**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Кафедра інформаційних систем**

**Звіт (алгоритми та складність)**

**“Дерево порядкової статистики.”**

**Предметна область : “Продукти”**

**Підготував студент**

**2 курсу**

**Групи К29**

**Опанюк Микита**

**Опис предметної області :**

«Вариант 7   
Предметная область  Продуктовый магазин   
Объекты  Категория продукта, Продукт   
Примечание  Продукты  в  магазине   
сгруппированы  по  категориям.   
Для  каждой  категории   
определено  множество   
продуктов.»

**Інтерфейс :** консоль.

В задачці використовується class market{}; - саме в цьому класі йде основна реалізація предметної області. А саме :

Клас працює з 2 типами структур :

struct type {

string group; - назва різновиду.

unsigned int hash\_key; - відповідно ключ, який підраховується ф-ю hash\_string.

vector<product\_element \*> array\_of\_products; - вектор вказівників на продукти вказаного типу.

}; - використовується для збереження різновидів продуктів.

struct product\_element {

string group; - назва різновиду продуктів, до якого відноситься вказаний.

string name; - відповідно ім’я.

double price; - ціна (для інформативності).

unsigned int hash\_key; - відповідно ключ, який підраховується ф-ю hash\_string.

};

vector<type \*> type\_of\_products; - масив типів продуктів в магазині, де кожен елемент масиву має вказівник на масив самих продуктів.

int check\_is\_product(int pos, string product); - перевіряє наявність вказаного продукту в магазині, де pos – це індекс елемента масива типів продуктів.

int check\_is\_type(string product); - перевіряє наявність відповідно типу продуктів в магазині.

void new\_data(); - додавання нового продукту з перевіркою на наявність типу цього продукту (якщо тип відсутній, то створюється новий), відповідно присутня перевірка того чи існує вже вказаний продукт.

void delete\_data(product\_element \*name); - видаляє з каталогу вказаний продукт (зменшує масив продуктів у відповідного типу)

void save\_changing(); - зберігає всі зміни каталогу, перезаписуючи файл, на якому зберігаються дані що до продуктів.

void read\_data\_from\_file(); - зчитує данні що до продуктів з файлу.

unsigned int hash\_string(string key); - хешує імена груп/продуктів для подальшої роботи з деревом порядкової статистики (в подальшому дане значення використовуватиметься як «ключ»).

2 інші класи відповідно використовуються в дереві

**“Дерево порядкової статистики.”**

**Опис Головного методу :**

1) Персистентна множина побудована на базі сбалансованого червоно-чорного дерева, для якого характерні наступні властивості :

а) Кожна вершина є або червоною, або чорною.

б) Корінь дерева є чорною вершиною.

в) Кожен лист дерева (son) є чорною вершиною.

г) Якщо вершина червона, то обидва його сини чорні.

д) Для кожної вершини всі прості шляхи від нього до листків, що є потомками даної вершини, має одну й ту ж кількість чорних вершин.

2)Node \***insert\_node**(**int** data, **unsigned** **int** num\_of\_version); - функція вставки елементів в дерево. Вставка стандартна, тобто рухаємось від кореня, порівнюючи ключі, до листків, збільшуючи ранг кожної вершини на 1, додаємо новий елемент у дерево, після цього перебалансуємо дерево за допомогою функції : **void** **insert\_balance**(Node \*x, **unsigned** **int** num\_of\_version);

3)При балансуванні розглядаються 5 випадків :

а) Чи є вказана вершина корнем дерева (відповідної версії дерева).

б) Чи батько вершини є чорним (в кінці функції це вказується).

в) Чи батько та дядько – червоні.

г) Батько – червоний, а дядько – чорний, крім того вершина є правим сином. (данний випадок зводиться до 5 випадку (д), тому як змінюються між собою позиції батька та сина)

д) Батько – червоний, дядько – чорний, вершина – лівий син. (Це буде поворот на право).

Відповідно під час поворотів вліво або вправо міняємо ранги вершин між собою при зміні їхніх позицій.

4) Видалення вершин відбувається таким чином :

а)**void** **delete\_node**(Node \*m, **unsigned** **int** num\_of\_version);

Рухаючись від кореня до вказаної вершини, зменшуємо ранги у вершин, що лежать на шляху на 1. Функція знаходить на заміну вершину, що є нащадком.

