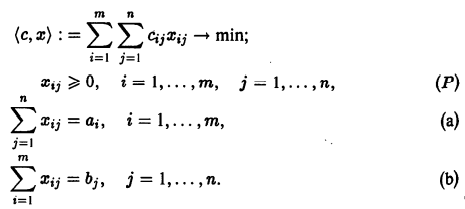
Лабораторная работа №1

**Транспортная задача. Метод минимальной стоимости**

Пусть в пунктах отправления А1 … Аm сосредоточено соответственно а1 … аm единиц некоторого однородного груза. Этот груз следует привезти в n пунктов назначения B1 … Bm, причем в каждый из них надлежит завезти соответственно b1 … bm единиц груза. Стоимость перевозки единицы груза из пункта Ai в пункт Bi равна cij.

Обозначая через xij количество единиц груза, предназначенного к отправке из пункта Ai в пункт Bi, получим задачу нахождения плана перевозок, при котором общая стоимость окажется минимальной:



Метод минимальной стоимости – основная идея:

Суть метода заключается в том, что из всей таблицы стоимостей выбирают наименьшую и в клетку, которая ей соответствует, помещают меньшее из чисел ai или bi. Затем из рассмотрения исключают либо строку, соответствующую поставщику, запасы которого полностью израсходованы, либо столбец, соответствующий потребителю, потребности которого полностью удовлетворены, либо и строку, и столбец, если израсходованы запасы поставщика и удовлетворены потребности потребителя. Из оставшейся части таблицы стоимостей снова выбирают наименьшую стоимость, и процесс распределения запасов продолжают, пока все запасы не будут распределены, а потребности удовлетворены.

* public static Matrix Transport(Vector a, Vector b, Matrix c)

a - количество товара у поставщиков;

b - количество товара у потребителя;

с - стоимость перевозки ij;

Дополнительный методы для решения транспортной задачи:

* private static bool Equaleble(Vector a, Vector b, Matrix c)
* private static bool AllEmpty(Vector a)
* private static List<double[]> FindMinInMatrix(Matrix m)
* private static List<double[]> FindNotNullMat(Matrix m)

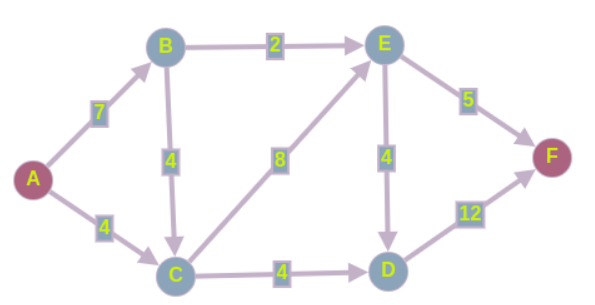
Лабораторная работа №2

**Алгоритм поиска max потока (Алгоритм Форда-Фалкерсона)**

Оптимальная задача на графы

**Постановка задачи:**

Имеется следующий ориентированный граф, в котором вес ребра обозначает пропускную способность между вершинами. *Нужно найти максимальный поток, который можно пропустить из истока A в сток F:*



Ориентированный граф

Определение 1:

          Сетью называется связный орграф без петель.

Определение 2:

          Потоком в сети называется некоторая функция, которая ставит в соответствие дуге некоторое число-вес дуги.

*Для определения потока в сети используют алгоритм Форда-Фалкерсона:*

а) ищем любую цепь из истока графа в сток;

б) каждой дуге приписываем возможный больший поток из истока в сток (записываем его через дробь с весом дуги; при этом поток не может превысить вес дуги, но может быть ему равен);

в) если поток становится равен весу дуги, то эта дуга является насыщенной, то есть через нее нельзя пройти при рассмотрении цепей в графе;

г) так перебираем все возможные цепи, пока станет невозможно попасть из истока в сток;

д) поток в сети будет равен сумме потоков всех дуг, инцидентных стоку графа (следует заметить, что сумма потоков всех дуг, инцидентных стоку графа равна сумме потоков всех дуг, инцидентных истоку графа).

