

Дискретная математика

ЛЕКЦИЯ 1

Зарецкий М.В.

ЧОУ ДПО ИТФИ

Введение

Дискретная математика - это часть математики, спецификой которой является дискретность - антипод непрерывности. Она включает: теорию чисел, алгебру, математическую логику, теорию множеств, комбинаторику, теорию графов и сетей, теорию алгоритмов, формальные грамматики, теорию игр, теорию кодирования и т.д.

Теория множеств

Создатель теории множеств немецкий математик Георг Кантор (1845 – 1918) писал: «Под многообразием или множеством я понимаю вообще всякое многое, которое можно мыслить как единое, т.е. всякую совокупность определенных элементов, которая может быть связана в одно целое с помощью некоторого закона».

Кратко это формулируют так:
«Множество — есть многое, мыслимое нами как единое». Но это не определение. Множество — одно из основных математических понятий, оно не определяется через другие. Мы должны его чувствовать интуитивно. Группа студентов, спортивная команда, стая птиц, букет цветов, колония микробов — различные примеры множеств. Но множества могут состоять и из совершенно разнородных предметов.

Способы задания множеств

● Множества будем обозначать заглавными буквами A, B, \dots, X, \dots , а их элементы – строчными буквами a, b, \dots, x, \dots . То, что a – элемент множества A обозначается так: $a \in A$. Противоположное утверждение обозначается так: $a \notin A$. Множества бывают конечными, если количество элементов в них конечно, и

—

Рассмотрим способы задания множеств.

1. Конечное множество можно задать перечислением его элементов. Например,
 $A = \{1, 3, 5, 7, 9\}$.

2. Множество можно задать с помощью некоторого характеристического свойства P : $A = \{x | P(x)\}$. Например,
 $M = \{x | x \in N, x : 3\}$, множество натуральных чисел, кратных 3 – бесконечное множество, которое нельзя задать перечислением.

\emptyset - пустое множество.

3. Множество можно задать с помощью некоторого характеристического свойства $P: A = \{x | P(x)\}$. Например,

$M = \{x | x \in N, x : 3\}$ можно описать с помощью процедуры, определяемой двумя правилами, : 1) $3 \in M$, 2) если $m \in M$, то $m + 3 \in M$.

К порождающим процедурам относится получение новых множеств из уже имеющихся, т.е. операции над множествами.

Операции над множествами

● Рассматриваем множества A, B, \dots, X, \dots .
Множество A является подмножеством множества B (записываем так: $A \subset B$), если каждый элемент A является элементом B . Считается что, $\emptyset \subset A$ для любого A .

Два множества называются **равными**, если они состоят из одних и тех же

Формально это определение можно

записать так: $A = B \Leftrightarrow \begin{cases} A \subset B \\ B \subset A \end{cases}$.

Для наглядного изображения множеств и операций над ними применяются диаграммы Эйлера-Венна (Leonard Euler, 1707 – 1783, John Venn, 1834 – 1923).

Например, утверждение $A \subset B$ изображается диаграммой Эйлера-Венна так (рис. 1):

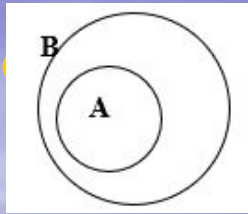


Рис. 1. Иллюстрация $A \subset B$

Объединением множеств A и B называется множество $A \cup B$, элементы которого содержатся в A или в B или в обоих множествах одновременно (рис. 2).

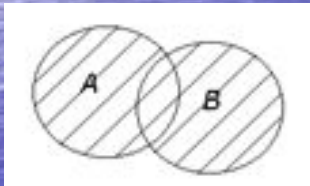


Рис. 2. Объединение множеств

Пересечением множеств A и B называется множество $A \cap B$, элементы которого содержатся в обоих множествах одновременно (рис. 3).

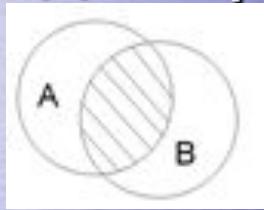


Рис. 3. Пересечение множеств

Разностью множеств A и B называется множество $A \setminus B$, элементы которого содержатся в A и не содержатся в B (рис. 4).

