NP-складність і NP-повнота. Приклади наближених алгоритмів для NP-повних задач.

Питання

- NP-складність задач.
- NP-повнота задач.
- Приклади наближених алгоритмів для NP-повних задач.

Задача пошуку (search problem)

- ***Задача пошуку (search problem)*** задається алгоритмом C, який отримує на вході умову I та кандидата на розв'язок S і має час роботи, обмежений поліномом від |I|.
- **S** називається **рішенням (solution)**, якщо C (S,I)=true.

Стосовно класів задач P, NP:

- NP – клас всіх задач пошуку.
- P – клас задач пошуку, рішення для яких може бути швидко знайдено (за поліноміальний час).

Теза Черча - Тьюринга

- Будь-яка обчислювана функція обчислюється машиною Тьюринга.

Поліноміальні алгоритми існують для багатьох задач.

- Багато важливих задач *для транспортних потоків* («Математика транспортних потоків») допускають поліноміальні алгоритми.
- *Алгоритми, пов'язані з Інтернетом* («Математика інтернету») є алгоритмами лише в широкому сенсі, так як самі задачі за своєю суттю динамічні і розподілені.
- *Всі алгоритми, що використовуються в криптографії* – поліноміальні.
- *Алгоритми, що використовуються для стиснення даних*, також є поліноміальними. Боротьба йде за поліпшення швидкості кодування і декодування всередині класу P - як і у випадку «великих даних», різниця між лінійними і, скажімо, квадратичними алгоритмами виявляється досить відчутною.

Задача «від прогулянок по Кенігсбергу до реконструкції геному».

- *Перша половина XVIII століття*. Великий математик Леонард Ейлер розв’язує «задачу про Кенігсбергські мости» - доводить, що в Кенігсберзі, розташованому на берегах річки і двох її островах, не можна було пройти по кожному з семи мостів, що існували в той час, рівно один раз і повернутися після цього у вихідну точку. Подібний шлях на відповідному графі називається ейлеровим циклом. У задачі про існування ейлерова циклу критерій можливості розв'язання дуже простий - з кожної вершини графа має виходити парне число ребер. Та й задача знаходження ейлерова циклу (ЗЕЦ) вирішується відносно швидко навіть для дуже великого графа.
- *Друга половина XIX століття*. Математик Вільям Гамільтон розглядає зовні схожу на ЗЕЦ задачу: знайти на графі замкнутий шлях (гамільтонів цикл), що проходить через кожну вершину по одному разу (ЗГЦ).
- *Друга половина XX століття.* Було встановлено, що ЗГЦ (на відміну від ЗЕЦ) є представником класу задач, для яких ефективні алгоритми рішення невідомі.
- *Кінець XX століття - XXI століття*. В середині 1990-х років був секвенірован геном бактерії, в 2001 році - людини. Робота була тривалою і дорогою, оскільки алгоритми суперкомп'ютерів грунтувалися на ЗГЦ. В останнє десятиліття математиками були розроблені швидкодіючі методи збирання, пов'язані з ЗЕЦ, і тепер біологи готуються до вирішення фундаментальної задачі: для кожного виду ссавців провести збірку генома.

- Твердження: $\mathrm{P} \subseteq \mathrm{PC}$. Доказ: оскільки детерміновані машини Тьюринга є частковим випадком не детермінованих.

- Отже, до детермінованих класів складності відноситься клас P – це множина мов, що розпізнаються за поліноміальний час. Формально він визначається так:

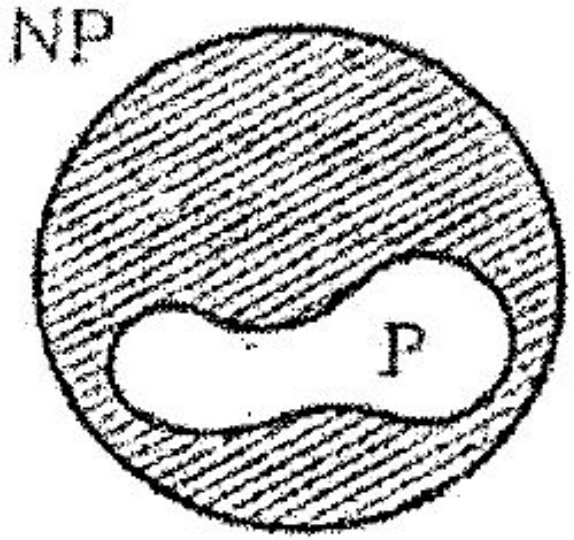
$$\mathbf{P} = \cup_{c=1}^{\infty} \mathbf{DTIME}(n^c).$$

