

Київський національний університет імені Тараса Шевченка

КРЕНЕВИЧ А.П.

**Методичні вказівки
до лабораторних занять із дисципліни
«Алгоритми і структури даних»**

для студентів механіко-математичного факультету

Київ – 2018

УДК 519.942+550

Рецензенти:
доктор фіз.-мат. наук, професор
доктор фіз.-мат. наук,

*Рекомендовано до друку вченою радою механіко-математичного
факультету
(протокол № __ від __ _____ 201__ року)*

Крєневич А.П.

Методичні вказівки до лабораторних занять із дисципліни «Алгоритми і структури даних» для студентів механіко-математичного факультету – К.: ВПЦ "Київський Університет", 2018. – __ с.

Посібник містить перелік завдань для самостійної роботи з дисципліни «Алгоритми і структури даних», що викладається студентам механіко-математичного факультету Київського національного університету імені Тараса Шевченка. Він містить завдання для засвоєння основних понять цього курсу, таких як рекурентні співвідношення, рекурсія, аналіз складності алгоритмів, лінійні рекурсивні структури даних, графи, дерева, динамічне програмування тощо.

Для студентів механіко-математичного факультету та викладачів, які проводять заняття з курсу «Алгоритми і структури даних».

ЗМІСТ

ВСТУП.....	4
ВИМОГИ ДО ОФОРМЛЕННЯ ЛАБОРАТОРНИХ РОБІТ	5
ЛАБОРАТОРНА РОБОТА 1. СКЛАДНІСТЬ АЛГОРИТМІВ	7
ЛАБОРАТОРНА РОБОТА 2. ЛІНІЙНИЙ ТА БІНАРНИЙ ПОШУК.....	13
ЛАБОРАТОРНА РОБОТА 3. РОЗВ’ЯЗАННЯ РІВНЯНЬ МЕТОДОМ БІЕКЦІЇ.	17
ЛАБОРАТОРНА РОБОТА 4. БІНАРНИЙ ПОШУК ПО ВІДПОВІДІ.	19
ЛАБОРАТОРНА РОБОТА 5. ХЕШУВАННЯ ТА ХЕШ-ТАБЛИЦІ.....	22
ЛАБОРАТОРНА РОБОТА 6. ХЕШ-ТАБЛИЦІ – РОЗВ’ЯЗАННЯ КОЛІЗІЙ МЕТОДОМ ЛАНЦЮЖКІВ.	25
ЛАБОРАТОРНА РОБОТА 7. СОРТУВАННЯ – АЛГОРИТМИ ЗІ СКЛАДНІСТЮ N^2.	26
ЛАБОРАТОРНА РОБОТА 8. СОРТУВАННЯ – ШВИДКІ АЛГОРИТМИ.....	29
ЛАБОРАТОРНА РОБОТА 9. РЕКУРСІЯ, ПОВНИЙ ПЕРЕБІР, МЕТОД «РОЗДІЛЯЙ І ВОЛОДАРЮЙ».....	32
ЛАБОРАТОРНА РОБОТА 10. РЕКУРСІЯ, ПОВНИЙ ПЕРЕБІР, МЕТОД «РОЗДІЛЯЙ І ВОЛОДАРЮЙ».	33
ЛАБОРАТОРНА РОБОТА 11. СТЕК - БАЗОВІ АЛГОРИТМИ.	34
ЛАБОРАТОРНА РОБОТА 12. СТЕК ТА ЙОГО ЗАСТОСУВАННЯ.....	35
ЛАБОРАТОРНА РОБОТА 13. ЧЕРГИ ТА ЗВ’ЯЗНІ СПИСКИ. ЇХНЕ ЗАСТОСУВАННЯ	37
ЛАБОРАТОРНА РОБОТА 14. ДЕРЕВА. АЛГОРИТМИ НА ДЕРЕВАХ.	38
ЛАБОРАТОРНА РОБОТА 15. БІНАРНІ ДЕРЕВА. БІНАРНІ ДЕРЕВА ПОШУКУ.	44
ЛАБОРАТОРНА РОБОТА 16. БІНАРНА КУПА ТА ДЕРЕВО ВІДРІЗКІВ.	45
ЛАБОРАТОРНА РОБОТА 17. АЛГОРИТМИ НА НЕ ЗВАЖЕНИХ ГРАФАХ.	50
ЛАБОРАТОРНА РОБОТА 18. АЛГОРИТМИ НА ЗВАЖЕНИХ ГРАФАХ	52
ЛАБОРАТОРНА РОБОТА 19. ЛАБІРИНТИ.....	53
СПИСОК ЛІТЕРАТУРИ ТА ДОДАТКОВИХ ДЖЕРЕЛ	55

