Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Лабораторна робота №3

“Реалізувати В-дерево”

Виконав студент 2-го курсу

Групи ІПС-22

Каширець Роман Віталійович

2025

**Завдання**

Реалізувати В-дерево

**Теорія**

***Б-дерева****([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0" \o "Англійська мова) B-tree) — це*[*збалансована деревоподібна*](https://uk.wikipedia.org/wiki/%D0%97%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE)*структура даних, яка підтримує відсортовані дані та дозволяє здійснювати пошук,*[*послідовний доступ*](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D1%96%D0%B4%D0%BE%D0%B2%D0%BD%D0%B8%D0%B9_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF)*, вставки та видалення в*[*логарифмічному часі*](https://uk.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D0%BE%D0%B2%D0%B0_%D1%81%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D1%96%D1%81%D1%82%D1%8C)*. Б-дерево узагальнює*[*двійкове дерево пошуку*](https://uk.wikipedia.org/wiki/%D0%94%D0%B2%D1%96%D0%B9%D0%BA%D0%BE%D0%B2%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D0%BE%D1%88%D1%83%D0%BA%D1%83)*, допускаючи*[*вузли*](https://uk.wikipedia.org/wiki/%D0%92%D1%83%D0%B7%D0%BE%D0%BB_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))*з більше ніж двома дочірніми елементами.*[*[2]*](https://uk.wikipedia.org/wiki/%D0%91-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE#cite_note-FOOTNOTEComer1979-2)*На відміну від інших самобалансуючих двійкових дерев пошуку, Б-дерево добре підходить для систем зберігання даних, які читають і записують відносно великі блоки даних, таких як бази даних і файлові системи. Б-дерево забезпечує ефективне збереження інформації на магнітних дисках та інших пристроях з прямим доступом. Б-дерева схожі на*[*червоно-чорні*](https://uk.wikipedia.org/wiki/%D0%A7%D0%B5%D1%80%D0%B2%D0%BE%D0%BD%D0%BE-%D1%87%D0%BE%D1%80%D0%BD%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE)*, різниця в тому, що в Б-дереві вузол може мати багато дітей, на практиці до тисячі, залежно від характеристик використовуваного диска. Завдяки цьому*[*константа*](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D0%B0%D0%BD%D1%82%D0%B0)*в оцінці O(log n) для висоти дерева менша, ніж для*[*червоно-чорних дерев*](https://uk.wikipedia.org/wiki/%D0%A7%D0%B5%D1%80%D0%B2%D0%BE%D0%BD%D0%BE-%D1%87%D0%BE%D1%80%D0%BD%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE)*. Як і червоно-чорні дерева, Б-дерева дозволяють реалізувати багато операцій з*[*множинами*](https://uk.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%B0)*розміру n за час O(log n).*

*Вузол x, який зберігає n[x] ключів, має n[x]+1 дітей. Ключі, що зберігаються в x служать границями, що розділяють всіх його нащадків на n[x]+1 груп; за кожну групу відповідає один з нащадків x. При пошуку в Б-дереві ми порівнюємо шуканий ключ з n[x] ключами, що зберігаються в x, і за результатами порівняння вибираємо одного з n[x]+1 нащадків.*

**Алгоритм**

**1. Алгоритм порівняння комплексних чисел**

**Функція CompareTo(ComplexInt a, ComplexInt b):**

1. Обчислити модуль a: |a| = √(a.Real² + a.Imaginary²)

2. Обчислити модуль b: |b| = √(b.Real² + b.Imaginary²)

3. Якщо |a| > |b|, повернути 1

4. Якщо |a| < |b|, повернути -1

5. Якщо модулі рівні:

- Якщо a.Real > b.Real, повернути 1

- Якщо a.Real < b.Real, повернути -1

6. Повернути 0 (числа ідентичні)  
**2. Алгоритм пошуку в B-дереві**Функція Search(node, key):

1. Ініціалізувати i = 0

2. Поки i < кількості ключів у node та key > node.Keys[i]:

- i = i + 1

3. Якщо i < кількості ключів у node та key == node.Keys[i]:

- Повернути node (знайдено)

4. Якщо node - листок:

- Повернути null (не знайдено)

5. Рекурсивно викликати Search(node.Children[i], key)  
**3. Алгоритм вставки в B-дерево**Функція Insert(key):

1. Якщо корінь повний (кількість ключів == 2t-1):

- Створити новий корінь s

- Зробити старий корінь дочірнім вузлом s

- Розділити старий корінь (SplitChild)

- Вставити key в s (InsertNonFull)

2. Інакше:

- Вставити key в корінь (InsertNonFull)

Функція InsertNonFull(node, key):

1. Якщо node - листок:

- Знайти позицію i для вставки

- Вставити key в node.Keys на позицію i

2. Інакше:

- Знайти дочірній вузол i, куди потрібно вставити

- Якщо дочірній вузол i повний:

\* Розділити його (SplitChild)

\* Визначити, в який з нових дочірніх вузлів вставляти

- Рекурсивно викликати InsertNonFull для дочірнього вузла i

Функція SplitChild(parent, childIndex):

1. Взяти дочірній вузол child за індексом childIndex

2. Створити новий вузол newChild

3. Перемістити середній ключ з child в parent

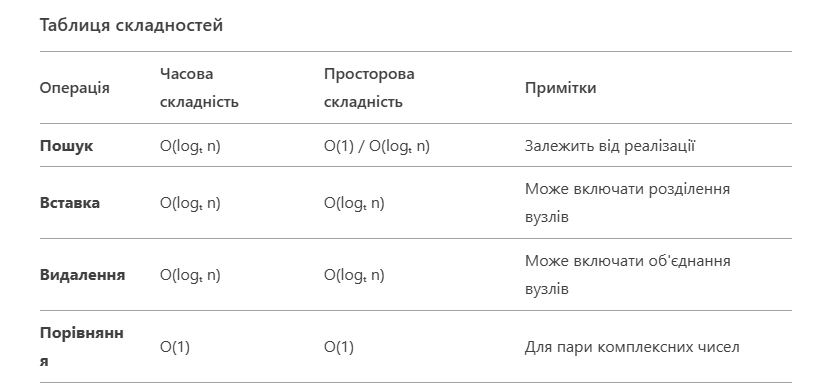
4. Перемістити праву половину ключів з child в newChild

5. Якщо child не листок:

- Перемістити праву половину дочірніх вузлів з child в newChild

6. Додати newChild як дочірній вузол parent після child

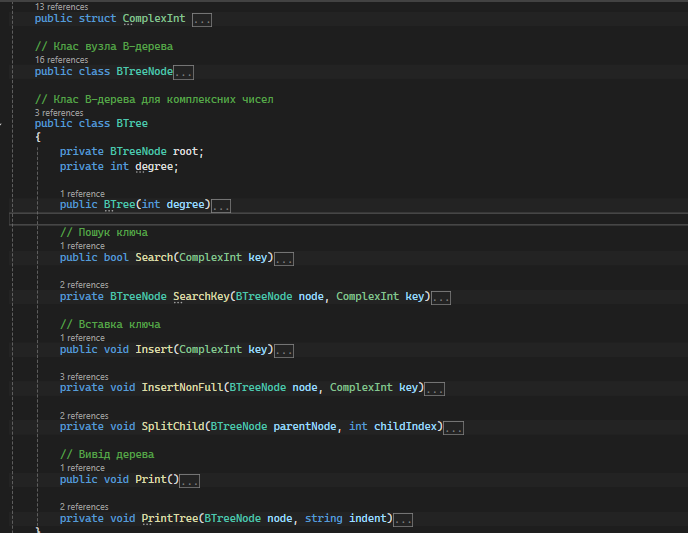
**Складність алгоритму**

****

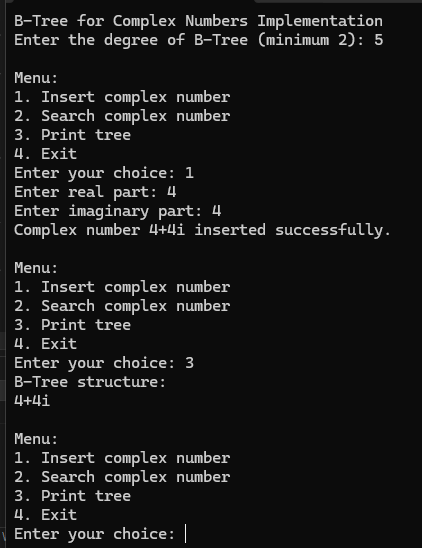
**Мова програмування**

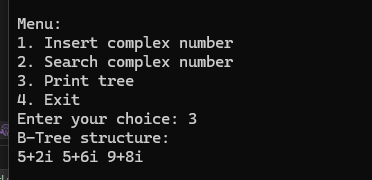
C#

**Модулі програми**



**Інтерфейс користувача**





**Висновки**

Ця реалізація B-дерева є ефективним рішенням для роботи з впорядкованими колекціями комплексних чисел, зберігаючи всі переваги класичної структури даних та забезпечуючи зручний інтерфейс для користувача.

**Література**

* **Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.** *"Introduction to Algorithms"* (3rd Edition) – MIT Press, 2009.  
  *(Розділи про B-дерева, їх складність та застосування.)*
* **Knuth, D.E.** *"The Art of Computer Programming, Volume 3: Sorting and Searching"* – Addison-Wesley, 1998.  
  *(Детальний опис B-дерев, їх варіацій та оптимізацій.)*
* **Sedgewick, R., Wayne, K.** *"Algorithms"* (4th Edition) – Addison-Wesley, 2011.  
  *(Практичний підхід до реалізації дерев у програмуванні.)*