- Існують більш високі класи. Крім класу P вивчають інші класи, які визначаються порядком зростання часу роботи на машині Тьюринга:
  - $\mathrm{QP} = \mathrm{DTIME}(2^{\mathrm{polylog}(n)}) = \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{\log^c n})$ (квазиполиномиальное время);
  - $\mathrm{SUBEXP} = \mathrm{DTIME}(2^{n^{o(1)}}) = \bigcap_{\varepsilon>0} \mathrm{DTIME}(2^{n^{\varepsilon}})$ (субэкспоненциальное время);
  - $\mathrm{E} = \mathrm{DTIME}(2^{O(n)}) = \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{cn})$ (линейно-экспоненциальное время);
  - $\mathrm{EXP} = \mathrm{DTIME}(2^{\mathrm{poly}(n)}) = \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{n^c})$ (экспоненциальное время, иногда его также называют EXPTIME);
  - $\mathrm{EE} = \mathrm{DTIME}(2^{2^{O(n)}}) = \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{2^{cn}})$ (дважды линейно-экспоненциальное время);
  - $\mathrm{EEXP} = \mathrm{DTIME}(2^{2^{\mathrm{poly}(n)}}) = \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{2^{n^c}})$ (дважды экспоненциальное время);
  - и т.д.

- До класу NP відносяться недетерміновані машини Тьюрінга. Формально недетерміновані обчислення проводяться на недетермінованій машині Тьюринга.

- Клас NP є класом всіх задач розпізнавання, які можуть бути вирішені недетермінованими алгоритмами за поліноміальний час.

NP-повнота



Якщо P не збігається з NP, то відмінність між P і NP\P дуже істотна. Всі задачі з P можуть бути вирішені поліноміальними алгоритмами, а всі задачі з NP\P складнорозв’язувані. Тому, якщо P≠NP, то для кожної конкретної задачі П∈NP важливо знати, яка з двох можливостей реалізується.

Поняття поліноміальної зведеності

Основна ідея умовного підходу заснована на понятті поліноміальної зведеності.

Будемо говорити, що має місце поліноміальна зведеність мови $L_1 \subseteq \sum^*_1$ до мови $L_2 \subseteq \sum^*_2$, якщо існує функція f: $\rightarrow$, що задовольняє, двом умовам:

1. Існує ДМТ-програма, що обчислює f p тимчасової складністю, обмеженою поліномом.

2. Для будь-якого $x \in \sum^*_1$, $x \in L_1$ в тому і тільки в тому випадку, якщо $f(x) \in L_2$.

Якщо L_1 поліноміально зводиться до L_2, то будемо писати $\mathbf{L_1 \propto L_2}$

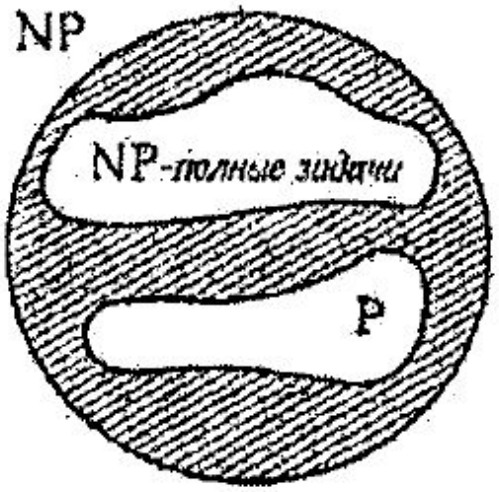
- **Лемма 1.** Якщо $L_1 \propto L_2$, то з $L_2 \in P$ слідує, що $L_1 \in P$. (Еквівалентне твердження: з L_1 P слід, що L_2 P).

- Якщо $П_1$ і $П_2$ - задачі розпізнавання, а e_1 і e_2 - їх схеми кодування, то будемо писати $П_1 \propto П_2$ (щодо заданих схем кодування), якщо існує поліноміальна зведеність мови $L[П_1, e_1]$ к $L[П_2, e_2]$.

- Таким чином, на рівні задач поліноміальної зведеності задачі розпізнавання $П_1$ до задачі розпізнавання $П_2$ означає наявність функції $f: D_{n1} \rightarrow D_{n2}$, що задовольняє двом умовам:
- (1) f обчислюється поліноміальним алгоритмом;
- (2) для всіх $I \in D_n$, $I \in Y_{n1}$, тоді і тільки тоді, коли $f(I) \in Y_{n2}$

- **Лемма 2.** Якщо $L_1 \propto L_2$ та $L_2 \propto L_3$, $L_1 \propto L_3$.