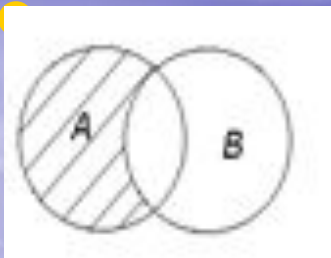


Рис. 4. Разность множеств

Симметрической разностью множеств A и B называется множество

$$A \Delta B = (A \setminus B) \cup (B \setminus A) \text{ (рис. 5).}$$

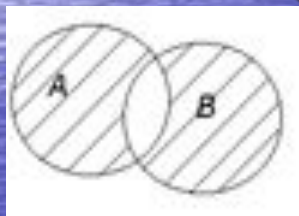


Рис. 5. Симметрическая разность

Если все рассматриваемые множества являются подмножествами одного определенного множества, то это множество называется универсальным, обозначается E и изображается прямоугольником.

Дополнением множества A называется разность $E \setminus A$ и обозначается \bar{A} (рис. 6).

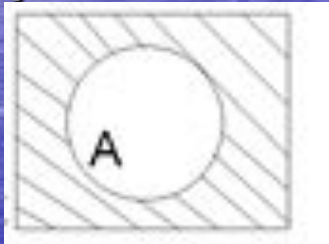


Рис. 6. Дополнение

Декартовым произведением множеств A и B называется множество $A \times B$ упорядоченных пар (a, b) , где $a \in A$ и $b \in B$. Кратко это записывается так:
 $A \times B = \{(a, b) | a \in A, b \in B\}$.

Аналогично определяется:

$$A_1 \times \dots \times A_n = \{(a_1, \dots, a_n) | a_1 \in A_1, \dots, a_n \in A_n\}.$$

Декартово произведение нескольких множеств используется в теории реляционных баз данных.

Каждый объект (сущность) характеризуется упорядоченной последовательностью значений реквизитов (кортежем). Каждый реквизит принимает значение из определенного множества - домена. Таким образом, объект определен на декартовом произведении доменов.

Введенные операции над множествами обладают следующими свойствами, которые доказываются взаимным включением.

1. $A \cup A = A, A \cap A = A$ – идемпотентность.
2. $A \cup B = B \cup A, A \cap B = B \cap A$ – коммутативность.
3. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C), A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ – ассоциативность.
4. $A \cup (B \cap C) = (A \cup B) \cap (A \cup C), A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ - дистрибутивность
5. $A \cup (A \cap B) = A, A \cap (A \cup B) = A$ - поглощение.

6. $A \cup \emptyset = A, A \cap \emptyset = \emptyset$

$A \cup E = E, A \cap E = A$ - действия с константами.

7. $A \cup \bar{A} = E, A \cap \bar{A} = \emptyset$ - действия с дополнением.

8. $\overline{\bar{A}} = A$ - инволютивность.

9. $\overline{(A \cup B)} = \bar{A} \cap \bar{B}, \overline{(A \cap B)} = \bar{A} \cup \bar{B}$ - законы де Моргана (Augustus de Morgan, 1806 - 1871).

10. $(A \cap B) \cup (A \cap \bar{B}) = A,$

$(A \cup B) \cap (A \cup \bar{B}) = A$ - склеивание.

11. $(A \cap C) \cup (B \cap \bar{C}) \cup (A \cap B) = (A \cap C) \cup (B \cap \bar{C}),$

$(A \cup C) \cap (B \cup \bar{C}) \cap (A \cup B) = (A \cup C) \cap (B \cup \bar{C})$ - обобщенное склеивание.

12. $A \cup (\bar{A} \cap B) = A \cup B,$

$A \cap (\bar{A} \cup B) = A \cap B$

$$13. A \setminus B = A \cap \overline{B}.$$

$$14. A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C),$$
$$A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C).$$

О декартовом произведении.

$$A \times B \neq B \times A,$$

$$(A \times B) \times C \neq A \times (B \times C),$$

$$(A \cup B) \times C \neq (A \times C) \cup (B \times C),$$

$$(A \cap B) \times C \neq (A \times C) \cap (B \times C).$$

- Характеристические функции
Характеристической функцией $\chi_A(x)$ множества $A \subset E$ называется функция, отображающая E в двухэлементное множество $\{0, 1\}$:

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \text{ (рис. 7).}$$

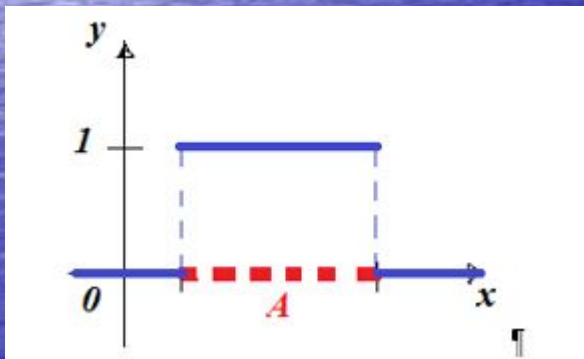


Рис 7 Характеристическая функция

Рассмотрим свойства характеристических функций.

