**Вопросы к экзамену по курсу «Математическое программирование»**

**Теоретическая часть**

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

**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.

Решение задачи математического программирования осуществляется в 4 этапа.

1. Построение математической модели.
2. Классификация задачи.
3. Выбор метода решения.
4. Вычисление.

В общем виде **модель задачи математического программирования** выглядит следующим образом:

|  |  |
| --- | --- |
| , , где | |
|  | – искомая, в общем случае векторная, величина; |
|  | – область определения искомой величины; |
|  | – функция цели (функция определяющая значение критерия оптимальности); |

В зависимости от природы множества  и вида функции  задачи математического программирования **классифицируются** как задачи

* дискретного программирования (комбинаторная оптимизация) ( конечно или счетно);
* целочисленного программирования ( подмножество множества целых чисел);
* линейного программирования (– линейная функция,  – может быть определено с помощью линейных неравенств);
* нелинейного программирования (– нелинейная функция и/или в описании  присутствует хотя бы одна нелинейная функция);

векторная оптимизация (– векторная функция).

**Метод решения** задачи математического программирования определятся в зависимости от исходных данных.

Смежные дисциплины:

Модели

Модели

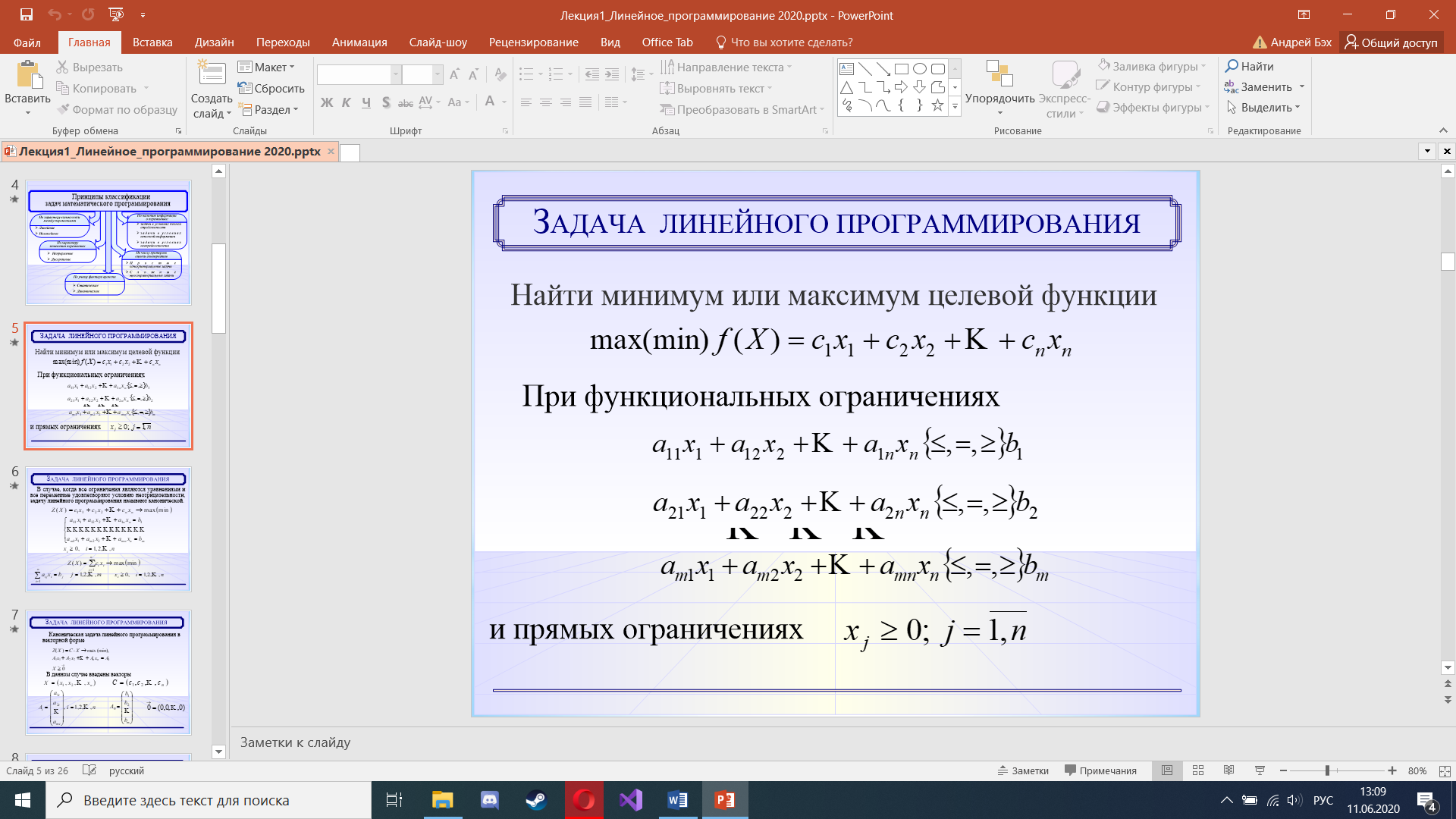
Методы оптимизации

1. **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:



В случае, когда все ограничения являются уравнениями и все переменные удовлетворяют условию неотрицательности, задачу лп называют канонической.

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

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.

