УДК 517.174

Чичканъ С. М.

АЛГОРИТМ ПЕРЕВІРКИ ГРАФІВ НА ІЗОМОРФНІСТЬ З ВИКОРИСТАННЯМ ЛОКАЛЬНИХ ІНВАРІАНТІВ

Одним із способів зниження часової складності перевірних задач є впорядкування множини вхідних даних, що дає змогу ідентифікувати їх незалежно від порядку зберігання чи переліку. Для цього кожному елементу множини приписується чисельна характеристика (вага). Якщо метою перебору є встановлення еквівалентності двох множин і всім елементам множини можна приписати різну вагу, швидке сортування обох множин за час $O(N\log_2 N)$ практично розв 'язує задачу, оскільки після впорядкування залишається за час $0(N)$ перевірити еквівалентність упорядкованих масивів, які містять елементи множини. У цій роботі наведено алгоритм перевірки графів на ізоморфність, що базується на впорядкуванні вершинних інваріантів локального порядку.

Вступ

Графом $G(E,V)$ називається об'єднання множини вершин $E = \{1,2,...,i,...,N\}$ і множини ребер, тобто упорядкованих пар вершин $V = \{(i,j)\}$ [1]. Як і в кожній множині, нумерація вершин не має значення і використовується з метою ідентифікації ребер та спрощення виконання операцій над графами. Тому кожна перестановка номерів вершин задає взаємно однозначне відображення ізоморфізму G у G, якщо одночасно з переходом до нової нумерації заміна номерів вершин також проводиться і в множині ребер графа. Так само несуттєвим є порядок переліку ребер у множині ребер. Тому граф $G(E,V)$, в якому відношення між номерами вершин і номерами ребер не мають принципового значення, можна розглядати як клас еквівалентності $G(E,V) = \{G(E_i,V_j)\}$, де індекси i і j незалежно пробігають усі можливі перестановки номерів вершин та ребер. Цей спосіб цілком аналогічний побудові поняття геометричної фігури, що не розрізняє конгруентні фігури між собою. Кожен такий клас містить $P < N!L!$ графів, де N і L – загальна кількість вершин і ребер, оскільки серед незалежних перестановок номерів ребер і номерів вершин можуть зустрічатись такі, які одночасно переставляють номери вершин і номери суміжних їм ребер таким чином, що номери нових вершин і номери суміжних ребер знову відповідають одні одним.

Проблема ізоморфізму виникає в різноманітних обчислювальних задачах, де вихідні графи задані у вигляді упорядкованих множин вершин і ребер, є проміжними результатами обчислень, або виникають в результаті розбиття чи композиції інших графів.

Локальні та глобальні інваріанти

Як було показано в роботах [2, 3], для графів загального вигляду не існує алгоритму встановлення ізоморфізму в термінах теорії автоматів, істотно швидшого, ніж простий перебір усіх перестановок вершин. У загальному вигляді алгоритм перевірки графів на ізоморфність описано у [7]. Для окремих класів графів існують методи приведення повної перебірної задачі до задачі з поліноміальною часовою складністю [8]. Деякі з цих методів базуються на використанні локальних інваріантів – величин, що визначаються локальним оточенням кожної вершини та глобальних інваріантів – величин, що характеризують граф у цілому. Якщо два графи ізоморфні, їх множини інваріантів збігаються.

Локальні інваріанти характеризують локальну структуру графа в невеликій околиці кожної вершини. Різні множини локальних інваріантів з необхідністю свідчать про те, що графи не ізоморфні. Для двох випадково вибраних графів імовірність збігу множин інваріантів досить низька, і хоча застосування цієї необхідної умови не відміняє перебір усіх перестановок вершин, при послідовній перевірці ізоморфності різних пар графів середній час виконання завдання суттєво зменшується.

Найпростішими інваріантами є загальна кількість вершин і загальна кількість ребер у графах [4, с. 25]. Якщо кількість вершин або кількість ребер не збігаються, то графи не ізоморфні.

Використання складніших інваріантів вимагає представлення графа у вигляді якої-небудь структури даних, придатної для зберігання в ОП комп'ютера. У цій роботі використовується матриця суміжності вершин $A \in M_{n,n}$: $a_{ij} = \{1$, якщо існує

ребро (i,j), 0 в інших випадках}. Така матриця не дає можливості відрізнити кратні ребра від однократних, а також ланку графа, тобто невпорядковане ребро (i,j), від пари орієнтованих ребер (i,j) та (j,i), оскільки в обох випадках $a_{ij} = a_{ji} = 1$. Тому надалі розглядатимуться орграфи, в яких ланки не відрізняються від пари спряжених ребер.