ВСТУП

ВИМОГИ ДО ОФОРМЛЕННЯ ЛАБОРАТОРНИХ РОБІТ

1. Основна ціль лабораторної роботи довести викладачу, що студент засвоїв матеріал на достатньому рівні.
2. Кожна лабораторна роботи містить
 - a. перелік контрольних запитань, що стосуються теоретичного матеріалу;
 - b. варіанти індивідуальних/групових завдань лабораторних робіт для практичного виконання;
 - c. посилання на допоміжну літературу для закріплення теоретичного матеріалу і стане в нагоді під час виконання практичних завдань.
3. На вибір студентів надано перелік практичних завдань по кожній темі, серед яких вони мають можливість самостійно вибрати бажаний варіант. Обраний номер завдання студенти записують у спеціальний хмарний документ, наданий викладачем, ціль якого рівномірно розподілити завдання між студентами групи. Кілька студентів однієї групи, що можуть обрати один варіант не може перевищувати 2, у випадку самостійної роботи, та 3-х, у випадку роботи у команді.
4. Кожна лабораторна робота оформлюється у паперовому та електронному вигляді.
5. Паперове оформлення здійснюється за допомогою текстового процесору Word (чи аналогічного) або редактора презентацій PowerPoint (чи аналогічного) та має містити такі частини:
 - a. Титульний аркуш, що містить назву навчальної дисципліни, номер лабораторної роботи, номер варіанту, ПІБ студента (студентів, якщо спільна лабораторна робота), що виконав роботу, спеціальність, курс, номер групи.
 - b. Умову задачі. Якщо задача береться з електронних джерел, то вказати посилання на задачу. Умова задачі має містити приклади вхідних та вихідних даних.
 - c. Аналіз задачі та опис алгоритму, яким пропонується розв'язувати задачу.
 - d. Приклад роботи алгоритму на модельному прикладі (взятому з умови задачі або придуманому самостійно).
 - e. Програмну реалізацію (основні моменти).
 - f. Висновки: якщо робота програми перевірялася за допомогою електронних систем, то обов'язково вказати

посилання на систему перевірки разом з задачею (наприклад, <https://www.e-olymp.com/uk/problems/3966>), логін (логіни) виконавців лабораторної роботи та результат перевірки (наприклад, 100%). Якщо результат не 100%, то обов'язково зазначити у чому проблема (не пройшов, по часу, помилки виконання, тощо).

6. Електронна частина, складається з
 - a. Файлу (файлів), з яких друкується паперовий примірник лабораторної роботи.
 - b. Файлів вихідного коду програми, що розв'язує поставлену задачу.
 - c. Файлів вхідних даних (за необхідності) – текстові файли, що містять вхідні дані задачі. Вони мають бути названі згідно з правилом – «input.txt» (якщо один файл) та «input01.txt», «input02.txt» і т.д. – якщо кілька файлів вхідних даних.
7. Електронний варіант надсилається на електронну адресу викладача, що викладає практичні заняття не пізніше ніж напередодні лабораторного заняття, на якому студент має її захищати.
8. Захист лабораторної роботи здійснюється студентом (студентами) одним з двох способів:
 - a. Індивідуальна співбесіда з викладачем, протягом якої викладач ставить запитання по теоретичному матеріалу теми лабораторної роботи (з переліку контрольних запитань), вислуховує ідею алгоритму, що розв'язує поставлену задачу, знайомиться з деталями реалізації алгоритму та перевіряє коректність розв'язку.
 - b. Презентація розв'язаної задачі з усіма деталями реалізації для всіх студентів навчальної групи. У цьому випадку роздрукований варіант не вимагається.

ЛАБОРАТОРНА РОБОТА 1. Складність алгоритмів

Контрольні запитання

- 1.1. Наведіть означення алгоритму.
- 1.2. Що таке комп'ютерна програма? Що таке реалізація алгоритму?
- 1.3. У чому полягає аналіз алгоритму?
- 1.4. Що таке елементарна операція? Наведіть кілька прикладів.
- 1.5. Що таке час виконання програми?
- 1.6. Що означає фраза: час виконання програми сталий/логарифмічний/лінійний/поліноміальний/експоненціальний?
- 1.7. Наведіть означення таких понять як найкращий/найгірший час виконання програми. Що таке час виконання програми в середньому.
- 1.8. Наведіть означення О-оцінки: що означає $f=O(g)$?
- 1.9. Наведіть означення Омега-оцінки: що означає $f=\Omega(g)$?
- 1.10. Наведіть означення Тета-оцінки: що означає $f=\Theta(g)$?
- 1.11. Наведіть основні правила визначення асимптотичної поведінки часу виконання.

