Дискретная математика

ЛЕКЦИЯ 2

Зарецкий М.В.
ЧОУ ДПО ИТФИ

Метод математической индукции

Рассуждения бывают общие и частные.
Натуральное число, сумма цифр которого делится на 3, делится на 3 - общее.
Число 741 делится на 3 - частное.
Переход от общего утверждения к частному называется дедукцией.
Дедукция широко используется в математике.

Переход от частных утверждений к общему называется **индукцией.**
Индукция лежит в основе получения нового знания.
Индукция бывает полной (все частные случаи рассмотрены) и неполной (на основании ряда частных случаев делается общий вывод). Неполная индукция может привести к ошибкам.
Рассмотрим пример полной индукции.

Существуют правильные многогранники следующих видов (таб. 1).

Таб. 1 Правильные многогранники

№	Вид	Вершины	Ребра	Грани
1	Тетраэдр	4	6	4
2	Гексаэдр	8	12	6
3	Октаэдр	6	12	8
4	Додекаэдр	20	30	12
5	Икосаэдр	12	30	20

Пусть В, Г, Р — число вершин, граней и ребер правильного многогранника. Перебором легко доказать, что В+Г=Р+2. Это полная индукция

Многочлен Эйлера $n^2 + n + 41$ дает простые числа при $n = 0,1,2, \ldots, 39$. Можно ли на этом основании сделать вывод, что эта формула позволяет получить простое число при любом n? При $n = 40$ получаем:

$40^2 + 40 + 41 = 40(40 + 1) + 41 = 41^2$.

Долгое время считалось, что выражение $991n^2 + 1$ ни при каких натуральных n не является точным квадратом. Затем обнаружилось 29-значное число n, пр котором выражение дает точный квадрат

Заблуждение Ферма (Pierre de Fermat, 1601 – 1665) $2^{2^n}+1$- простое число. При $n=0,1,2,3,4$ это так. Но при $n=5$ получаем $2^{2^5}+1=2^{32}+1$. Это число делится на 641. Проверьте!

Перейдем к изучению **математической индукции**.

Метод математической индукции основан на аксиоме (принципе математической индукции).

Утверждение, зависящее от натурального числа n, справедливо для всех n , если:

1. оно справедливо при $n = 1$ (базис индукции);
2. из справедливости его при $n = k$ следует его справедливость при $n = k + 1$ шаг индукции.

Рассмотрим пример. Надо доказать, что:

$$1 \cdot 2 + 2 \cdot 3 + \cdots + n \cdot (n + 1) = \frac{n \cdot (n + 1) \cdot (n + 2)}{3}$$

1. Базис индукции. При $n = 1$ формула верна, так как $1 \cdot 2 = 2 = \frac{1 \cdot 2 \cdot 3}{3}$.
2. Шаг индукции. Пусть k – произвольное натуральное число и формула справедлива при $n = k$, то есть

$$1 \cdot 2 + 2 \cdot 3 + \cdots + k \cdot (k+1) = \frac{k \cdot (k+1) \cdot (k+2)}{3}.$$

Докажем, что формула имеет верна при $n = k + 1$, то есть

$$1 \cdot 2 + 2 \cdot 3 + \cdots + (k+1) \cdot (k+2) = \frac{(k+1) \cdot (k+2) \cdot (k+3)}{3}.$$

Для этого достаточно вычислить $(k+1) \cdot (k+2) + \frac{k \cdot (k+1) \cdot (k+2)}{3}$. Почему?

Элементы комбинаторики

В самых разных ситуациях приходится решать вопросы о том, сколько комбинаций, удовлетворяющих тем или иным условиям, можно получить из имеющихся объектов. Например, руководителю проекта необходимо разделить работу между программистами. Сколько возможно вариантов расписания занятий в университете.

Мы познакомимся с несколькими основными задачами комбинаторики, понимание которых необходимо программисту. Начнем с основных правил.

Правило сложения или правило или. Если элемент a может быть выбран n способами, а элемент b может быть выбран m способами, то выбрать **или** a, **или** b можно $m + n$ способами.

Пример. Студенту требуется начать работу с языком Python. Он может воспользоваться услугами двух облачных сервисов (Google Colaboratory или RStudio Cloud) или установить на своем компьютере среду разработки (Anaconda, PyCharm, Visual Studio с нагрузкой Python, RStudio). Сколькими способами он может начать работу с Python, если он выбирает только один вариант: или использует облачную среду, или устанавливает среду разработки на свой компьютер?

