Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання №5

“ Персистентна множина”

Виконав студент 2-го курсу

Групи К-29

Лавров Сергій Сергійович

2021

**Предметна область**

Футбол

Об'єкти: Команди, Гравці

Примітка: Маємо команди. Для кожної команди визначені гравці

**Завдання**

Реалізація персистентної множини на основі червоно-чорного дерева. Час роботи вставки і видалення в найгіршому випадку і об‘єм необхідної пам’яті мають бути О(log n).

**Теорія**

Персистентні динамічні множини

• Зберігають свої попередні версії (і доступ до них) в процесі внесення змін.

• Може зберігатися тільки остання версія або всі існуючі попередні.

• Персистентними можна зробити різні структури даних.

• Для ефективної реалізації просте копіювання не підходить.

**Алгоритми**

Робимо модифікацію червоно-чорного дерева. Зберігаємо вершини дерева в яких відбулась зміна дерева для економії пам’яті.

**Складність**

Червоно-чорне дерево має складність О(lg n) для всіх операцій.Час роботи вставки і видалення в найгіршому випадку і об‘єм пам’яті О(log n). Для цього при вставці треба створювати під дерево старої версії вузла, який буде зберігатися в певній змінні, після чого може відбуватися зміна основної структури. Аналогічно для операції видалення.

**Мова програмування**

С++

**Модулі програми**

class Team, class Player – класи для реалізації предметної області.

Для класу Team перевизначив оператори <, >, =>, == для роботи з деревом, і оператор << для простішого виводу в консоль.

struct treeNode – вузел дерева

int ID індетифікатор множини

T key ключ вершини

Color color колір вершини

treeNode\* parent вказівник на батька

treeNode\* left вказівник на лівого сина

treeNode\* right вказівник на правого сина

int generation номер версії в множині

class RedBlackTree клас червоно чорного дерева

treeNode<T>\* root корінь

vector<treeNode<T>\*> persistentRoots корні збережених під дерев

void leftRotate(treeNode<T>\* x) поворот вліво

void rightRotate(treeNode<T>\* x) поворот вправо

Фукнції підтримки властивостей червоно-чорного дерева

void insertFixup(treeNode<T>\* z)

void insertFixupPersistent(treeNode<T>\* z)

void removeFixup(treeNode<T>\* x);

void removeFixupPersistent(treeNode<T>\* x);

Допоміжні функції для отримання вершин:

treeNode<T>\* treeSuccessor(treeNode<T>\* x);

treeNode<T>\* treeSuccessorPersistent(treeNode<T>\* x);

treeNode<T>\* treeMinimum(treeNode<T>\* x);

treeNode<T>\* treeMinimumPersistent(treeNode<T>\* x, treeNode<T>\* pre)

treeNode<T>\* treeMaximum(treeNode<T>\* x);

treeNode<T>\* treeSearch(treeNode<T>\* x, const T& key);

Функції для виводу дерева в консоль

string getText(treeNode<T>\* x);

void getGraphInfo(treeNode<T>\* x, string& text);

void getGraphInfoPersistent(treeNode<T>\* x, string& text, int gen) RedBlackTree(const initializer\_list<T>& list, bool isPersistent = false)

void skipBack() повернення до попередньої версії

template<typename U>

friend ostream& operator<<(ostream& os, RedBlackTree<U>& tree);

void insertPersistent(const T& item) вставка елемента

treeNode<T>\* treeSearchPersistent(const T& key) пошук елемента

treeNode<T>\* removePersistent(const T& item) видалення елемента

**Інтерфейс користувача**

Користувач може вводити наступні команди в консоль:

addteam ‘назва команди’ ‘кількість гравців’ ‘імена’…

Створює нову команду і додає її до розширюваного дерева

insertplayers ‘назва команди’ ‘кількість гравців’ ‘імена’…

Шукає команду за назвою і вставляє в неї нових гравців

search ‘назва команди’

Знаходить команду в дереві і виводить її гравців в консоль

erase ‘назва команди’

