**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Кафедра інформаційних систем**

**Алгоритми та складність**

**Завдання № 3**

**Звіт**

**Виконав:**

Студент групи К-29

Михед Олександр Михайлович

**Київ-2018**

# Умова завдання

Реалізувати розширюване дерево.

# Розширюване дерево

## **Опис алгоритму**

а) Спочатку створюється кореневий елемент, до якого в ході роботи програми додаються дочірні елементи.

б) Опрацьовується дерево елементів.

в) Інваріант: значення кореневого елемента, є медіанним значенням серед значень всіх елементів. Лівий дочірній елемент завжди менший батьківського. Правий дочірній елемент завжди більший батьківського.

## **Аналіз алгоритму**

O(n) – найкращий випадок, коли всі елементи однакові, в цьому випадку не відбудеться додавання нових елементів.

O(nlogn) – найгірший випадок, коли всі елементи різні і впорядковані таким чином, що щоразу алгоритм проходить до найнижчого елемента дерева. Так як висота дерева дорівнює log2(n), то спуск елемента коштує log2(n). Таких елементів n, тому складність дорівнює O(nlogn).

**Cкладність пам’яті:** O(1), тому що для роботи алгоритму не потрібні ніякі допоміжні структури даних, усе відбувається в одному дереві

Час роботи при сортуванні N елементів

|  |  |
| --- | --- |
| N | Time(s) |
| 1000 | ≈0 |
| 10000 | 0,005 |
| 50000 | 0,025 |
| 100000 | 0,05 |

Отже, при вхідних даних розміру n, час роботи програми описується рівнянням y=1E−7x

Використана пам’ять при сортуванні N елементів

|  |  |
| --- | --- |
| N | Memory(KB) |
| 1000 | 428 |
| 10000 | 876 |
| 50000 | 2916 |
| 100000 | 4668 |

Відношення використаної пам’яті до часу роботи програми при сортуванні N елементів

|  |  |
| --- | --- |
| N | Memory(KB)/Time(s) |
| 1000 | ~ |
| 10000 | 175200 |
| 50000 | 116640 |
| 100000 | 93360 |

Зі зростанням часу роботи програми використання пам’яті збільшується, оскільки постійно добавляються нові елементи до дерева.

**Висновок**

Швидкодія алгоритму дуже висока, швидко відбувається додавання нових елементів. Із збільшенням кількості унікальних елементів значно збільшується об’єм використовуваної пам’яті. Це дерево належить до класу «саморегульованих дерев», які підтримують необхідний баланс галуження дерева, щоб забезпечити виконання операції пошуку, додавання і видалення за логарифмічний час від числа елементів, що зберігаються. Це реалізується без використання яких-небудь додаткових полів у вузлах дерева (як, наприклад, в Червоно-чорних деревах або АВЛ-деревах, де у вершинах зберігається, відповідно, колір вершини і глибина піддерева). Замість цього «розширювальні операції» (splay operation), частиною яких є повороти дерева, які виконуються при кожному зверненні до дерева. Облікова вартість з розрахунку на одну операцію з деревом складає O(log n).

Даний алгоритм дозволяє швидше знаходити ті елементи котрі недавно використовувались.