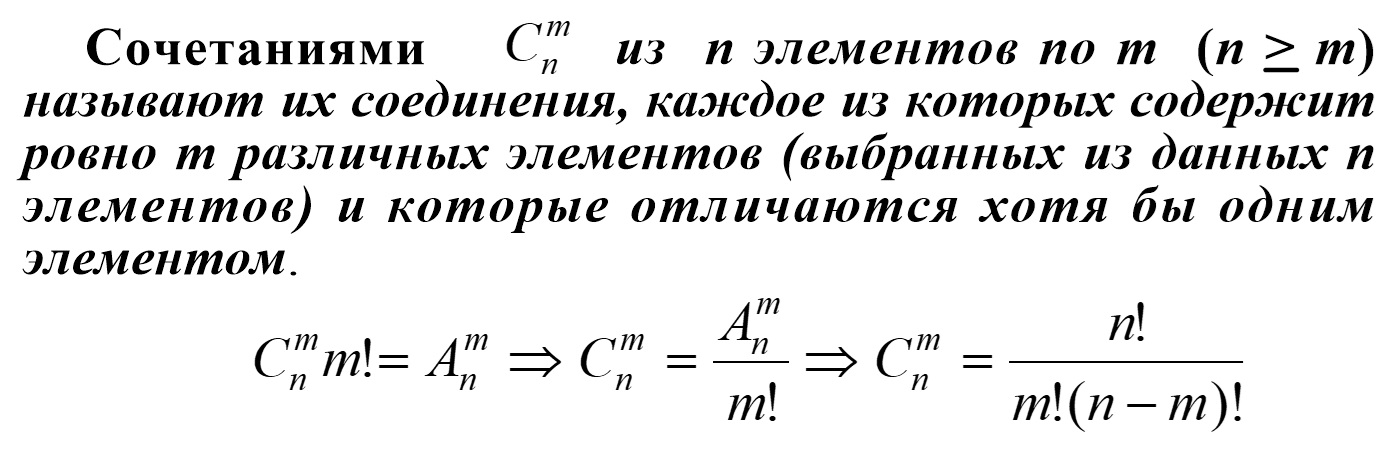


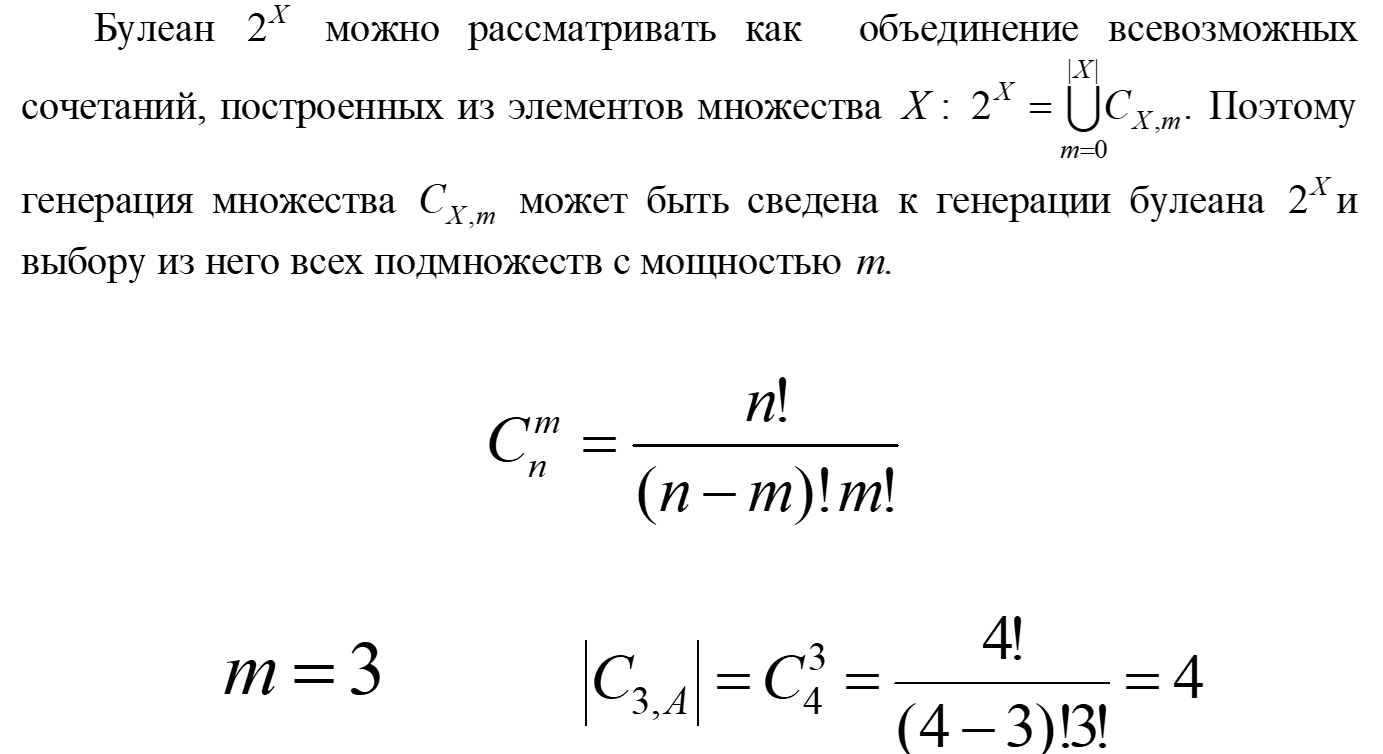


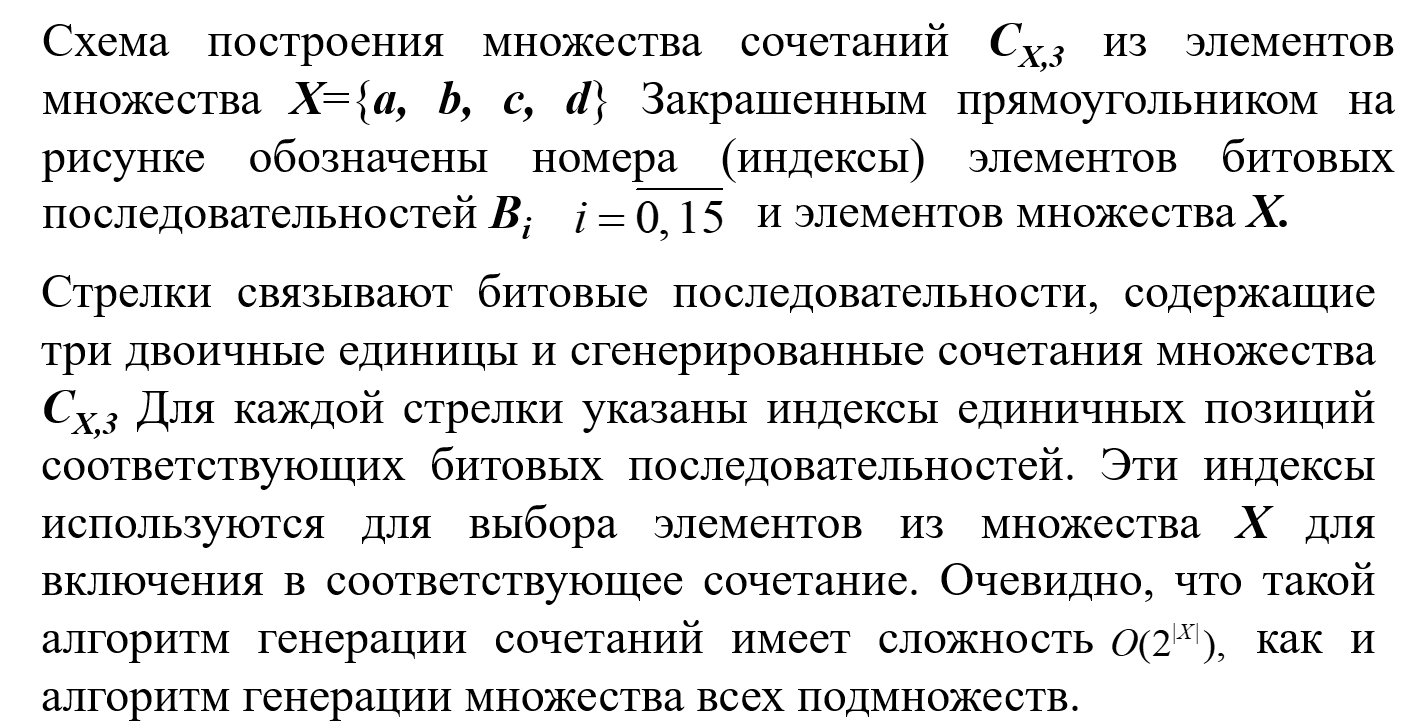
С помощью данного генератора решается задача о рюкзаке:

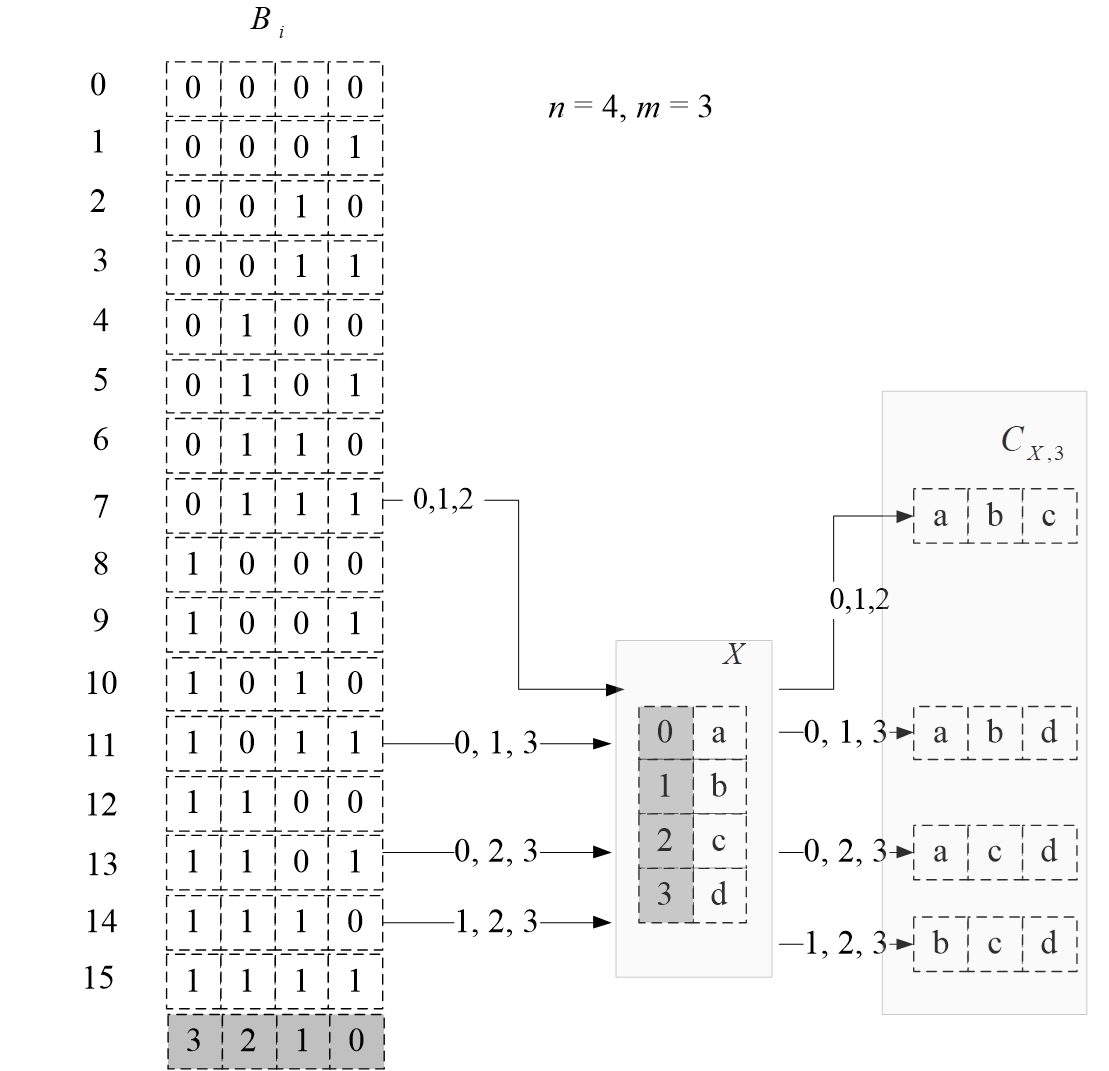


**8. Комбинаторные методы решения оптимизационных задач. Генерация сочетаний.**

****

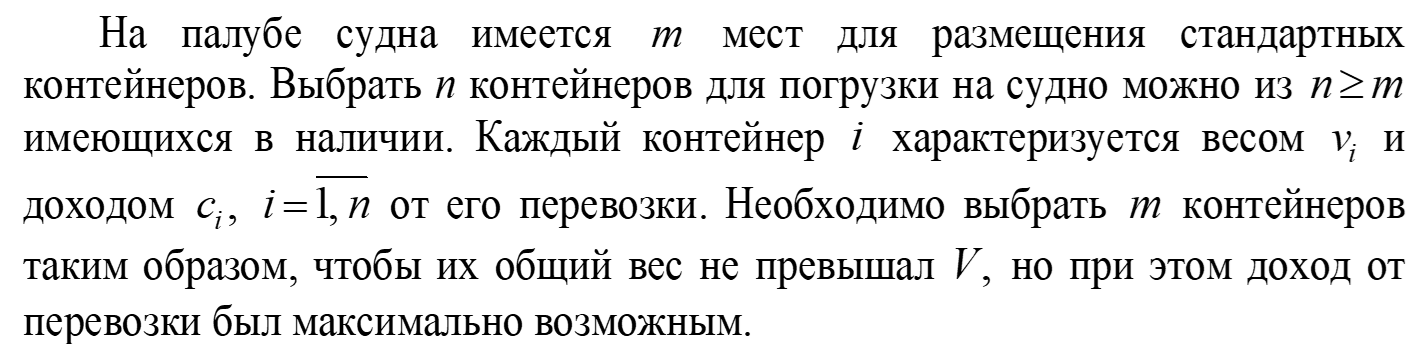




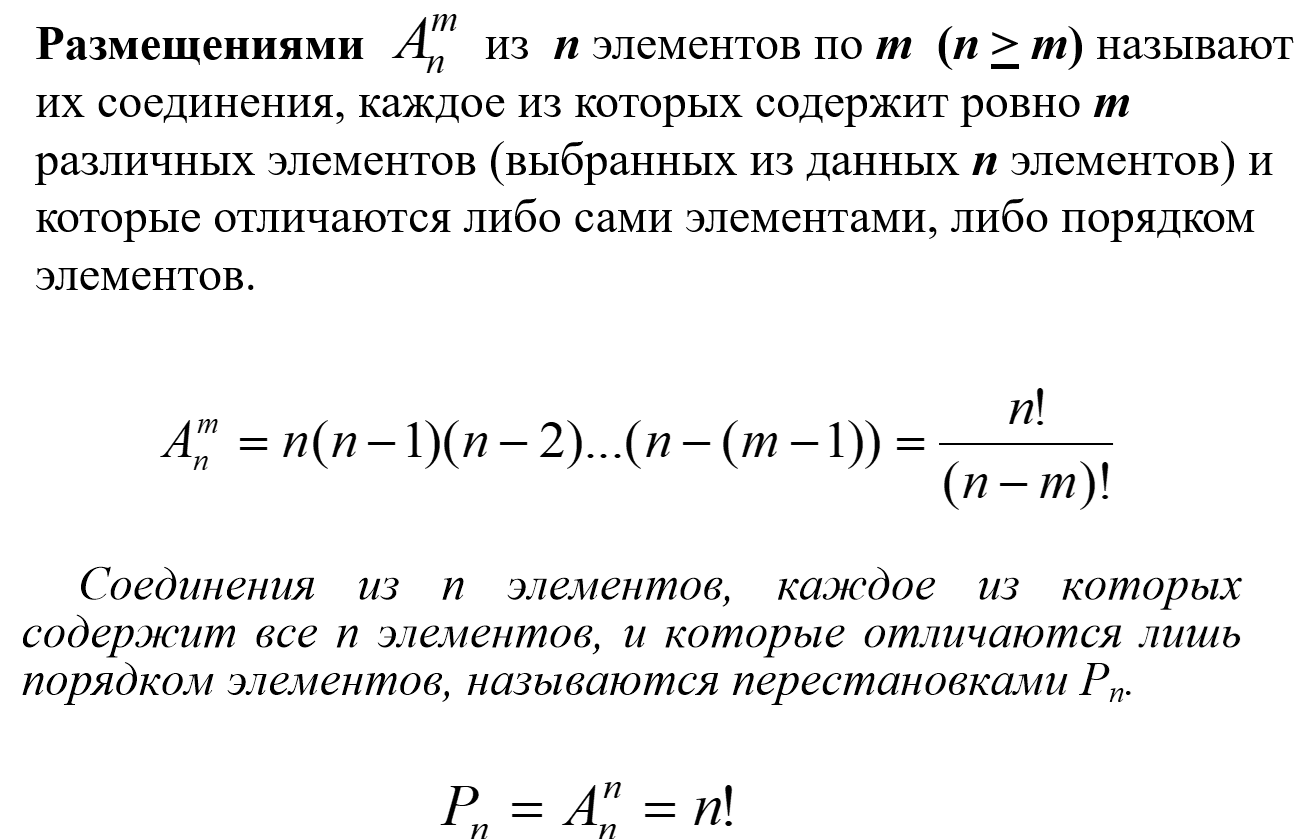




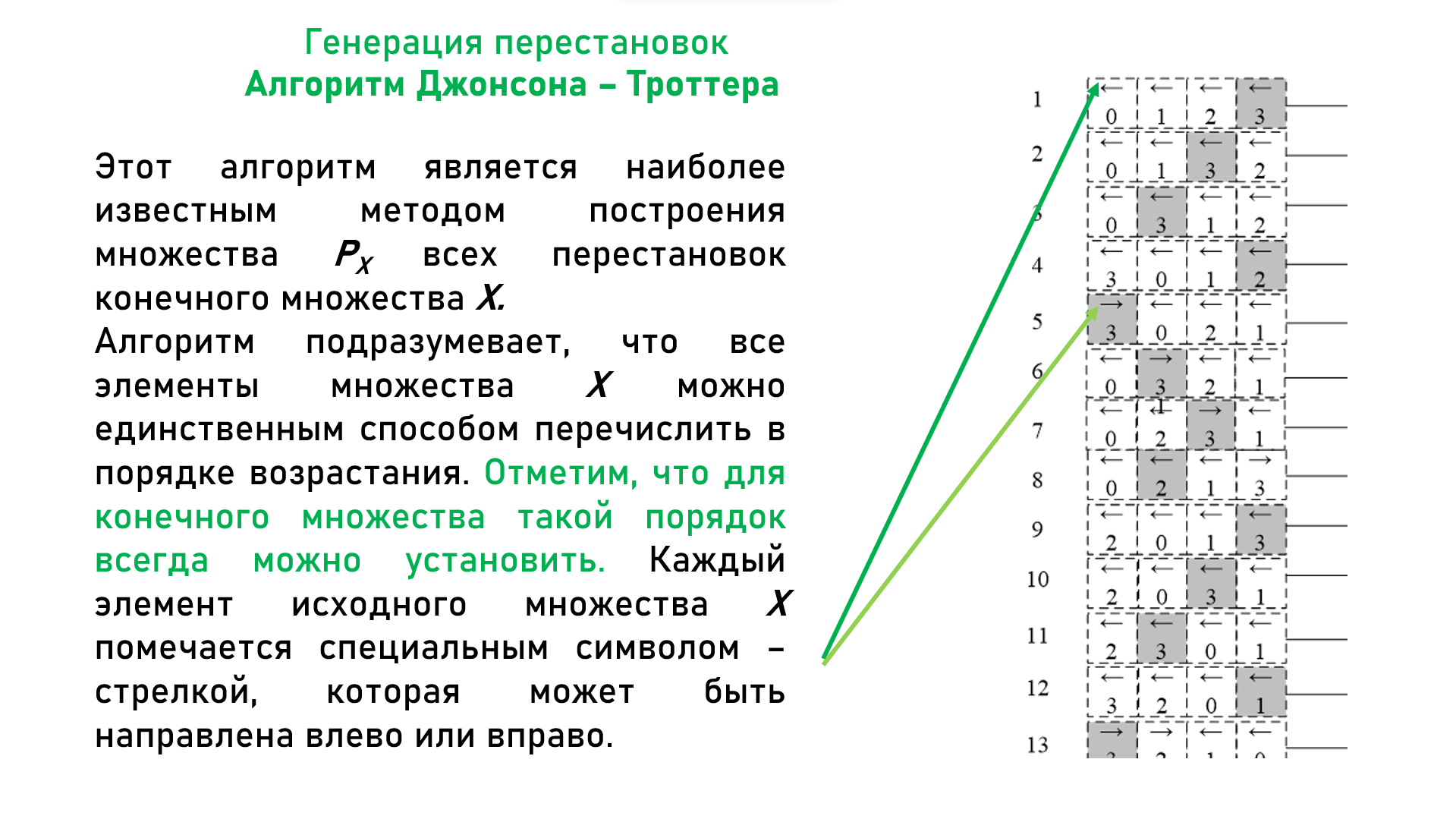
Данным генератором решается задача об оптимальной загрузке судна.