В перебалансувані дерева після видалення вершини розглядаються в загальному 6 випадків:

а) Вершина, що є заміною (нехай дана вершина “N” - заміна на новій позиції) – новий корінь. (Нічого не треба робити, властивості дерева зберігаються).

б) Чи брат вершини N червоний (тоді змінюємо колір батька на червоний, виконуємо поворот дерева вліво).

в) Якщо батько і брат вершини N, а також діти брата – чорні (тоді колір брата змінюється на червоний).

г) Брат та його діти – чорні, проте батько – червоний (батько стає чорним, брат стає червоним).

д) Брат – чорний, лівий син брата – червоний, правий син брата – чорний. N – лівий син (лівий син брата = SL, стає батьком брата S, S стає правим сином SL, тобто виконано поворот направо).

е) Брат – чорний, правий син брата – червоний, N – лівий син (виконуємо поворот дерева наліво, перемальовуємо вершини для виконання властивостей дерева).

Функція Delete\_balance перевіряє всі ці умови та відновлює властивості дерева.

(відповідно змінюємо ранги під час поворотів вліво або вправо).

**Опис алгоритму з предметною областю:**

Внесено до файлу 4 продукти : ковбаса, шоколад, цукерка «Ромашка», курятина.

Алгоритми основується на побудові дерева з піддеревами. Основне дерево :

1)Будується за алгоритмом вище, реалізований для типів продуктів.

struct type {

node\_colour colour;

string group; - назва групи

unsigned int hash\_key; - ключ для побудови дерева

unsigned int rank; - ранг

RB\_tree\_product \*new\_tree; - вказівник на піддерево продуктів.

type \*left;

type \*right;

type \*parent;

};

class RB\_tree\_group {} – основний клас, в якому описаний метод побудови дерева. Під час побудови дерева «типів» для кожного елемента будується піддерево «продукти» :

struct products {

node\_colour colour;

string name; - назва продукту

string group; - тип продуктів

double price; - ціна

unsigned int hash\_key; - ключ для побудови дерева

unsigned int rank; - ранг

products \*left;

products \*right;

products \*parent;

};

class RB\_tree\_product {} – клас піддерева «продукти», методи.

Відповідно, для користувача подається запит на пошук групи в дереві, а вже потім подається запит на пошук самого продукту в піддереві.

**Тестовий приклад :**

1. Зчитується з файлу «products.txt» інформація про продукти :

4

candies;boom;14.95 429

candies;chocolate bar;16.65 1287

candies;cake;10.59 404

meat;chicken;33.99 725

1. vector<type \*> type\_of\_products; - вноситься інформація про продукти. А саме до вектору додаються 2 нових елементи, які матимуть вказівники на відповідного типу продукти.
2. В подальшому вектор type\_of\_products використовується для побудови основного дерева RB\_tree\_group main\_tree(main\_data);, де main\_data – клас, що зберігає цей вектор. main\_tree представляє із себе червоно-чорне дерево порядкової статистики, елементами якого є типи продуктів : candies та meat. Тобто дерево в даному випадку складається всього з 2 вершиню
3. Під час додавання вершини (приклад) candies починаємо побудову піддерева за вказівником RB\_tree\_product new\_tree; Це дерево матиме елементи = продукти : boom, cake, chocolate bar. Відповідно, користуючись функцією hash\_number() отримуємо ключі відповідних імен для побудови дерева. Після побудови піддерева, відбувається виведення дерева та виклик меню редагування вказаного піддерева (додати вершину, видалити вершину, перевірити вершину за рангом). Після виходу з меню редагування повертаємось до корку 3) (продовжуємо додавати вершини до основного дерева «груп»).
4. Після побудови основного дерева груп та всіх піддерев відбувається виведення дерева та виклик меню редагування для основного дерева (видалення вершини, перевірка за рангом). Після завершення редагування завершується робота програми.

**Плюси, порівняно зі звичайним червоно-чорним деревом:** Збільшення функціональності дерева, що дозволяє розв’язувати певний тип задач.

**Використані джерела :** Кормен “Алгоритми. Побудова та Аналіз”.