**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Кафедра інформаційних систем**

**Алгоритми та складність**

**Завдання № 3**

**Звіт**

Артюхов Яків

Група К-28

**⦁ Умова завдання.**

Реалізувати розширюване дерево (splay tree).

Розширюване дерево (англ. splay tree) є двійковим деревом пошуку, у якому підтримується збалансованість. 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)} O(log n).

Розширюване дерево придумали Роберт Тарьян і Данієль Слейтор в 1985 році.

**⦁ Опис алгоритму.**

Splay (розширення)

Основна операція дерева полягає в переміщенні вершини в корінь за допомогою послідовного виконання трьох, наведених нижче, операцій: Zig, Zig-Zig і Zig-Zag.

1.Zig: Node is child of root (the node has no grandparent). Node is either a left child of root (we do a right rotation) or node is a right child of its parent (we do a left rotation).

2.Node has both parent and grandparent. There can be following subcases.

а) Zig-Zig and Zag-Zag Node is left child of parent and parent is also left child of grand parent (Two right rotations) OR node is right child of its parent and parent is also right child of grand parent (Two Left Rotations).

b) Zig-Zag and Zag-Zig Node is left child of parent and parent is right child of grand parent (Left Rotation followed by right rotation) OR node is right child of its parent and parent is left child of grand parent (Right Rotation followed by left rotation).

Search (пошук елемента)

Пошук виконується як в звичайному двійковому дереві пошуку. При знаходженні елементу запускаємо Splay для нього.

Insert (додавання елемента)

Following cases to insert a key k in splay tree.

Аllocate memory for new node and compare root’s key with k.

a) If k is smaller than root’s key, make root as right child of new node, copy left child of root as left child of new node and make left child of root as nullptr.

b) If k is greater than root’s key, make root as left child of new node, copy right child of root as right child of new node and make right child of root as nullptr.

Delete (видалення елемента)

Following cases to delete a key k from splay tree.

The key k is present.

Split the tree into two trees Tree1 = root’s left subtree and Tree2 = root’s right subtree and delete the root node.

Let the root’s of Tree1 and Tree2 be Root1 and Root2 respectively.

If Root1 is NULL: Return Root2.

Else, Splay the maximum node (node having the maximum value) of Tree1.

After the Splay procedure, make Root2 as the right child of Root1 and return Root1.

**⦁ Аналіз алгоритму.**

*Амортизована вартість будь-якої операції на розширюваному дереві становить*  *O(log(n)), де*  *це кількість вузлів у дереві. Будь-яка операція на дереві може зайняти O(n) часу, але, як правило, також робить дерево більш збалансованим, так що з плином часу середня вартість операції э O(log(n)).*

*Для отримання заявленої оцінки O(log(n)) використаємо метод потеніцалів. По-перше визначимо вагу, суму і функцію рангу для кожного вузла:*

* *Кожен вузол X має вагу w(X)>0 (з метою аналізу, може бути будь-якою)*
*  *(вага вузла дорівнює сумі ваг усіх вузлів його піддерева)*
* 

*Визначимо потенціал усього розширюваного дерева в будь-який момент i як суму рангів у дереві:*



*Потенціал вимірює наскільки збалансованим є дерево. Чим менше потенціал тим ліпше збалансоване дерево. Амортизаційна вартість операції складається з дійсної вартості плюс зміна потенціалу дерева.*

Амортизована вартість одного кроку розширення

Нехай  буде ранг  перед розширенням і нехай  буде ранг  після розширення. Тоді амортизаційна вартість розширення

у випадку ZIG 

у випадках ZIG-ZIG і ZiG-ZAG 