Для реализации данной задачи были созданы 2 класса:

Public class MaxFlow – класс с реализаций алгоритма

     Public class FindPath – дополнительный класс

Основной метод:

public int getMaxFlow(int[,] graph, int endPoint)– Алгоритм Форда-Фалкерсона

# Лабораторная работа №3

## ***Оптимальные задачи на графах.***

## ***Задача Коммивояжера (Метод ближайшего соседа)***

**Постановка задачи:**

Пункты обхода плана последовательно включаются в маршрут, причем каждый очередной включаемый пункт должен быть ближайшим к последнему выбранному пункту среди всех остальных, ещё не включенных в состав маршрута.

***Алгоритм решения включает следующие шаги.***

*1 шаг*. Выбрать первую вершину.

*2 шаг.* Для выбранной вершины найти наиболее близкую к ней. Если данная вершина уже вошла в маршрут, то она блокируется, и выбирается следующая по степени близости.

*3 шаг.* Повторять шаг 2 до тех пор, пока маршрут (путь) не пройдет через все вершины.

*4 шаг.* Выбрать вторую вершину и повторить шаги 2–3.

*5 шаг.* Процесс считается законченным, когда гамильтоновы контуры построены для всех вершин, выбранных в качестве начальных.

*6 шаг.* Рассчитать длину каждого из полученных маршрутов и выбрать минимальное значение.

Метод:

* public static void Bliz2(Graph gr,Vertex start)– алгоритм ближайшего соседа

# Лабораторная работа №4

## ***Решение задач методом динамического программирования***(Задача о рюкзаке)

Задача о рюкзаке (англ. Knapsack problem) — дано N предметов, ni предмет имеет массу wi>0 и стоимость pi>0. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины W (вместимость рюкзака), а суммарная стоимость была максимальна.

Пусть A(k,s) есть максимальная стоимость предметов, которые можно уложить в рюкзак вместимости s, если можно использовать только первые k предметов, то есть {n1,n2,…,nk}, назовем этот набор допустимых предметов для A(k,s).

A(k,0)=0

A(0,s)=0

Найдем A(k,s). Возможны 2 варианта:

Если предмет k не попал в рюкзак. Тогда A(k,s) равно максимальной стоимости рюкзака с такой же вместимостью и набором допустимых предметов {n1,n2,…,nk−1}, то есть A(k,s)=A(k−1,s)

Если k попал в рюкзак. Тогда A(k,s) равно максимальной стоимости рюкзака, где вес s уменьшаем на вес k-ого предмета и набор допустимых предметов {n1,n2,…,nk−1} плюс стоимость k, то есть A(k−1,s−wk)+pk

A(k,s)={A(k−1,s),A(k−1,s−wk)+pk,bk=0bk=1

То есть: A(k,s)=max(A(k−1,s),A(k−1,s−wk)+pk)

Стоимость искомого набора равна A(N,W), так как нужно найти максимальную стоимость рюкзака, где все предметы допустимы и вместимость рюкзака W.

Восстановим набор предметов, входящих в рюкзак

Будем определять, входит ли ni предмет в искомый набор. Начинаем с элемента A(i,w), где i=N, w=W. Для этого сравниваем A(i,w) со следующими значениями:

Максимальная стоимость рюкзака с такой же вместимостью и набором допустимых предметов {n1,n2,…,ni−1}, то есть A(i−1,w)

Максимальная стоимость рюкзака с вместимостью на wi меньше и набором допустимых предметов {n1,n2,…,ni−1} плюс стоимость pi, то есть A(i−1,w−wi)+pi

Заметим, что при построении A мы выбирали максимум из этих значений и записывали в A(i,w). Тогда будем сравнивать A(i,w) c A(i−1,w), если равны, тогда ni не входит в искомый набор, иначе входит.

Метод динамического программирование всё равно не позволяет решать задачу за полиномиальное время, потому что его сложность зависит от максимального веса. Задача о ранце (или задача о рюкзаке) — одна из NP-полных задач комбинаторной оптимизации.

Методы:

public static int Bag(List<int[]> pi, int vob)

Дополнительные методы:

private static int FindMaxiesNext(Matrix m,List<int> list)- Нахождение следующего максимума

private static bool HaveList(int i,List<int> lsit )- есть ли предмет ?

# Лабораторная работа №5

## ***Теория игр***

Ситуации, в которых сталкиваются интересы двух сторон и результат любой операции, осуществляемой одной из сторон, зависит от действий другой стороны, называются конфликтными.

Математическая модель конфликтной ситуации называется игрой, а математическая теория, помогающая принимать рациональные решения в конфликтной ситуации, − ***теорией игр***.