Матриця суміжності A орієнтованого графа загального вигляду несиметрична, тобто $A^T \neq A$, і не обов'язково нормальна, тобто $A^T A \neq AA^T$. Два ізоморфних графи G і H мають подібні матриці суміжності $A(G)$, $A(H)$, що відрізняються порядком нумерації вершин, тобто

$$A(G) = S\,A(H)\,S^T \qquad (1)$$

для певної матриці перестановок номерів вершин S, яка перенумеровує вершини G таким чином, що після цього множина ребер G стає еквівалентною множині ребер H. Інваріантами будуть будь-які величини, які зберігаються після перестановки номерів вершин.

Оскільки матриці перестановок ортогональні, тобто $SS^T = I$ (одиничній матриці), будь-які величини, обчислені за правилами матричного множення матриці суміжності самої на себе, будуть інваріантними з точністю до перестановки індексів. Скалярні інваріанти не залежать від індексів і тому тотожно рівні для ізоморфних графів і мають глобальний характер. Тому будь-яке слово виду

$$Tr(W(A, A^T)) =$$
$$= Tr(A^{m_1}(A^T)^{n_1} A^{m_2}(A^T)^{n_2} \ldots A^{m_k}(A^T)^{n_k}), m_k, n_k \geq 0 \quad (2)$$

задає скалярний інваріант.

Як легко пересвідчитись підстановкою, при заміні $A(G)$ на $SA(H)S^{-1}$, де S – матриця перестановки номерів вершин, що переводить матрицю $A(H)$ у $A(G)$, вираз типу (2) для графа G переходить у вираз для графа H. Кількість різних інваріантів типу (2) визначається кількістю незалежних виразів з різними наборами ступенів $\{m_i, n_i\}$. Якщо обмежити обчислення загальним ступенем $K = \Sigma m_i + \Sigma n_i$, кількість різних слів виду (2) визначатиметься кількістю способів розкладу числа K на доданки, тобто в практичному плані обчислення добутків матриць для великих K мало чим відрізняється за часовою складністю від перебору перестановок вершин.

Більший інтерес становлять векторні локальні інваріанти, оскільки їх упорядкування можливе внаслідок швидких методів сортування. Такі інваріанти можна розбити на інваріанти 0, 1, 2 та вищих порядків залежно від кількості матриць суміжності у добутку. Таким чином, інваріант порядку 0 – це діагональна частина матриці суміжності $D_0 = \mathrm{Diag}(A)$. Вектор D_0 містить кількість петель при вершині, тобто тотожно рівний 0 для орграфів без петель.

Інваріант другого порядку можна утворити кількома способами, оскільки несиметрична матриця A не комутує з транспонованою. Таким чином можна утворити 3 вектори:

$$\begin{aligned} D_{11} &= \mathrm{Diag}(AA) = (AA)_{ii} = a_{ij}a_{ji} = a_{ji}a_{ij} = \\ &= \mathrm{Diag}\left(A^T A^T\right), \\ D_{12} &= \mathrm{Diag}\left(AA^T\right)() = \left(AA^T\right)_{ii} = a_{ij}a_{ij}, \\ D_{13} &= \mathrm{Diag}\left(A^T A\right) = \left(A^T A\right)_{ii} = a_{ji}a_{ji}. \end{aligned} \qquad (3)$$

Покажемо, що значення таких векторних інваріантів не змінюються в результаті перестановок номерів вершин. Нехай перестановка S перенумеровує вершини графа G таким чином, що нумерації G і H збігаються: $A(G) = SA(H)S^T$. Тоді, наприклад, $D_{11}(H) = \mathrm{Diag}(A(H)A^T(H)) = \mathrm{Diag}(S^T A(G)SS^T A^T(G)S) = \mathrm{Diag}(S^T A(G)A^T(G)S) = s_{Ji}a_{Jk}a_{kl}s_{li} = s_{Ji}a_{Jk}a_{kL}s_{Li} = s_{Ji}a_{Jk}a_{kJ}s_{Ji} = S^T\mathrm{Diag}(a_{Jk}a_{kJ})$. В останніх рівностях підсумовування за індексами j, l замінене значенням елементів A для індексів, що відповідають єдиному ненульовому елементу s_{Ji} в i-му стовпчику матриці перестановок S.