- Лемма 2 стверджує, що це відношення є відношенням еквівалентності, а також, що відношення $\propto$ визначає часткове впорядкування класів еквівалентності мов, що виникають (задач розпізнавання).



- Мова L називається ***NP-повною***, якщо L∈NP і будь-яка інша мова L'∈NP зводиться до мови L.

- **Радача розпізнавання П називається NP-повної**, якщо П∈NP і будь-яка інша задача розпізнавання П∈NP зводиться до П.

- Якщо хоча б одна NP-повна задача може бути вирішена за поліноміальний час, то і всі задачі з NP також можуть бути вирішені за поліноміальний час.

- Якщо хоча б одна задача з NP складнорозв’язувана, то і все NP-повні задачі складнорозв’язувані.

- Будь-яка NP-повна задача П має властивість: якщо P≠NP, то П∈NP\P. Точніше, П∈P тоді і тільки тоді, коли P=NP.

- Зауважимо, що NP не просто поділяється на дві області: клас P і клас NP-повних задач. Якщо P відрізняється від NP, то повинні існувати задачі з NP, нерозв'язні за поліноміальний час і не є NP-повними.

Приклади наближених алгоритмів для NP-повних задач

- Алгоритм, який повертає рішення, близькі до оптимальних, називається ***наближеним алгоритмом.***

Методи розв'язання NP-повних задач

- **Наближені та евристичні методи** – застосування евристик для вибору елементів рішення.
- **Псевдополіноміальні алгоритми** – підклас динамічного програмування.
- **Метод локальних покращень** – пошук більш оптимального рішення в околиці деякого поточного рішення.
- **Метод гілок і меж** – відкидання свідомо неоптимальних рішень цілими класами відповідно до деякої оцінки.
- **Метод випадкового пошуку** – представлення вибору послідовністю випадкових виборів.

Метод локальних покращень

- Розпочати з довільного рішення.
- Для покращення поточного рішення застосувати до нього будь-яке перетворення із заданої сукупності перетворень. Це покращене рішення стає поточним рішенням.
- Повторити зазначену процедуру до тих пір, поки жодне з перетворень із заданої сукупності не дозволить поліпшити поточне рішення.

- Якщо задана сукупність перетворень включає всі перетворення, то ми отримаємо точне (**глобально-оптимальне**) рішення.
- На практиці сукупність перетворень обмежують. За допомогою них з ряду довільних рішень отримують **локально-оптимальні** рішення і вибирають з них

Метод гілок та меж

- Вирішуючи дискретну екстремальну задачу, розіб'ємо множину всіх можливих варіантів на класи і побудуємо оцінки для них.
- В результаті стає можливим відкидати рішення цілими класами, якщо їх оцінка гірше деякого рекордного значення.

Дискретна екстремальна (на мінімум) задача в загальному вигляді:

- Нехай задано дискретну множину A і визначено на ньому певна функція f. Позначимо мінімум функції f на X як F(X).

- Потрібно знайти $x_0 \in A$: $f(x_0)=F(A)$.

Алгоритм методу:

- Розіб'ємо множину A на підмножини A_i і на кожному з них знайдемо нижню оцінку Ф.
- Для елементів множини A будемо обчислювати значення функції f і запам'ятовувати найменше в якості рекордного значення f *.
- Все підмножини, у яких оцінка вище f *, об'єднаємо в підмножину A_0, щоб в подальшому не розглядати.
- Оберемо будь-яке з множин A_i, i> 0. Розіб'ємо цю множину на більш дрібні підмножини. При цьому ми будемо продовжувати покращувати рекордне значення f *.
- Цей процес триває до тих пір, поки не будуть переглянуті всі множини A_i, i> 0.

Метод випадкового пошуку

- Зазвичай вибір рішення можна уявити послідовністю виборів.
- Якщо робити ці вибори за допомогою будь-якого *випадкового механізму*, то рішення знаходиться дуже швидко, так що можна знаходити рішення багаторазово і запам'ятовувати "рекорд", тобто найкраще з тих рішень, що зустрічалися.
- Цей наївний підхід суттєво покращується, коли вдається врахувати у випадковому механізмі перспективність тих чи інших виборів, тобто комбінувати випадковий пошук з еврестичним методом і методом локального пошуку.
- Такі методи застосовуються, наприклад, при складанні розкладів літаків в аеропорті.