Модуль 3. NP-трудные задачи

Лекция 12

# NP-трудные задачи. Изоморфизм.

# Сложность задачи

Для заданной задачи могут существовать алгоритмы разной сложности.

? Сложность задачи = сложность самого быстрого алгоритма, решающего эту задачу.

Теорема Блюма об ускорении: существует задача, для которой любой решающий её алгоритм можно экспоненциально ускорить.

# Классы сложности

***Класс сложности*** = множество задач, для каждой из которых существует решающий её алгоритм, имеющий указанную сложность.

P = множество задач, решаемых за полиномиальное время.

К сожалению, для многих практически важных задач пока не известны полиномиальные алгоритмы. Это те самые ***вычислительно сложные задачи***.

# NP-трудные задачи

Класс NP (Non-deterministic Polynomial)

- Распознавательные задачи, решаемые недетерминированным алгоритмом за полиномиальное время.
- Задачи, для которых решение может быть проверено (детерминированным алгоритмом) за полиномиальное время при наличии *сертификата*.

# NP-трудные задачи

## Полиномиальная сводимость

Задача $R$ полиномиально сводится к задаче $Q$ <=> существует алгоритм $A_R$, обращающийся к алгоритму $A_Q$ (алгоритм для $Q$), и решающий задачу $R$ за полиномиальное время <u>без учёта времени работы</u> $A_Q$.

# NP-трудные задачи

## NP-трудные задачи

– Задача называется NP-трудной, если к ней полиномиально сводится любая задача $Q \in NP$.

– Задача NP-полна, если она NP-трудна и принадлежит классу NP.

# NP-трудные задачи

- $P \subseteq NP$. $P = NP$ ????
- Все NP-полные задачи полиномиально эквивалентны, т. е. полиномиально сводятся друг к другу.
- Ни для одной NP-полной задачи не известен полиномиальный алгоритм. И маловероятно, что будет обнаружен в обозримом будущем.

# Что делать?

1) Решать долго (за экспоненциальное время)
2) Пытаться сократить время решения для входов, ожидаемых на практике
   – Полиномиальные в среднем алгоритмы
   – Параметризованные алгоритмы
3) Решать приближённо
   – С гарантированной *оценкой точности*
   – Без гарантий, но как правило достаточно точно
   – С некоторой *вероятностью* ошибки

# Метод полного перебора

**Полный перебор (Brute Force)**

- Последовательно генерировать все возможные решения
- Для каждого сгенерированного решения x выполнять проверку на допустимость / оптимальность: *Process*(x)

# Генерация перестановок

Задача: для заданного $n$ сгенерировать и обработать все перестановки степени $n$.

Решение:

1. Храним в массиве $A[1..n]$.
2. Инициализация: $\forall\, i \quad A[i] := i$.
3. Для всех $k$ последовательно переставляем $A[k]$ с элементами в позициях $1,\ldots, k-1$.

# Генерация перестановок

Вызов: ProcessPermutations(*A*,*n*)

<u>ProcessPermutations(*A*,*k*)</u>

if *k* = 1 then Process(*A*)

else

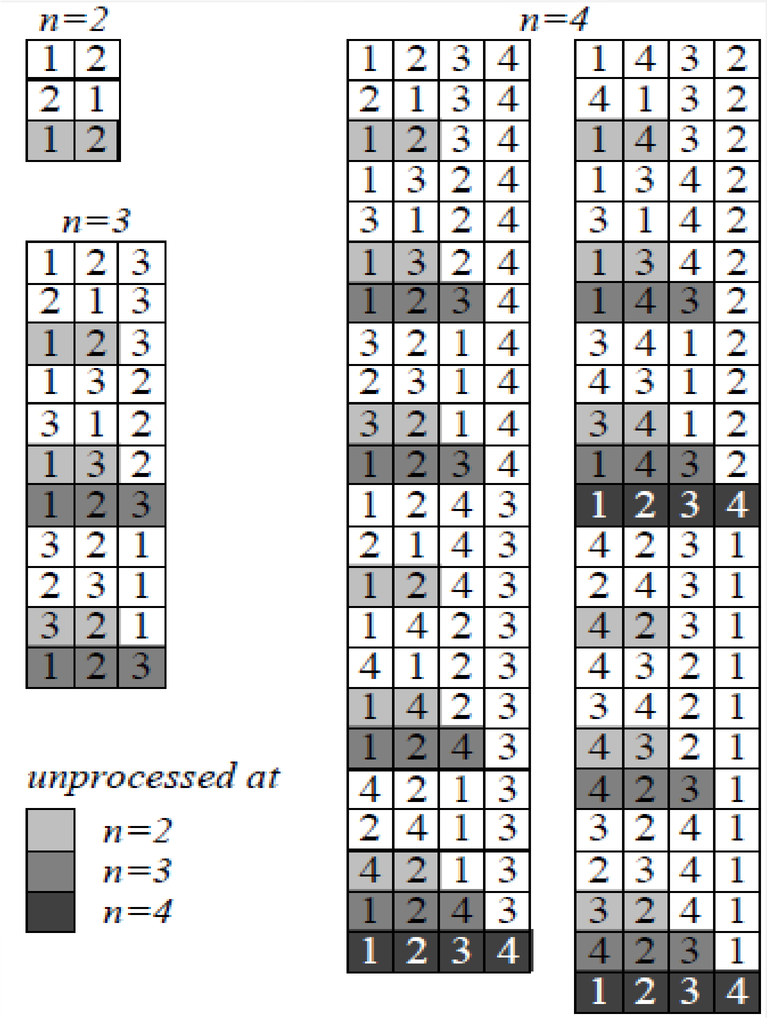
    ProcessPermutations(*A*, *k*-1);