Завдання для аудиторної та самостійної роботи

- 1.1. Визначте час виконання фрагментів програм заданих нижче

a)

```
1 i = 0
2 while i < n:
3     k += 1
4     i += 1
```

b)

```
1 i = 1
2 while i < n:
3     k += 1
4     i = i * 2
```

c)

```
1 i = n - 1
2 while i != 0:
3     k += 1
4     i = i // 2
```

d)

```
1 i = 0
2 while i < n:
3     if i % 2 == 0:
4         k += 1
5     i += 1
```

e)

```
1 i = 0
2 while i < n:
3     j = 0
4     while j < n:
5         k += 1
6         j += 1
7     i += 1
```

f)

```
1 i = 0
2 while i < n:
3     j = i
4     while j < n:
5         k += 1
6         j += 1
7     i += 1
```

g)

```

1  i = 0
2  while i < n:
3      j = 0
4      while j < i * i:
5          k += 1
6          j += 1
7      i += 1

```

h)

```

1  i = 0
2  while i < n:
3      j = n
4      while j != 0:
5          k += 1
6          j //= 3
7      i += 1

```

1.2. Визначте час виконання програми у явному вигляді, якщо для нього відоме рекурентне співвідношення

- a) $T(n) = \begin{cases} 1, & n = 0; \\ T(n-1) + 1, & n \geq 1. \end{cases}$
- b) $T(n) = \begin{cases} 1, & n \leq a, a > 0; \\ T(n-a) + 1, & n \geq a. \end{cases}$
- c) $T(n) = \begin{cases} 1, & n = 0; \\ 2T(n-1) + 1, & n \geq 1. \end{cases}$
- d) $T(n) = \begin{cases} 1, & n = 0; \\ 2T(n-1) + n, & n \geq 1. \end{cases}$
- e) $T(n) = \begin{cases} 1, & n = 1; \\ T(n/2) + 1, & n \geq 2. \end{cases}$
- f) $T(n) = \begin{cases} 1, & n = 1; \\ 2T(n/2) + 1, & n \geq 2. \end{cases}$
- g) $T(n) = \begin{cases} 1, & n = 1; \\ 2T(n/2) + n, & n \geq 2. \end{cases}$

Зауваження. У рекурентних співвідношеннях вище операція «/» означає цілочисельне ділення.

1.3. Нехай $f(n) = 3n^2 - n + 4$. Користуючись означенням покажіть що

- a) $f(n) = O(n^2)$, b) $f(n) = \Omega(n^2)$.

1.4. Нехай $f(n) = 3n^2 - n + 4$ та $g(n) = n \log n + 5$. Покажіть що

$$f(n) + g(n) = O(n^2).$$

1.5. Нехай $f(n) = \sqrt{n}$ та $g(n) = \log n$. Покажіть що

$$f(n) + g(n) = O(\sqrt{n}).$$

1.6. Для кожної пари функцій $f(n)$ та $g(n)$, зазначених у таблиці визначте яке із співвідношень має місце $f(n) = O(g(n))$ чи $g(n) = O(f(n))$

$f(n)$	$g(n)$
$10n$	$n^2 - 10n$
n^3	$n^2 \log n$
$n \log n$	$n + \log n$

$\log n$	$\sqrt[k]{n}$
$\ln n$	$\log n$
$\log(n + 1)$	$\log n$
$\log \log n$	$\log n$
2^n	10^n
n^m	m^n
$\cos(n\pi/2)$	$\sin(n\pi/2)$
n^2	$(n \cos n)^2$

1.7. Знайдіть асимптотичний час виконання програми у явному вигляді, якщо для нього відоме рекурентне співвідношення

- a) $T(n) = \begin{cases} O(1), & n = 0; \\ aT(n-1) + O(1), & n \geq 1, a > 1. \end{cases}$
- b) $T(n) = \begin{cases} O(1), & n = 0; \\ aT(n-1) + O(n), & n \geq 1, a > 1. \end{cases}$
- c) $T(n) = \begin{cases} O(1), & n = 0; \\ aT([n/a]) + O(1), & n \geq 1, a \geq 2. \end{cases}$
- d) $T(n) = \begin{cases} O(1), & n = 0; \\ aT([n/a]) + O(n), & n \geq 1, a \geq 2. \end{cases}$

1.8. Доведіть співвідношення

- a) $\sum_{i=0}^n i = O(n)$ b) $\sum_{i=0}^n i^2 = O(n^2)$ c) $\sum_{i=0}^n i^3 = O(n^4)$
- d) $\sum_{i=0}^n a^i = O(n)$ e) $\prod_{i=1}^n \frac{1}{1+i} = O(n)$ f) $\prod_{i=1}^n \frac{1}{1+i^2} = O(n^2)$
- g) $\prod_{i=1}^n \frac{1}{1+i!} = O(n)$ h) $\prod_{i=1}^n \frac{1}{1+i^m} = O(nm)$ i) $\prod_{i=1}^n \frac{1}{1+i^i} = O(n^2)$

1.9. Чи можна описати алгоритм для кожної з задач підрахунку суми наведених нижче, асимптотична складність яких буде $O(1)$? Якщо так, то наведіть фрагмент такої програми.

