УДК 004.043

ОРГАНІЗАЦІЯ ЕФЕКТИВНОГО СХОВИЩА ДАНИХ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Лементова Є.О.
Науковий керівник – к.т.н, доц. Білова Т. Г.
Харківський національний університет радіоелектроніки, каф. ІУС
м. Харків, Україна
тел. +38(050) 136-88-57, email: yevheniia.lementova@nure.ua.

During software testing results of it are needed to save rationally. If tests are built as classification trees, it is got general ones. They need a lot of memory. Therefore it’s necessary to use a data structure where searching, addition/ deletion of elements are performed quickly. After analyzing common data structures as arrays, linked lists, binary and ternary trees the last ones have the best results. Despite the fact that average complexities of searching in binary and ternary trees are the same the last ones don’t need to do sorting after it. Thus they can be used for processing results of testing.

Тестування програмного забезпечення потребує раціонального зберігання наборів тестових задач [1]. При побудові тестів на основі дерева класифікації спочатку виділяється набір факторів (аспектів), що мають найбільший вплив на поведінку тестувальної системи. Далі визначається розбиття значень аспектів на групи. Базові аспекти утворюють вершини дерева класифікації, безпосередньо пов'язані з його корінням. Їхні класи та вторинні аспекти прив'язуються до базових аспектів, тощо, поки всі суттєві фактори не будуть знайдені та класифіковані.

Відомо, що зберігання сильнорозголуженого дерева, яким є класифікаційне дерево, потребує значних витрат пам’яті. Рішенням даної проблеми є застосування ефективної структури даних, яка реалізує швидкий пошук та додавання нових елементів.

При організації зберігання результатів тестування у масиві перевагою буде швидкий доступ до його елементів за індексом. Двійковий або інтерполяційний пошук в масиві має логарифмічну складність, але дані повинні бути впорядкованими, що призводить до квадратичної складності його реалізації. Операції вставки та видалення на масиві також дають лінійну складність через необхідність зсуву елементів. Тож, на великих обсягах даних час обробки суттєво збільшується.

Організація зберігання набору тестових задач у вигляді списку призводить до константної складності операцій вставки та видалення, але пошук займатиме лінійний час, що є неефективним в рамках вирішення даної задачі.

Оптимізованою структурою даних вважають бінарне дерево пошуку. Складність пошуку в бінарному дереві O(h), де h – висота дерева. Після операції додавання (або видалення) елементів висота бінарного дерева

змінюється, а отже зростає і складність пошуку. Як зазначено в [2], сортування бінарного дерева пошуку з метою покращення його структури втрачає сенс, коли операції додавання елемента та пошуку його на дереві виконуються одночасно. Тоді ефективність може скласти $n \times \log_2 n$ [3].

Тернарні дерева – дерева пошуку, що мають не більше трьох нащадків. Пошук у них починається з кореня. Ключ порівнюється з рядком, та розглядається три випадки: якщо код є меншим, то пошук продовжується у лівому піддереві, якщо дорівнює ключу – у середньому піддереві, якщо більше – в правому піддереві.

При зберіганні результатів тестування у вигляді тернарних дерев фактичний час виконання пошуку дорівнюватиме $O(L \times \log n)$, де L – довжина найдовшого рядка в структурі даних, а n – кількість рядків. Складність визначається як $O(\log n)$ порівнянь, кожне з яких вимагає часу O(L).

Ще однією перевагою використання тернарних дерев для зберігання даних великого обсягу, порівнюючи з бінарними деревами, є наявність операції рівності. За допомогою чого відбувається утворення нових гілок, які знаходять вільні місця на дереві. А в бінарних деревах заповнення новими елементами у більшості випадків можливе завдяки збільшенню глибини дерева, а як наслідок збільшення часу виконання програми.

Порівнявши ефективність операцій вставки та пошуку для зберігання результатів тестових програмних модулів в різних структурах даних можна зробити висновок, що використання списків та асоціативних масивів є нераціональним через лінійну складність в операціях пошуку та вставки/видалення, відповідно. Бінарні дерева дають логарифмічну складність, якщо вони врівноважені. Слід врахувати й той факт, що бінарні дерева можуть втрачати свою ефективність, коли операції додавання елемента та пошуку його на дереві виконуються одночасно. Тернарні дерева мають середню складність пошуку, порівняну з ефективністю на двійковому врівноваженому дереві, але без виконання операції сортування або врівноважування дерева, що робить їх найбільш перспективними для використання при обробці результатів тестування.

Список використаних джерел:

1. Kulyamin V. V. Testirovanie na osnove moduley. Vzyato 15 lyutogo 2022 roku з http://panda.ispras.ru/~kuliamin/lectures-mbt/Lecture05.pdf

2. Knut D. (1978) Iskusstvo programmirovaniya dlya EVM. Sortirovka i poisk. Moskva. Mir. T. 3.

3. Kovartsev A. N., Popova-Kovartseva D. A., & Goroshkova E. E. (2016) *Ispolzovanie ternarnyih derevev dlya hraneniya dannyih vyichislitelnogo eksperimenta. Informatsionnyie tehnologii i nanotehnologii*, 1044-1050.