Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Катедра «Комп'ютерна інженерія та програмування»

ЗВІТ

про виконання лабораторної роботи №9

з навчальної дисципліни «Алгоритми та структури даних»

Варіант 9

Виконав студент:

Ульянов Кирило Юрійович
Група: КН-1023b

Перевірив:

старший викладач

Бульба С.С.

Харків-2024

1 Мета роботи

Набути навичок та закріпити знання при виконанні операцій пошуку.

2 Хід роботи

Розробити та налагодити програму, в якій реалізувати два алгоритми пошуку відповідно до завдання. Порівняти алгоритми за часом роботи.

На етапі тестування для кожного з алгоритмів (варіанти 4 – 15) визначити кількість порівнянь у наборі даних з різною кількістю елементів (20, 100, 1000, 10000) визначити час пошуку, заповнити таблицю за формою табл. 9.1, побудувати графіки, зробити висновки. При роботі з текстом (варіанти 1 – 3) змінювати довжину текста та зразка.

Моє завдання: Лінійний та лінійний з бар’єром пошуки у лінійному списку.

Для виконання лабораторної був використаний універсальний список з лабораторної 7.

2.1 Реалізація функції лінійного пошуку

```
int linear_search_linked_list(Node *head, void *key, CompareFunc comparer)
{
  int comparisons = 0;
  Node *current = head;

  while (current)
  {
    comparisons++;
    if (comparer(current->data, key) == 0)
    {
      return comparisons;
    }
    current = current->next;
  }
  return comparisons;
}
```

2.2 Реалізація функції лінійного пошуку з бар'єром

```
int linear_search_with_barrier(Node **head, void *key, CompareFunc
    comparer)
{
  if (!*head)
  {
    return 0;
  }

  insert_head_linked_list(head, key);
  int comparisons = 0;
  Node *current = *head;

  while (current)
  {
    comparisons++;
    if (comparer(current->data, key) == 0)
    {
      break;
    }
    current = current->next;
  }

  delete_node_linked_list(head, key, comparer, NULL);
  return comparisons;
}
```

2.3 Реалізація додаткових методів для списку з цілими числами

```c
int int_comparer(const void *a, const void *b)
{
  return *(int *)a - *(int *)b;
}

void fill_linked_list(Node **head, int size)
{
  for (int i = 0; i < size; i++)
  {
    int *value = (int *)malloc(sizeof(int));
    *value = generateRandomInt(1, 10000);
    insert_head_linked_list(head, value);
  }
}

void free_int_data(void *data)
{
  free(data);
}
```

2.4 Реалізація лабораторної програми

Було додано додаткову функцію для комфортного тестування та обробки результатів роботи функції.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "linked_list.h"
#include "general_utils.h"

void test_search(Node **head, void *key, CompareFunc comparer, int (*
    search_func)(Node *, void *, CompareFunc))
{
  clock_t start = clock();
  int comparisons = search_func(*head, key, comparer);
  clock_t end = clock();

  double time_taken_ms = ((double)(end - start)) / CLOCKS_PER_SEC * 1000;

  if (comparisons > 0 && comparisons <= 10000)
  {
    printf("Key found. Comparisons: %d \n\033[34 m\033[0m Time: \033[32m%.3
        f\033[0m milliseconds\n", comparisons, time_taken_ms);
  }
  else
  {
    printf("Key not found. Comparisons: %d \n\033[34 m\033[0m Time: \033[32
        m%.3f\033[0m milliseconds\n", comparisons, time_taken_ms);
  }
}
```