В результаті вектор $D(H) = S^T D(G)$.

Вектори D_{12} і D_{13} різні, оскільки в загальному випадку матриця A не є нормальною. Як нескладно перевірити, елементи D_{11} дорівнюють валентностям відповідних вершин, тобто кількостям пар ребер виду $(i,j),(j,i)$ для заданої вершини i. Елементи вектора D_{12} дорівнюють напівступеням виходу вершин, а елементи вектора D_{13} – напівступеням входу, тобто рядковим і стовпчиковим сумам елементів A. Таким чином, для підрахунку векторів D_{12} і D_{13} не потрібно множити матриці, достатньо виконати N операцій підсумовування по N вершинах.

Додатково можна використати ще 3 інваріанти 1-го порядку, які дають кількість різних шляхів довжини 2, що виходять з вершини, заходять у вершину або проходять через вершину:

$$\begin{aligned} (D_{14})_i &= \sum_{j,k} a_{ij} \cdot a_{jk}, (D_{15})_i = \\ &= \sum_{j,k} a_{kj} \cdot a_{ji}, (D_{16})_i = \sum_{j,k} a_{ki} \cdot a_{ij}. \end{aligned} \qquad (4)$$

Обчислення інваріантів 1-го порядку вимагає не більше $O(N^2)$ операцій.

Існує 4 інваріанти 2-го порядку виду (3), які відповідають кількості орієнтованих трикутників з вершинами в даній вершині та комбінаціям пар ребер (ланок) і орієнтованих ребер або

орієнтованих ребер і шляхів довжини 2, що виходять із заданої вершини:

$$D_{21} = \mathrm{Diag}(AAA) = \mathrm{Diag}(A^T A^T A^T) = a_{ij}a_{jk}a_{ki},$$
$$D_{22} = \mathrm{Diag}(AAA^T) = \mathrm{Diag}(AA^T A^T) = a_{ij}a_{kj}a_{ik},$$
$$D_{23} = \mathrm{Diag}(AA^T A) = \mathrm{Diag}(A^T AA^T) = a_{ji}a_{jk}a_{ki}, \quad (5)$$
$$D_{24} = \mathrm{Diag}(A^T AA) = \mathrm{Diag}(A^T A^T A) = a_{ji}a_{kj}a_{ki}.$$

Так само, як і у випадку інваріантів 1-го порядку, набір можна поповнити додатковими векторами, що містять кількості шляхів довжини 3, пов'язаних з даною вершиною. Обчислення інваріантів другого порядку вимагає не більше ніж $O(N^3)$ операцій. Аналогічно можна отримати 10 інваріантів 3-го порядку типу (3), (5), елементи яких відповідатимуть кількостям «квадратів» різних комбінацій пар ланок або ланок і пар орієнтованих ребер, та додаткові вектори, елементи яких містять кількості шляхів довжини 4, які проходять через задану вершину, включаючи комбінації пар шляхів сумарної довжини 4.

Обчислення локальних інваріантів порядку k вимагає не більше $O(N^{k+1})$ операцій і так само, як і для скалярних інваріантів глобального порядку типу (2), для великих k і великих графів не становить значного інтересу.

У загальному випадку діагональні елементи матриці A^k дорівнюють кількості орієнтованих циклів довжини k, включаючи цикли з самоперетинами. Оскільки максимальна довжина C простого циклу не перевищує n, а циклу із самоперетинами $C \sim n^2$, принципово достатньо проаналізувати локальні інваріанти від 0 до $n-1$ порядку в неорієнтованому графі і від 0 до $\sim n^2$ – в орієнтованому. Точніша оцінка на основі теореми Персі дана в [5]. Таким чином, використання всіх можливих локальних інваріантів потребує порядку $O(N^N) - O(N^{N^2})$ операцій і за часовою складністю не відрізняється від перебору всіх перестановок вершин.