- a) $\sum_{i=0}^n i$ b) $\sum_{i=0}^n a^i$ c) $\sum_{i=0}^{\infty} a^i, |a| \leq 1$

1.10. Чи можна описати алгоритм для кожної з задач підрахунку суми наведених нижче, асимптотична складність яких буде $O(n)$? Якщо так, то наведіть фрагмент такої програми.

a) $\sqrt{2 + \sqrt{2 + \dots + \sqrt{2}}}, (n \text{ коренів});$

b) $y = x^{2^n} + x^{2^{n-1}} + \dots + x^4 + x^2 + 1;$

c) $\left(1 + \frac{1}{1^1}\right)\left(1 + \frac{1}{2^2}\right) \dots \left(1 + \frac{1}{n^n}\right);$

d) $1 + \sin x + \dots + \sin^n x.$

1.11. Припустимо, що n, m та k невід'ємні цілі числа і методи e, f, g та h мають такі характеристики:

- Час виконання у найгіршому випадку методу $e(n, m, k) \in O(1)$ і повертає значення з проміжку від 1 до $(n + m + k)$.
- Час виконання у найгіршому випадку методу $f(n, m, k) \in O(n + m)$.
- Час виконання у найгіршому випадку методу $g(n, m, k) \in O(m + k)$.
- Час виконання у найгіршому випадку методу $h(n, m, k) \in O(n + k)$.

Визначте асимптотичні оцінки виконання програм у найгіршому випадку в термінах O —«великого» для фрагментів програм зазначених нижче:

- a)
1 `f(n, 10, 0)`
2 `g(n, m, k)`
3 `h(n, m, 1000000)`
-
- b)
1 `for i in range(e(n, 10, 100)):`
2 `f(n, 10, 0);`
-
- c)
1 `for i in range(n):`
2 `f(n, m, k);`
-
- d)
1 `for i in range(n):`
2 `for j in range(i, n):`
3 `f(n, m, k);`

1.12. Визначте асимптотичну оцінку виконання функції у найгіршому випадку в термінах O —«великого» для функції:

```
def f(n):
    sum = 0
    for i in range(1, n + 1):
        sum = sum + i
    return sum
```

Що є результатом виконання наведеної функції для заданого натурального числа n ? Чи можна оптимізувати цю функцію, покращивши її асимптотичну оцінку?

1.13. Нехай $f(n)$ функція визначена у вправі 1.12. Розглянемо функцію

```
def g(n):
    sum = 0
    for i in range(1, n + 1):
        sum = sum + i + f(i)
    return sum
```

Визначте асимптотичну оцінку виконання функції $g(n)$ у найгіршому випадку в термінах O —«великого». Що є результатом виконання функції $g(n)$ для заданого натурального числа n . Чи можна оптимізувати цю функцію, покращивши її асимптотичну оцінку?

1.14. Нехай $f(n)$ функція визначена у вправі 1.12, а функція $g(n)$ – у вправі 1.13. Розглянемо функцію

```
def h(n):
    return f(n) + g(n)
```

Визначте асимптотичну оцінку виконання функції $h(n)$ у найгіршому випадку в термінах O —«великого». Що є результатом її виконання для заданого натурального числа n . Чи можна оптимізувати цю функцію, покращивши її асимптотичну оцінку?

1.15. Визначте асимптотичну оцінку виконання функції у найгіршому випадку в термінах O —«великого» для функції:

```
def f(n):
    k = 0
    i = n - 1
    while i != 0:
        k += 1.0 / i
        i = i / 2
    return k
```

Що є результатом виконання наведеної функції для заданого натурального числа n ?

1.16. Опишіть функції натурального аргументу n , час виконання яких у найгіршому випадку має асимптотику:

- a) $O(n)$ b) $O(n^2)$ c) $O(n^3)$
d) $O(\log n)$ e) $O(n \log n)$ f) $O(2^n)$

1.17. Опишіть функції натуральних аргументів n та m , час виконання яких у найгіршому випадку має асимптотику:

- a) $O(n + m)$ b) $O(m \log n)$ c) $O(n^m)$

1.18. Натуральне число називається паліндромом, якщо його запис читається однаково зліва направо і справа наліво (наприклад, 1, 393, 4884). Скласти програму, що визначає, чи є задане натуральне число n паліндромом асимптотична складність якої $O(n)$, де n – кількість цифр у числі.

1.19. Числами трибоначчі називається числова послідовність $\{T_k: k \geq 0\}$, задана рекурентним співвідношенням третього порядку:

$$T_0 = 0, T_1 = T_2 = 1, T_k = T_{k-1} + T_{k-2} + T_{k-3}, \quad k \geq 3.$$

Опишіть функції для обчислення T_n за допомогою рекурентного співвідношення та використовуючи рекурсію. Обчисліть асимптотичну складність кожного з варіантів. Порівняйте абсолютний час виконання (у секундах) обох варіантів для знаходження T_{10}, T_{20}, T_{50} .

