

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 2 з дисципліни
«Проектування алгоритмів»

„Проектування структур даних”

Виконав(ла)

IT-03 Чабан Антон Євгенович
(шифр, прізвище, ім'я, по батькові)

Перевірів

Головченко М.Н.
(прізвище, ім'я, по батькові)

Київ 2021

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи проектування та обробки складних структур даних.

2 ЗАВДАННЯ

Відповідно до варіанту (таблиця 2.1), записати алгоритми пошуку, додавання, видалення і редагування запису в структурі даних за допомогою псевдокоду (чи іншого способу по вибору).

Записати часову складність пошуку в структурі в асимптотичних оцінках.

Виконати програмну реалізацію невеликої СУБД, з функціями пошуку (алгоритм пошуку у вузлі структури згідно варіанту таблиця 2.1, за необхідності), додавання, видалення та редагування записів (запис складається із ключа і даних, ключі унікальні і цілочисельні, даних може бути декілька полів для одного ключа, але достатньо одного рядка фіксованої довжини). Для зберігання даних використовувати структуру даних згідно варіанту (таблиця 2.1).

Заповнити базу випадковими значеннями до 10000 і зафіксувати середнє (із 10-15 пошуків) число порівнянь для знаходження запису по ключу.

Зробити висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	Структура даних
19	В-дерево $t=50$, бінарний пошук

3.1 Псевдокод алгоритмів

B-TREE-SEARCH(x, k)

```

1   $i \leftarrow 1$ 
2  while  $i \leq n[x]$  and  $k > key_i[x]$ 
3      do  $i \leftarrow i + 1$ 
4  if  $i \leq n[x]$  and  $k = key_i[x]$ 
5      then return  $(x, i)$ 
6  if  $leaf[x]$ 
7      then return NIL
8  else DISK-READ( $c_i[x]$ )
9      return B-TREE-SEARCH( $c_i[x], k$ )

```

B-TREE-CREATE(T)

```

1   $x \leftarrow \text{ALLOCATE-NODE}()$ 
2   $leaf[x] \leftarrow \text{TRUE}$ 
3   $n[x] \leftarrow 0$ 
4  DISK-WRITE( $x$ )
5   $root[T] \leftarrow x$ 

```

B-TREE-SPLIT-CHILD(x, i, y)

```

1   $z \leftarrow \text{ALLOCATE-NODE}()$ 
2   $leaf[z] \leftarrow leaf[y]$ 
3   $n[z] \leftarrow t - 1$ 
4  for  $j \leftarrow 1$  to  $t - 1$ 
5      do  $key_j[z] \leftarrow key_{j+t}[y]$ 
6  if not  $leaf[y]$ 
7      then for  $j \leftarrow 1$  to  $t$ 
8          do  $c_j[z] \leftarrow c_{j+t}[y]$ 
9   $n[y] \leftarrow t - 1$ 
10 for  $j \leftarrow n[x] + 1$  downto  $i + 1$ 
11     do  $c_{j+1}[x] \leftarrow c_j[x]$ 
12  $c_{i+1}[x] \leftarrow z$ 
13 for  $j \leftarrow n[x]$  downto  $i$ 
14     do  $key_{j+1}[x] \leftarrow key_j[x]$ 
15  $key_i[x] \leftarrow key_t[y]$ 
16  $n[x] \leftarrow n[x] + 1$ 
17 DISK-WRITE( $y$ )
18 DISK-WRITE( $z$ )
19 DISK-WRITE( $x$ )

```

```

B-TREE-INSERT( $T, k$ )
1   $r \leftarrow \text{root}[T]$ 
2  if  $n[r] = 2t - 1$ 
3      then  $s \leftarrow \text{ALLOCATE-NODE}()$ 
4           $\text{root}[T] \leftarrow s$ 
5           $\text{leaf}[s] \leftarrow \text{FALSE}$ 
6           $n[s] \leftarrow 0$ 
7           $c_1[s] \leftarrow r$ 
8          B-TREE-SPLIT-CHILD( $s, 1, r$ )
9          B-TREE-INSERT-NONFULL( $s, k$ )
10 else B-TREE-INSERT-NONFULL( $r, k$ )

```

```

B-TREE-INSERT-NONFULL( $x, k$ )
1   $i \leftarrow n[x]$ 
2  if  $\text{leaf}[x]$ 
3      then while  $i \geq 1$  and  $k < \text{key}_i[x]$ 
4          do  $\text{key}_{i+1}[x] \leftarrow \text{key}_i[x]$ 
5               $i \leftarrow i - 1$ 
6           $\text{key}_{i+1}[x] \leftarrow k$ 
7           $n[x] \leftarrow n[x] + 1$ 
8          DISK-WRITE( $x$ )
9  else while  $i \geq 1$  and  $k < \text{key}_i[x]$ 
10     do  $i \leftarrow i - 1$ 
11      $i \leftarrow i + 1$ 
12     DISK-READ( $c_i[x]$ )
13     if  $n[c_i[x]] = 2t - 1$ 
14         then B-TREE-SPLIT-CHILD( $x, i, c_i[x]$ )
15         if  $k > \text{key}_i[x]$ 
16             then  $i \leftarrow i + 1$ 
17     B-TREE-INSERT-NONFULL( $c_i[x], k$ )

```

3.2 Часова складність пошуку

Ми використовуємо бінарний пошук в ноді, тому часова складність $O(\log n)$

3.3 Програмна реалізація

3.3.1 Вихідний код

[*Вихідний код завантажено на GitHub*](#)

3.3.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для додавання і пошуку запису.

