Лекція 12

Тема 8. Теорія обчислень

Алгоритми і обчислювальна складність

План лекції

- Складність алгоритмів. Поліноміальні і експоненціальні алгоритми
- Класи задач *P* та *NP*
- Теорема Кука
- Приклади *NP* - повних задач

Складність алгоритмів

У класичній теорії алгоритмів задачу вважають розв'язною, якщо існує алгоритм (машина Тьюрінга, наприклад), який її розв'язує. Проте для реалізації деяких алгоритмів за будь-яких розумних з точки зору фізики припущеннях щодо швидкості виконання елементарних кроків може потребуватись більше часу, ніж за сучасними поняттями існує всесвіт. Тому виникає потреба конкретизувати поняття розв'язності і надати йому оціночний, кількісний характер. Потрібно ввести такі характеристики алгоритмів, котрі дозволяли б судити про можливість і доцільність їхнього практичного застосування.

З міркувань зручності у теорії складності дискретних задач прийнято розглядати тільки *задачі розпізнавання властивостей*. У цих задачах формулюють питання, які допускають одну з двох відповідей – „**так**” або „**ні**”. Дуже багато задач дискретної математики можна сформулювати саме у такій формі. Наприклад, у задачі про комівояжера, якщо її сформулювати як задачу розпізнавання властивостей, *потрібно виявити, чи існує гамільтонів цикл, довжина якого не перевищує задану межу B*.

Критерієм ефективності в теорії складності є часова складність алгоритму як функція вхідної довжини задачі. Спочатку дамо неформальне пояснення понять, які далі використовуватимемо.

Вхідна довжина задачі з конкретними числовими даними (або *довжини входу*) – це кількість символів у ланцюжку, котрим описується дана задача в прийнятій схемі кодування. Наприклад, задачу про комівояжера можна закодувати послідовністю чисел, які відповідають віддалям між містами. Якщо розглядається деякий алгоритм на графі $G = (V, E)$, то довжиною входу можуть бути, наприклад, $|V|$, $|E|$, $|V| + |E|$.

Часова складність (у подальшому – просто *складність*) алгоритму при заданій вхідній довжині n – це максимальна кількість кроків (операцій), що виконуються вибраною моделлю обчислювального пристрою при розв’язуванні заданим алгоритмом задач даного типу з вхідною довжиною n з різними конкретними числовими даними. Зрозуміло, що для однієї й тої самої задачі можуть існувати як повільні, так і швидкі алгоритми.

Зазначимо, що складність залежить від моделі алгоритмів, яку ми використовуємо. Кількість кроків буде меншою, якщо на кожному кроці буде виконуватись більше роботи.

Проте фундаментальні поняття, такі, як поліноміальна складність, у значній мірі не залежать від вибору моделі алгоритму. Ми виберемо машину Тьюрінга як модель алгоритму.

Алгоритм називають *поліноміальним*, якщо його складність при довільному n не перевищує $p(n)$, де p – деякий поліном, n – вхідна довжина. Алгоритми, складність яких не допускає такої оцінки, називають *експоненціальними*. Отже, до експоненціальних відноситься й алгоритм, складність якого оцінюється, наприклад, функцією $n^{\ln n}$, яка не є експонентою.

Класи задач *P* та *NP*

Вважають, що задача є добре розв'язною, якщо для її розв'язування існує поліноміальний алгоритм. Клас усіх таких задач позначають як *P* (Polynomial).

При формальному означенні ми задамо функцію складності, пов'язану з машиною Тьюрінга *T*, рівністю

$$f_T(n) = \max\{m \mid T \text{ зупиняється після } m \text{ кроків на вході } w \text{ з } |w| = n\}.$$

При зупинці одержують відповідь „так" чи „ні". Ми вважаємо для спрощення, що *T* зупиняється, тобто досягає заключного стану, для всіх входів. Зрозуміло, що це не так по відношенню до довільної машини Тьюрінга. Задача належить класу *P*, якщо існує така машина Тьюрінга *T*, що $f_T(n) \le p(n)$ для всіх n.