1. **Симплекс-метод решения задачи линейной оптимизации. Алгоритмы нахождения опорного(базисного) и оптимального решений.**

Суть этого метода заключается в том, что:

* вначале получают допустимый вариант, удовлетворяющий всем ограничениям, но необязательно оптимальный (так называемое начальное опорное решение);
* оптимальность достигается последовательным улучшением исходного варианта за определенное число этапов (итераций);
* нахож­дение начального опорного решения и переход к следующему опорному решению проводятся на основе применения метода Жордана-Гаусса для системы линейных уравнений в канонической форме, в которой должна быть предварительно записана исходная задача линейного программирования;
* на­правление перехода от одного опорного решения к другому выбирается при этом на основе критерия оптимальности (целевой функции) исходной задачи.

Определение базисных решений



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

Транспортные задачи – специальный класс задач линейного программирования. 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.

 – количество поставщиков продукции;

 – количество потребителей продукции;

 – индекс для поставщиков;

 – индекс для потребителей;

,  – наличие продукции у каждого поставщика;

,  – потребность в продукции каждого потребителя;

 – стоимость доставки продукции единицы продукции от  - поставщика к -потребителю.

**Этапы:**

1. Построение начального базисного решения : метод северо-западного угла, метод наименьшей стоимости (минимального элемента), метод Фогеля.
2. Итеративный процесс поиска оптимального решения (метод потенциалов).

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.

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



30 135 200 20 90

1. **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.

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

Метод ветвей и границ сводится к построению корневого дерева T

Метод ветвей и границ – это общий алгоритмический метод решения задач комбинаторной оптимизации. По существу, метод является вариацией полного перебора с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений. В основе метода лежат две процедуры: процедура ветвления, позволяющая разбивать множество допустимых решений на непересекающиеся подмножества, и процедура вычисления нижней или верхней границы.

Данный метод часто применяется для решения задачи о коммивояжере.

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

Пусть  – конечное множество мощности  **Множество всех подмножеств** множества  называется ***булеаном***  и обозначается  Количество элементов булеана множества  вычисляется по формуле

 (1)

Алгоритм генерации множества всех подмножеств основывается на взаимно однозначном соответствии между элементами булеана  множества  и всеми целыми числами множества , записанными в двоичном виде. Следует сказать, что алгоритм имеет сложность  и поэтому реально может применяться для множеств с небольшой мощностью. Например, генерация булеана множества мощностью  на компьютере с тактовой частотой процессора 3 ГГц займет более 100 лет.

3:

Построение элементов булеана множества  сводится к следующему алгоритму:

1. Пронумеровать элементы заданного множества  начиная с нуля.
2. Сформировать битовую последовательность  состоящую из  двоичных нулей. Пронумеровать элементы этой последовательности справа налево, начиная с нуля.
3. Последовательно выполнить шаги 4 и 5 алгоритма  раз.
4. Выбрать из множества  элементы с номерами  для которых  Полученное подмножество будет являться элементом булеана  В первом случае не будет выбран ни один элемент (пустое подмножество) множества  так как исходная последовательность  состоит только из нулей.
5. Интерпретируя битовую последовательность как целое положительное число, увеличить это число на единицу.

4:



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

Поэтому генерация множества  может быть сведена к генерации булеана и выбору из него всех подмножеств с мощностью 

**** (2)

** **



1. **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.

Pn=n!

Построение множества всех перестановок с помощью алгоритма Джонсона – Троттера сводится к следующей процедуре:

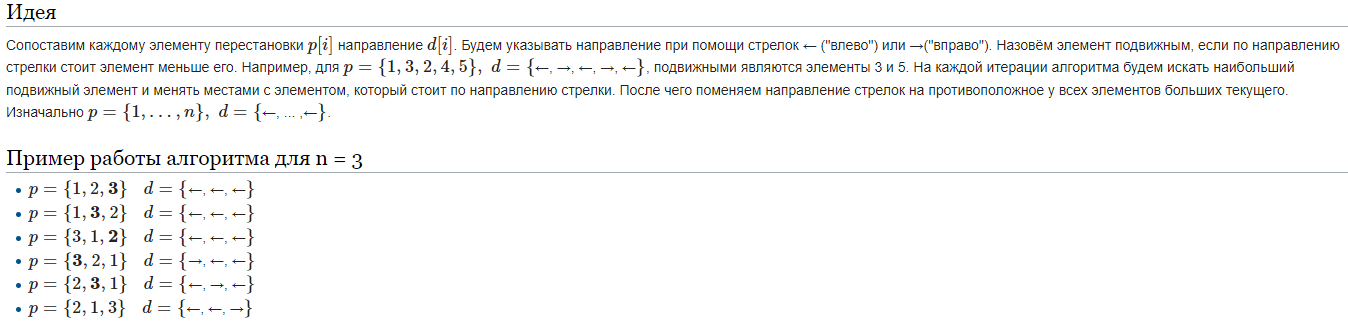
1. Построить первую перестановку. Первая перестановка – это последовательность всех элементов множества  перечисленных в порядке возрастания. Стрелки всех элементов последовательности направлены влево.
2. Найти наибольший мобильный элемент в текущей перестановке. Если в последовательности нет мобильного элемента, то построены все перестановки элементов множества  – алгоритм закончил свою работу.

3. Поменять местами наибольший мобильный элемент и элемент, на который указывает стрелка наибольшего мобильного элемента.

