Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Алгоритми та складність

Завдання №1

“Оптимальне бінарне дерево пошуку”

Виконав студент 2-го курсу

Групи К-29

Пупов Нікіта Андрійович

2020

**Завдання**:

Оптимальне бінарне дерево пошуку(динамічне програмування)

**Теорія:**

Розглянемо множину S = {e[1], e[2], ..., e[n]}, що складається з n різних елементів таких, що e[1] <e[2]<... <e[n]. Розглянемо бінарне дерево пошуку, що складається з елементів S. Чим частіше проводиться запит до елементу, тим ближче він повинен розташовуватися до кореня.

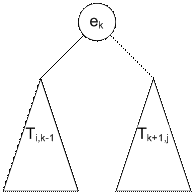
Вартістю cost доступу до елементу e[i] з S в дереві будемо називати значення cost (e[i]), яка дорівнює кількості ребер на шляху, який з'єднує корінь з вершиною, що містить елемент. Маючи частоту запитів до елементів з S, (f (e[1]), f (e[2]), ..., f (e[n])), визначимо загальну вартість дерева наступним чином:

f (e[1]) \* cost (e[1]) + f (e[2]) \* cost (e[2]) + ... + f (e[n]) \* cost (e[n])

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

**Алгоритм**

Нехай T[i, j] дорівнює вартості оптимального бінарного дерева пошуку, яке можна побудувати з елементів e[i], e[i + 1], ..., e[j]. Очевидно, що T[i, i] = 0 (вартість дерева пошуку з однієї вершини дорівнює нулю). Для i <j має місце рекурентність:



Елемент e[k] ставимо в корені. Вартість побудови лівого піддерева дорівнює  , правого  . При цьому оскільки корінь лівого піддерева знаходиться на один рівень нижче e[k], то для вартості лівого піддерева необхідно додати суму частот всіх його елементів, тобто значення  . Аналогічно при підрахунку вартості правого піддерева слід додати  .

При i > j покладемо T[i,j] = 0.

Таким чином, для побудови дерева ми заповнюємо таблицю dp, де dp[i,j] зберігає індекс кореня та вартість для піддерева з вершин з індексами i...j

При кожному зверненні до вершини, показник frequency збільшується. Потім на основі цього показника ми можемо перебудувати дерево та зробити його оптимальним.

**Мова програмування** С++

**Структура програми**

Програма складається з головного класу алгоритму OptBinTree, який працює з об’єктами типу File.

**Модулі програми:**

* **int minimizeTree();**

Перебудовує дерева, роблячи його найоптимальнішим.

Складність: О(n^2)

* **File getFile(string name);**

Знаходить файл по його імені.

Складність O(logn)

* **void addFile(File file);**

Додає файл

Складність: O(logn)

**Матеріали**

<https://site.ada.edu.az/~medv/acm/Docs%20e-olimp/Volume%2016/1522.htm>

<https://www.intuit.ru/studies/courses/648/504/lecture/11469?page=2>