```
25
26  void task1()
27  {
28    srand(time(NULL));
29
30    int sizes[] = {20, 100, 1000, 10000};
31    int num_sizes = sizeof(sizes) / sizeof(sizes[0]);
32
33    for (int i = 0; i < num_sizes; i++)
34    {
35      int size = sizes[i];
36      Node *head = NULL;
37
38      printf("\n\033[34mTesting for list size: %d\033[0m\n", size);
39      fill_linked_list(&head, size);
40
41      int key = generateRandomInt(1, 1000);
42      printf("Searching for key: %d\n\n", key);
43
44      printf("\033[34 m\033[0m Linear Search \033[34 m\033[0m\n");
45      test_search(&head, &key, int_comparer, linear_search_linked_list);
46      puts("");
47      printf("\033[34 m\033[0m Linear Search with Barrier \033[34 m\033[0m\n"
            );
48      test_search(&head, &key, int_comparer, (int (*)(Node *, void *,
            CompareFunc))linear_search_with_barrier);
49      puts("\n------------------------------");
50
51      destroy_linked_list(&head, free_int_data);
52    }
53
54    return 0;
55  }
```

2.5 Результати роботи програми:

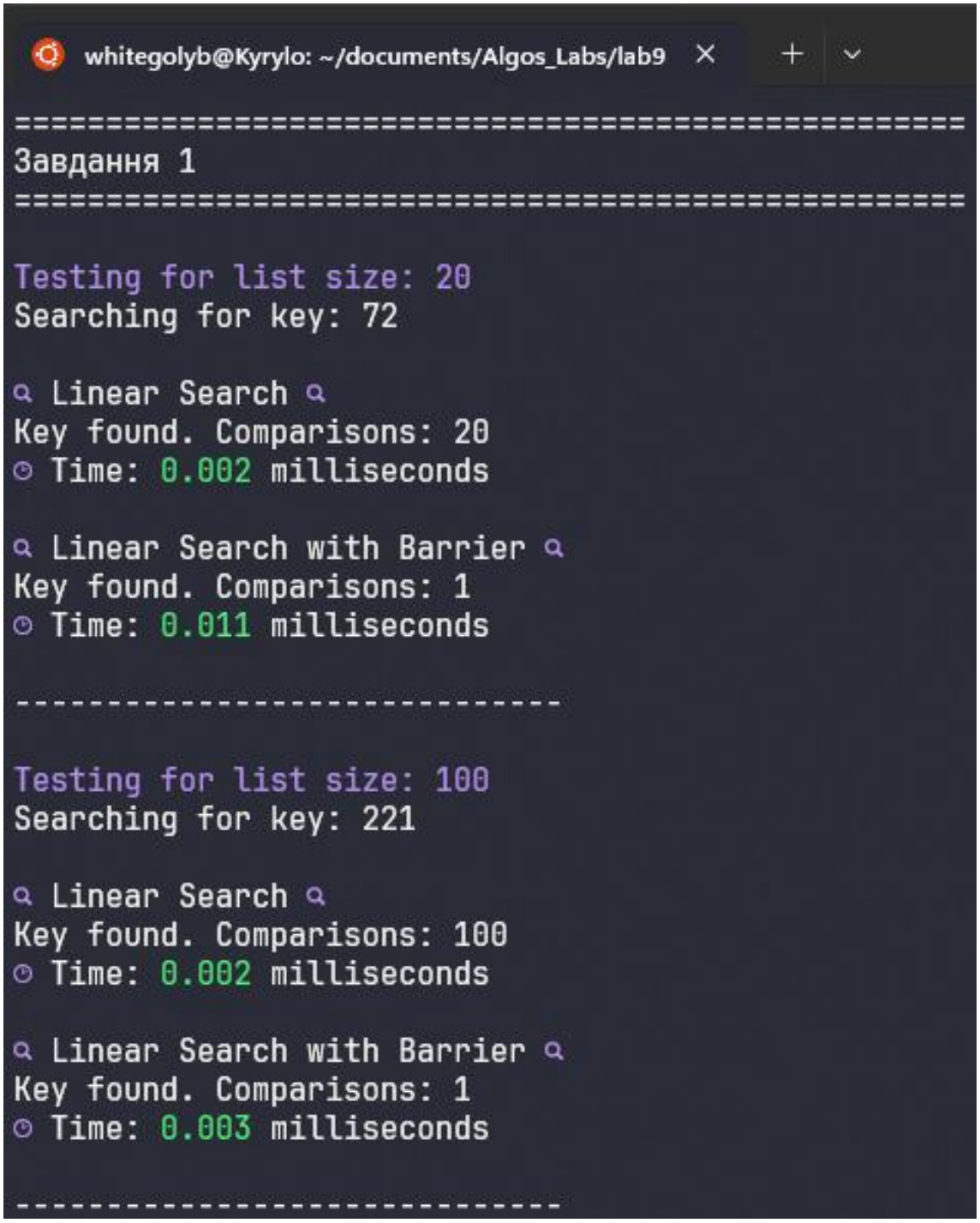


Рис. 1. Результат пошуку для списку розміром 20 та 100

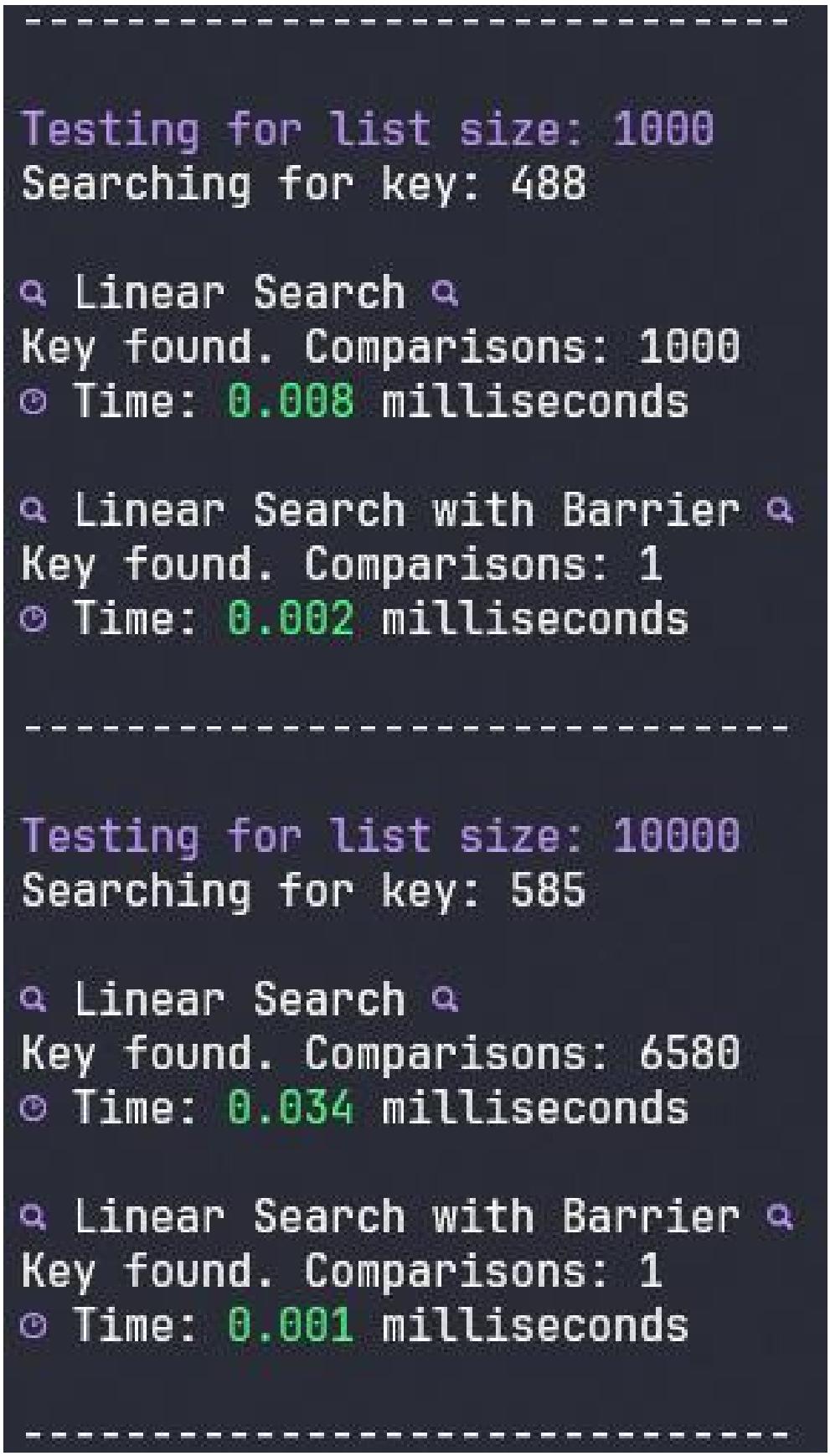


Рис. 2. Результат пошуку для списку розміром 1000 та 10000

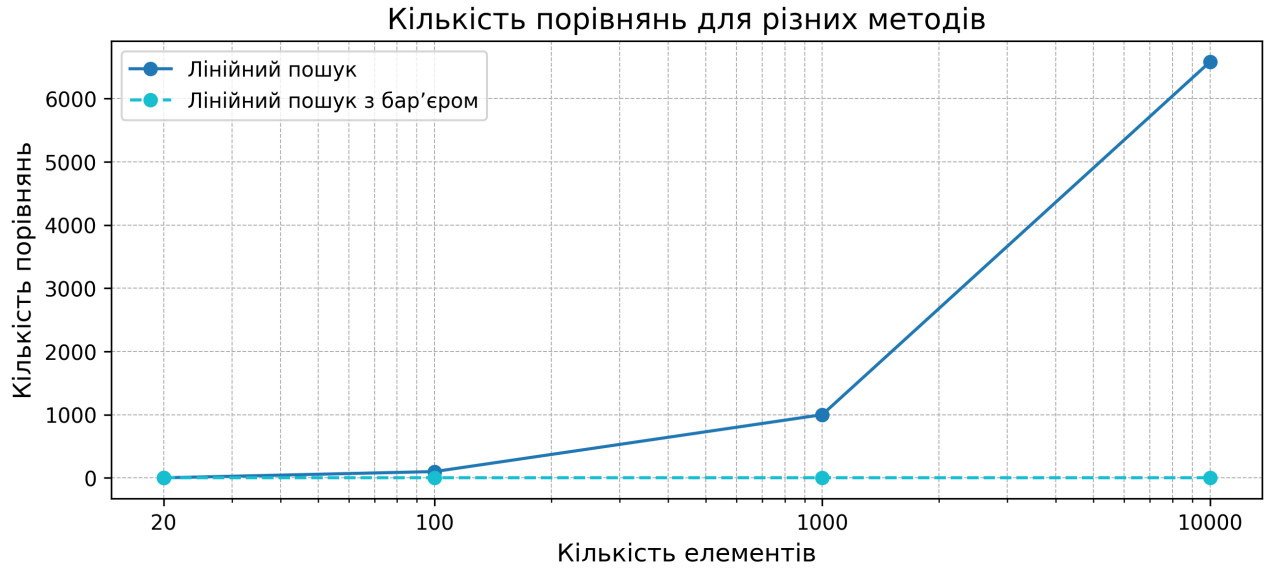
Таблиця 1. Таблиця результату для лінійного пошуку

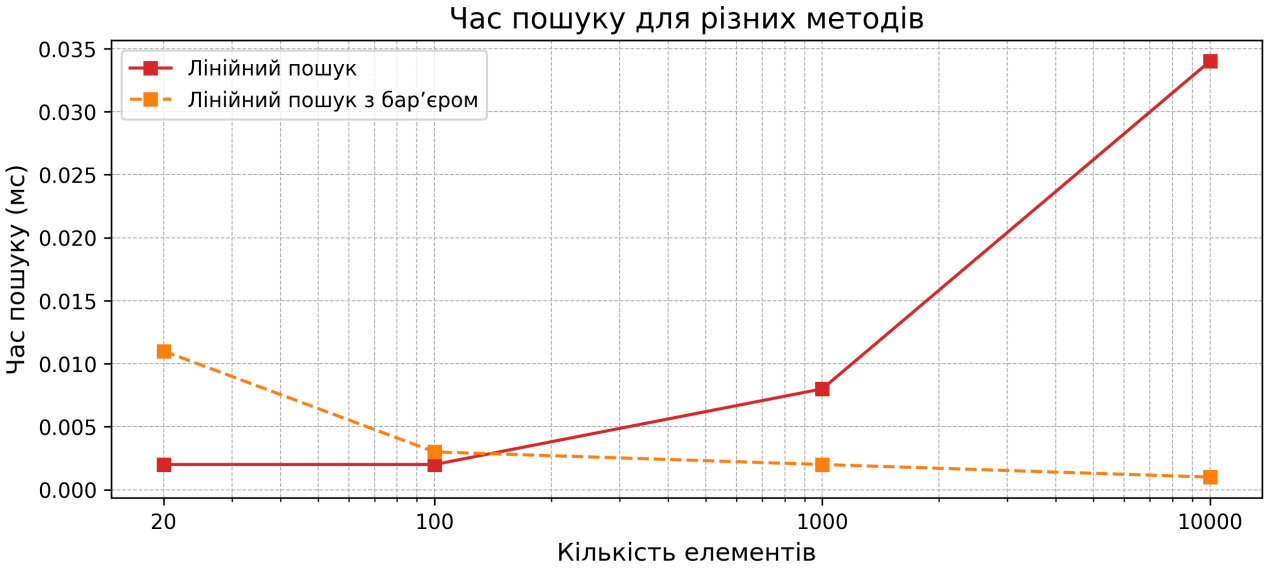
Кількість елементів	20	100	1000	10000
Кількість порівнянь	1	100	1000	6580
Час пошуку	0,002	0,002	0,008	0,034

Таблиця 2. Таблиця результату для лінійного пошуку з бар’єром

Кількість елементів	20	100	1000	10000
Кількість порівнянь	1	1	1	1
Час пошуку	0,011	0,003	0,002	0,001

Графіки побудовані на основі таблиць





3 Висновки

В ході виконання лабораторної роботи було розроблено два методи пошуку ключа у лінійному списку та порівняно їх результати роботи. Порівняння представлено у вигляді таблиць та графіків за параметрами співпадінь та часу виконання відповідно.

Алгоритми лінійного пошуку краще за все підходять для списків, можна зазначити що пошук з бар’єром веде себе більш стабільніше за порівняннями при збільшенні обсягу списку та швидше за часом ніж лінійний.