4. Найти все элементы, большие, чем мобильный элемент (если они есть) и изменить их стрелки на противоположное направление.

5. Перейти к пункту 2.

Схема алгоритма генерации множества всех перестановок множества  приведена на рис. 1.





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

Множество размещений  из  по  можно получить, перестановкой элементов всех сочетаний множества . Другими словами, для получения множества размещений  требуется сначала сгенерировать все сочетания по  элементов из множества  а затем все перестановки элементов для каждого сочетания.



Сначала записываем 012-abc, потом меняем 2-3, 1-2, … пока не получим повтор. Потом берем 013-abd, …

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

Метод решения задачи оптимизации, реализующей рекурсивный алгоритм с перекрывающимися подзадачами, в котором каждая такая подзадача решается один раз, а ее результат сохраняется для последующего применения, называется ***динамическим программированием***.

Динамическое программирование - это метод решения сложных задач, разбивая их на более простые подзадачи и сохраняя результаты этих подзадач для последующего использования.



1. **Основные приложения динамического программирования. Обзор задач, решаемых методами динамического программирования.**

Метод решения задачи оптимизации, реализующей рекурсивный алгоритм с перекрывающимися подзадачами, в котором каждая такая подзадача решается один раз, а ее результат сохраняется для последующего применения, называется ***динамическим программированием***.

Им решаются: задача о рюкзаке, задача о расстановке скобок при перемножении матриц, задача о наибольшей общей последовательности, задача о расстоянии Левенштейна

1. **Рекурсивные алгоритмы.**

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

Неразрывно с понятием рекурсивного алгоритма связано понятие рекурсивной функции. Существует два определения этого понятия.

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.

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

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

Граф – это математическая модель, с помощью которой удобно представлять бинарное отношение.

Рассмотрим множество , которое будем назвать ***множеством*** ***вершин***, и набор  упорядоченных пар , называемый ***множеством дуг***. Пара  называется ***графом***. Запись  обозначает граф с именем , состоящий из множества вершин  и множества дуг .

Две соединенные дугой вершины графа, называются ***смежными вершинами***. Например,  и  смежные вершины графа , т.к. есть дуга .

Считается, что дуга  ***выходит из вершины***  и ***входит в вершину*** . При этом говорят, что вершины  и  ***инцидентны*** дуге , а дуга  инцидентна вершинам  и . Например, вершины  и  инцидентны дуге , выходящей из  и входящей в .

Вершины  и  дуги  называют соответственно ***начальной*** и ***конечной*** вершинами этой дуги.

Дуги, которые выходят и входят в одну и ту же вершину, называются ***петлями***. Например, дуга  является петлей.

Вершины, не имеющие смежных, называются ***изолированными вершинами***. Например,  – изолированная вершина.

Очевидно, что множество дуг  можно интерпретировать как бинарное отношение, а  – множество, на котором это бинарное отношение строится.

Если множество дуг  не является симметричным отношением, то такой граф называется ***ориентированным графом***. Граф, изображенный на рис. 1.1, является ориентированным графом.

Если множество дуг – симметричное отношение, то соответствующий граф называется ***неориентированным*** ***графом***. Симметричность отношения предполагает, что для каждой дуги  графа найдется дуга , имеющая противоположное направление.

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



Поиск кратчайшего пути между двумя вершинами графа является одной из часто используемых в приложениях задач. Наиболее известными способами решения этой задачи являются ***алгоритмы Дейк-стры***, ***Беллмана*** *–* ***Форда*** и ***Флойда – Уоршолла***. Здесь будет рассмотрен только алгоритм Дейкстры

Алгоритм последовательно преобразовывает исходный граф  в дерево кратчайших путей . ***Дерево кратчайших путей*** – это граф , обладающий следующими свойствами:

1. ;
2.  – ориентированное дерево с корнем ;
3.  – множество достижимых вершин графа  из ;
4. для каждой вершины  путь из  в  в дереве  является кратчайшим путем в графе .
5. **Математические основы сетевого планирования. Максимальные пути между вершинами графа.**

***Максимальным путем  называется путь с максимальным весом ***. В общем случае, в одном графе может быть несколько максимальных путей.

Построение максимального пути ****** во взвешенном ориентированном графе  возможно, если в нем нет контуров с положительным весом. Если в графе есть такой контур, то некоторые пути могут иметь сколь угодно большой вес, т.к. каждый обход контура увеличивает вес пути на величину веса этого контура.

Будем предполагать, что граф  безконтурный и его вершины пронумерованы так, что у любой дуги  конечная вершина всегда имеет больший номер, чем начальная: . Выше рассматривалась процедура ранжирования, основанная на алгоритме Демукрона, позволяющая привести любой безконтурный граф такому виду. Пусть все вершины графа  имеют номера от  до .

Рассматриваемый алгоритм основывается на рекуррентном выражении , где  – множество начальных вершин дуг, входящих в вершину . При этом предполагается, что для всех вершин  ранга  значение .

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

Будем называть ***проектом*** деятельность, имеющую начало и конец во времени и направленную на достижение определенного результата. Результатом проекта может быть определенный продукт (построенное здание, разработанное программное обеспечение, собранное оборудование и т.д.) или услуга (транспортировка грузов, обучение персонала, медицинское обслуживание и т.п.). Часто успешное выполнение проекта называют ***реализацией проекта*** или просто говорят – ***процесс реализован***.