    for *i* = *k*-1 downto 1 do

        Поменять A[*k*] и A[*i*]

        ProcessPermutations(*A*, *k*-1);

        Поменять A[*k*] и A[*i*]

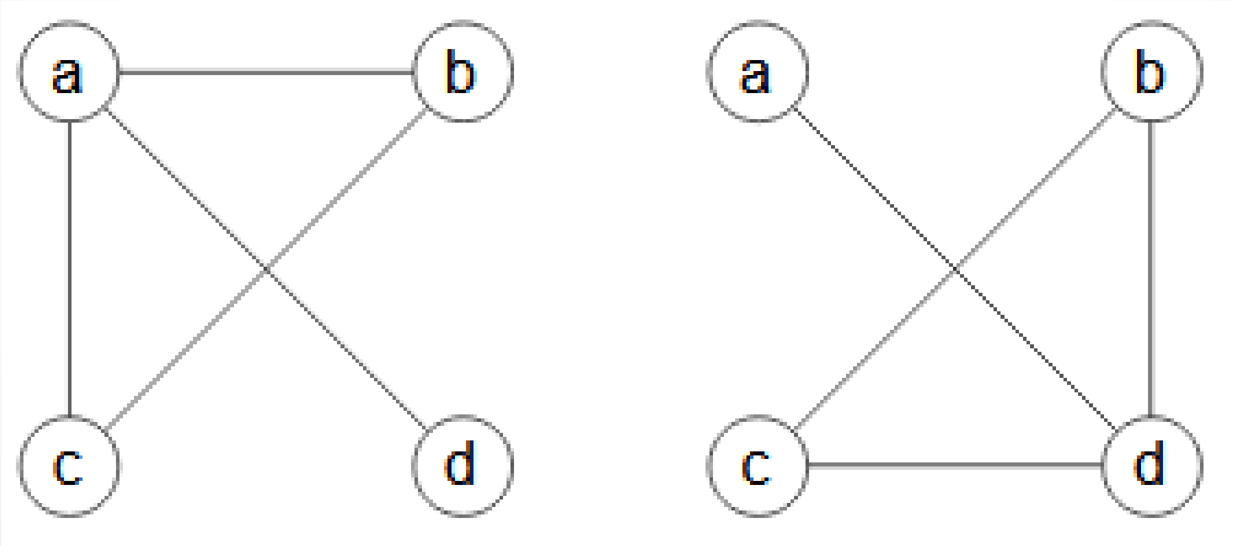


# Изоморфизм графов

# Изоморфизм графов

**<u>Определение.</u>** Графы $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ *равны*, если их множества вершин и множества рёбер совпадают: $V_1 = V_2, E_1 = E_2$.