Решение. Выбрать облачную среду студент может 2 способами ($n = 2$), выбрать среду для стационарной разработки - 4 способами ($\mathrm{m} = 4$).

Следовательно, выбрать или облачную, или стационарную среду можно $n + m = 2 + 4 = 6$ способами.

Примечание. Среда разработки RStudio, предназначенная в первую очередь для разработки на языке R, позволяет работать и с языком Python.

Правило умножения (правило «и», основное правило комбинаторики). Если элемент a может быть выбран n способами и для каждого из таких выборов элемента a другой элемент b может быть выбран m способами, то выбор пары элементов (a, b) может быть осуществлен $m \cdot n$ способами.

Пример. Пусть студенту требуется выбрать пару («Облачная среда разработки», «Стационарная среда разработки». Сколько имеется вариантов?

Размещения, сочетания, перестановки

Пусть имеется n различных элементов. При выборе m элементов из имеющихся n принято говорить, что они образуют соединения из n элементов по m. **Размещениями** из n элементов по m $(m \leq n)$ называются соединения из n элементов по m, отличающиеся друг от друга составом элементов или порядком

Пример. Запишем все размещения из четырех элементов A, B, C, D по два:
$AB, BA, AC, CA, AD, DA, BC, CB, BD, DB, CD, DC.$
Получили 12 размещений из 4 элементов по 2.

Количество размещений из из n элементов по m обозначают A_n^m. Мы установили, что $A_4^2 = 12$. В общем случае:
$A_n^m = n \cdot (n-1) \cdot \dots \cdot (n-m+2) \cdot (n-m+1)$.

Эту формулу можно представить так:
$A_n^m = \frac{n!}{(n-m)!}$.

Пример. Сколько различных трехзначных чисел можно составить из цифр 1,2,3,4,5, если ни одна цифра не должна повторяться?

$A_5^3 = 5 \cdot 4 \cdot 3 = 60$. Тот же ответ можно получить и так: $A_5^3 = \frac{5!}{(5-3)!} = \frac{5 \cdot 4 \cdot 3 \cdot 2!}{2!} = 60$.

Если $m = n$, размещения отличаются друг от руга только порядком элементов. Такие размещения называются перестановками.

Количество перестановок из n элементов обозначается P_n. Получим формулу для этой величины: $P_n = A_n^n = \frac{n!}{(n-n)!} = \frac{n!}{0!} = n!$

Напомню, что по определению $0! = 1$.

Пример. Группа из 10 студентов использует на занятии 10 компьютеров. Программа генерирует список студентов с указанием номера компьютера. Сколько возможно вариантов? Найдите!

Сочетаниями из n элементов по m $(m \leq n)$ называются соединения из n элементов по m, отличающиеся друг от друга составом элементов.

Пример. Запишем все сочетания из четырех элементов A, B, C, D по два: AB, AC, AD, BC, BD, CD.

Получили 6 сочетаний из 4 элементов по 2.

Количество сочетаний из n элементов по m обозначается C_n^m. Встречается и такое обозначение: $\binom{n}{}$

Обратите внимание на расположение n и m в формулах – что «вверху» и что внизу в каждой из них. Пишется именно так!

Получим формулу для количества сочетаний из из n элементов по m. Выбрать m из n разных элементов можно C_n^m способами. Имеется $\mathrm{m}!$ возможностей упорядочения элементов каждого из сочетаний. Таким образом, $A_n^m = m!\, C_n^m$.

Следовательно, $C_n^m = \frac{A_n^m}{m!} = \frac{n!}{m!\cdot(n-m)!}$

Пример. Отдел в программистской фирме состоит из 10 человек. Сколько имеется вариантов создания группы из 4 человек для выполнения задания?

Решение. Надо найти величину

$$C_{10}^{4} = \frac{10!}{4!\cdot 6!} = \frac{10\cdot 9\cdot 8\cdot 7\cdot 6!}{4!\cdot 6!} = \frac{10\cdot 9\cdot 8\cdot 7}{4\cdot 3\cdot 2} = 10\cdot 3\cdot 7 = 210.$$

Обратите внимание на то, что не нужно сначала вычислять числитель и знаменатель, а только потом делить. Придется выполнить много ненужных вычислений. НО о компьютерной реализации – потом.

- Сочетания и бином Ньютона

Отметим сначала следующее:

$$C_n^m = C_n^{n-m}$$

$$C_n^0 = C_n^n = 1$$

$$C_n^1 = n$$

$$C_{n+1}^m = C_n^{m-1} + C_n^m.$$

Самостоятельно проверьте все эти утверждения!