Как правило, проект представляется в виде ряда элементарных работ, которые называют ***операциями***. Некоторые операции могут быть выполнены только после завершения одной или нескольких других операций. В этом случае говорят о ***зависимости*** операций. Совокупность операций проекта и их зависимостей называется ***комплексом операций***.

Каждой операции комплекса соответствует два момента времени: начала и окончания операции. Эти моменты называются ***событиями***. Событие в отличие от операции не имеет протяженности по времени и является просто фиксацией факта начала или окончания операции. Если операции  предшествует операция  (или по другому – операция  ***опирается*** на ), а операция  имеет только одну последующую операцию, то событие, соответствующее окончанию операции *b*, будет являться одновременно событием, соответствующим началу операции .

Как правило, для обозначения событий используют целые положительные числа. При таком обозначении каждой операции соответствует пара , где  – номер ***начального***, а  ***– конечного события*** операции. При этом говорят, что ***операция инцидентна событиям***  и , и наоборот, ***события***  и  ***инциденты операции***. Часто для обозначения операции используют пару  номеров инцидентных событий.

Различают три вида событий комплекса операций: исходное, промежуточное и завершающее. ***Исходным событием*** называется событие, которое не является конечным ни для одной операции комплекса. ***Завершающим событием*** называется событие, которое не является начальным ни для одной операции комплекса. Все остальные события комплекса операций называются ***промежуточными***.

В основе метода сетевого планирования и управления лежит понятие сетевого графика. ***Сетевой график*** представляет собой взвешенный ориентированный корневой графбез контуров (ациклический) и изолированных вершин, который построен по определенным правилам.

Рассмотрим сначала отдельно операцию Z4, имеющую начальное событие 6 и конечное 7. Она непосредственно предшествует операциям Z6, Z7 и Z8. Первоначальное построение приводит к графу, изображенному на рис. 1. Вершины графа (события) здесь изображены овалами, а дуги (операции) сплошными линиями со стрелками. Дуги соединяют вершины, соответствующие начальному и конечному событиям соответствующей операции.



Рис. . Построение фрагмента сетевого графика для операций Z4, Z6, Z7 и Z8

Интуитивно ясно, что событие 7 должно предшествовать событиям 10, 12 и 14. Но в перечне (табл. 3.2) нет операций, связывающих эти события. В таких случаях необходимо расширить комплекс операций, добавив ***фиктивные*** операции, позволяющие отразить недостающие логические связи между событиями. На рис. 2 изображен предыдущий граф (рис. 1), в который добавлены три дуги, соответствующие фиктивным операциям F1, F2 и F3. Как правило, фиктивные операции изображаются на рисунках штриховой линией. Дальше всегда будем придерживаться такого обозначения.



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

Связный подграф  графа  являющийся деревом и содержащий все его вершины, называется покрывающим деревом, или остовным деревом.

В общем случае один граф может иметь несколько покрывающих деревьев.

Покрывающее дерево  графа , имеющее минимальную сумму длин его ребер, называется минимальным покрывающим деревом, или минимальным остовным деревом.

В общем случае граф может иметь несколько минимальных покрывающих деревьев.

Наиболее известными алгоритмами построения минимального остовного дерева являются алгоритмы Крускала (Джозеф Крускал (1928–2010) − американский математик) и Прима (Роберт Прим (род. 1921) − американский математик).

Построение минимального остовного дерева в этих алгоритмах осуществляется поэтапно. Начинается построение с пустого множества ребер  На каждом этапе множество  пополняется одним ребром, причем так, что множество ребер  всегда остается подмножеством ребер искомого минимального остовного дерева. Задача сводится к поиску необходимого ребра (обычно его называют безопасным ребром) на каждом шаге алгоритма.

Алгоритмы Крускала и Прима относятся к классу алгоритмов, называемых жадными алгоритмами. Такое название эти алгоритмы получили за стратегию, заключающуюся в принятии на каждом шаге локально оптимального решения (жадного решения), в предположении, что такая стратегия приведет к конечному оптимальному решению.

**Алгоритм Крускала** работает по тому же принципу. Отличие только в порядке объединения вершин в пошагово формирующемся минимальном покрывающем дереве.

Первоначально в алгоритме Крускала неориентированный связный граф  с заданной на его ребрах весовой функцией   разбивается на максимальное количество подграфов, каждый из которых является деревом. Очевидно, что каждое такое дерево будет представлять собой одну вершину графа, а общее количество таких подграфов будет 

Далее из множества  ребер графа  поочередно в порядке возрастания длины выбираются ребра. При этом возможны два случая:

1) концевые вершины лежат в разных подграфах разбиения;

2) обе концевые вершины лежат в одном подграфе разбиения.

В первом случае из двух подграфов, которые можно соединить выбранным ребром, образуется один общий, включающий все вершины и ребра этих подграфов, а также новое связующее ребро. Очевидно, что такое объединение не может образовать циклы, а следовательно, объединенный подграф тоже является деревом.

Во втором случае не выполняется никаких новых построений.

Из условия связности исходного графа  очевидно, что итогом работы такого алгоритма будет дерево, связывающее все вершины графа (т. е. остовное дерево).

Минимальность остовного дерева следует из того, что на каждом его шаге выбиралось безопасное ребро для совокупности ребер двух подграфов.

