Фізико-технічний інститут *кафедра ММЗІ*

Теорія складності **Осінь 2021**

Матеріал 1: Інтуїтивне поняття алгоритму

1.1 Інтуїтивне поняття алгоритму

Теорія алгоритмів (англ. theory of computation) виникла в 30-х роках XX століття як розділ математичної логіки, оскільки перші найчисельніші застосування теорія алгоритмів мала саме в математичній логіці. На даний момент теорія алгоритмів вважається окремим розділом математики, який вивчає загальні властивості алгоритмів, але теорія алгоритмів є тісно пов'язаною з математичною логікою, і, в першу чергу, численнями формальних теорій.

Фактично алгоритми є відомими людству достатньо давно, але перші відомі алгоритми, як, наприклад, алгоритм Евкліда є правилами виконання арифметичних дій з числами. В таких алгоритмах чітко визначені об'єкти (натуральні числа), до яких застосовується алгоритм, та елементарні кроки (додавання, віднімання, множення та ділення), необхідні для виконання алгоритму. Поступово алгоритми стали застосовувати й до інших об'єктів. *Областю застосовності* алгоритму називають сукупність тих об'єктів, до яких його можна застосувати. Цю сукупність об'єктів також називають *множиною початкових даних* або *множиною вхідних даних*. Робота кожного алгоритму відбувається шляхом виконання послідовності певних елементарних дій, які називають *кроками (алгоритму)*, а процес їхнього виконання називають *алгоритмічним процесом*. Завершення алгоритмічного процесу призводить до отримання певного об'єкту на завершальному кроці, який називають *результатом (роботи алгоритму)*. Множину, яка містить всі можливі результати роботи алгоритму, називають *областю значень* або *множиною вихідних даних*. Якщо результатом роботи алгоритму при застосуванні до об'єкту α з множини вхідних даних є об'єкт β, то кажуть, що *алгоритм перетворює об'єкт α в об'єкт β*. Алгоритмічний процес може ніколи на завершити свою роботу при застосуванні до певного об'єкту вхідних даних. Якщо алгоритмічний процес при застосуванні до певного об'єкту вхідних даних завершується за скінченну кількість кроків, то кажуть, що алгоритм є *застосовним* до цього об'єкту вхідних даних, інакше — алгоритм є *незастосовним* до цього об'єкту вхідних даних.

Неформально поняття алгоритму можна визначити як це процес послідовної покрокової обробки об'єкту з області застосовності алгоритму, який може закінчити свою роботу, тоді результатом його роботи є об'єкт, який отримали

на останньому кроці; або ніколи не закінчити свою роботу, тоді вважають , що результат його роботи є невизначеним. Таке неформальне означення алгоритму ще називають *інтуїтивним поняттям алгоритму*.

Якщо є заданим скінченний алфавіт і над ним визначені мови множини вхідних даних та множини вихідних даних, є заданою множина всіх можливих елементарних кроків, а також є заданим скінченний алфавіт та визначена мова над ним всіх можливих описів алгоритму, то кажуть, що є визначеною *модель обчислень* або *алгоритмічна система*.

Будь-які об'єкти можна описати за допомогою формальної мови над деяким скінченним алфавітом. Тому будь-який алгоритм є покроковим перетворенням слів над деяким скінченним алфавітом в слова над цим же алфавітом, вважаючи, що, можливо, не всі слова входять до множини вхідних даних. Тому про алгоритм кажуть, що він *обчислює функцію* f (або *обчислює значення функції* f), якщо його область застосування збігається з областю визначення функції f, і алгоритм перетворює будь-який об'єкт α зі своєї області застосування в об'єкт $f(\alpha)$. З кожним алгоритмом асоціюється одна функція, яку він обчислює. Довільну функцію називають *обчислювальною*, якщо існує алгоритм, який її обчислює. Тому поняття алгоритму і обчислювальної функції є тісно пов'язаними.

Будь-який алгоритм має опис з використання певного скінченного алфавіту, отже представляється певним словом скінченної довжини над деякою формальною мовою. Це означає, що загальна кількість всіх алгоритмів, а, отже, і обчислювальних функцій, є зліченою. В свою чергу, навіть множина всіх функцій виду $\mathbb{N} \to \mathbb{N}$ є незліченою. З цього випливає, що деякі функції є необчислювальними. Відповідно задачі називають *розв'язними*, якщо вони мають алгоритм розв'язку, та *нерозв'язними* — в іншому випадку.

В 1900 році математик Давид Гільберт (англ. David Hilbert) склав список математичних задач, які не той момент не мали розв'язку. Згодом з'ясувалося, що десята проблема Гільберта, задача розв'язку довільного діофантового рівняння, є нерозв'язною. Щоб довести це довелося побудувати нову математичну теорію.

Інтуїтивне поняття алгоритму визначає декілька загальних властивостей поняття алгоритм.

Обов'язкові фундаментальні властивості алгоритму:

- *скінченність* (або *об'єктивність*);
  Має бути можливість записати довільний алгоритм у вигляді скінченного слова над деяким скінченним алфавітом. Повністю прочитати або записати нескінченний опис алгоритму не має жодної можливості, оскільки йде мова про конструктивні дії.
- *дискретність*;
  Незалежно від моделі обчислень процес виконання будь-якого алгоритму є дискретним процесом, який складається з елементарних кроків. Також

під дискретність підпадає представлення вхідних даних у вигляді слів над скінченним алфавітом.

- *детермінованість*;
  Незалежно від моделі обчислень опис алгоритму однозначно визначає послідовність елементарних кроків. При застосуванні одного й того ж алгоритму до тих самих вхідних даних завжди маємо отримати такий самий результат. Вплинути на результат одного алгоритму може тільки зміна вхідних даних.

Необов'язкові властивості алгоритму:

- *масовість алгоритму*;
  Іноді від алгоритму обов'язково вимагають, щоб множина вхідних даних була нескінченною. В більшості випадків дійсно саме такі алгоритми зустрічаються на практиці і викликають найбільший інтерес. Але масовість це, скоріше, властивість самої задачі, а не алгоритму, і тому не може бути обов'язковою властивістю алгоритму. Існують задачі, які не є масовими, тобто множина можливих екземплярів цієї задачі є скінченною, і мають певний алгоритм розв'язку. Наприклад, пошук 52-го простого числа Мерсенна або побудова відомої комп'ютерної програми, яка друкує вираз "Hello world".
- *правильність*;
  Незалежно від моделі обчислень з кожним алгоритмом асоціюється певна функція, яку він обчислює. І, якщо існує декілька алгоритмів, які обчислюють одну й ту саму функцію, то потрібні чіткі критерії, який з алгоритмів вважати правильним. Ці критерії мають залежати від конкретної моделі обчислень, тому ця властивість не може бути фундаментальною для загального поняття алгоритм.
- *завершуваність*;
  Іноді вважають, що алгоритм має завжди завершувати свою роботу, тобто бути застосовним до будь-якого об'єкту множини вхідних даних. З точки зору практичного застосування було б добре, щоб всі алгоритми мали властивість, але в загальному випадку це не так. Існує багато алгоритмів, які є незастосовними до певних об'єктів множини вхідних даних, або навіть до всіх об'єктів множини вхідних даних. З точки зору комп'ютерних програм таким прикладом є програма, яка містить тільки "вічний цикл".