1. $\chi_A^n(x) = \chi_A(x)$ для любого натурального n .

2. $\chi_{A \cap B}(x) = \chi_A(x) \cdot \chi_B(x)$.

3. $\chi_{A \cup B}(x) = \chi_A(x) + \chi_B(x) - \chi_A(x) \cdot \chi_B(x)$.

4. $\chi_{\bar{A}}(x) = 1 - \chi_A(x)$.

5. $\chi_{A \setminus B}(x) = \chi_A(x) - \chi_A(x) \cdot \chi_B(x)$.

6. $\chi_{A \Delta B}(x) = \chi_A(x) + \chi_B(x) - 2 \cdot \chi_A(x) \cdot \chi_B(x)$.

7. $\chi_{A \times B}(x, y) = \chi_A(x) \cdot \chi_B(y)$.

Докажем свойство 2.

$$\chi_{A \cap B}(x) = \begin{cases} 1, & x \in A \text{ и } x \in B, \\ 0, & x \in A \text{ и } x \notin B, \\ 0, & x \notin A \text{ и } x \in B, \\ 0, & x \notin A \text{ и } x \notin B. \end{cases} = \chi_A(x) \cdot \chi_B(x).$$

Характеристические функции используются для доказательства теоретико-множественных тождеств. Тождество верно тогда и только тогда, когда характеристические функции его правой и левой частей совпадают.

Докажем свойство 2.

$$\chi_{A \cap B}(x) = \begin{cases} 1, & x \in A \text{ и } x \in B, \\ 0, & x \in A \text{ и } x \notin B, \\ 0, & x \notin A \text{ и } x \in B, \\ 0, & x \notin A \text{ и } x \notin B. \end{cases} = \chi_A(x) \cdot \chi_B(x).$$

Характеристические функции используются для доказательства теоретико-множественных тождеств. Тождество верно тогда и только тогда, когда характеристические функции его правой и левой частей совпадают.

● Пусть требуется доказать тождество

$$A \cup (A \cap B) = A.$$

Построим характеристическую функцию левой части этого тождества.

$$\begin{aligned}\chi_{A \cup (A \cap B)}(x) &= \chi_A(x) + \chi_{A \cap B}(x) - \chi_A(x) \cdot \chi_{A \cap B}(x) = \\ &= \chi_A(x) + \chi_A(x) \cdot \chi_B(x) - \chi_A(x) \cdot \chi_A(x) \cdot \chi_B(x) = \\ &= \chi_A(x) + \chi_A(x) \cdot \chi_B(x) - \chi_A(x) \cdot \chi_B(x) = \chi_A(x).\end{aligned}$$

Итак, доказано, что характеристическая функция левой части равна характеристической функции правой части. Тождество доказано.

Работа с множествами в Python

- Множество в Python – неупорядоченная последовательность уникальных элементов. Рассмотрим примеры задания множества в Python (рис. 8).

```
✓ [20] a,b,c,d={1,2,3,7,9,3},[1,2,3,7,9,3],(1,2,3,7,9,3),'123793'  
0  
сек. print(a,type(a))  
b,c,d=set(b),set(c),set(d)  
print(b,type(b))  
print(c,type(c))  
print(d,type(d))
```

- Рис. 8. Задание множеств.

После выполнения фрагмента кода убеждаемся, что множества содержат только уникальные элементы.

Приведем примеры создания пустого множества (рис. 9).

```
✓ [22] empty1,empty2,empty3,empty4={},[],(),''
0
сек. empty1,empty2,empty3,empty4 = set(empty1),set(empty2),set(empty3),set(empty4)
print(empty1,type(empty1))
print(empty2,type(empty2))
print(empty3,type(empty3))
print(empty4,type(empty4))
```

Рис. 9. Варианты создания пустого множества

Сразу же возникает вопрос, как добавить элемент в множество. Напомню, что множества в языке Python являются объектами и, следовательно, обладают методами.

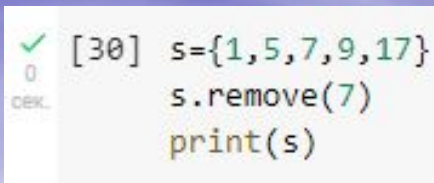

```
✓ [24] s = {1,3,7}
0 s.add(9)
сек. print(s)
s.add(3)
print(s)
```

Рис. 10. Добавление элемента к множеству

Итак, для добавления элемента к множеству служит метод `add(<элемент>)`. Рассмотрите пример. Объясните результат работы.

