

Національний технічний університет України
"Київський політехнічний інститут ім. Ігоря Сікорського"

Навчально-науковий Фізико-технічний інститут
Кафедра математичних методів захисту інформації

ЛАБОРАТОРНА РОБОТА 1: РЕАЛІЗАЦІЯ МНОЖИНИ

ВАРІАНТ 1

Виконав студент
групи ФІ-23
Чуй Тимофій

Опис реалізації

У варіанті 1 потрібно було використовувати як базовий тип даних список, для спрощення коду у моєму випадку він двозв'язний; на швидкість операцій це не впливає (для однозв'язного можна було б записати потрібні сусідні вершини як сторонні вказівники). Список дозволяє зберігати об'єкти різних типів, залишив основні: *int*, *float*, *char*, *string*; проте вони можуть бути довільним за умови можливості порівняння, їх легко можна додати за необхідності до множини допустимих (вузол визначається з використанням шаблону класу *variant*, тому достатньо лише дописати потрібний тип).

Базові операції, здебільшого, реалізовані стандартним способом: лінійний пошук і потрібна операція в залежності від результату. З особливостей, метод *set_insert* написаний у двох варіантах, оскільки іноді можна не виконувати перевірку наявності в множині і витратити трохи менше часу на деякі додаткові операції. З тією ж метою метод *set_search* повертає не булеве значення, а вказівник на знайдений вузол, який можна використовувати надалі (наприклад, у *set_delete*). Щодо вилучення елемента, воно не просто перенаправляє вказівники, а додатково вивільняє пам'ять, виділену під вузол - як наслідок, *set_clear* має викликати *set_delete* для кожного елемента. Так як пошук починається з початку списку, кожний такий виклик закінчується на першому елементі, але все ж очистка множини вимагає час $\Theta(n)$.

Додаткова логіка так само використовує лінійний пошук, який я намагався уникати, де було можливо, за рахунок *set_unsafe_insert*. Усі методи, котрі приймають на вхід дві множини і мають повернути третю, залишають вхід без змін і повертають вказівник на результат. Варто сказати, що методи додатково покриті юніт-тестами, усі вони пройшли.

Безпосередньо заміри часу я проводив на випадкових множинах, згенерованих за допомогою вихра Мерсена з випадковим початковим значенням: намагався наблизити якомога ближче розподіл до рівномірного, щоб особливості генерації не впливали на результат. Для спрощення в тестові множини входять лише типи *int*, *float*, *string* - в такому випадку менш ймовірно неспівпадіння типів даних (що дозволяє не порівнювати значення і потенційно зменшує час), але, загалом, це не має сильно вплинути. Тестових сценаріїв було 10, з розмірами множин від 25 до 1025 з кроком 100; перевірялися пошук в двох випадках і обчислення різниці (оскільки остання не використовує спрощення додавання елементів); кожен сценарій повторювався 1000 раз.

Код - [тут](#); за умови наявності бібліотеки *gtest* його достатньо для автономної компіляції.

Результати експериментів

Далі наведено графіки залежності часу виконання від розміру множини, для *set_search* у мікросекундах, для *set_difference* у мілісекундах.

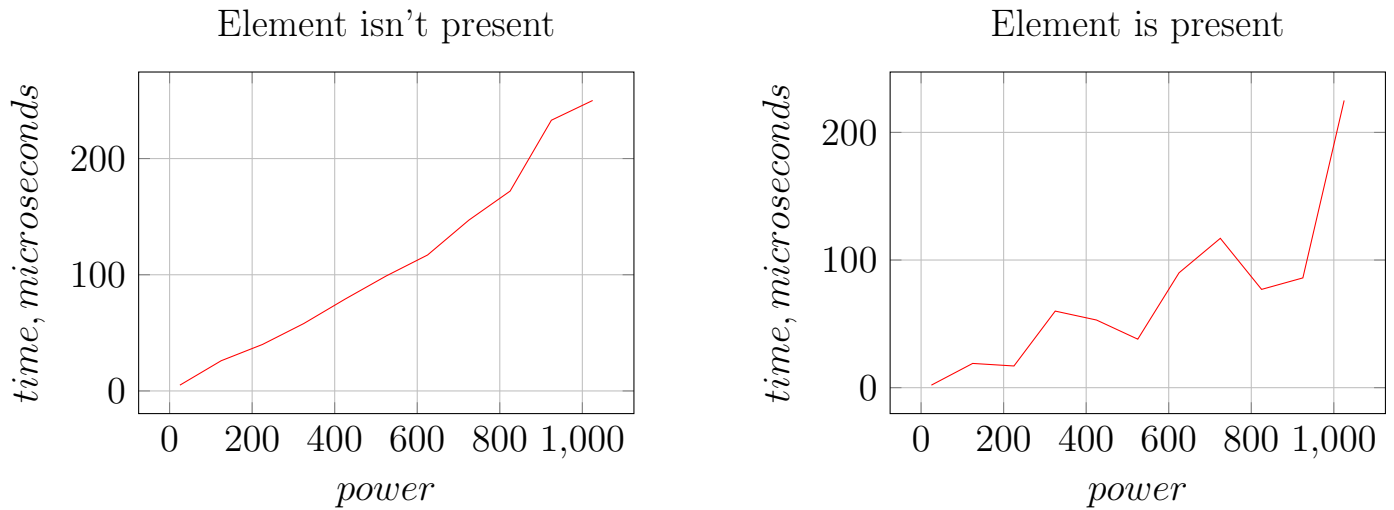


Рис. 1: Search tests

Загалом, перші графіки відповідають очікуванам: пошук наявного елементу займає у більшості тестів приблизно вдвічі менше часу, ніж відсутнього; для пошуку відсутнього залежність близька до лінійної. Не зважаючи на це, у двох випадках результати виділяються серед інших. Перший - пошук відсутнього елементу при потужності множини 925, різниця між часом пошуку при цьому сценарії і попередньому помітно велика; потім графік вирівнюється до очікуваного. Найбільш ймовірне пояснення - структура тестової множини і підвищення складності через складність порівнянь елементів. Другий - при потужності 325: пошук існуючого елементу зайняв на 2 мікросекунди більше, ніж відсутнього. Тут я не впевнений у причині, може бути як те саме, що і у минулому випадку, так і погрішність через відносно невелику кількість експериментів за сценарієм.

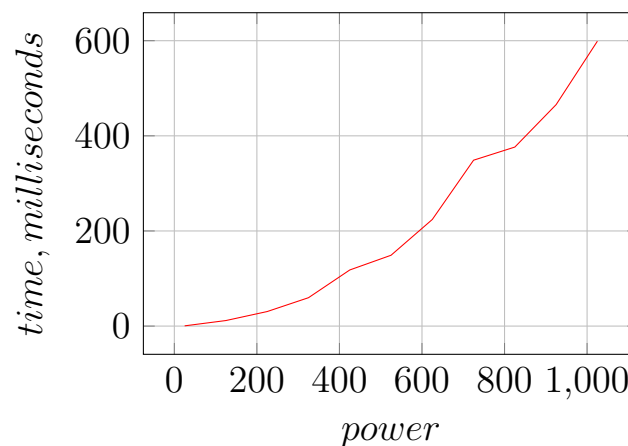


Рис. 2: Difference test

Тут результат краще: залежність явно не лінійна, близька до поліноміальної; єдина проблемна точка - 725, яка дещо відходить від очікуваного. Знову, проблема скоріш за все у структурі.