Опис алгоритму

Таким чином, у результаті обчислення певної кількості локальних інваріантів ми отримуємо систему векторів $D = \{D_{11}, D_{12}, \ldots, D_{21}, D_{22}, \ldots\}$, елементи яких є локальними інваріантами і характеризують локальне оточення кожної вершини. В результаті перестановки S номерів вершин $A_1 = S^T AS$ значення елементів D також переставляються: $D_1 = S^T D$. Отже, впорядкування векторів локальних інваріантів за допомогою сортування із застосуванням індексного масиву генерує в елементах індексного масиву перестановку номерів вершин масиву локального інваріанта, яка відповідає такій перестановці номерів вершин графа, що вершини впорядковуються відповідно до порядку, встановленого в масиві D. Якщо значення локального інваріанта різні для різних вершин, таке впорядкування розв'язує проблему ідентифікації ізоморфних графів.

Справді, достатньо побудувати масиви $D(G)$ і $D(H)$ для обох графів G і H і упорядкувати їх значення з використанням індексних масивів. Тоді індексні масиви міститимуть 2 перестановки $S(G)$ і $S(H)$. Якщо сортовані вектори D різні, графи не ізоморфні. Якщо вони однакові, перестановки $S(G)$ і $S(H)$ задають необхідну перенумерацію вершин графів таким чином, що коли перенумеровані матриці суміжності $S(G)^T A(G)S(G)$ і $S(H)^T A(H)S(H)$ однакові, графи ізоморфні. Якщо матриці суміжності не однакові, то графи не ізоморфні, оскільки не існує іншої перестановки вершин, яка б зберегла значення локальних інваріантів та їх порядок.

На жаль, графи з усіма різними значеннями локальних інваріантів D_{11} не існують. Справді, в будь-якому графі з N вершинами з кожної вершини виходить не більше $N-1$ ребер. Тому цілочисельний масив D, що містить N різних елементів з максимальним значенням $N-1$, повинен містити недодатні елементи. У кращому випадку мінімальний елемент дорівнює 0, що відповідає ізольованій вершині і максимальній кількості щонайбільше $N-2$ ребер, що виходять з однієї вершини більшої компоненти зв'язності. З індукції очевидно, що подібний граф нульовий.

Таким чином, локальні інваріанти 1-го порядку з необхідністю містять кратні елементи. Нехай $\{n_1, n_2, \ldots, n_m\}$ – упорядкована система m груп різних значень D, кожна з яких містить n_i кратних значень. Після сортування масиву D таке розбиття відповідає послідовному розбиттю індексів масиву номерів вершин (індексів вершин) на інтервали $\{[1..n_1], [(n_1+1)..(n_1+n_2)], \ldots, [(N-n_m)..N]\}$. Як і раніше, якщо сортовані масиви $S(D)^T D(G)$ і $S^T D(H)$ різні, графи не ізоморфні. Якщо вони однакові, то модифіковані матриці суміжності $A_1(G) = S^T(G)A(G)S(G)$ і $A_1(H) = S^T(H)A(H)S(H)$ ізоморфних графів G і H можна перевести одна в одну шляхом додаткової перестановки $A_1(G) = S_1^T A_1(H)S_1$. Оскільки при цьому локальні інваріанти для вершин зберігаються, генеруючи різні перестановки S_1, достатньо переставляти індекси тільки всередині кожної групи.

Загальна кількість перестановок при цьому становитиме $P = n_1!n_2!n_3!\ldots n_m!$, де $\Sigma n_i = N$, тобто в загальному випадку $P \ll N!$ – кількості перестановок, які потрібно було б генерувати без попереднього впорядкування вершин графів.

Оскільки різні локальні інваріанти дають різні і в більшості випадків лінійно незалежні характеристики вершин, одночасне використання кількох локальних інваріантів збільшує подрібнення N на групи, в кожній з яких знаходяться вершини з однаковими значеннями кількох інваріантів зразу. Ідею впорядкування вершин за значеннями кількох параметрів було висловлено в [9]. Очевидно, що коли такі параметри лінійно незалежні, набору з N параметрів для кожної вершини достатньо для побудови канонічного упорядкування, яке зберігається після перестановки вершин і однозначно характеризує ізоморфні графи. У нашому випадку ми обмежуємось відносно невеликим $m < N$.