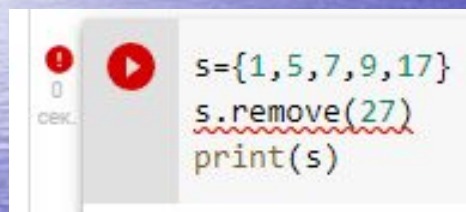
Рассмотрим теперь удаление элемента из множества.

Для этой цели предусмотрены два метода – `remove(<элемент>)` (рис. 11, 12) и `discard(<элемент>)` (рис. 13, 14).

A terminal window showing a successful execution of Python code. On the left, there is a green checkmark icon, the number '0', and the word 'сек.' (seconds). The code consists of three lines: `s={1,5,7,9,17}`, `s.remove(7)`, and `print(s)`.

```
✓ [30] s={1,5,7,9,17}
0      s.remove(7)
сек.   print(s)
```

Рис. 11 Удаление из множества находящегося в нем элемента
Выполните фрагмент кода. Убедитесь, что элемент 7 будет удален.

A terminal window showing an error during the execution of Python code. On the left, there is a red exclamation mark icon, the number '0', and the word 'сек.' (seconds). The code consists of three lines: `s={1,5,7,9,17}`, `s.remove(27)`, and `print(s)`. The number '27' in the second line is underlined with a red wavy line, indicating an error.

```
! [30] s={1,5,7,9,17}
0      s.remove(27)
сек.   print(s)
```

Рис. 12. Попытка удаления из множества отсутствующего в нем элемента
В этом случае получаем аварийное сообщение.


```
✓ [32] s={1,5,7,9,17}
0 s.discard(7)
сек. print(s)
```

Рис. 13. Удаление из множества существующего в нем элемента
Выполните фрагмент кода. Убедитесь, что элемент 7 будет удален.

```
✓ [33] s={1,5,7,9,17}
0 s.discard(27)
сек. print(s)
```

Рис. 14. Попытка удаления из множества отсутствующего в нем элемента
В этом случае просто ничего не происходит.

Рассмотрим создание копии множества.
Обратим внимание на типичную ошибку.

```
✓ [34] s={1,5,7,9,17}  
0     t=s  
сек.  t.add(31)  
      print(s)
```

Рис. 15. Неудачная попытка копирования множества

Убедитесь, что действия над копией (множеством *t*) приводят к изменению оригинала (множества *s*).

Для корректного копирования множеств надо использовать метод `copy()` (рис. 16).


```
✓ [35] s={1,5,7,9,17}  
0  
сек. t=s.copy()  
t.add(31)  
print(s)
```

Рис. 16. Корректное копирование множества

Для удаления элементов множества служит метод `clear()` (рис. 17).

```
✓ [39] s={1,5,7,9,17}  
0  
сек. s.clear()  
print(s)
```

Рис. 17. Очистка множества

Для определения количества элементов множества служит функция `len(<множество>)` (рис. 18).

```
✓ [42] s={1,5,7,9,17}
0      k=len(s)
сек.   print(k)
```

Рис. 18. Определение количества элементов множества

Обратите внимание. Данная функция не является методом объекта типа множество!

Мы можем в цикле перебрать все элементы множества (рис. 19).

```
✓ [51] s={1,5.09,'This is a set',(1,5),(7,9)}  
0  
сек.   for _ in s:  
        print(_)
```

Рис. 19. Циклический перебор элементов множества

Мы рассмотрели «технические» операции над множествами. Перейдем к рассмотрению теоретико-множественных операций

Эти операции могут выполняться с помощью «традиционных» функций и с помощью методов.

Объединение множеств (рис. 20).

```
✓ [57] p,q={1,17,5},{5,9,21}
0 сек. r,s=p|q,p.union(q)
print(p,q,r,s)

{1, 5, 17} {9, 21, 5} {1, 17, 5, 21, 9} {1, 17, 5, 21, 9}
```

Рис. 20. Объединение множеств

Объединение множеств выполняется с помощью операции “|” и метода `union(<множество>)`. При этом множества – операнды `p` и `q` не изменяются.