1.20. Послідовністю Падована називається числова послідовність $\{P_k: k \geq 0\}$, задана рекурентним співвідношенням третього порядку:

$$P_0 = P_1 = P_2 = 1, P_k = P_{k-1} + P_{k-3}, \quad k \geq 3.$$

Опишіть функції для обчислення P_n за допомогою рекурентного співвідношення та використовуючи рекурсію. Обчисліть асимптотичну складність кожного з варіантів. Порівняйте абсолютний час виконання (у секундах) обох варіантів для знаходження P_{10}, P_{20}, P_{50} .

ЛАБОРАТОРНА РОБОТА 2.

Лінійний та бінарний пошук.

Контрольні запитання

- 2.1. Які задачі допомагає розв'язати лінійний пошук? Наведіть алгоритм лінійного пошуку та його складність (у найгіршому випадку).
- 2.2. У чому полягає цілочисельний бінарний пошук? Для яких структур даних він може бути реалізований? Яка складність бінарного пошуку? Яка перевага бінарного пошуку у порівнянні з лінійним пошуком? Наведіть алгоритм бінарного пошуку.

Завдання для аудиторної роботи

- 2.1. Одиниці

<https://www.e-olymp.com/uk/problems/622>

На уроках інформатики вас, напевно, вчили переводити числа з одних систем числення у інші і виконувати інші подібні операції. Прийшов час продемонструвати ці знання. Знайдіть кількість одиниць у двійковому запису заданого числа.

Вхідні дані

У вхідному файлі міститься єдине ціле число n ($0 \leq n \leq 2000000000$).

Вихідні дані

Вихідний файл повинен містити одне число — кількість двійкових одиниць у запису числа n .

Вхідні дані #1

5

Вихідні дані #1

2

Вхідні дані #2

7

Вихідні дані #2

3

Зауваження. Скористайтесь побітовими операціями.

- 2.2. Реалізуйте інтерфейс для роботи з англійсько-українським словником та швидким пошуком перекладу. Реалізацію здійсніть у вигляді сукупності двох функцій описаних нижче. Функція

```
def addTranslation(eng, translation)
```

додає до словника англійське слово `eng` та його переклад `translation`. Пари `(eng, translation)` приходяться у порядку, що

відповідає лексикографічному порядку. Функція

```
def find(eng)
```

повертає переклад слова `eng` зі словника, якщо воно міститься у словнику, або порожній рядок у іншому разі.

Тестова програма `main.py`, файл `user.py`, що містить вищезгаданий інтерфейс та вхідні дані (текстові файли) розташовані за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Search/task1>

Завантажте всі файли, що містяться за цим посиланням, у одну папку та реалізуйте згаданий вище інтерфейс (файл `user.py`). Для перевірки правильності алгоритму запустіть файл `main.py`.

- 2.3. Реалізуйте алгоритм пошуку номеру найпершого входження до заданого масиву, заданого числа `x`. Якщо заданий елемент відсутній у списку - поверніть номер першого елементу, що більший за число `x`:

```
array[i] >= x
```

Алгоритм оформіть у вигляді функції

```
def bsearch_leftmost(array, x)
```

Гарантується, що вхідний масив `array` впорядкований за неспаданням.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Search/task2>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію, інтерфейс якої міститься у файлі `user.py`. Для перевірки правильності алгоритму запустіть файл `main.py`.

Завдання для самостійної роботи

- 2.4. Циклічні зсуви
<https://www.e-olymp.com/uk/problems/27>
- 2.5. Сортуння за зростом
<https://www.e-olymp.com/uk/problems/3607>
Вказівка: Використайте лінійний пошук для підрахунку відповіді.
- 2.6. Затятий колекціонер метеликів
<https://www.e-olymp.com/uk/problems/3966>
- 2.7. Мутанти
<https://www.e-olymp.com/uk/problems/3970>

Вказівка: Використайте бінарний пошук для знаходження відповіді для кожного вказаного інтервалу.

Додаткові завдання для самостійної роботи

- 2.8. Реалізуйте алгоритм пошуку номеру останнього входження до заданого масиву заданого числа x . Якщо такий елемент відсутній у списку - поверніть номер останнього елементу, що менший за число x . Зауважимо, що якщо всі елементи масиву менші за шукане число, програма має повертати значення -1 (мінус один).

`array[i] >= x`

Алгоритм оформіть у вигляді функції

```
def bsearch_rightmost(array, x)
```

Гарантується, що вхідний масив `array` впорядкований за неспаданням.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Search/task3>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію, інтерфейс якої міститься у файлі `user.py`. Для перевірки правильності алгоритму запустіть файл `main.py`.