Збільшення m забезпечує зменшення (в середньому) величин n_i і таким чином значно скорочує необхідну кількість перестановок. Використання кількох інваріантів одразу незначно ускладнює алгоритм, оскільки більшість стандартних процедур сортування має параметр-функцію, яка задає бажаний користувачеві спосіб порівняння елементів. Тому достатньо модифікувати функцію порівняння, що використовує значення одного масиву D, на функцію, яка використовує кілька таких масивів. Алгоритм упорядкування вершин за неспаданням для I масивів локальних інваріантів D_i буде такий [9]:

Вершина i менша за вершину j,
якщо $D_1(i) < D_1(j)$,
або $D_1(i) = D_1(j)$ та $D_2(i) < D_2(j)$,
або $D_1(i) = D_1(j)$, $D_2(i) = D_2(j)$ та $D_2(i) < D_3(j)$,
або ...

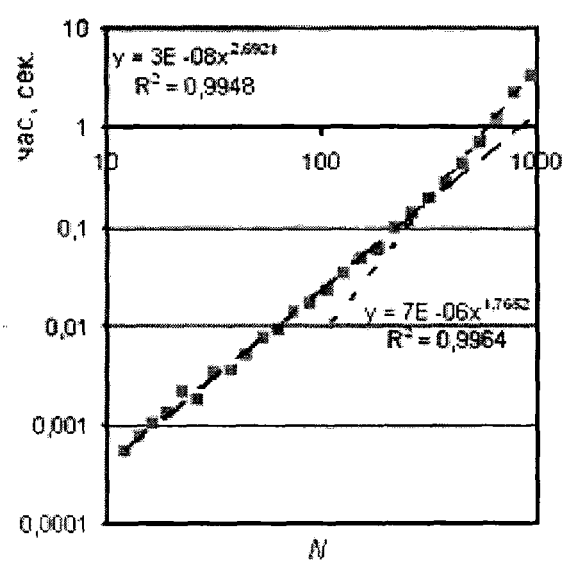

Рис. 1. Час розрахунку залежно від кількості вершин у графі. Результат усереднено для 100 випадкових графів, що відрізняються положенням 20 ребер, з однаковою кількістю вершин N і середньою кількістю ребер для однієї вершини 50 % від N.

Для графів G і H з однаковою кількістю вершин і ребер і випадковим розподіленням ребер між вершинами імовірність збігу одразу кількох інваріантів надзвичайно низька [7], що практично гарантує високу ефективність алгоритму.

Реалізація алгоритму та результати тестування

Алгоритм був реалізований мовою програмування C++. Генератор перестановок номерів вершин на множині підінтервалів індексів $\{[1..n_1], [(n_1 + 1)..(n_1 + n_2)], ..., [(N - n_m)..N]\}$ реалізовано шляхом послідовної генерації транспозицій на кожному з підінтервалів з переходом до наступного підінтервалу з досягненням перестановки останнього індексу в попередньому. Для кожної перестановки генератор викликає функцію порівняння графів. У цілому процедура мало чим відрізняється від описаної в [6].

Масиви D було об'єднано у масив з довільною кількістю елементів. Замість побудови окремого індексу для сортування масивів D в алгоритмі безпосередньо використовується індексний масив вершин графа.

Алгоритм тестували для випадкових графів зі змінною кількістю вершин N, а також для графів з фіксованою кількістю вершин $N = 50$ і змінною середньою кількістю ребер V, що належать кожній вершині. Для випадкових графів час встановлення ізоморфізму двох ізоморфних графів $\sim N^{1,75}$ для $N < 300$ і $\sim N^{2,7}$ для $N > 400$. Оскільки практично в кожному випадку алгоритм генерує розбиття на $N - N/2$ підінтервалів, час роботи визначається поліноміальною складністю обчислення локальних інваріантів та порівняння двох графів. Збільшення значення показника пов'язане зі значним збільшенням обсягу пам'яті, який займають графи. При обчисленнях на процесорах типу Pentium та Power PC це призводить до переповнення кеш-пам'яті 2-го рівня і різкого зниження продуктивності за рахунок збільшення кількості операцій кеш-пам'яті. У цілому результати відповідають часовій оцінці, отриманій за допомогою пакета nauty [7], в якому окрім локальних інваріантів 1-го порядку враховуються деякі інші, наприклад кількість k-клік, до яких входить кожна вершина, для невеликих k.

На рис. 1 наведено середній час, витрачений на обчислення локальних інваріантів та генерацію всіх перестановок залежно від кількості вершин для щільних неізоморфних графів з $V \sim N/2$. Для порівняння з відомими результатами [7] в наших обчисленнях графи збігалися з точністю до випадкової перестановки 20 пар ребер.