Інший важливий клас *NP* (Nondeterministic Polynomial) формально описується як клас задач, які є поліноміально розв'язними на недетермінованому обчислювальному пристрої. *Недетермінована машина Тьюрінга* має декілька можливостей для своєї поведінки при зчитуванні букви. Відповідно вхідне слово призводить до декількох обчислень. Це можна собі уявити так, що машина робить здогади про розв'язок задачі або використовує довільну кількість паралельних процесорів. Для кожного входу w розглядається найкоротше *успішне обчислення* $S(w)$, тобто обчислення, яке приводить до заключного стану з відповіддю „так". Функцію складності недетермінованої машини Тьюрінга T означують так

$$f_T(n) = \max\{1,\ m \mid \textit{в обчисленні } S(w) \textit{ } m \textit{ кроків для } w \textit{ з } |w| = n\}.$$

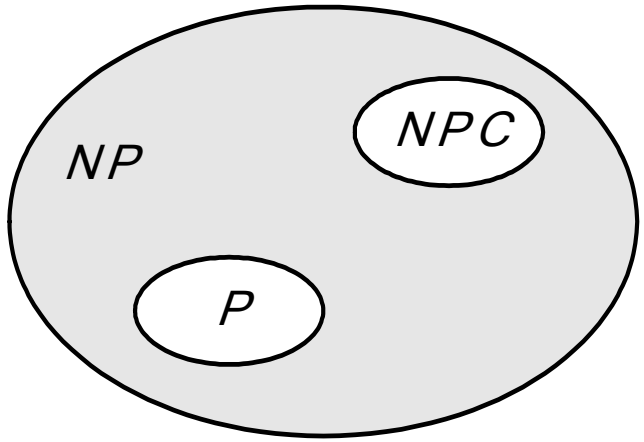
Розглядають пару $\{1, m\}$, бо для деяких n, можливо, взагалі немає таких входів із довжиною n, котрі приводять до заключного стану з відповіддю „так".

Задача належить класу *NP*, якщо існує така недетермінована машина Тьюрінга T, що $f_T(n) \le p(n)$ для всіх n.

Поняття поліноміальної розв'язності на недетермінованій машині можна неформально пояснити *поняттям можливості поліноміальної перевірки*: здогад про розв'язок задачі є можливість *перевірити* за допомогою *поліноміального* алгоритму.

Очевидно, що $P \subset NP$. Головне питання теорії складності алгоритмів полягає у доведенні або спростуванні гіпотези про те, що $P \neq NP$. Якщо ця гіпотеза справджується, то для розв'язання найбільш складних задач класу *NP* поліноміальних алгоритмів не існує. Нині широко розповсюджена думка, що $P \neq NP$, хоча доведення цієї гіпотези немає. Тим не менше, при сучасному рівні знань, мабуть, розумніше працювати за домовленості, що $P \neq NP$.

Задача вважається найбільш складною в класі *NP*, якщо до неї з поліноміальною складністю зводиться будь-яка задача з *NP*. Такі задачі називають *NP-повними*. Прикладами *NP*-повних задач є задача про наявність гамільтонового циклу в графі, задача комівояжера, задача про ізоморфний підграф, задача про існування в графі незалежної множини вершин, яка має потужність не меншу від заданої, задача про розфарбування графа в задану кількість кольорів та багато інших задач. Клас *NP*-повних задач будемо позначати як *NPC*. Гіпотетичне співвідношення між класами *P*, *NP* та *NPC* зображено на рисунку. Класи *P* та *NPC* містяться в *NP* (що очевидно), і, можна гадати, не перетинаються і не покривають всього *NP*.



Теорема Кука

Першою *NP*-повною задачею стала задача розпізнавання з логіки; її називають ВИКОНУВАНІСТЬ і формулюють так.

ВИКОНУВАНІСТЬ.

Умова. Дано множину булевих змінних X і формулу F від цих змінних у кон'юнктивній нормальній формі.

Запитання. Чи виконувана формула F, тобто, чи існує інтерпретація, у якій формула F набуває значення Т?

Теорема (Кук, 1971). Задача ВИКОНУВАНІСТЬ *NP*-повна.

Доведення цієї теореми доволі складне і ґрунтується на побудові недетермінованої машини Тьюрінга.

Приклади NP-повних задач

Якби всі доведення *NP*-повноти були так само складні, як доведення *NP*-повноти задачі ВИКОНУВАНІСТЬ , то дуже сумнівно, щоб список *NP*-повних задач міг би бути настільки великим, який він нині. Однак, якщо відома одна *NP*-повна задача, то процедура доведення *NP*-повноти інших задач сильно спрощується. Для доведення *NP*-повноти задачі $\Pi \in NP$ достатньо показати, що якусь із відомих *NP*-повних задач Π' можна звести до Π за поліноміальний час.

Отже, у подальшому процес доведення *NP*-повноти задачі розпізнавання буде складатись з таких кроків:

1) доведення того, що $\Pi \in NP$;

2) доведення, що деяка одна відома *NP*-повна задача Π' зводиться до Π за поліноміальний час.

Приклад 1. ІЗОМОРФНИЙ ПІДГРАФ.

У м о в а . Задані два графа $G_1=(V_1, E_1)$ та $G_2=(V_2, E_2)$.

П и т а н н я . Чи правильно, що G_1 містить підграф, який ізоморфний графу G_2?