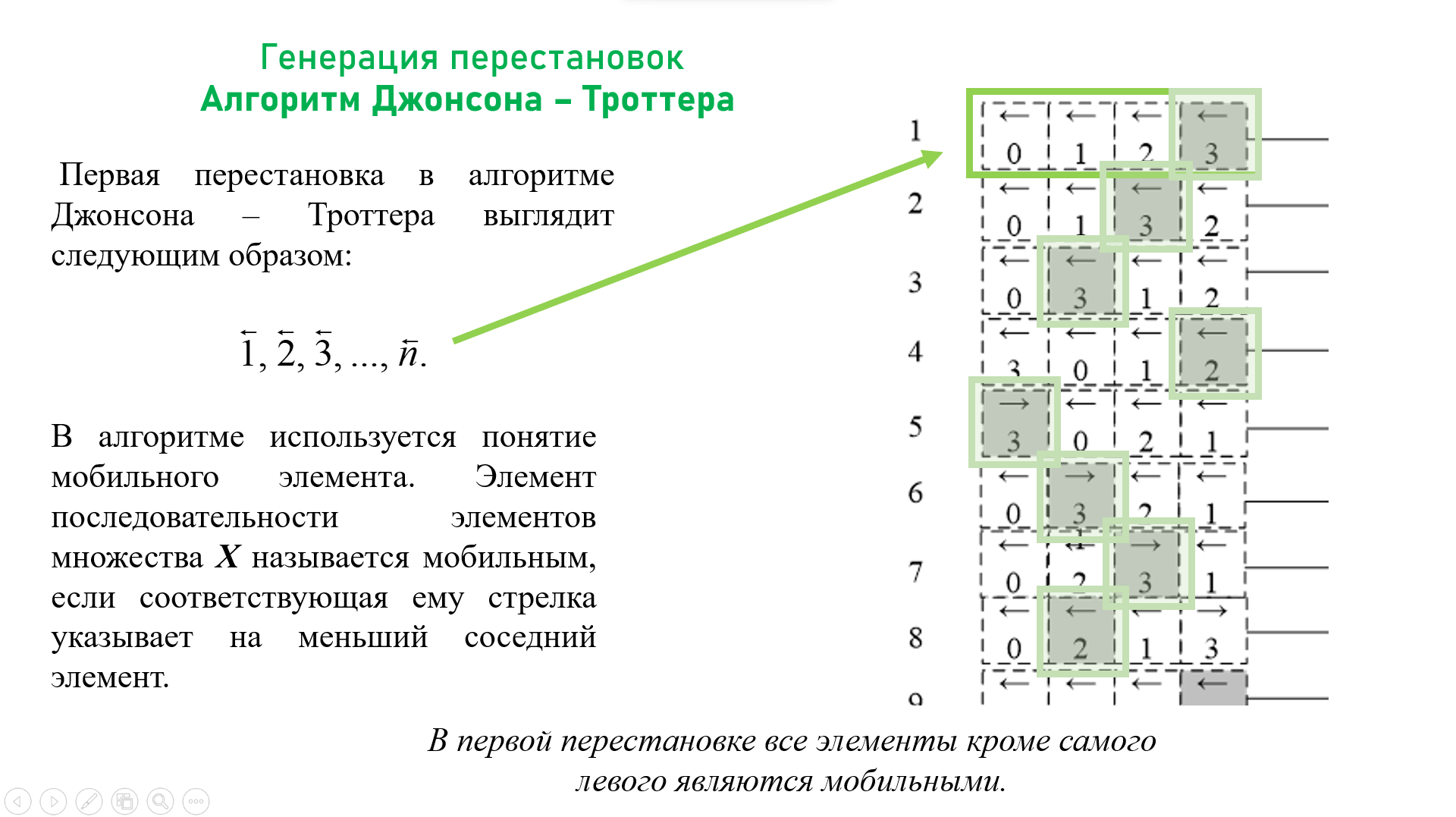


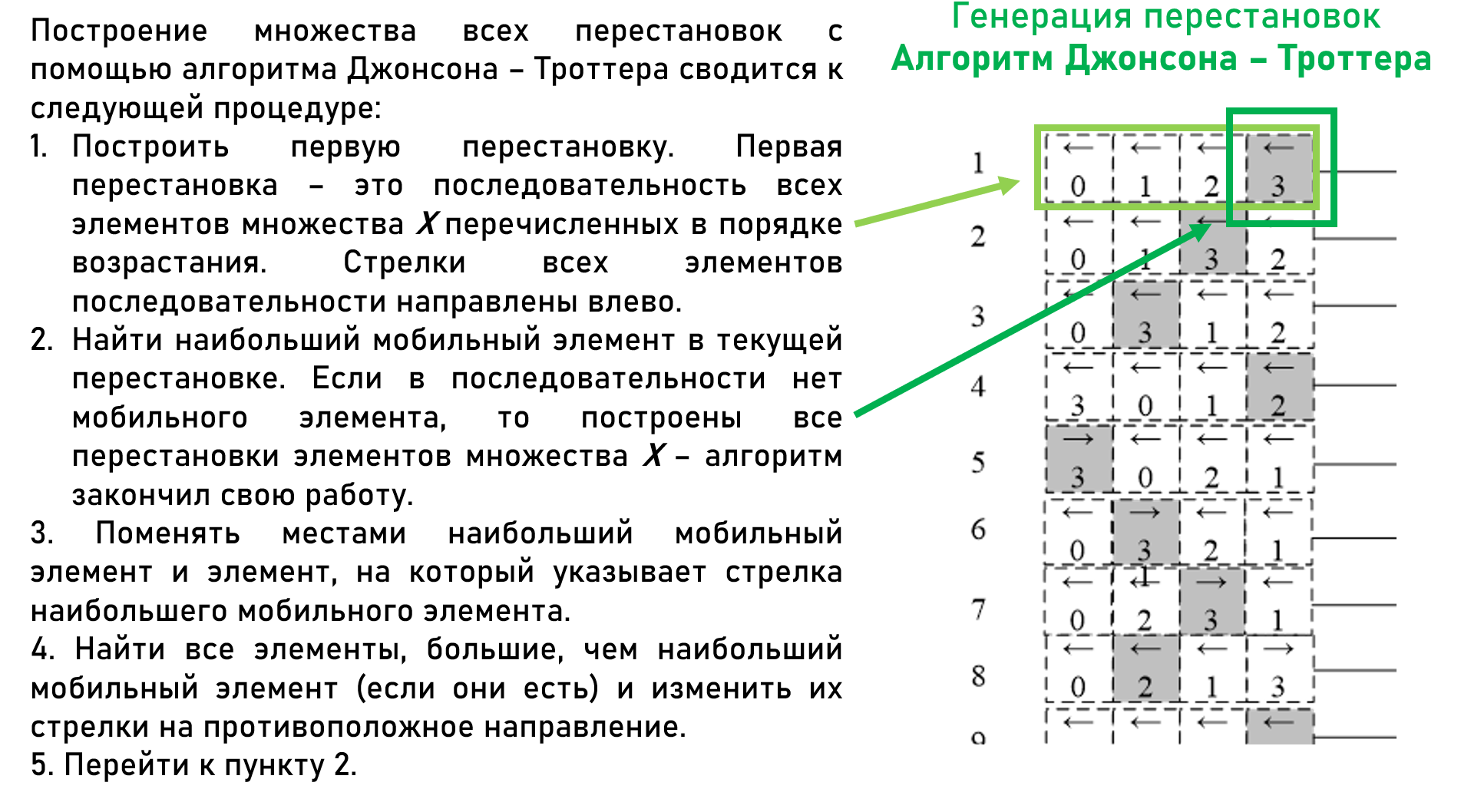
**9. Комбинаторные методы решения оптимизационных задач. Генерация перестановок.**