Видаляє команду з дерева за назвою

print

Виводить дерево в консоль

Back

Повернутися до попередньої версії

exit

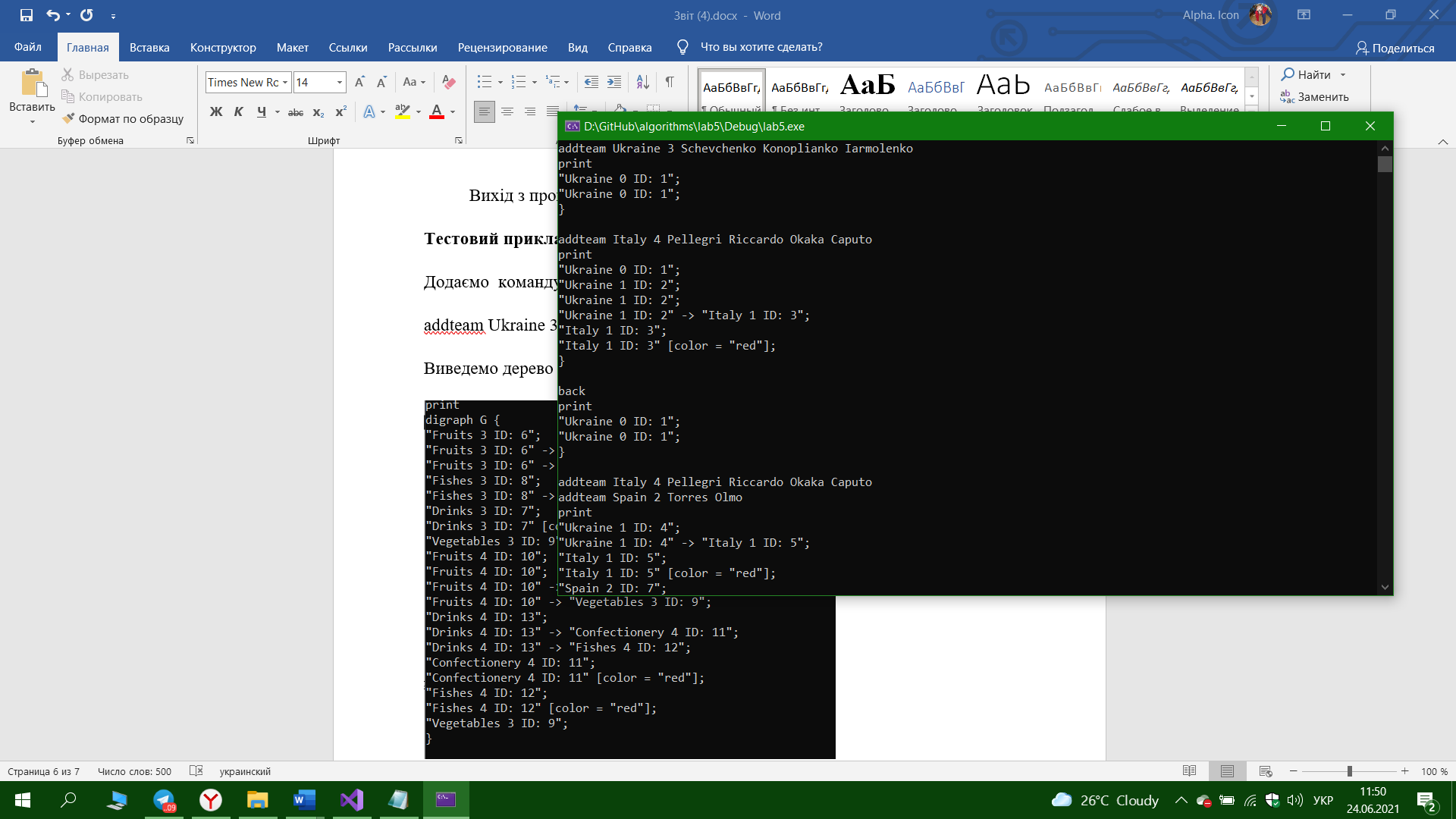
Вихід з програми

**Тестовий приклади**

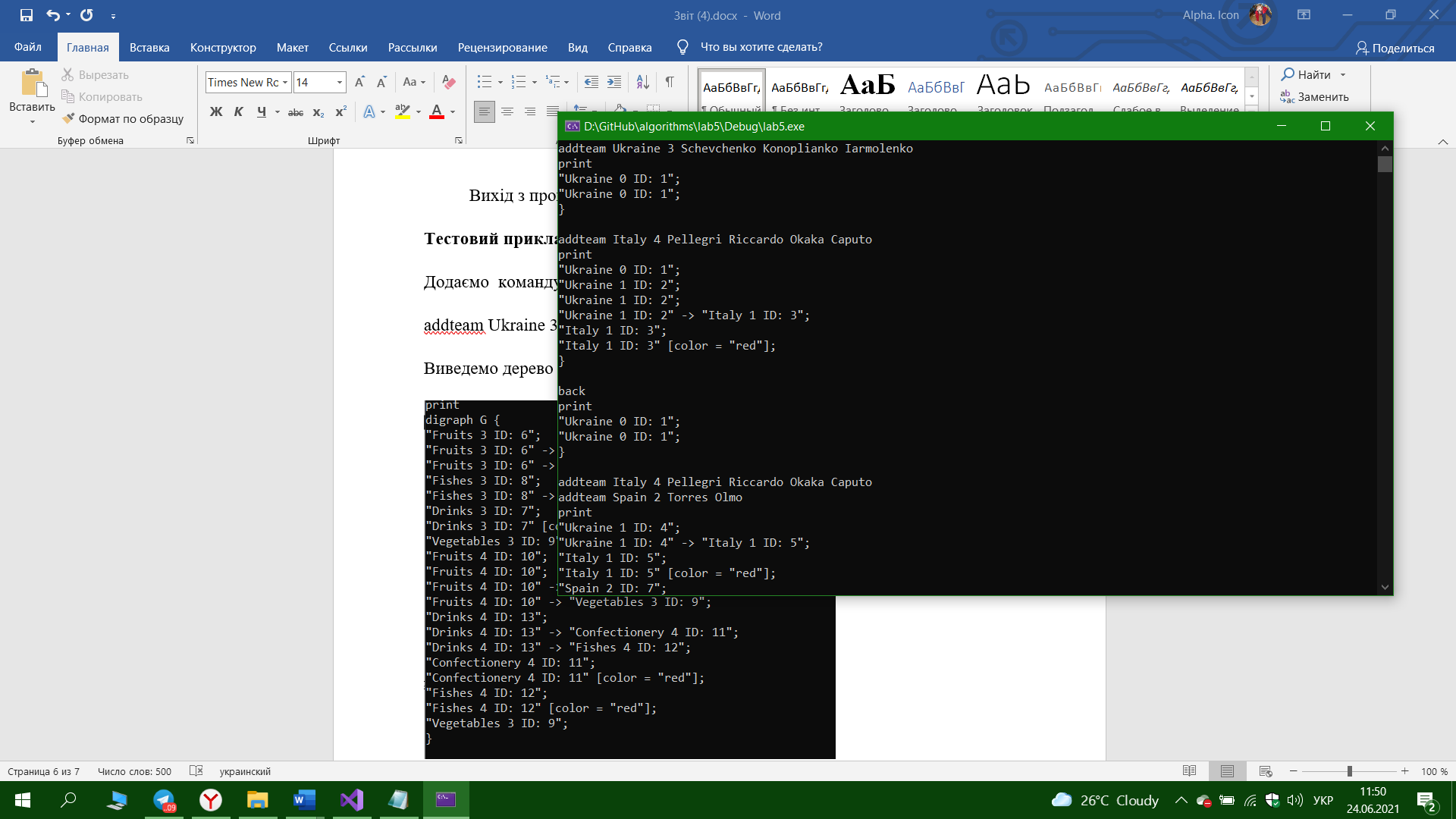
Додаємо команду

addteam Ukraine 3 Schevchenko Konoplianko Iarmolenko

Виведемо дерево



Додамо ще команду и виведемо. Відмінимо додавання і виведемо



**Висновки**

Реалізована персистентна множина на основі червоно-чорного дерева дозволяє зберігати попередні стани структури, що дозволяє швидко отримати попередню версію множини.

**Література**

* Лекція №4