**Лекция 9**

**Основы комбинаторики**

*(Подробный разбор: от перестановок до реальных задач)*

**1. Что такое комбинаторика?**

**Комбинаторика** — это раздел математики, изучающий способы подсчёта и упорядочивания объектов.

**Где применяется?**

* Генерация паролей.
* Оптимизация маршрутов (например, задача коммивояжёра).
* Анализ вероятностей (лотереи, карточные игры).
* Программирование (алгоритмы, обработка данных).

**2. Основные понятия**

**2.1. Перестановки (Permutations)**

**Определение:** Упорядоченная выборка **всех** элементов из множества.

**Формула:**  
Pn=n!   
где n!=1⋅2⋅3⋅...⋅n (читается как «эн факториал»).

**Пример:**  
Сколько способов расставить книги А, Б, В на полке?

* Варианты: АБВ, АВБ, БАВ, БВА, ВАБ, ВБА → 6 способов.
* По формуле: 3!=6.

**Особенности:**

* Все элементы участвуют в перестановке.
* Порядок важен: АБВ ≠ БАВ.

**Код на Python:**

import math

n = 3

print(math.factorial(n)) # 6

**2.2. Размещения (Arrangements)**

**Определение:** Упорядоченная выборка **k** элементов из **n**.

**Формула:**  
A\_n^k=n!/(n−k)!

**Пример:**  
Сколько двузначных чисел можно составить из цифр 1, 2, 3 без повторений?

* Варианты: 12, 13, 21, 23, 31, 32 → 6 чисел.
* По формуле:   
  A\_3^2=3!/(3−2)!=6/1=6

**Особенности:**

* Порядок важен: 12 ≠ 21.
* Элементы не повторяются.

**Код на Python:**

python

Copy

Download

from itertools import permutations

items = [1, 2, 3]

k = 2

print(list(permutations(items, k)))

# Результат: [(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)]

**2.3. Сочетания (Combinations)**

**Определение:** Неупорядоченная выборка **k** элементов из **n**.

**Формула:**  
C\_n^k=n! / k!(n−k)!

**Пример:**  
Сколько способов выбрать 2 фрукта из яблока, банана и апельсина?

* Варианты: яблоко+банан, яблоко+апельсин, банан+апельсин → 3 способа.
* По формуле: C\_3^2 = 3! / (2! \* 1!) = 3

**Особенности:**

* Порядок не важен: яблоко+банан = банан+яблоко.

**Код на Python:**

from itertools import combinations

print(list(combinations(items, 2)))

# Результат: [(1,2), (1,3), (2,3)]

**4. Практические задачи**

**4.1. Задача о пароле**

**Условие:** Сколько паролей длиной 4 символа можно составить из букв A, B, C, если символы могут повторяться?

**Решение:**

* Каждый символ — 3 варианта.
* Всего: 34=8134=81.

**4.2. Задача о команде**

**Условие:** В группе 10 человек. Сколько способов выбрать 3 из них для участия в конкурсе, если порядок выбора не важен?

**Решение:**

* Используем сочетания: C\_10^3=120.

**4.3. Задача о вероятности**

**Условие:** В лотерее 50 билетов, 5 выигрышных. Какова вероятность, что 2 случайно выбранных билета окажутся выигрышными?

**Решение:**

* Число благоприятных исходов: C\_5^2=10.
* Всего исходов: C\_50^2=1225.
* Вероятность: 10 / 1225≈0.00816.

**5. Комбинаторика в программировании**

**5.1. Генерация всех вариантов**

**Пример:** Найти все подмножества множества [1, 2, 3].

from itertools import chain, combinations

def all\_subsets(items):

return chain.from\_iterable(combinations(items, r) for r in range(len(items)+1))

print(list(all\_subsets([1, 2, 3])))

# Результат: [(), (1,), (2,), (3,), (1,2), (1,3), (2,3), (1,2,3)]

**5.2. Оптимизация алгоритмов**

Комбинаторные методы используются в:

* Поиске кратчайших путей (графы).
* Шифровании данных (перебор ключей).
* Машинном обучении (выбор признаков).

**6. Итоги**

1. **Перестановки** — порядок всех элементов.
2. **Размещения** — упорядоченные выборки.
3. **Сочетания** — неупорядоченные выборки.
4. **Повторения** меняют формулы!
5. **Python-библиотеки** (itertools, math) упрощают расчёты.