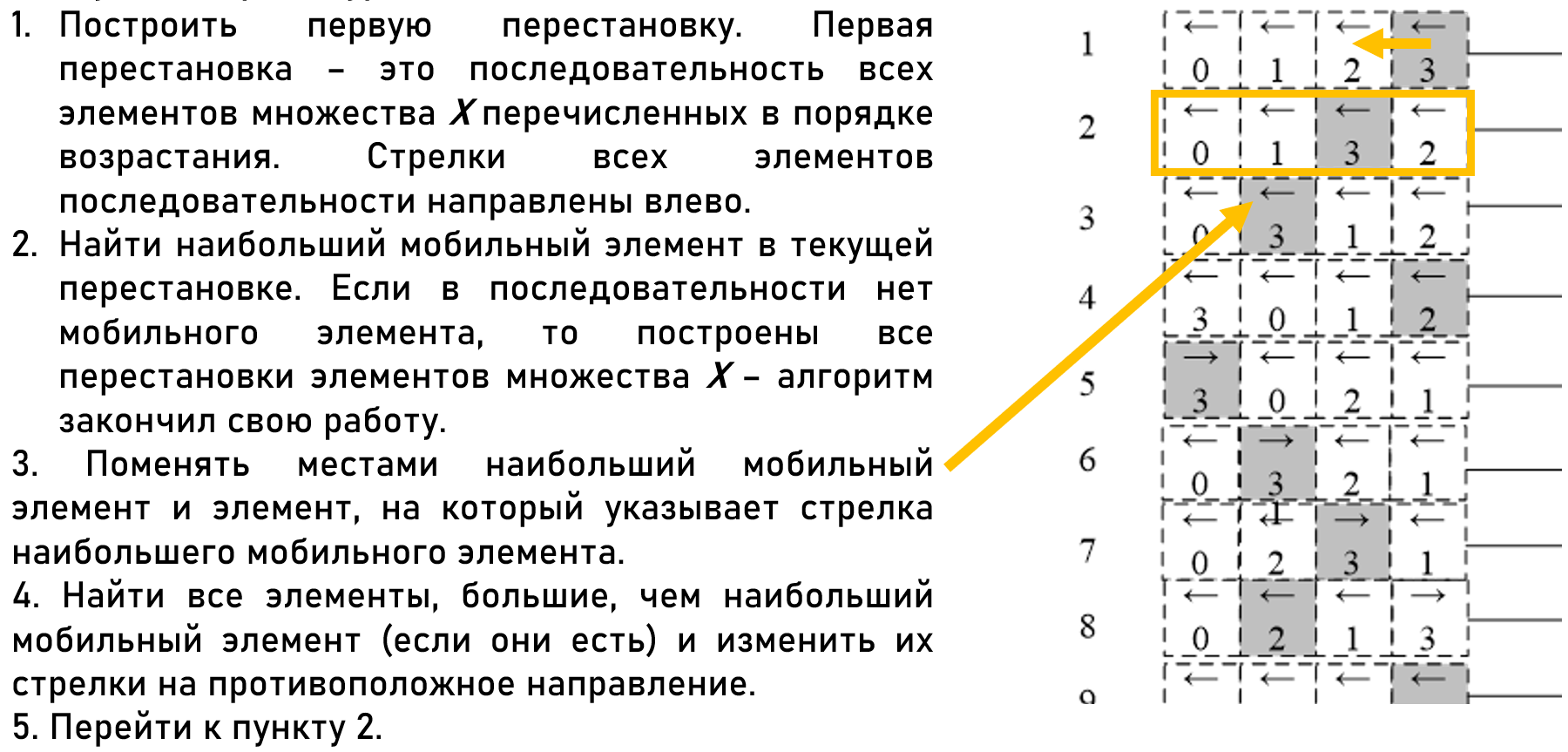
****

Наиболее известным методом построения множества  всех перестановок конечного множества  является ***алгоритм Джонсона – Троттера***. Алгоритм подразумевает, что все элементы множества  можно единственным способом перечислить в порядке возрастания.