- 2.9. Знайдіть кількість входжень заданого числа x до масиву цілих чисел `array`. Алгоритм оформіть у вигляді функції

```
def counter(array, x)
```

Гарантується, що вхідний масив `array` впорядкований за неспаданням.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Search/task4>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію, інтерфейс якої міститься у файлі `user.py`. Для перевірки правильності алгоритму запустіть файл `main.py`.

- 2.10. Реалізуйте алгоритм лінійного пошуку всіх елементів списку, що є степенями двійки та визначте асимптотичну складність отриманого алгоритму.
- 2.11. Припустимо, що список містить елементи, для яких визначена операція \leq (менше або рівно) і елементи у цьому списку розташовані

у порядку зростання. Скориставшись цією властивістю списку, запропонуйте алгоритм лінійного пошуку та оцініть його час виконання. Чи буде такий алгоритм оптимальнішим ніж класичний лінійний пошук?

2.12. Реалізуйте алгоритм бінарного пошуку з використанням рекурсії.

ЛАБОРАТОРНА РОБОТА 3.

Розв'язання рівнянь методом бісекції.

Контрольні запитання

- 3.1. Для чого використовується дійсний бінарний пошук? Наведіть приклад задачі. Чи є відмінність у реалізації дійсного бінарного пошуку від цілочисельного? Які основні підходи застосовуються для моменту завершення пошуку? У чому їхні переваги та недоліки у порівнянні один з одним? Наведіть алгоритм дійсного бінарного пошуку.

Завдання для аудиторної роботи

- 3.1. Для монотонної на відрізку $[a, b]$ функції f розв'яжіть рівняння

$$f(x) = c. \quad (3.1)$$

Реалізацію здійсніть у вигляді сукупності двох функцій описаних нижче. Функція

```
def solve(f, c, a, b):
```

Для неспадної на відрізку $[a, b]$ функції f розв'язує рівняння (3.1), а функція

```
def solve_decreasing(f, c, a, b)
```

відповідно для незростаючої на відрізку $[a, b]$ функції f розв'язує рівняння (3.1). Для визначення моменту завершення пошуку використайте всі три підходи (точність по аргументу, точність по значенню та безпосереднє сусідство двох чисел з плаваючою крапкою). Порівняйте отримані результати.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Search/task5>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадані вище функції, інтерфейси яких містяться у файлі `user.py`. Для перевірки правильності алгоритму запустіть файл `main.py`.

Завдання для самостійної роботи

- 3.2. Квадратний корінь

<https://www.e-olymp.com/uk/problems/3968>

- 3.3. Знайдіть найменше $x \in [0,10]$, що
$$f(x) = x^3 + x + 1 > 5$$
- 3.4. На відрізку $[1.6, 3]$ знайдіть корінь рівняння
$$\sin x = \frac{x}{3}.$$
- 3.5. На відрізку $[0,2]$ знайдіть корінь рівняння
$$x^3 + 4x^2 + x - 6.$$

ЛАБОРАТОРНА РОБОТА 4.

Бінарний пошук по відповіді.

Контрольні запитання

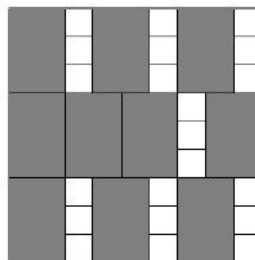
4.1. У чому полягає концепція бінарного пошуку по відповіді?

Завдання для аудиторної роботи

4.1. Дипломи

<https://www.e-olymp.com/uk/problems/3969>

Коли Петя вчився в школі, він часто брав участь в олімпіадах з інформатики, математики та фізики. Так як він був досить здібним хлопчиком і старанно вчився, то на багатьох із цих олімпіад він отримував дипломи. До закінчення школи у нього накопичилося n дипломів, причому, як виявилось, всі вони мали однакові розміри: w - в ширину і h - у висоту.



Зараз Петя навчається в одному з кращих університетів і живе в гуртожитку зі своїми одногрупниками. Він вирішив прикрасити свою кімнату, повісивши на одну зі стін свої дипломи за шкільні олімпіади. Так як до бетонної стіни прикріпити дипломи досить важко, він вирішив купити спеціальну дошку з коркового дерева, щоб прикріпити її до стіни, а до неї - дипломи. Для того, щоб ця конструкція виглядала більш красиво, Петя хоче, щоб була квадратною і займала якомога менше місця на стіні. Кожен диплом повинен бути розміщений строго в прямокутнику w на h . дипломи забороняється повертати на 90 градусів. Прямокутники, які відповідають різним дипломам, не повинні мати спільних внутрішніх точок.

Потрібно написати програму, яка обчислить мінімальний розмір сторони дошки, яка буде потрібно Пете для розміщення всіх своїх дипломів.

Вхідні дані

Вхідний файл містить три цілих числа w , h , n ($1 \leq w, h, n \leq 109$).