Построение минимального остовного дерева с помощью алгоритма **Прима** начинается с выбора произвольной вершины исходного неориентированного связного графа. Выбранная вершина окрашивается.

На последующих шагах просматриваются все окрашенные вершины и анализируются все ребра исходного графа, у которых одна концевая вершина окрашена, а другая нет. Среди всех таких ребер выбирается ребро с наименьшей длиной. Неокрашенная вершина этого ребра окрашивается, а само оно добавляется в формируемое минимальное остовное дерево.

Алгоритм заканчивает свою работу, когда все вершины графа станут окрашенными. Сформированное множество выбранных ребер будет составлять искомое минимальное остовное дерево.

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

Алгоритм подразумевает, что задана исходная (стартовой) вершина, и основывается на простом правиле: при выборе очередной вершины предпочтение отдается ближайшей. При этом считается, что все дуги графа имеют единичную длину. Сначала посещается стартовая вершина, затем все вершины, смежные ей (т. е. находящиеся на расстоянии 1), после чего вершины, находящиеся на расстоянии 2 от стартовой и т.д.

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

Как и для поиска в ширину, задается стартовая вершина. Алгоритм описывается следующим образом: для каждой не пройденной вершины, начиная со стартовой, необходимо найти все смежные вершины и повторить поиск для каждой.

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

Часто с помощью графов описывается некоторая последовательность связанных действий. Например, если поставить в соответствие каждой странице сайта вершину графа, а дуге – ссылку (возможность перехода) с одной страницу на другую, то такой граф будет отображать модель (карту) сайта.

Другой пример: применение графов для описания проектов. Как правило, в этом случае вершинам графа соответствуют события проекта (завершение этапа), дугам – операции. Такой граф описывает комплекс связанных операций проекта.

Если графы включают много связанных вершин, то модели, основанные на них, становятся трудно воспринимаемыми. В некоторых случаях вершины графа можно упорядочить таким образом, что вид графа становится более структурированным.

**Топологическая сортировка** − это процедура упорядочивания вершин ориентированного графа, не имеющего циклов (ациклического графа). В результате топологической сортировки для вершин графа определяется такой порядок, что если их расположить на рисунке в соответствии с этим порядком сверху вниз, то дуги будут направлены только от верхних вершин к нижним. Обычно после выполнения топологической сортировки вершины переименовываются (перенумеровываются) в соответствии с полученным порядком. После такого переименования граф обладает свойством: начальная вершина каждой дуги имеет номер (имя) меньший, чем номер конечной вершина этой дуги.

Наиболее известны два способа топологической сортировки графа: алгоритмы Демукрона и алгоритм применяющий поиск в глубину.

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

***Сеть*** – это ориентированный граф , каждому ребру  которого поставлено в соответствие число , называемое пропускной способностью, а также выделено две вершины - исток и  - сток, .

***Поток*** – это функция ,  обладающая тремя свойствами:

1. ;

2.  (кососимметричность);

3. , 

***Величина потока *** это 

***Разрез  сети***  называется разбиение множества  на две части  и  такое, что , , , .

***Пропускная способность разреза*** ****** – это сумма пропускных способностей дуг соединяющих вершины в  и .

***Минимальный разрез сети*** – это разрез с минимальной пропускной способностью.

Теорема Форда-Фалкерсона: В любой сети максимальная величина потока из истока  в сток  равна минимальной пропускной способности разреза отделяющего  от .

Потоки в сетях - это концепция, используемая для измерения и оптимизации передачи информации или ресурсов через сеть. Она может быть применена к различным ситуациям, таким как передача данных через компьютерные сети или передача жидкости через трубопроводы.

Задача о максимальном потоке состоит в нахождении наибольшего количества информации или ресурсов, которые можно передать через сеть от источника к стоку. Источник - это место, откуда начинается передача, а сток - место, куда должны попасть передаваемые элементы.

Теорема Форда-Фалкерсона является одной из основных теорем, связанных с задачей о максимальном потоке. Она утверждает, что максимальный поток в сети равен минимальной пропускной способности любого разреза в сети.

Разрез в сети - это разделение вершин сети на два множества: множество вершин, которые находятся до разреза (обычно связаны с источником), и множество вершин, которые находятся после разреза (обычно связаны со стоком). Пропускная способность разреза определяется суммой пропускных способностей всех ребер, которые пересекают разрез.

Теорема Форда-Фалкерсона гарантирует, что максимальный поток в сети можно найти путем нахождения минимального разреза в сети. Это означает, что при поиске максимального потока можно искать разрезы и определять их пропускные способности, чтобы найти максимальное количество информации или ресурсов, которые можно передать через сеть.

1. **Потоки в сетях. Задача о максимальном потоке. Алгоритм Форда-Фалкерсона.**

АЛГОРИТМ ФОРДА-ФАЛКЕРСОНА

1. Обнуляем все потоки. Остаточная сеть изначально совпадает с исходной сетью.
2. В остаточной сети находим любой путь из источника в сток. Если такого пути нет, останавливаемся.
3. Пускаем через найденный путь (он называется увеличивающим путём или увеличивающей цепью) максимально возможный поток:

* На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью cmin.
* Для каждого ребра на найденном пути увеличиваем поток на cmin, а в противоположном ему - уменьшаем на cmin.
* Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.