****

****

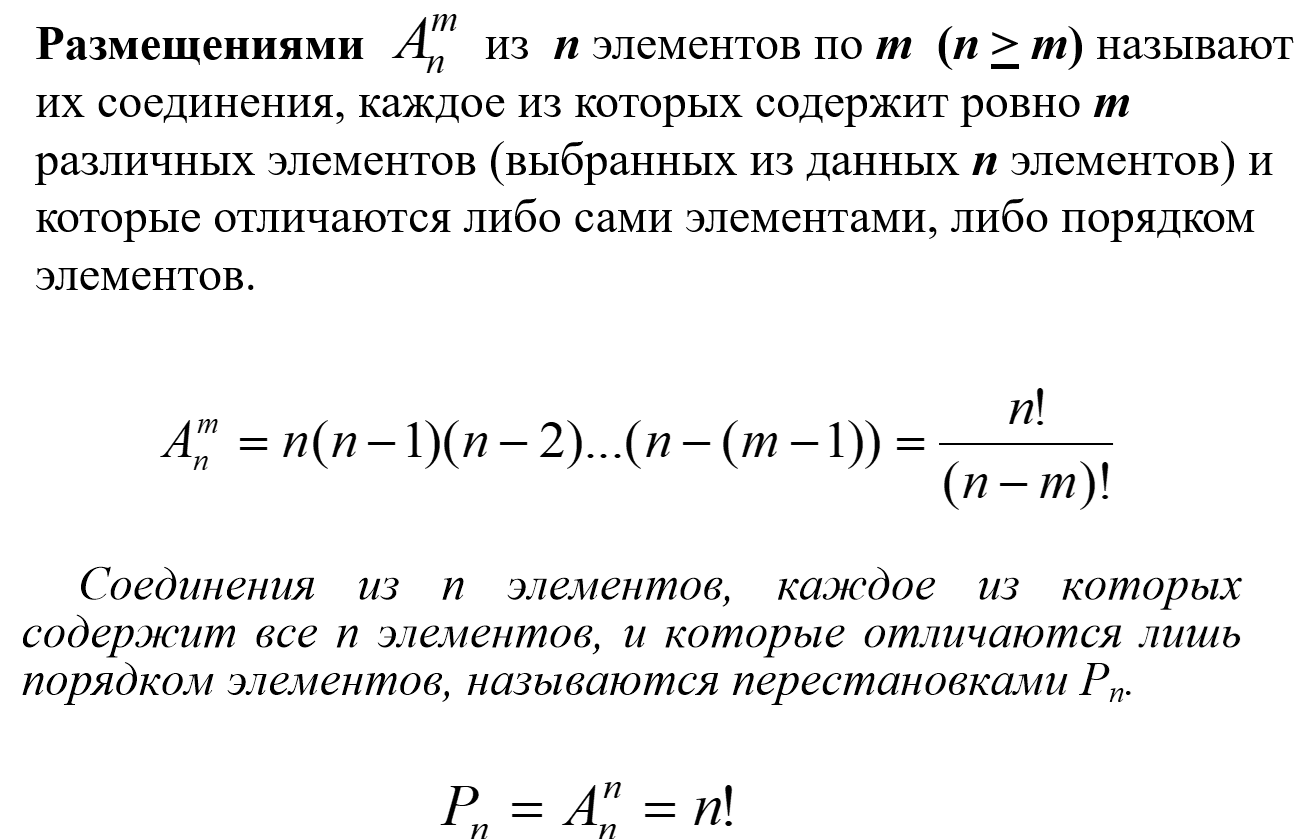
****

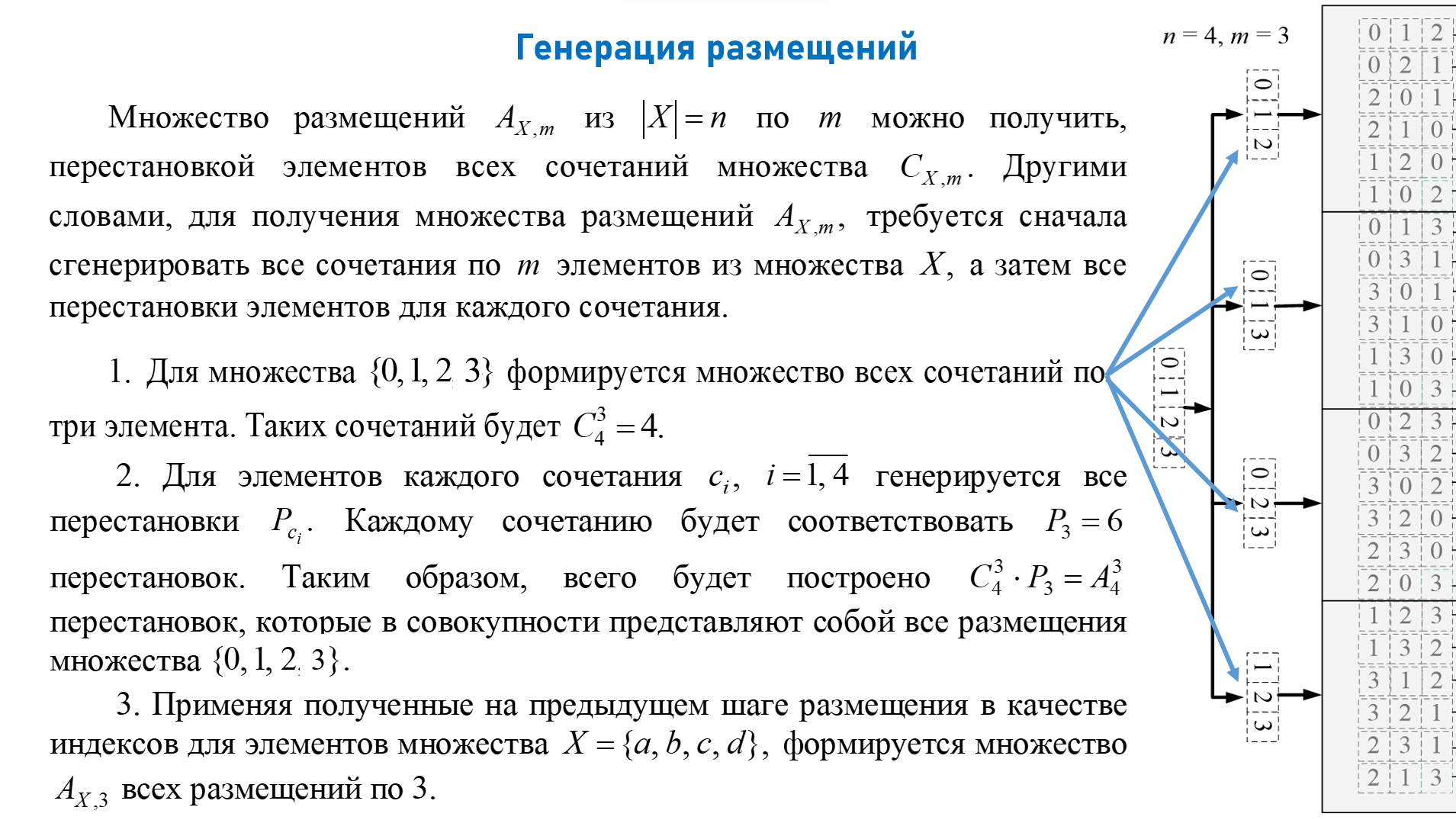
****

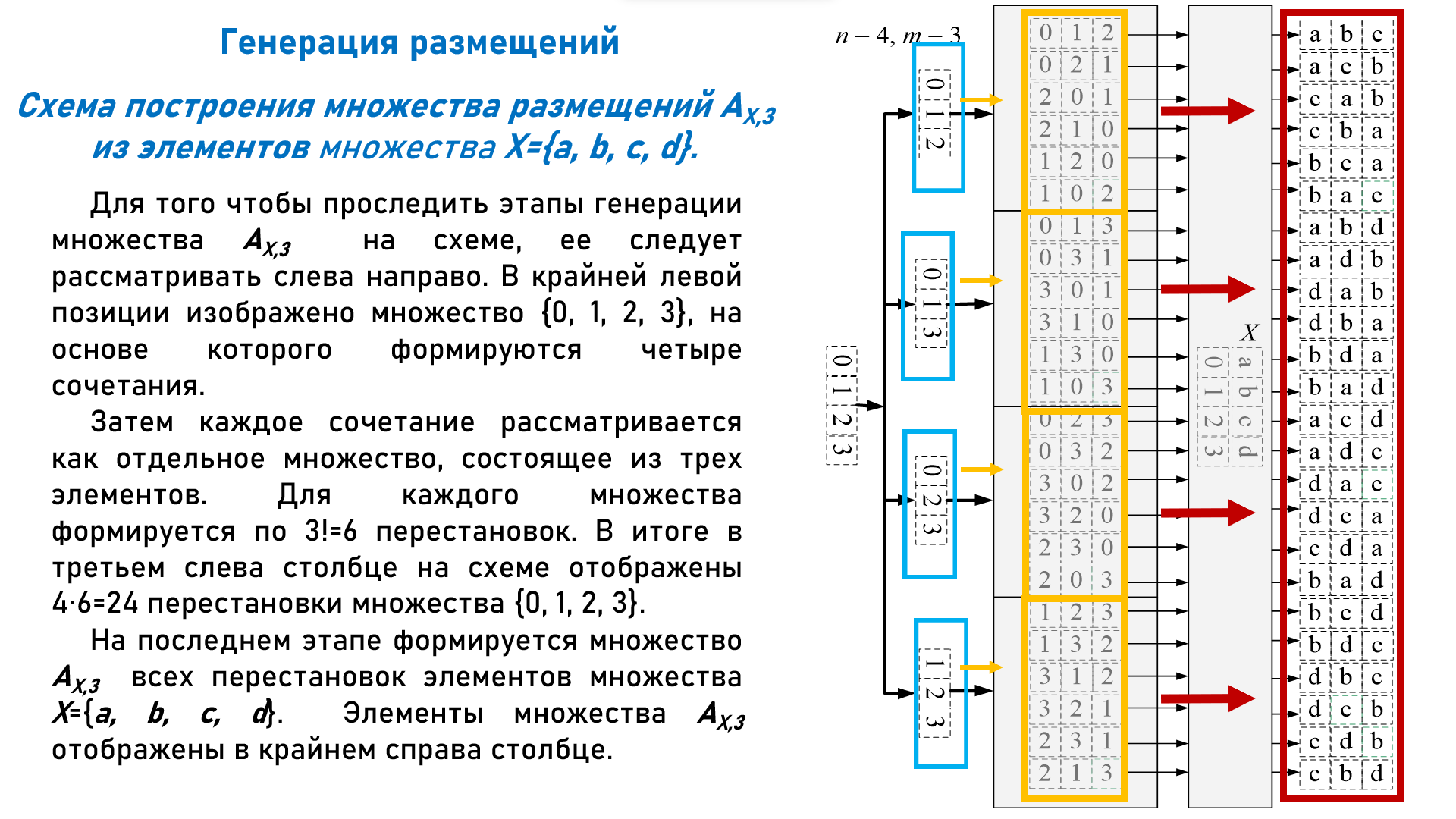


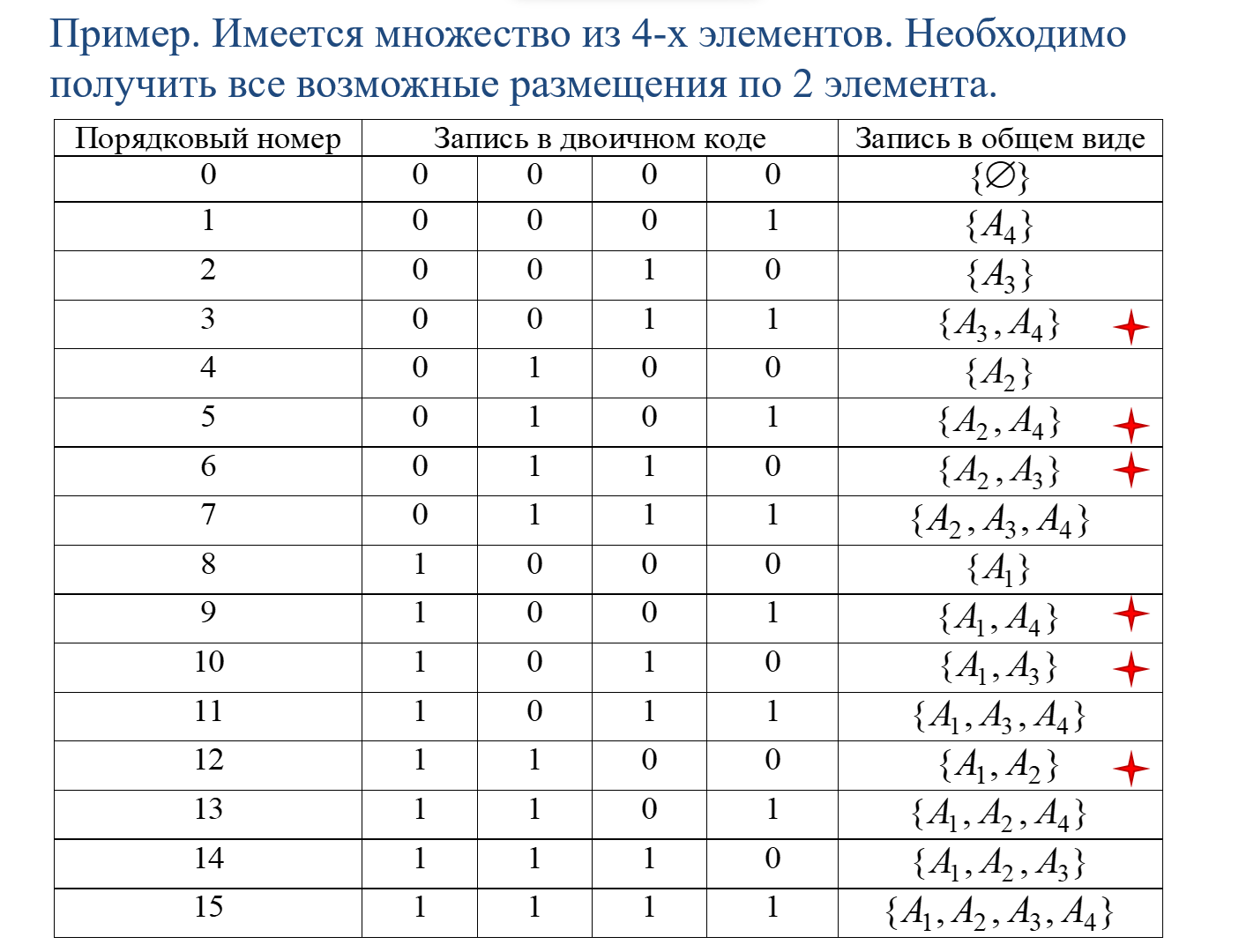
C помощью данного генератора решается задача о коммивояжере

