

Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра системного аналізу та теорії прийняття рішень

Звіт
з лабораторної роботи № 1
на тему:
«Рядки і послідовності»

Студента другого курсу
групи К-23(2)
Міщука Романа Андрійовича
Факультету комп'ютерних наук
та кібернетики

Київ – 2023

1. Пошук точного підрядка в рядку

Опис алгоритму: знаходження точного входження рядка x в рядок s . Для цього використовується алгоритм Моріса-Прата. Він є покращенням наївного алгоритму. Це покращення досягається за допомогою попереднього обрахування довжини зрушення у випадку виявлення розбіжності зразка.

Результат роботи:

```
C:\Users\mario\Documents\Documents\University\OOП  
Enter task number (1-10): 1  
Enter string s : abcde  
Enter string x : bcd  
Index of occurrence (0-indexed) : 1  
  
Enter task number (1-10): |
```

0 <u>1</u> 234
abcde
bcd

2. Нечіткий пошук (близькі за відстанню)

Опис алгоритму: знаходження входжень зразку x у рядок y , за умови, що входження може мати k відмінностей від зразка. Для цього використовується алгоритм Ландау-Вішкіна. Алгоритм ґрунтується на перефразуванні наївного методу динамічного програмування, що дозволяє використовувати суфіксні дерева для пришвидшення. Проте через це необхідно буде перед запуском алгоритму побудувати дерево для $y\#x\$$.

Результат роботи:

```
Enter task number (1-10): 2
Enter string y : abcde
Enter string x : abd
Enter int k : 1
Indexes of occurrences ends (0-indexed) :
1 2 3

Enter task number (1-10): |
```

0 <u>1</u> 234
abcde
ab <u>d</u>

012 <u>3</u> 4
ab <u>c</u> de
ab <u>d</u>

0123 <u>4</u>
abc <u>d</u> e
ab d

3. Перевірка на підпоследовність

Опис алгоритму: перевірка, чи можливо видаленням деякої кількості символів із рядка у отримати рядок х. Алгоритм реалізовано простим скануванням символів у рядку у.

Результат роботи:

```
Enter task number (1-10): 3
Enter string y : abcde
Enter string x : bcd
x IS in y

Enter task number (1-10): 3
Enter string y : abcde
Enter string x : abf
x is NOT in y

Enter task number (1-10): |
```

01234
abcde
bcd

01234
abcde
abf

4. Загальні підпоследовності. Відстань

Опис алгоритму: знаходження відстані (кількості відмінностей) між рядками a та b. Застосовується алгоритм Вагнера-Фішера. Для виконання завдання алгоритм методом динамічного програмування рахує відстані між різними комбінаціями префіксів a та b.

Результат роботи:

```
Enter task number (1-10): 4
Enter string a : abc
Enter string b : ac
Distance between a and b: 1

Enter task number (1-10): 4
Enter string a : abc
Enter string b : ba
Distance between a and b: 2

Enter task number (1-10): |
```

012	012
a b c	a b c
a c	b a

5. Пошук lis та lcs

lis – longest incremental subsequence – найбільша зростаюча підпослідовність;

lcm – longest common subsequence – найбільша спільна підпослідовність;

Опис алгоритмів:

Для знаходження *lis* послідовності чисел *a*, застосовується алгоритм Робінсона-Шенстеда. Він ґрунтується на застосуванні методу динамічного програмування (зберігання найменшого члену зростаючих підпослідовностей певної довжини) для прискорення роботи.

Для знаходження *lcm* двох послідовностей чисел *x* та *y* також використовується метод динамічного програмування. Він рахує довжину спільної підпослідовності для різних комбінацій префіксів *x* та *y*.

Результат роботи:

```
Enter task number (1-10): 5
Select one of the options:
  1. lis
  2. lcs
: 1
Computing lis
Enter numerical sequence a
(space is delimiter) : 1 2 3 5 6 1 0 -5 1 4 3 4 5
Answer: 1 2 3 4 5

Enter task number (1-10): 5
Select one of the options:
  1. lis
  2. lcs
: 2
Computing lcs
Enter numerical sequence x
(space is delimiter) : 1 2 3 4
Enter numerical sequence y
(space is delimiter) : 1 10 3 10
Answer: 2
```

6. Максимальний повторюваний підрядок

Опис алгоритму: знаходження такого підрядку максимальної довжини, який би повторювався хоча б 2 рази. Ця задача є досить легко виконуваною, використовуючи методи, реалізовані в пункті 2 лабораторної роботи. Адже завдання відповідає пошуку внутрішньої вершини суфіксного дерева рядку. Критерієм пошуку є довжина підрядку, якому відповідає ця вершина.

Результат роботи:

```
Enter task number (1-10): 6
Enter string s : abdaskbdja
Longest repeated subsequence : bd

Enter task number (1-10): |
```

abdaskbdja

7. Загальні елементи двох масивів

Опис алгоритму: знаходження таких чисел, які б одночасно належали послідовностям а та b. Для розв'язання використовується сортування елементів послідовностей, з подальшим ітеруванням по ним.

Результат роботи:

```
Enter task number (1-10): 7
Enter numerical sequence a
(space is delimiter) : 1 2 3 4
Enter numerical sequence b
(space is delimiter) : 5 3 8
Common elements of a and b : 3
```


8. Бінарний пошук

Опис алгоритму: знаходження індекса числа to_find у відсортованій послідовності чисел.

Результат роботи:

```
Enter task number (1-10): 8
Enter numerical sequence list
(space is delimiter) : 1 2 3 4 4 4 6 6 6 7
Enter int to_find : 6
Enter int left (enter -1 for default val) : -1
Enter int left (enter -1 for default val) : -1
Index of to_find (zero indexed) : 7
```

0	1	2	3	4	5	6	<u>7</u>	8	9
1	2	3	4	4	4	6	6	6	7
							6		

9. Інтерполяційний пошук

Опис алгоритму: знаходження індекса числа `to_find` у відсортованій послідовності чисел. Схожий за принципом роботи на бінарний пошук.

Основна відмінність полягає у стрибках, які робить цей алгоритм.

Інтерполяційний пошук ділить відрізок не навпіл, а інтерполює його (вважаючи послідовність арифметичною прогресією) і переходить ймовірного індекса елемента. Це дозволяє прискорити пошук.

Результат роботи:

```
Enter task number (1-10): 9
Enter numerical sequence list
(space is delimiter) : 1 2 3 4 4 4 6 6 6 7
Enter int to_find : 6
Enter int left (enter -1 for default val) : -1
Enter int left (enter -1 for default val) : -1
Index of to_find (zero indexed) : 7
```

0	1	2	3	4	5	6	<u>7</u>	8	9
1	2	3	4	4	4	6	6	6	7
						6			

10. Бінарний пошук з визначенням найближчих вузлів

Опис алгоритму: знаходження індекса числа `to_find` у відсортованій послідовності чисел, за зразком бінарного пошуку. Проте крім послідовності та числа, цей алгоритм також отримує додаткову інформацію у вигляді приблизного місця знаходження числа `guess_index`. Це дозволяє запускати бінарний пошук на вузькому відрізку, і, відповідно, прискорити алгоритм.

Результат роботи:

```
Enter task number (1-10): 10
Enter numerical sequence list
(space is delimiter) : 1 2 3 4 4 4 6 6 6 7
Enter int to_find : 6
Enter int guess_index : 5
Index of to_find (zero indexed) : 8
```

[illegible]