1. Возвращаемся на шаг 2.
2. **Задачи нелинейного программирования. Основные алгоритмы решения.**

Нелинейное программирование – раздел математического программирования, изучающий методы решения экстремальных задач с нелинейной целевой функцией и (или) областью допустимых решений, определенной нелинейными ограничениями. К нелинейному программированию относят квадратичное, дробное, выпуклое, дискретное, целочисленное и геометрическое программирование.

В общем виде задачу нелинейного программирования можно сформулировать так:

*F*(*х*)→min (max)

Среди большого числа вычислительных алгоритмов нелинейного программирования значительное место занимают:

* различные варианты градиентных методов (метод проекции градиента, метод условного градиента и т. п.);
* методы штрафных функций;
* методы барьерных функций;
* метод модифицированных функций Лагранжа и др.

1. **Постановка задачи векторной оптимизации.**

Эффективность функционирования экономической системы оценивается, как правило, несколькими критериями. Математической формой критерия эффективности в оптимизационных экономико-математических задачах является целевая функция.

Пусть имеется  критериев, которые можно записать в виде целевых функций , где . Поскольку , то для простоты в дальнейшем будем предполагать, что все целевые функции максимизируются. Задача многокритериальной оптимизации в этом случае запишется

 (1)

; (2)

. (3)

Если точки максимума , определенные при решении задач по каждому критерию  не совпадают, то решение задачи (1)-(3) может быть только компромиссным. В области допустимых значений задачи находится область компромиссов. При перемещении из одной точки области компромиссов в другую, невозможно одновременное улучшение всех критериев. Принадлежащие области компромиссов планы называются эффективными или оптимальными по Парето (по имени итальянского экономиста, впервые сформулировавшего проблему многокритериальной оптимизации и принцип оптимальности).

План  оптимален по Парето, если он допустим и не существует другого плана  для которого



и хотя бы для одного критерия выполняется строгое неравенство.

К задачам векторной оптимизации приходят в следующих случаях:

1. Качество моделируемого процесса нужно оценить с точки зрения нескольких показателей. Это могут быть прибыль, себестоимость, рентабельность и т.д.
2. Моделируемые процесс представляет собой составляющую нескольких процессов (частей), и каждая из этих частей имеет свой критерий качества.
3. Моделируемые процесс расчленяется на несколько шагов и на каждом шаге его качество определяется своей функцией. (Например, на отдельных временных промежутках)

Постановка задачи векторной оптимизации сводится к поиску наилучших значений для нескольких переменных, при условии соблюдения некоторых ограничений и достижении оптимальных значений для нескольких целевых функций. Вместо того, чтобы найти единственное оптимальное решение, векторная оптимизация стремится найти набор решений, которые обеспечивают компромисс между целевыми функциями.

Конкретная постановка задачи векторной оптимизации включает определение целевых функций, которые нужно минимизировать или максимизировать, переменных, которые нужно оптимизировать, а также набора ограничений, которым должны удовлетворять переменные.

Цель состоит в том, чтобы найти решения, которые находятся на Парето-фронте, то есть наилучшие компромиссные значения переменных, где нельзя улучшить одну целевую функцию, не ухудшив другие. По сути, векторная оптимизация стремится найти набор решений, которые представляют оптимальное соотношение между всеми целевыми функциями.

Формально, задача векторной оптимизации может быть сформулирована следующим образом:

Минимизировать (или максимизировать) вектор целевых функций:

f(x) = [f₁(x), f₂(x), ..., fₙ(x)],

при условии, что вектор переменных x принадлежит допустимому множеству X и удовлетворяет ограничениям:

g(x) ≤ 0, h(x) = 0,

где:

* f(x) = [f₁(x), f₂(x), ..., fₙ(x)] - вектор целевых функций, которые требуется минимизировать или максимизировать.
* x = [x₁, x₂, ..., xₘ] - вектор переменных, которые нужно оптимизировать.
* g(x) ≤ 0 - набор неравенств-ограничений, которые должны выполняться.
* h(x) = 0 - набор равенств-ограничений, которые должны выполняться.

Оптимальное решение задачи векторной оптимизации называется Парето-оптимальным или эффективным, если не существует другого решения, которое одновременно улучшает значения всех целевых функций.

1. **Методы решения задач векторной оптимизации.**

Методы решения задач многокритериальной оптимизации можно подразделить на четыре группы:

* + - * методы, основанные на свертывании критериев;
      * методы, использующие ограничения на критерии;
      * методы целевого программирования;
      * методы, основанные на отыскании компромиссного решения.

Вместо исходной многокритериальной задачи в соответствии с выбранным методом, формируется замещающая задача. В состав замещающей задачи входит один критерий, а к исходной системе ограничений добавляется одно или несколько дополнительных ограничений. Решение замещающей задачи называется субоптимальным.