**10. Комбинаторные методы решения оптимизационных задач. Генерация размещений.**

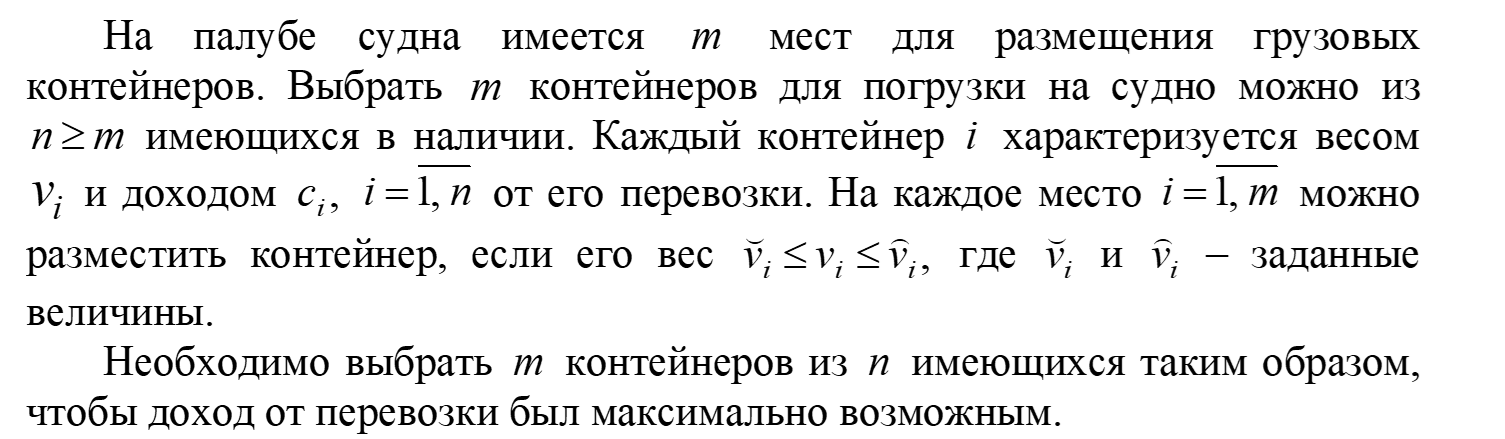
****







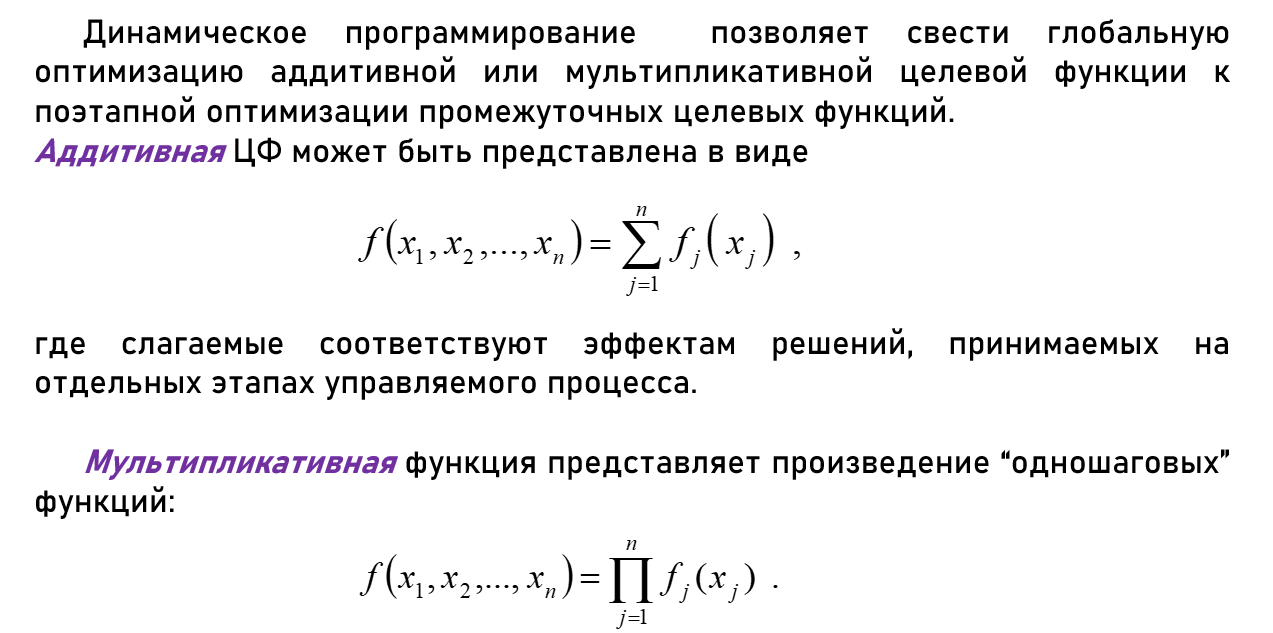
Генератор размещений решает задачу об оптимальном размещении контейнеров на судне



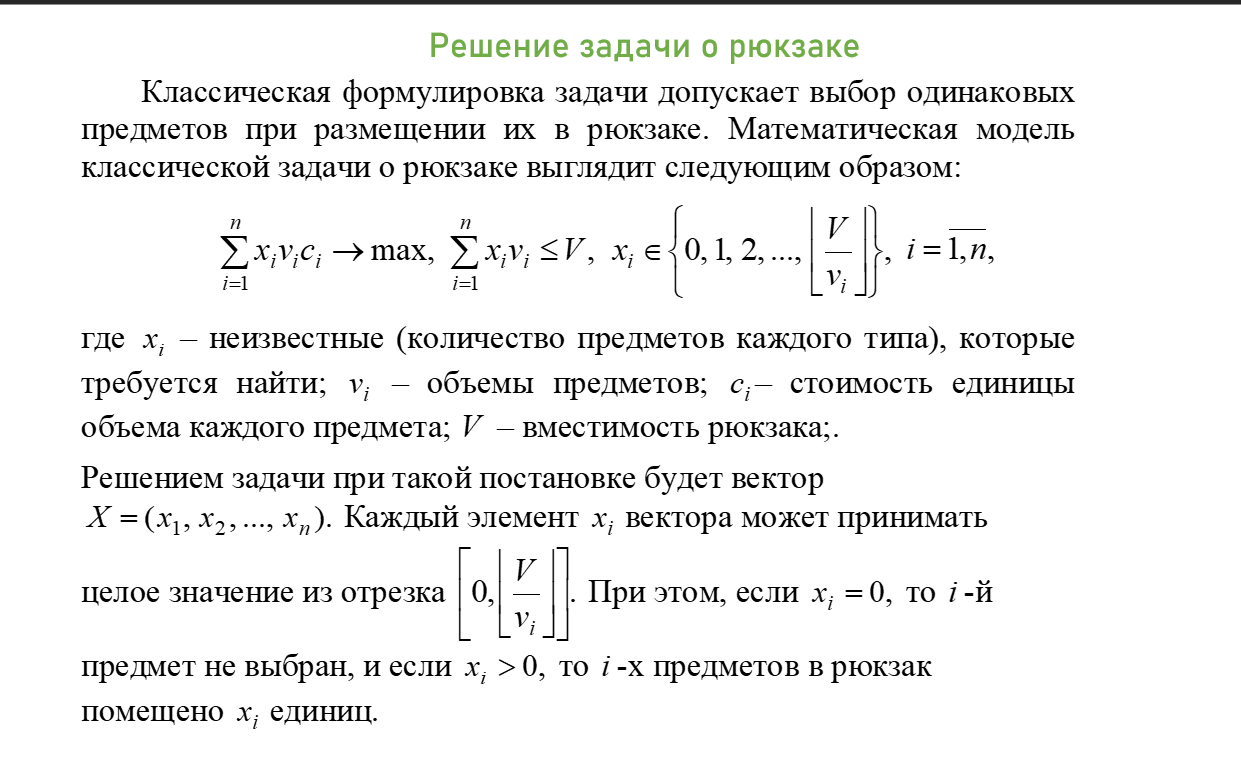
**11. Динамическое программирование. Вычислительная схема решения задачи динамического программирования (на примере решения задачи о рюкзаке).**

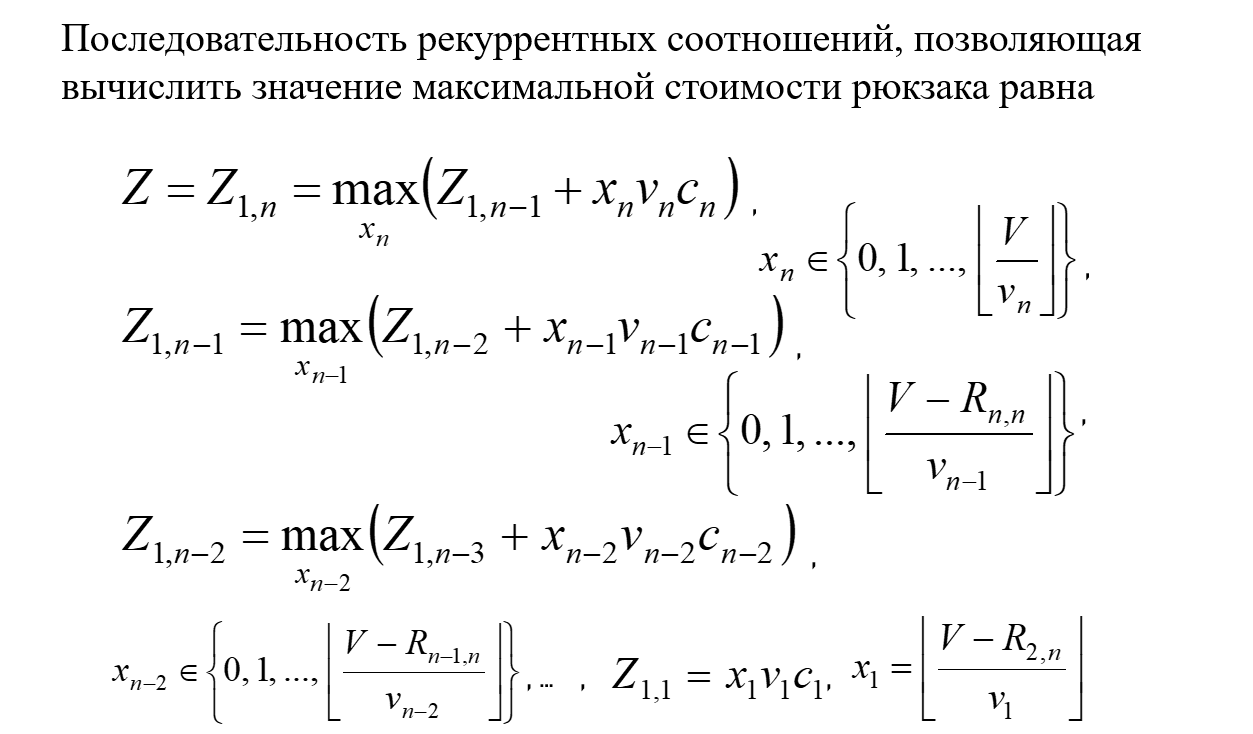
Динамическое программирование используется при оптимальном планировании управляемых процессов и наиболее эффективно в случае многошаговых или многоэтапных процессов принятия решений.