Задачу розпізнавання, яка відповідає задачі комівояжера, можна сформулювати так.

Приклад 2. КОМІВОЯЖЕР.

У м о в а . Задано скінченну множину $C=\{c_1, c_2, \ldots, c_m\}$ міст, віддаль $d(c_i, c_j)\in N$ для кожної пари міст $c_i, c_j \in C$ і межу $B \in N$.

П и т а н н я . Чи існує шлях, який починається в місті $c_{s(1)}$, проходить через кожне місто точно один раз, повертається в $c_{s(1)}$ і довжина якого не перевищує B? Іншими словами, чи існує така перестановка $(c_{s(1)}, c_{s(2)}, \ldots, c_{s(m)})$ елементів множини C, що

$$\sum_{i=1}^{m-1} d\left(c_{s(i)}, c_{s(i+1)}\right) + d\left(c_{s(m)}, c_{s(1)}\right) \le B?$$

Останній приклад слугує також для ілюстрації важливого прийому – побудови задачі розпізнавання з відповідної оптимізаційної задачі.

Приклад 3. ГАМІЛЬТОНІВ ЦИКЛ.

У м о в а . Задано граф $G=(V, E)$.

П и т а н н я . Чи правильно, що G містить гамільтонів цикл?

Приклад 4. КЛІКА.

Умова. Дано граф $G=(V, E)$ та натуральне число k, $k \le |V|$.

Запитання. Чи містить граф кліку потужністю k, тобто таку підмножину $V' \subset V$, що $|V'|=k$ і будь-які дві вершини з V' з'єднані ребром з E?

Приклад 5. НЕЗАЛЕЖНІСТЬ.

Умова. Дано граф $G=(V, E)$ і натуральне число k, $k \le |V|$.

Запитання. Чи містить G незалежну множину потужністю k, тобто, чи існує підмножина $V' \subset V$, така, що $|V'|=k$ і жодні дві вершини з V' не з'єднані ребром з E?

Приклад 6. ВЕРШИННЕ ПОКРИТТЯ.

Умова. Дано граф $G=(V, E)$ і натуральне число k, $k \le |V|$.

Запитання. Чи існує в графі G вершинне покриття потужністю k, тобто така підмножина $V' \subset V$, що $|V'| = k$ і для кожного ребра $\{u, v\} \in E$ хоча б одна з вершин u чи v належить V'?

Приклад 7. ДОМІНАНТНА МНОЖИНА.

Умова. Дано граф $G=(V, E)$ і натуральне число k, $k \leq |V|$.

Запитання. Чи існує в графі G домінантна множина потужністю k, тобто така підмножина $V' \subset V$, що $|V'|=k$ і для всіх вершин $u \in V \setminus V'$ існує така вершина $v \in V'$, що $\{u, v\} \in E$?

Приклад 8. РОЗФАРБУВАННЯ ВЕРШИН ГРАФА.

Умова. Дано граф $G=(V, E)$ і натуральне число k, $k \leq |V|$.

Запитання. Чи існує розфарбування графа G в k кольорів?

Приклад 9. СУМА ЕЛЕМЕНТІВ ПІДМНОЖИНИ.

Умова. Дано скінченну множину натуральних чисел A і натуральне число k.

Запитання. Чи існує підмножина $A' \subset A$, сума елементів якої дорівнює k?

Є задачі, які належать до класу *NP*, для розв'язування яких до цього часу не знайдено поліноміальних алгоритмів, і невідомо, чи є вони *NP*-повними. Найбільш значуща з них – задача про ізоморфізм графів.

ІЗОМОРФІЗМ ПРОСТИХ ГРАФІВ.

Умова. Задано прості графи $G_1 = (V_1, E_1)$ і $G_2 = (V_2, E_2)$.

Запитання. Чи ізоморфні графи G_1 та G_2, тобто чи існує бієкція $\varphi: V_1 \to V_2$ така, що

$\{u, v\} \in E_1$ тоді й лише тоді, коли $\{\varphi(u), \varphi(v)\} \in E_2$, $\forall\, u, v \in V_1$?

Нарешті, відзначимо таку обставину. Багато задач, які зустрічаються на практиці, не є задачами розпізнавання, і, отже, не належать до класу *NP*. У той же час до багатьох із них удається звести деякі *NP*-повні задачі. У цій ситуації корисним виявляється таке означення.

Задачу називають *NP-важкою*, якщо до неї поліноміально зводиться деяка *NP*-повна задача.

Теорія NP-повноти окрім теоретичного, має й чисто практичне значення. Довівши, що задача NP-важка, ми одержуємо достатні причини, щоб відмовитись від пошуку ефективного й точного алгоритму. Подальші зусилля можуть бути спрямовані на одержання наближеного розв'язку.