**Задания для выполнения на компьютере с использованием языка программирования С++**

1. Реализуйте пример замера продолжительности процесса вычисления на основе цикла. Проведите необходимые эксперименты и постройте график зависимости продолжительности процесса вычисления от количества циклов (в Excel).
2. Разработать программу, реализующую работу генератора подмножеств заданного множества.
3. Разработать программу, реализующую работу генератора сочетаний.
4. Разработать программу, реализующую работу генератора перестановок.
5. Разработать программу, реализующую работу генератора размещений.
6. Разработать программу, реализующую решение задачи о коммивояжере (расстояния для 10 городов сгенерировать случайным образом в пределах 10 – 300 км, 3 расстояния между городами задать бесконечными).
7. Разработать программу, реализующую решение упрощенной задачи о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, стоимость предметов 5 – 55 у.е.; количество предметов – 18 шт.).
8. Разработать программу, реализующую решение об оптимальной загрузке судна (веса контейнеров сгенерировать случайным образом: ограничение по общему весу – 1500 кг., количество мест на судне для контейнеров – 5, количество контейнеров 25, веса контейнеров 100 – 900 кг., доход от перевозки 10 – 150 у.е.).
9. Разработать программу, реализующую решение об оптимальной загрузке судна с условием центровки (веса контейнеров сгенерировать случайным образом: количество мест на судне для контейнеров – 5, количество контейнеров 8, веса контейнеров 100 – 200 кг., доход от перевозки 10 – 100 у.е.; минимальный вес контейнера для каждого места 50 – 120 кг, максимальный вес контейнера для каждого места 150 – 850 кг).
10. Разработать программу исследования зависимости времени вычисления необходимого для решения задачи коммивояжера (5–12 городов) и построить соответствующий график.
11. Разработать программу исследования зависимости времени вычисления необходимого для упрощенной задачи о рюкзаке (количество предметов 12 – 20 шт.) и построить соответствующий график.
12. Разработать программу исследования зависимости времени вычисления необходимого для решения задачи об оптимальной загрузке судна (количество мест на судне – 6 шт., количество контейнеров 25 – 35 шт.) и построить соответствующий график.
13. Разработать программу исследования зависимости времени вычисления необходимого для решения задачи об оптимальной загрузке судна об оптимальной загрузке судна с условием центровки (количество мест на судне для контейнеров 4 – 10 шт.) и построить соответствующий график.
14. Разработать программу вычисления дистанции Левенштейна для слов «антимонопольный» и «антикоррупционный» рекурсивным методом. Найти время выполнения для данного алгоритма.
15. Разработать программу вычисления дистанции Левенштейна для слов «антимонопольный» и «антикоррупционный» методом динамического программирования. Найти время выполнения для данного алгоритма.
16. Разработать программу вычисления наибольшей общей подпоследовательности для Х= «ALDC» и Y= «LADCMI» рекурсивным методом. Найти время выполнения для данного алгоритма.
17. Разработать программу вычисления наибольшей общей подпоследовательности для Х= «ALDC» и Y= «LADCMI» методом динамического программирования. Найти время выполнения для данного алгоритма.
18. Разработать программу для решения задачи об оптимальной расстановке скобок при умножении нескольких матриц рекурсивным методом (сгенерировать последовательность  случайным образом). Найти время выполнения для данного алгоритма.
19. Разработать программу для решения задачи об оптимальной расстановке скобок при умножении нескольких матриц методом динамического программирования (сгенерировать последовательность  случайным образом). Найти время выполнения для данного алгоритма.
20. Разработать программу генерации ориентированного графа из 7-ми вершин случайным образом, а также его представления матричным и списковым способом.
21. Разработать программу генерации ориентированного графа из 7-ми вершин случайным образом, а также создать функцию **BFS** обхода вершин графа, используя метод поиска в ширину.
22. Разработать программу генерации ориентированного графа из 7-ми вершин случайным образом, а также создать функцию **DFS** обхода вершин графа, используя метод поиска глубину.
23. Разработать программу генерации ориентированного графа из 7-ми вершин случайным образом, а также создать функцию **TFS** для топологической сортировки графа.

**Практическая часть**

1. Получить все подмножества множества, состоящего из 4-х элементов А,В,С,D. По какой формуле вычисляется количество подмножеств?
2. Получить все сочетания по 3 множества, состоящего из 4-х элементов А,В,С,D. По какой формуле вычисляется количество сочетаний?
3. Получить множество всех перестановок множества, состоящего из 3-х элементов А,В,С. По какой формуле вычисляется количество перестановок?
4. Получить множество размещений, по 2 элемента из множества А,В,С,D. По какой формуле вычисляется количество размещений?
5. Решить задачу о коммивояжере методом ветвей и границ с количеством городов равным 4. Расстояния между городами в км. заданы в матрице *С*:

*С*=.

1. Решить задачу о коммивояжере методом ветвей и границ с количеством городов равным 4. Расстояния между городами в км. заданы в матрице С:

*С*=.

1. Решить задачу о коммивояжере методом ветвей и границ с количеством городов равным 4. Расстояния между городами в км. заданы в матрице *С*:

*С*=.

1. Вычислить дистанцию Левенштейна на основе рекурсивного алгоритма между словами «сом» и «домен».
2. Вычислить дистанцию Левенштейна на основе рекурсивного алгоритма между словами «эркер» и «акр».
3. Вычислить дистанцию Левенштейна на основе рекурсивного алгоритма между словами «дом» и «сонет».
4. Определить длину наибольшей общей подпоследовательности последовательностей «*ACBDAB*» и «*BDCABA*» на основе алгоритма динамического программирования.
5. Определить длину наибольшей общей подпоследовательности последовательностей «*ACEFAB*» и «*DCABEF*» на основе алгоритма динамического программирования.
6. Ориентированный граф представлен в виде матрицы смежности *М*:

*М*=.

Отобразить данный граф в обычном виде и в виде списка смежных вершин.