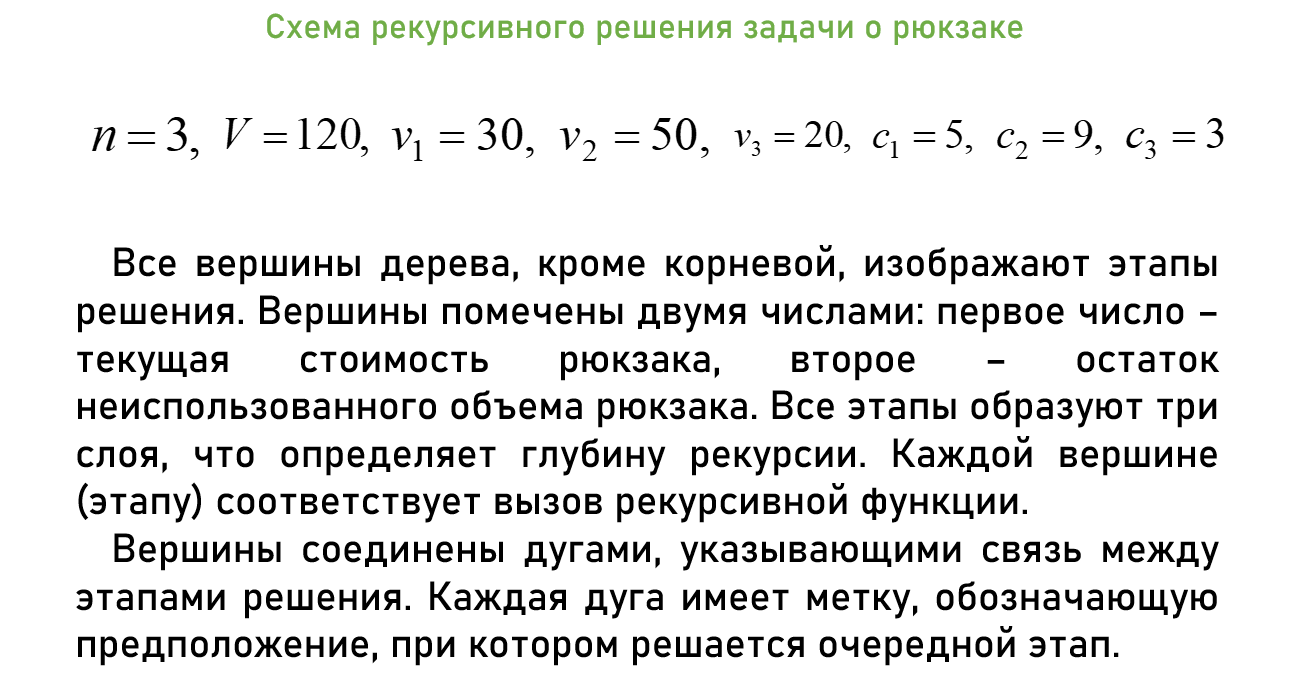
Задача ДП состоит в поиске оптимального управления, переводящего систему из начального состояния в конечное, и обеспечивающего экстремум целевой функции.

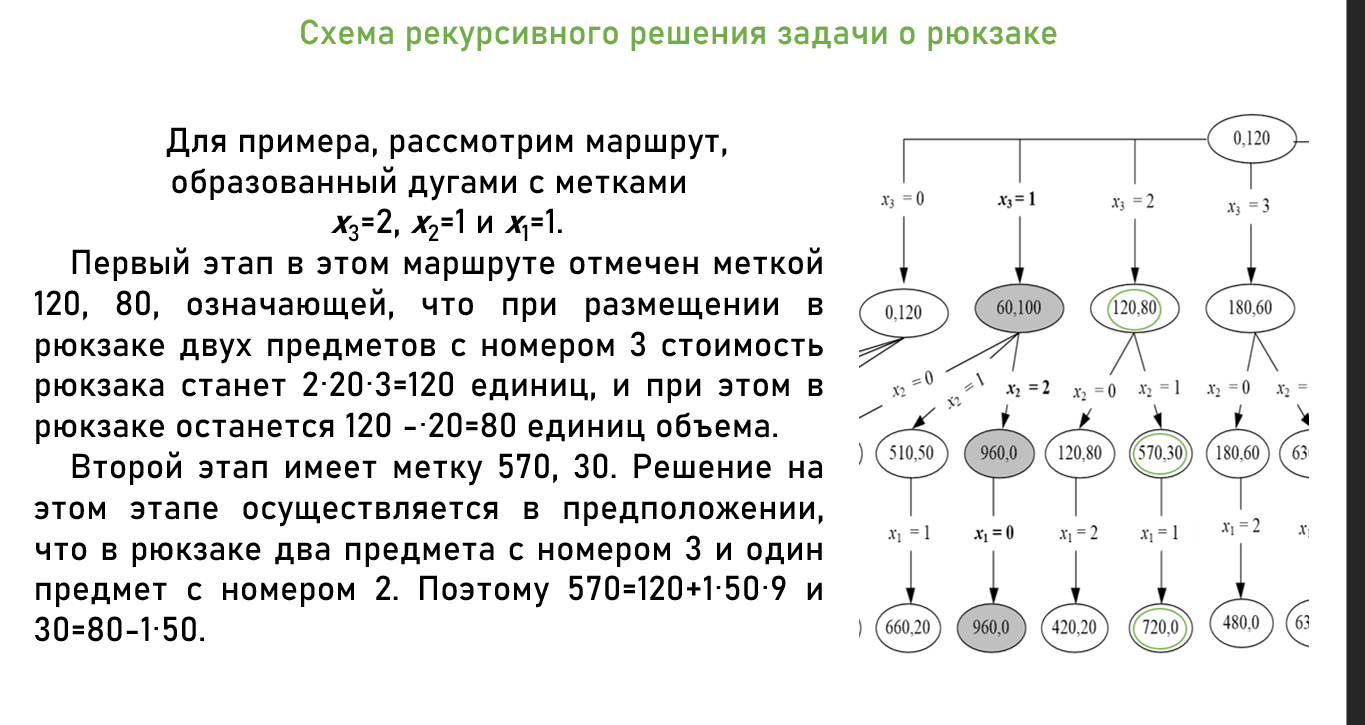


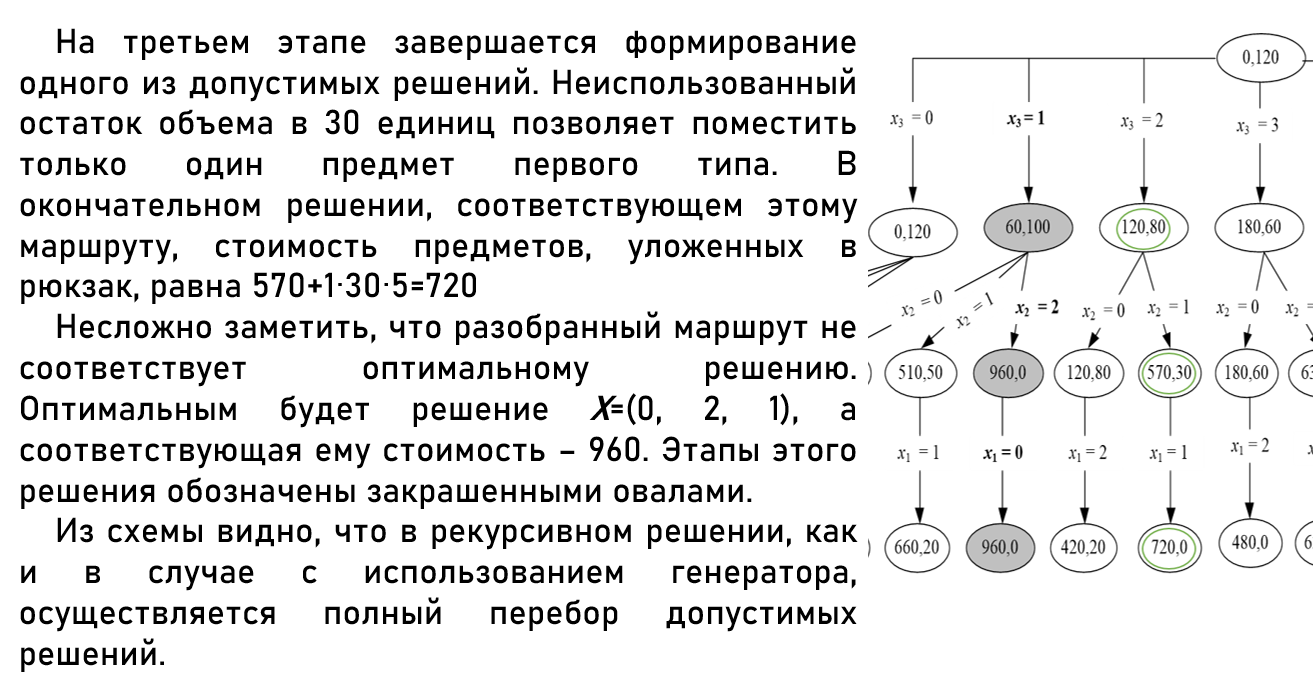
К преимуществам метода динамического программирования по сравнению с “классическими” методами оптимизации относятся более высокая скорость расчетов и широкая область применимости. В частности, для него некритично требование линейности и дифференцируемости функций **fj**(**xj**) и функция выигрыша может быть задана не в аналитическом, а в табличном виде. Другими словами, метод применим и при решении задач нелинейного и дискретного программирования.

