**Алгоритми та складність**

**Завдання № 4**

**Оптимальне бінарне дерево пошуку**

**Звіт**

**Виконала:**

**Гур’янова Дар’я група к-28**

**Київ-2018**

**Оптимальне бінарне дерево пошуку**

**•** Умова завдання. Реалізувати оптимальне бінаре дерево пошуку.

• **Основна ідея**

Дано масив частот p[0…n-1], де p[i] це кількість звернень до елемента i. Треби побудувати бінарне дерево пошуку з цих ключів, щоб повна вартість дерева була найменшою.

Дерево, яке має найменшу вартість, вважається найкращим для пошуку елементів з n. Саме тому воно називається Оптимальним Бінарним Деревом Пошуку.

Цей алгоритм завдяки методу динамічного програмування дозволяє представити данні оптимальним чином і підрахувати математичне сподівання вартості пошуку в оптимальному бінарному дереві пошуку за наявності таблиці ймовірностей по кожному ключу .

Так як ми реалізуємо цей алгоритм завдяки методу динамічного програмування нам знадобиться допоміжний масив e[n][n] , щоб зберігати вартість піддерев. Так, у комірці масиву e[i][j] , i ≤ j будемо зберігати вартість дерева, що містить елементи з ключами від i-ого до j-ого включно, очевидно, що комірка масиву e[0][n-1] містить наймешу ціну нашого дерева. Для побудови дерева треба ще один допоміжний масив root[n][n], який у комірці root[i][j] , i ≤ j буде зберігати корінь оптимального дерева , що містить елементи з ключами від i-ого до j-ого включно.

Спочатку нам відомо, що вартість дерева. що має одну вершину це є частота звернень до цієї вершини, потім на кожній ітерації будемо знаходити оптимальні дерева, прчинаючи з дерев, що містять 2 вершини, 3 вершини і так далі. Нехай на якійсь ітерації треба знайти оптимальне дерево, що містить ключі від i-ого до j-ого включно, ми ставимо кожен ключ k, такий що k ≥ i і j ≥ k як корінь дерева і ми знаємо оптималні дерева що містять вершини від i-ого до (k - 1)-ого ключі та від (k + 1)-ого до j-ого ключі, ставимо їх як ліве піддерево і праве піддерева відповідно і рахуємо вартість (варість піддерев ми вже знаємо), так ми знайдемо потрібне оптимальне дерево.

**• Реалізація алгоритму.**

Реалізовано на С++.

**• Висновки.**

Цей алгоритм виконується за О(n^3) часу.В результаті ми отримуємо бінарне дерево пошуку на даних з вірогідностями q i p.

**• Використані джерела.**

Т. Кормен «Алгоритмы, построение и анализ».