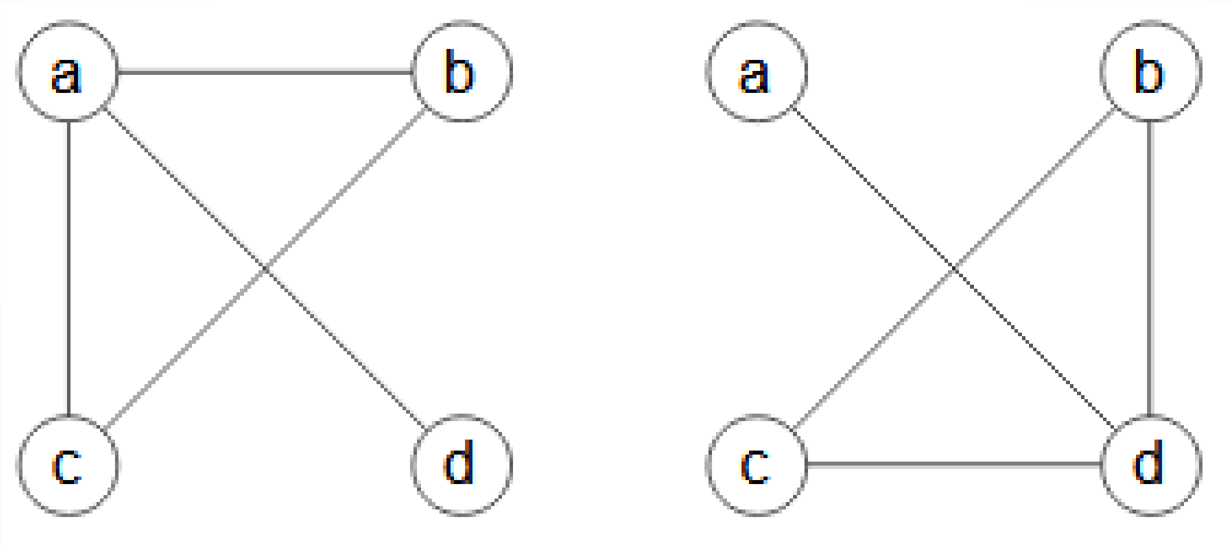
Равенство графов можно определить по матрице смежности.



# Изоморфизм графов

Более «мягкое» понятие сходства графов отражается отношением *изоморфизма* графов.

**Определение.** Графы $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ *изоморфны*, если существует взаимно однозначное отображение $\varphi: V_1 \to V_2$, такое, что $(u, v) \in E_1 \Leftrightarrow \big(\varphi(u), \varphi(v)\big) \in E_2$.



# Изоморфизм графов

Задача изоморфизма графов:

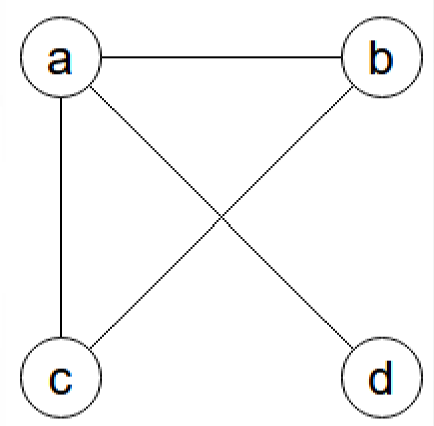
- Дано: два графа $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$.
- Вернуть: изоморфны ли графы $G_1$и $G_2$?

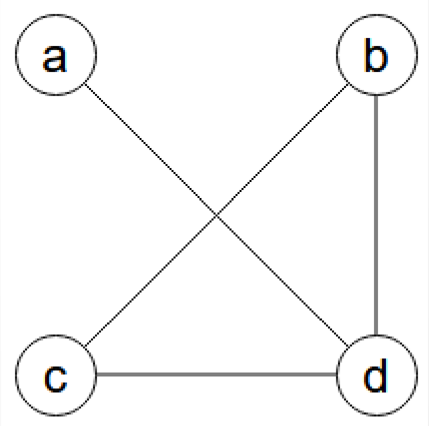
То есть, это распознавательная задача.

Стандартное обозначение изоморфизма графов: $G_1 \cong G_2$.

# Изоморфизм графов

Взаимно однозначное отображение конечного множества – это аналог перестановки. Поэтому проверить изоморфизм можно полным перебором всех перестановок из $n(= |V_1|)$ элементов.




| $v$ | $\varphi(v)$ |
|---|---|
| a | d |
| b | b |
| c | c |
| d | a |

# Изоморфизм графов

Сложность переборного алгоритма: $O(n!)$.
По формуле Стирлинга

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

получаем временную сложность переборного алгоритма: $O(n^n)$.

На текущий момент, для задачи проверки изоморфизма графов не построен полиномиальный алгоритм, но и не доказана её NP-полнота.

# Изоморфизм подграфов

Задача изоморфизма *подграфов*:

- Дано: два графа $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$, $|V_1| > |V_2|$.
- Вернуть:
    1) Распознавательный вариант: содержит ли $G_1$ подграф $H$, изоморфный ли графу $G_2$ ($H \cong G_2$)?
    2) Вычислительный вариант: найти в $G_1$ подграф $H$, изоморфный ли графу $G_2$ ($H \cong G_2$), или сообщить, что такого подграфа нет.

Задача изоморфизма подграфов является NP-трудной.