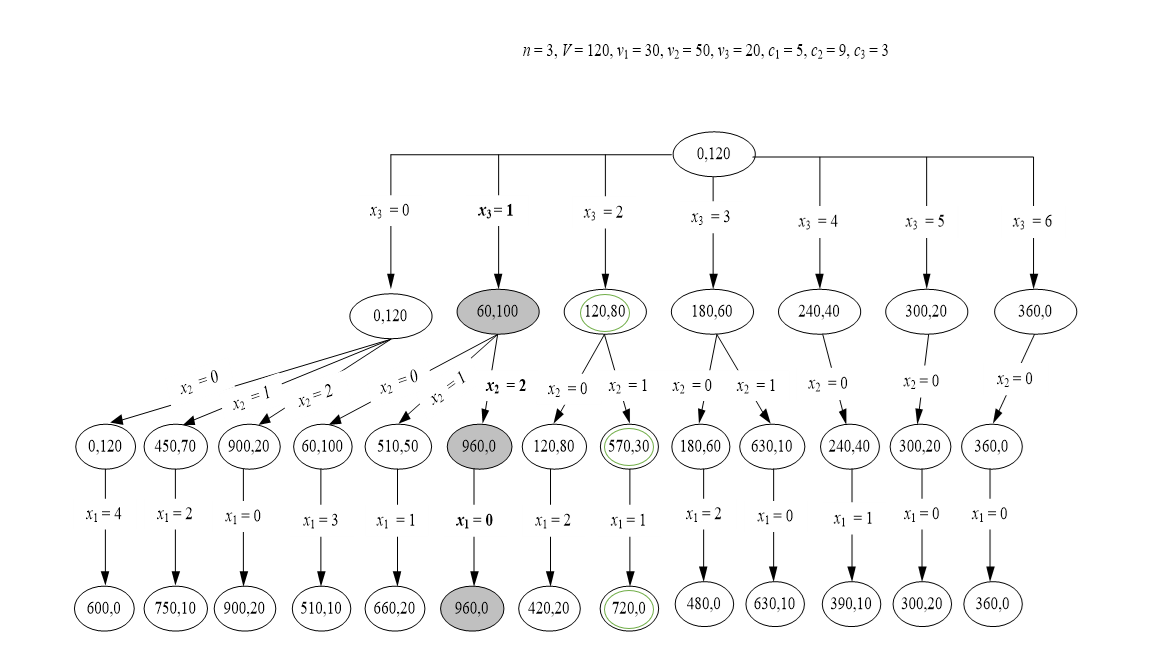












**12. Рекурсивные алгоритмы.**

Многие оптимизационные алгоритмы основаны на принципе разбиения основной задачи на подзадачи, каждая из которых повторяет основную, но входные их данные таковы, что область допустимых решений становится меньше.

***Рекурсивный алгоритм*** – это алгоритм, решающий задачу путем сведения ее к решению одной или нескольких таких же задач, но в сокращенном их варианте.

Существует два определения понятия рекурсивной функции .

Первое определение рекурсивной функции относится к теории вычислимости и является синонимом понятия вычислимой функции, т. е. функции, для вычисления значения которой можно указать алгоритм.

Второе определение, которое и будет использоваться здесь, происходит из области теории программирования.

***Рекурсивная функция*** – это функция, которая вызывает саму себя. Рекурсивный алгоритм может быть записан в виде рекурсивной функции.

Классическими примерами рекурсивных функций являются функции для вычисления факториала, чисел Фибоначчи и наибольшего общего делителя с помощью алгоритма Эвклида.

Рекурсивную функцию всегда можно преобразовать в цикл, и, наоборот любой цикл можно представить в виде рекурсивной функции.

Рекурсивная запись алгоритма, как правило, не дает выигрыша в скорости его работы. Скорее наоборот, так как вызов любой функции связан с сохранением и восстановлением контекста вызывающей функции, что является затратной по времени операцией.

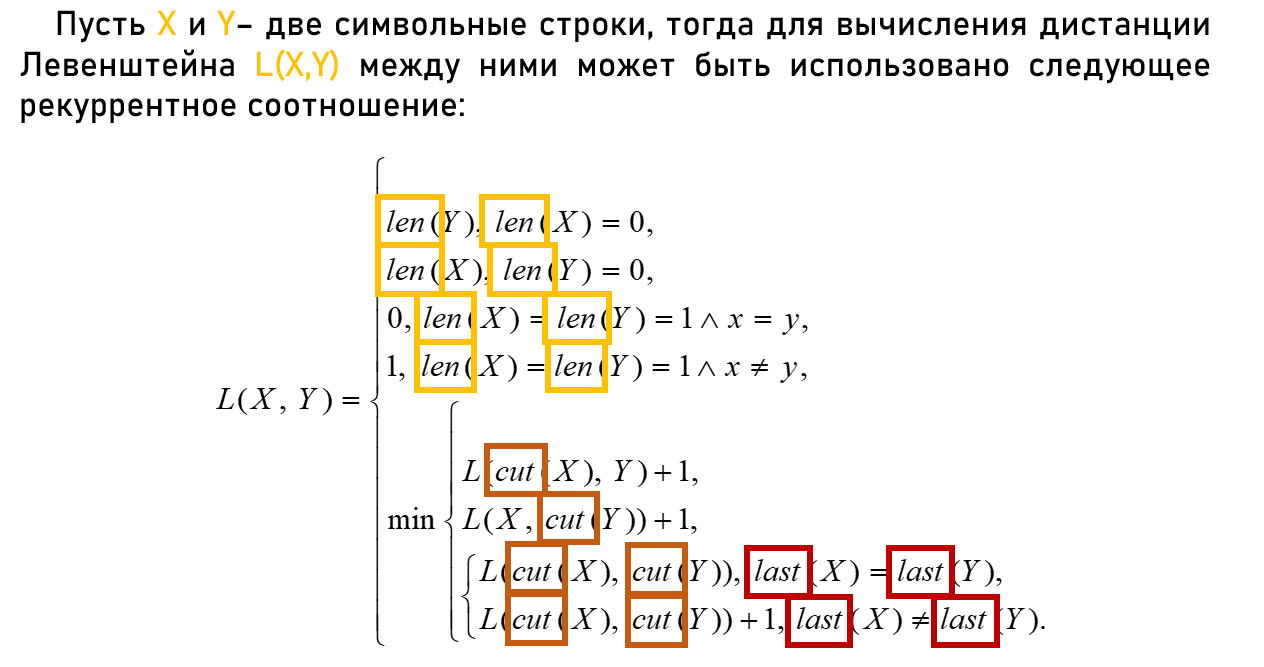
Кроме того, для хранения контекста операционной системой резервируется специальная секция памяти, называемая системным стеком.

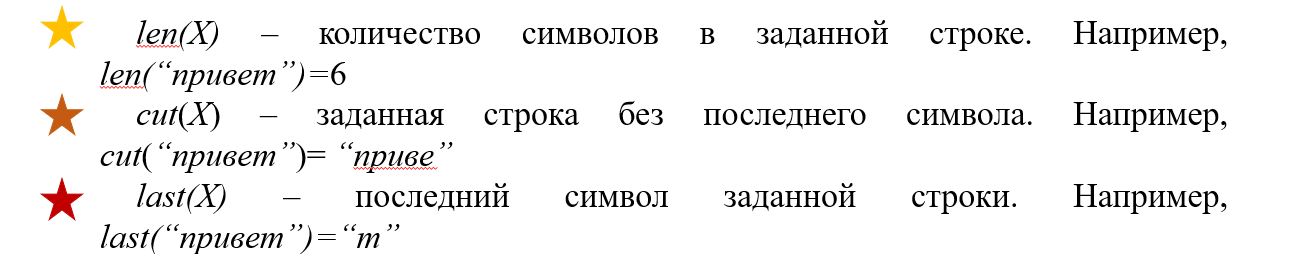
Если цепочка вызовов функций является длинной (иногда говорят о большой ***глубине рекурсии***), то это может привести к переполнению стека. Например, при вычислении факториала числа 25 глубина рекурсии достигает значения 24.

Часто рекурсивные функции, применяемые для решения оптимизационных задач, используют более одного рекурсивного вызова, каждый из которых работает приблизительно с половиной входных данных. Такую схему решения называют ***«разделяй и властвуй»***.

***Дистанция Левенштейна (расстояние Левенштейна, редакционное расстояние, дистанция редактирования)*** определяется между двумя строками и равна минимальному количеству операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Расстояние Левенштейна активно применяется для исправления ошибок в поисковых системах, в текстовых редакторах, а также в биоинформатике.





Пример:

1. L(“ель”, “дрель”) = min
2. L(“ел”, “дрель”) = min
3. L(“ель”, “дрел”) = min
4. L(“ел”, “дрел”) = min
5. = min

L(“ ”, ”дрель”) = 5, L(“”, “дрел”) = 4

1. L(“e”, ”дрел”) =min

L(“ ”, “дрел”) = 4, L(“ ”,” дре”) = 3

1. L(“ель”, ”дре”) = min
2. = min
3. L=(“ель”, ”др”) = min
4. L=(“ель”, ”д”) = min

L=(“ель”, ””) = 3, L=(“ел”, ””) = 2

1. L=(“е”, ”дре”) = min

L=(“”, ”дре”) = 3, L=(“”, ”др”) = 2

1. L=(“ел”, ”др”) = min

L=(“е”, ”д”) = 1

1. L=(“е”, ”др”) = min

L(“”, “др”) = 2, L(“е”, “д”) = 1, L(“”, “д”) = 1

1. L=(“ел”, ”д”) = min

L(“е”, “д”) = 1, L(“ел”, “”) = 2, L(“е”, “”) = 1

1. L=(“ел”, ”д”) = min(2,3,2) = 2
2. L=(“е”, ”др”) = min(3,2,2) = 2
3. L=(“ел”, ”др”) = min(2,2,2) = 2
4. L=(“е”, ”дре”) = min(4,2,3) = 2
5. L=(“ель”, ”д”) = min(3,4,3) = 3
6. L=(“ель”, ”др”) = min(3,4,3) = 3
7. = min(3,3,3) = 3
8. L(“ель”, ”дре”) = min(4,4,3) = 3
9. L(“e”, ”дрел”) =min(5,3,4) = 3
10. = min(6,4,5) = 4
11. L(“ел”, “дрел”) = min(4,4,2) = 2
12. L(“ель”, “дрел”) = min(3,5,4) = 3
13. L(“ел”, “дрель”) = min(5,3,4) = 3
14. L(“ель”, “дрель”) = min(4,4,2) = 2

**13. Математические основы сетевого планирования. Основные понятия теории графов.**