Для розріджених графів з $V \ll N$ характерний збіг локальних інваріантів першого і навіть дру-

гого порядку. В цьому випадку необхідне обчислення інваріантів вищого порядку, що призводить до значних витрат часу на обчислення векторів D порядку 4–5. Для сучасних комп'ютерів з тактовою частотою ~1–2 ГГц виконання однієї операції займає ~0,25–1 нс. Таким чином, для графів з ~1000 вершин будь-який алгоритм матиме задовільну продуктивність, лише якщо його часова складність не перевищує $O(N^3)$. Отже, використання даного алгоритму для великих графів обмежене інваріантами 3–4-го порядку.

На рис. 2 наведено середній час обчислень для графів з 50 вершинами і різною середньою кількістю ребер на одну вершину. Середня кількість перестановок змінювалась від 10^{10} для графів із середньою валентністю вершини 1–3 та 47–49 до 2–100 для графів з середньою валентністю вершини 4–6 та 44–46. В ділянці проміжних значень валентності масив векторів D звичайно визначає єдину можливу конфігурацію.

Такий результат узгоджується із запропонованим алгоритмом. Для графів з малою валентністю вершин (або дуже великою, тобто малою валентністю доповнення графа до повного) кількість можливих значень локальних інваріантів

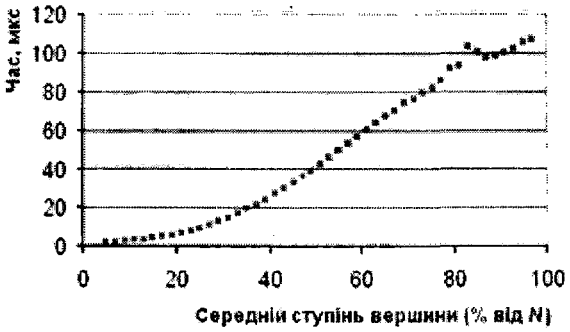

Рис. 2. Час розрахунку залежно від ступеня насиченості графа

невелика. Даний алгоритм не дає виграшу в кількості перестановок, якщо всі значення локального інваріанта однакові. Це типово для сильно зв'язних регулярних графів, циклів та графів, що містять один великий цикл довжини ~N з невеликою кількістю додаткових вузлів. Більш того, існує широкий і цікавий клас графів, у яких збігаються локальні інваріанти перших порядків. Серед них назвемо структурні графи молекул полімерів зі складних мономерів. У таких графах ланцюгові та розгалужені розташування мономерів не відрізняються або практично не відрізняються локальним порядком.

1. *Зыков А. А.* Основы теории графов.– М.: Наука, 1987.
2. *Журавлев Ю. И.* Оценка сложности локальных алгоритмов для переборных экстремальных задач на конечных множествах.– ДАН СССР, 158(1964).– № 5.– С. 1018–1021.
3. *Яблонский С. В.* Об алгоритмических трудностях решения задач / Сб. «Проблемы кибернетики» 2.– М.: Физматгиз, 1959.– С. 75–121.
4. *Мелихов А. Н.* Ориентированные графы и конечные автоматы.– М.: Наука, 1971.
5. *Хорн Р., Джонсон Ч.* Матричный анализ.– М.: Мир, 1989.
6. *Липский В.* Комбинаторика для программистов.– М.: Мир, 1988.
7. *Fortin Scott.* The graph isomorphism problem / Technical Report TR-96-20.– University of Alberta, 1996.
8. *Eppstein D.* Subgraphs isomorphism in planar graphs and related problems // Journal of Graph Algorithms and Applications, v. 3.– № 3.– P. 1–27.
9. *Brendan D. McKay.* Practical graph isomorphism // Congressus Numerantium, v. 30.– P. 45–87 (1981).

S. M. Chychkan

ALGORITHM FOR GRAPH ISOMORPHISM BASED ON LOCAL INVARIANTS

Data sorting and arrangement is one of the important methods, which help to narrow calculation time in combinatoric algorithms. When such a method is applicable, one could start from initial data set and assign each data set member an unique number – «weight» or «colour» in graph theory. When the aim of calculation consists in equating 2 data sets, quick sort of both sets with time ~$O(N\log N)$ by weights solves the problem, because afterward one would need to compare ordered arrays only. The current paper partially solves the isomorphism problem by introducing sorting technique for a number of different local invariants for graph vertices.