Рассмотрим операцию добавления элементов одного множества в другое (рис. 21, 22).

```
✓ 0 сек. p,q={1,17,5},{5,9,21,17}
p |= q
print(p,q)

{1, 17, 5, 21, 9} {9, 21, 5, 17}
```

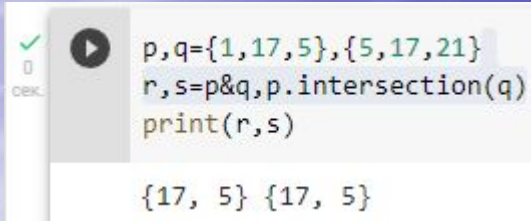
Рис. 21. Добавление элементов множества q в множество p (1 способ)

```
✓ [63] 0 сек. p,q={1,17,5},{5,9,21,17}
p.update(q)
print(p,q)

{1, 17, 5, 21, 9} {9, 21, 5, 17}
```

Рис. 22. Добавление элементов множества q в множество p (2 способ)

Пересечение множеств также выполняется двумя способами (рис. 23).



```
p,q={1,17,5},{5,17,21}
r,s=p&q,p.intersection(q)
print(r,s)
```

{17, 5} {17, 5}

Рис. 23. Пересечение множеств

Пересечение множеств выполняется с помощью операции “&” и метода `intersection(<множество>)`. При этом множества – операнды `p` и `q` не изменяются.

Рассмотрим еще два способа реализации пересечения множеств (рис. 24, 25).


```
✓ [66] p,q={1,17,5},{5,17,21}
0
сек. p &= q
      print(p,q)
```

Рис. 24. Построение пересечения множеств (способ 1).

```
✓ [67] p,q={1,17,5},{5,17,21}
0
сек. p.intersection_update(q)
      print(p,q)
```

Рис. 25. Построение пересечения множеств (способ 2).

Рассмотрим два способа построения разности множеств (рис. 26).

```
✓ [69] p,q={1,17,5,19},{5,17,21}
0
сек. r,s=p-q,p.difference(q)
print(r,s)
```

Рис. 26. Разность множеств

Существуют еще два способа получить разность множеств - с помощью операции «-» и метода

`difference_update(<множество>)`.

Студенты смогут изучить их самостоятельно.

Такой же набор способов существует для получения симметрической разности.


```
✓ [70] p,q={1,17,5,19},{5,17,21}
0
сек. r,s=p^q,p.symmetric_difference(q)
print(r,s)

{1, 19, 21} {1, 19, 21}
```

Рис. 27. Симметричная разность множеств

Существуют еще два способа получить симметричную разность множеств - с помощью операции « \wedge » и метода `symmetric_difference_update(<множество>)`. Студенты смогут изучить их самостоятельно.

Рассмотрим операции сравнения множеств.

```
✓ 0 сек. ▶ p,q,r={1,17,5,19},17,39  
s,t = q in p,r in p  
print(s,t)
```

True False

Рис. 28 Проверка принадлежности элемента множеству

```
✓ 0 сек. [72] p,q,r={1,17,5,19},{19,1,5,17},{19,2,5,17}  
s,t = p == q,p == r  
print(s,t)
```

Рис. 29. Проверка равенства множеств

```
✓ 0 сек. ▶ p,q={1,17,5,19},{1,5,17}  
s,t = q <= p, q.issubset(p)  
print(s,t)
```

True True

Рис. 30. Проверка того, что множество является подмножеством другого

В итоге рассмотренной операции возвращается True и в случае равенства множеств-операндов. При использовании операции «<» d в случае равенства множеств возвращается False.

Операции «>=»,issuperset(<множество>), «>» предназначены для проверки включения второго множества-операнда в первое. Студенты могут изучить их самостоятельно.

Для проверки, является ли пересечение множеств пустым служит метод `isdisjoint(<множество>)` (рис. 30).

```
✓ [78] p,q,r={1,17,5,39},{19,1,5,17},{19,2,33,75}
0
сек. s,t = p.isdisjoint(q),p.isdisjoint(r)
print(s,t)
```

Рис. 31. Проверка наличия пустого пересечения множеств

Перечисление всех элементов множества часто неудобно или даже невозможно.

Рассмотрим некоторые способы генерации множеств.


```
✓ [81] p={_ for _ in [1,3,5,5,3,5,9]}  
0 print(p)  
DEK.
```

Рис. 32. Генерация элементов множества простым перечислением

Более интересна генерация множества с использованием программного кода (рис. 32).

```
✓ [81] p = {x for x in [1, 7, 9, 5, 15, 14, 21] if x % 3 ==0 }  
0 print(p)  
DEK.
```

Рис. 33. Генерация множества с использованием элементов программного кода