Конфликтующие стороны называются игроками, а действия, которые могут выполнять игроки, − стратегиями.

От реальной ситуации игра отличается тем, что в игре противники действуют по строго определенным правилам.

* ***Матричной игрой*** называется игра, осуществляемая по следующим правилам:

**1.** В игре участвуют два игрока;

**2.** Каждый из игроков обладает конечным набором стратегий;

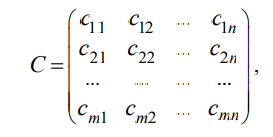
**3.** Игра заключается в том, что каждый из игроков, не имея информации о действиях противника, делает один ход (выбирает одну из своих стратегий). Результатом выбора игроками стратегий является выигрыш и проигрыш в игре.

**4.** И выигрыш, и проигрыш выражаются числами.

* Матричная игра называется игрой с ***нулевой суммой***, если в этой игре выигрыш одного игрока равняется проигрышу другого игрока.

Каждая матричная игра с нулевой суммой имеет платежную матрицу. Для того чтобы построить эту матрицу, обозначим одного из игроков символом A, а другого − символом B, и предположим, что A1, A2, Am − стратегии, которые может применять игрок A, а B1, B2, Bn − стратегии, которые может применять игрок B.

* Матричная игра, в которой у игрока A имеется m стратегий, а у игрока B − n стратегий, называется игрой типа m х n. Рассмотрим матрицу:



у которой элементы сij (i = 1, 2, ... m, ; j =1,2,... n, ) равны выигрышам игрока A (и проигрышам игрока B) при применении игроками стратегий Ai и Bj соответственно.

* Матрица C называется платежной матрицей игры.

**Типы стратегий:**

* **Чистая стратегия** дает полную определенность, каким образом игрок продолжит игру. В частности, она определяет результат для каждого возможного выбора, который игроку может придется сделать.
* **Смешанная стратегия** является указанием вероятности каждой чистой стратегии. Выбор осуществляется перед началом каждой игры и не меняется до ее конца.

***Смешанная стратегия*** игрока есть вероятностное распределение на множестве его чистых стратегий. В случае, когда игрок имеет только конечное число m чистых стратегий, смешанная стратегия представляет собой m-мерный вектор х = (х1, x2, …хm), удовлетворяющий условиям:



Если игроки участвуют в игре с матрицей А и 1-й игрок выбирает смешанную стратегию **x**, а 2-й – **y**, то ожидаемый выигрыш будет равен

https://lh5.googleusercontent.com/W-jP6VvoT7LyZQ9mk_t3OwPaSne4FD3WwiKVQ1f4NZdiVXse_jG8krk0r7NJCWo4R1h8hXyXrSDR_9WI5IweBbfkqBr13MfB8WF0yFDXxLHHdmxxANYwRPhbdxdLOddcyk7HPspGiQEWQ2s7rw

***Справедлива основная теорема матричных игр***: Любая матричная игра имеет решение в смешанных стратегиях

***Теорема о минимаксе***. Принцип минимакса: поступай так, чтобы при наихудшем для тебя поведении противника получить максимальный выигрыш, т.е. надо выбрать ту стратегию, при которой минимальный выигрыш максимален

Методы:

* *public static void TeoriaIgrFirst (Matrix A) – чистая стратегия*
* *public static double GameTheory (double[,] matrix) – смешанная стратегия*

Дополнительные методы для реализации, выше описанных методов:

* *private static double[] FindMinIG(Vector v)*
* *private static double[] FindMaxIG(Vector v)*

**Список литературы**

1. А.В. Аттетков, В.С. Зарубин, А.Н. Канатников «Введение в методы оптимизации».
2. Э. Ф. Галеев «Оптимизация: теория, примеры, задачи: учебное пособие».
3. К. Л. Саламов «Учебно-методическое пособие по разделу: Транспортная задача».
4. А. Станкевич «Задача о максимальном потоке»
5. В. И. Мудров «Задача о коммивояжере».
6. И.Л. Акулич «Математическое программирование в примерах и задачах».
7. Т. Кормен, Ч. Лейзерсон, Р. Ривест «Алгоритмы: построение и анализ».
8. Н.Ю. Прокопенко «Методы оптимизации».
9. Е. В. Елтошкина «Теория игр. Учебное пособие»
10. Н.Л. Леонова «Задачи линейного программирования и методы их решения».