Вихідні дані

У вихідний файл вивести відповідь до сформульованої задачі.

Вхідні дані #1

2 3 10

Вихідні дані #1

9

4.2. Дуже Легка Задача

<https://www.e-olymp.com/uk/problems/5102>

Сьогодні вранці журі вирішило додати у варіант олімпіади ще одну, Дуже Легку Задачу. Відповідальний секретар оргкомітету надрукував її умову в одному екземплярі, і тепер йому потрібно до початку олімпіади встигнути зробити ще n копій. У його розпорядженні є два ксерокси, один з яких копіює аркуш за x секунд, а другий за y . (Дозволяється використовувати як один ксерокс, так і обидва одночасно. Можна копіювати не лише з оригінала, але і з копії.) Допоможіть йому вияснити, який мінімальний час для цього потрібно.

Вхідні дані

Три натуральних числа n , x та y ($1 \leq n \leq 2 \cdot 10^8$, $1 \leq x, y \leq 10$).

Вихідні дані

Виведіть одне число – мінімальний час в секундах, необхідний для отримання n копій.

Вхідні дані #1

4 1 1

Вихідні дані #1

3

Вхідні дані #2

5 1 2

Вихідні дані #2

4

Завдання для самостійної роботи

4.3. Корови – в стійла

<https://www.e-olymp.com/uk/problems/5149>

4.4. Черепаха

<https://www.e-olymp.com/uk/problems/669>

4.5. Рисосховище

<https://www.e-olymp.com/uk/problems/2254>

4.6. Мотузочки

<https://www.e-olymp.com/uk/problems/3967>

4.7. Роботи

<https://www.e-olymp.com/uk/problems/161>

- 4.8. Автобус
<https://www.e-olymp.com/uk/problems/6132>

ЛАБОРАТОРНА РОБОТА 5. Хешування та хеш-таблиці.

Контрольні запитання

- 5.1. Що таке хешування?
- 5.2. Наведіть приклад хеш-функції.
- 5.3. Наведіть основні характеристики, яким має задовольняти хеш-функція
- 5.4. Що таке колізія хеш-функції?
- 5.5. Яка хеш-функція називається ідеальною?
- 5.6. Що таке асоціативний масив?
- 5.7. Яка структура даних називається хеш-таблицею? Які основні операції вона має підтримувати?
- 5.8. У чому полягає процес розв'язання колізій?
- 5.9. Які види хеш-таблиць бувають? Які відмінності між ними?
- 5.10. Яким чином реалізується процес видалення ключів у хеш-таблиці з відкритою адресацією?
- 5.11. Запропонуйте алгоритм збільшення слотів хеш-таблиці, що використовує лінійне зондування для розв'язання колізій, для випадку, якщо фактор завантаження таблиці перевищує деякий поріг.

Завдання для аудиторної роботи

- 5.1. Реалізуйте інтерфейс асоційованого масиву, ключами якого є цілі числа, а значеннями – рядки. Реалізацію здійсніть як хеш-таблицю з відкритою адресацією у вигляді сукупності функцій описаних нижче функцій. Функція

```
def init()
```

Викликається 1 раз на початку виконання програми. Функція

```
def set(key: int, value: str) -> None
```

встановлює значення value для ключа key. Якщо такий ключ відсутній у структурі – додає пару, інакше змінює значення для цього ключа. Функція

```
def get(key: int) -> str
```

за ключем key повертає значення зі структури (або None, якщо ключ відсутній у структурі). Нарешті функція

```
def delete(key: int) -> None
```

видаляє пару ключ-значення за заданим ключем, якщо такий ключ

міститься у таблиці.

Тестова програма розташована за посиланням:

github.com/krenevych/algo/tree/master/labs/T3/Hashtables/task1

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадані вище функції, інтерфейси яких містяться у файлі `user.py`. Для перевірки правильності алгоритму запустіть файл `main.py`.

- 5.2. Розглядається бібліотека художніх книг. У бібліотеці може міститися кілька книг одного автора. Реалізуйте каталог для цієї бібліотеки. Інтерфейс, що необхідно реалізувати наведено нижче.

```
def init():
    """ Викликається 1 раз на початку виконання програми. """
    pass

def addBook(author, title):
    """ Додає книгу до бібліотеки.
    :param author: Автор книги
    :param title: Назва книги
    """
    pass

def find(author, title):
    """ Перевірає чи міститься задана книга у бібліотеці.
    :param author: Автор
    :param title: Назва книги
    :return: True, якщо книга міститься у бібліотеці та
            False у іншому разі. """
    return False

def delete(author, title):
    """ Видаляє книгу з бібліотеки.
    :param author: Автор
    :param title: Назва книги
    """
    pass

def findByAuthor(author):
    """ Повертає список книг заданого автора.
    Якщо бібліотека не містить книг заданого автора,
    то підпрограма повертає порожній список.
    :param author: Автор
    :return: Список книг заданого автора у алфавітному порядку. """
    return []
```

Тестова програма розташована за посиланням:

github.com/krenevych/algo/tree/master/labs/T3/Hashtables/task2

Завантажте всі файли, що містяться за цим посиланням (у тому

ж числі папку data разом з усім її вмістом), у одну папку. Реалізуйте згадані вище функції, інтерфейси яких містяться у файлі user.py. Для перевірки правильності алгоритму запустіть файл main.py.