С помощью этих свойств можно получить таблицу Паскаля (рис. 1).

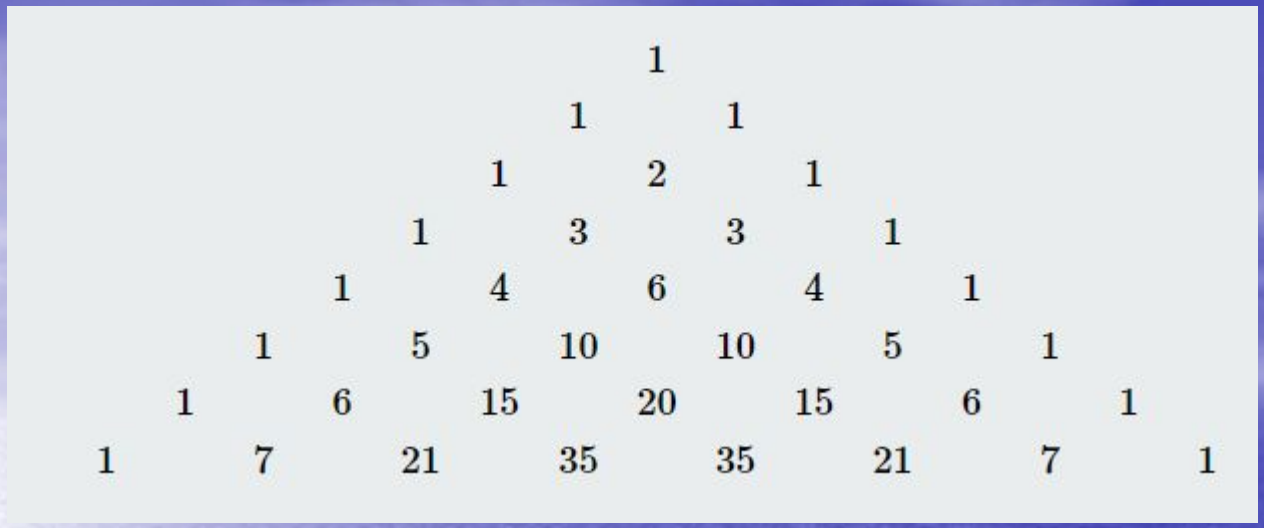


Рис. 1. Таблица Паскаля

Пусть n – номер строки в треугольнике, m – номер столбца в треугольнике (нумерация в обоих случаях начинается с 0). Элементы, расположенные на пересечении строки n и столбца m треугольника суть величины C_n^m.

Заметим, что числа, расположенные в строках треугольника суть коэффициенты полинома $(x+a)^0$ при $n=0$, $(x+a)^1$ при $n=1$, $(x+a)^2$ при $n=2$, $(x+a)^3$ при $n=3$ и т.д.

Можно доказать, что $(x+a)^n=\sum_{k=0}^{n} C_n^k x^k a^{n-k}$.

Эта формула называется формулой разложения бинома Ньютона. Ее коэффициенты C_n^k называются биномиальными коэффициентами.

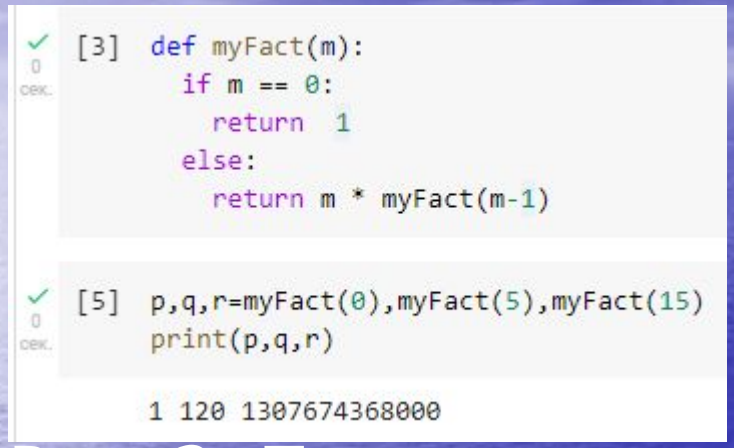
Формула бинома Ньютона доказывается методом математической индукции.