1. Ориентированный граф представлен в виде матрицы смежности *М*:

*М*=.

Реализовать алгоритм поиска в ширину для данного графа.

1. Ориентированный граф представлен в виде матрицы смежности *М*:

*М*=.

Реализовать алгоритм поиска в глубину для данного графа.

1. Ориентированный граф представлен в виде матрицы смежности *М*:

*М*=.

Реализовать алгоритм топологической сортировки для данного графа.

1. Построить кратчайший путь во взвешенном ориентированном графе G, представленном в виде матрицы:

*G*=.

1. Построить максимальный путь во взвешенном ориентированном графе G, представленном в виде матрицы:

*G*=.

1. Имеется сеть *G*. Дана пропускная способность дуг, представленная в виде матрицы *М* и поток *f*. Определить максимальный поток в сети G на основе алгоритма Форда-Фалкерсона.

*М*=, *f*=.

1. Имеется граф связанный неориентированный граф G, с заданной весовой функцией на множестве ребер. Построить минимальное покрывающее дерево на основе алгоритма Крускала.



1. Имеется граф связанный неориентированный граф G, с заданной весовой функцией на множестве ребер. Построить минимальное покрывающее дерево на основе алгоритма Прима.



1. Решить транспортную задачу с использованием метода потенциалов: имеется 5 поставщиков продукции и 4 потребителя. Потребности (в тоннах) составляют соответственно 28, 13, 52, 11 и 7. Запасы (в тоннах) – 25, 41, 32, 13. Затраты на перевозку продукции (в у.е.) заданы матрицей *С*:

С=.

1. Решить открытую транспортную задачу с использованием метода потенциалов: имеется 5 поставщиков продукции и 4 потребителя. Потребности (в тоннах) составляют соответственно 29, 13, 26, 10 и 7. Запасы (в тоннах) – 24, 39, 32, 23. Затраты на перевозку продукции (в у.е.) заданы матрицей *С*:

С=.

1. Решить задачу линейного программирования геометрическим методом. Фирма производит типографскую краску двух цветов: черную и синюю из сырья двух типов М1 и М2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Расход сырья (в тоннах) на тонну краски | | Максимально возможный ежедневный расход сырья |
| Черная | Синяя |
| Сырье М1 | 6 | 4 | 24 |
| Сырье М2 | 1 | 2 | 6 |
| Доход (в 1000$) на тонну краски | 5 | 4 |  |

Отдел маркетинга фирмы ограничил ежедневное производство синей краски до 2 т. (из-за отсутствия спроса), а также поставил условие, чтобы ежедневное производство синей краски не превышало более чем на тонну аналогичный показатель производства черной краски. Фирме требуется определить оптимальное (наилучшее) соотношение между видами выпускаемой продукции для максимизации общего ежедневного дохода.

1. Решить задачу линейного программирования геометрическим методом. Фирма производит типографскую краску двух цветов: черную и синюю из сырья двух типов М1 и М2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Расход сырья (в тоннах) на тонну краски | | Максимально возможный ежедневный расход сырья |
| Черная | Синяя |
| Сырье М1 | 2 | 4 | 30 |
| Сырье М2 | 3 | 1 | 18 |
| Доход (в 1000$) на тонну краски | 3 | 6 |  |

Отдел маркетинга фирмы ограничил ежедневное производство синей краски до 4 т. (из-за отсутствия спроса), а также поставил условие, чтобы ежедневное производство синей краски не превышало более чем на 2 т. аналогичный показатель производства черной краски. Фирме требуется определить оптимальное (наилучшее) соотношение между видами выпускаемой продукции для максимизации общего ежедневного дохода.

1. Найти точку условного экстремума функции  при ограничениях:



1. Решить задачу линейного программирования симплекс методом. Фирма производит типографскую краску двух цветов: черную и синюю из сырья двух типов М1 и М2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Расход сырья (в тоннах) на тонну краски | | Максимально возможный ежедневный расход сырья |
| Черная | Синяя |
| Сырье М1 | 6 | 4 | 24 |
| Сырье М2 | 1 | 2 | 6 |
| Доход (в 1000$) на тонну краски | 5 | 4 |  |

Отдел маркетинга фирмы ограничил ежедневное производство синей краски до 2 т. (из-за отсутствия спроса), а также поставил условие, чтобы ежедневное производство синей краски не превышало более чем на тонну аналогичный показатель производства черной краски. Фирме требуется определить оптимальное (наилучшее) соотношение между видами выпускаемой продукции для максимизации общего ежедневного дохода.

1. Решить задачу линейного программирования симплекс методом. Фирма производит типографскую краску двух цветов: черную и синюю из сырья двух типов М1 и М2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Расход сырья (в тоннах) на тонну краски | | Максимально возможный ежедневный расход сырья |
| Черная | Синяя |
| Сырье М1 | 2 | 4 | 30 |
| Сырье М2 | 3 | 1 | 18 |
| Доход (в 1000$) на тонну краски | 3 | 6 |  |

Отдел маркетинга фирмы ограничил ежедневное производство синей краски до 4 т. (из-за отсутствия спроса), а также поставил условие, чтобы ежедневное производство синей краски не превышало более чем на 2 т. аналогичный показатель производства черной краски. Фирме требуется определить оптимальное (наилучшее) соотношение между видами выпускаемой продукции для максимизации общего ежедневного дохода.