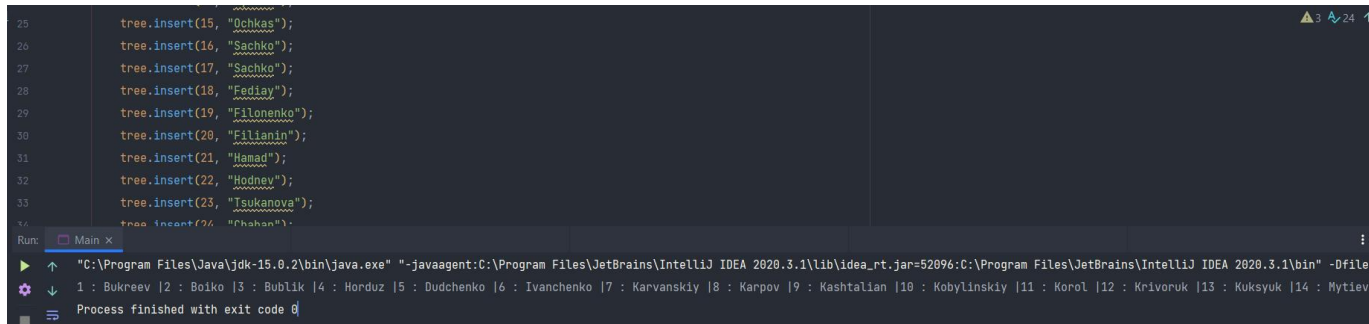


Рисунок 3.1 –Додавання запису

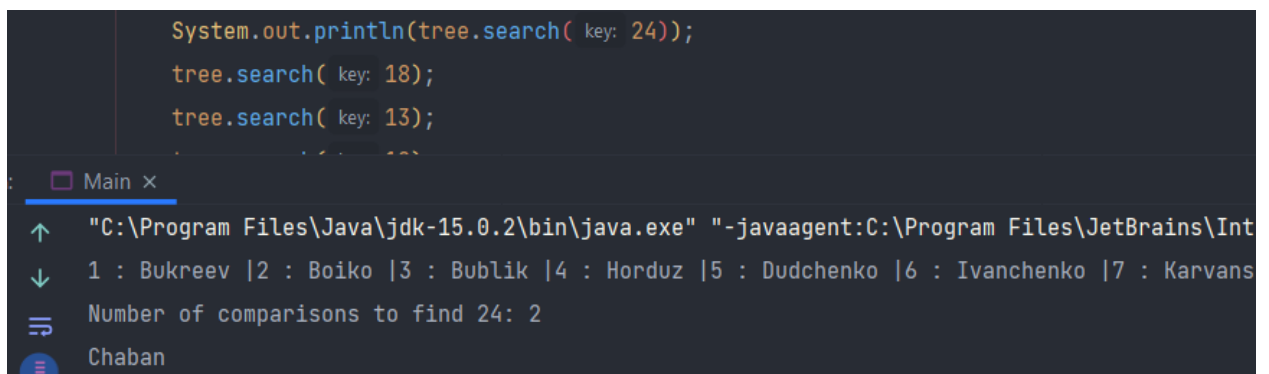


Рисунок 3.2 – Пошук запису

3.4 Тестування алгоритму

3.4.1 Часові характеристики оцінювання

В таблиці 3.1 наведено кількість порівнянь для 15 спроб пошуку запису по ключу.

Таблиця 3.1 – Число порівнянь при спробі пошуку запису по ключу

Номер спроби пошуку	Число порівнянь
1	Number of comparisons to find 24: 2
2	Number of comparisons to find 18: 4
3	Number of comparisons to find 13: 4
4	Number of comparisons to find 12: 3
5	Number of comparisons to find 8: 3
6	Number of comparisons to find 6: 4
7	Number of comparisons to find 15: 3
8	Number of comparisons to find 11: 4
9	Number of comparisons to find 1: 3
10	Number of comparisons to find 3: 2
11	Number of comparisons to find 25: 4
12	Number of comparisons to find 21: 1
13	Number of comparisons to find 2: 4
14	Number of comparisons to find 17: 2
15	Number of comparisons to find 19: 3

ВИСНОВОК

В рамках лабораторної роботи я ознайомився з такою структурою даних як B-Tree та її програмною реалізацією на практиці.