Биномиальные коэффициенты играют важную роль в теории вероятностей (биномиальный закон распределения случайной величины). В современной компьютерной графике используются кривые и поверхности Безье, основанные на полиномах Бернштейна, в которых используются биномиальные коэффициенты.

Комбинаторика в Python

Решение задач комбинаторики требует выполнения громоздких вычислений. Рассмотрим компьютерную реализацию этих вычислений на примере пакетов языка Python. Но для того, чтобы понять вычислительные аспекты комбинаторики, будем писать свои программы и сравнивать их с программами из пакетов Python.

Начнем с вычисления факториала. Тем более, что вычисление факториала – классический пример при изучении рекурсии.



```
[3]  def myFact(m):
       if m == 0:
         return  1
       else:
         return m * myFact(m-1)

[5]  p,q,r=myFact(0),myFact(5),myFact(15)
     print(p,q,r)

     1 120 1307674368000
```

Рис. 2. Программа для вычисления факториала и примеры ее использования

Вместо рекурсии можно использовать цикл.

Отметим, что факториал от аргумента 15 – это весьма большая величина.

Выполним те же вычисления с помощью функции из пакета NumPy (рис. 3).

```
[6]  p,q,r=np.math.factorial(0),np.math.factorial(5),np.math.factorial(15)
     print(p,q,r)

     1 120 1307674368000
```

Рис. 3. Вычисление факториала с помощью функции из пакета NumPy

В реальных задачах использование пакетов предпочтительнее.

Язык Python не ограничивает длину целого числа. Всегда ли это хорошо? Найдем количество перестановок для 50 объектов, равное 50! (рис. 4).

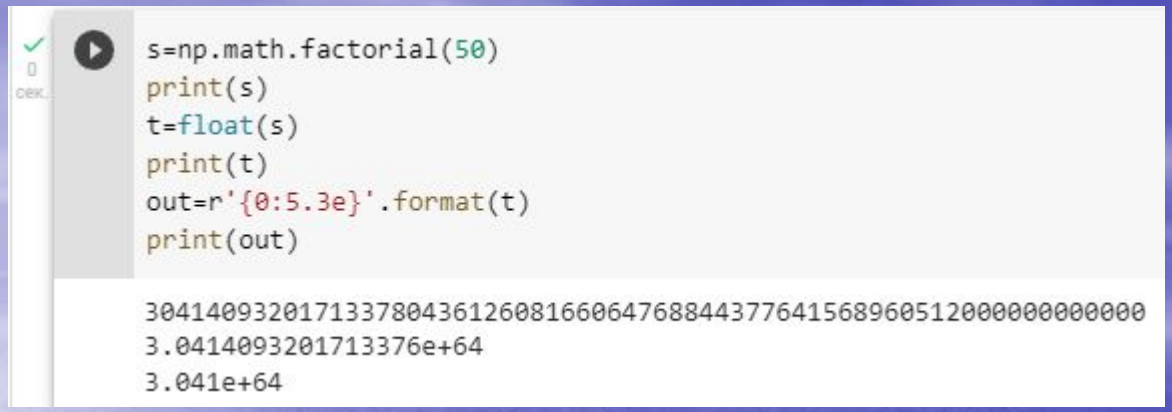


Рис. 4 Вычисление количества перестановок для 50 объектов

Мы получили целое число перемещений – точное, но малополезное. Преобразовав это число в вещественное, мы получили возможность оценить его. С помощью форматирования мы сделали вывод более удобным для оценки.

Более интересный подход к вычислению факториала реализован в научном пакете scipy.special (рис. 5)

```
[34] p,q,r=scipy.special.factorial(0,exact = True),scipy.special.factorial(5,exact=True),scipy.special.factorial(50)
     print(p,type(p),q,type(q),r,type(r))
     out=r'{0:5.3e}'.format(t)
     print(out)

     1 <class 'int'> 120 <class 'int'> 3.041409320171338e+64 <class 'numpy.float64'>
     3.041e+64
```

Рис. 5. Вычисление факториала функцией пакета scipy.special

Функция scipy.special.factorial принимает 2 аргумента. Первый аргумент – неотрицательное целое число, факториал которого вычисляется.

Второй аргумент exact указывает, вычисляется ли факториал в виде целого числа или в виде вещественного числа. Аргумент не является обязательным. Если он не задан, результат возвращается в виде вещественного числа. Если требуется, чтобы результат был возвращен в виде целого числа, при вызове функции указывается exact = True (см. рис. 5).

Функция binom из того же пакетане имеет аргумента exact и возвращает только вещественные значения.