Завдання для самостійної роботи

- 5.3. Англо-латинський словник
<https://www.e-olymp.com/uk/problems/4842>
- 5.4. Перший словник Енді
<https://www.e-olymp.com/uk/problems/1227>

Завдання для самостійної роботи підвищеної складності

- 5.5. Слова
<https://www.e-olymp.com/uk/problems/131>
- 5.6. Дешифратор
<https://www.e-olymp.com/uk/problems/2035>
- 5.7. Чорна скринька
<https://www.e-olymp.com/uk/problems/1225>

ЛАБОРАТОРНА РОБОТА 6. Хеш-таблиці – розв’язання колізій методом ланцюжків.

Контрольні запитання

- 6.1. Поясніть концепцію методу ланцюжків для розв’язання колізій при реалізації хеш-таблиць.

Завдання для аудиторної роботи

- 6.1. Розв’яжіть задачу 5.1 реалізуючи хеш-таблицю з розв’язанням колізій методом ланцюжків.
- 6.2. Розв’яжіть задачу 5.2 реалізуючи хеш-таблицю з розв’язанням колізій методом ланцюжків.

Завдання для самостійної роботи

- 6.3. Англо-латинський словник
<https://www.e-olymp.com/uk/problems/4842>
- 6.4. Перший словник Енді
<https://www.e-olymp.com/uk/problems/1227>
- 6.5. Слова
<https://www.e-olymp.com/uk/problems/131>

Завдання для самостійної роботи підвищеної складності

- 6.6. Словник нецензурних слів
<https://www.e-olymp.com/uk/problems/1236>
- 6.7. Дешифратор
<https://www.e-olymp.com/uk/problems/2035>
- 6.8. Хеш-таблиця
<https://www.e-olymp.com/uk/problems/7273>

ЛАБОРАТОРНА РОБОТА 7.

Сортування – алгоритми зі складністю n^2 .

Контрольні запитання

- 7.1. Наведіть ідею алгоритму сортування обміном (бульбашкового сортування).
- 7.2. Наведіть ідею алгоритму сортування вибором. У чому його перевага перед алгоритмом сортування обміном?
- 7.3. Наведіть ідею алгоритму сортування вставкою.

Завдання для аудиторної роботи

- 7.1. Реалізуйте алгоритм сортування обміном (бульбашкове сортування). Реалізацію здійсніть у вигляді функції

```
def sort(array):  
    """ Сортування масиву  
    :param array: Вхідний масив даних, що треба відсортувати. """  
    pass # TODO: реалізуйте відповідний алгоритм тут
```

що отримує на вхід масив array та сортує його.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Sort/task1>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію у файлі user.py. Для перевірки правильності алгоритму запустіть файл main.py.

Зауваження: Для визначення розміру масиву вхідних даних, встановіть відповідне значення змінній N., що описана у початку модуля user.py.

```
N = 10000 # Кількість елементів масиву.  
          # Використовується у головній програмі для генерування  
          # масиву з випадкових чисел  
          # Для повільних алгоритмів сортування з асимптотикою  
          #  $n^2$  рекомендується використовувати  
          # значення не більше 10к
```

- 7.2. Модифікуйте алгоритм бульбашкового сортування, реалізований під час виконання задачі 7.1, перевіркою на кожному проході чи відсортований вже масив. Порівняйте час виконання обох програм.
- 7.3. Реалізуйте алгоритм сортування вибором. Реалізацію здійсніть у вигляді функції

```
def sort(array):
    """ Сортвання масиву
    :param array: Вхідний масив даних, що треба відсортувати. """
    pass # TODO: реалізуйте відповідний алгоритм тут
```

що отримує на вхід масив array та сортує його. Порівняйте час виконання цього алгоритму сортування з алгоритмами реалізованими у попередніх задачах.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Sort/task1>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію у файлі user.py. Для перевірки правильності алгоритму запустіть файл main.py.

- 7.4. Реалізуйте алгоритм сортування вставкою. Реалізацію здійсніть у вигляді функції

```
def sort(array):
    """ Сортвання масиву
    :param array: Вхідний масив даних, що треба відсортувати. """
    pass # TODO: реалізуйте відповідний алгоритм тут
```

що отримує на вхід масив array та сортує його. Порівняйте час виконання цього алгоритму сортування з алгоритмами реалізованими у попередніх задачах.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Sort/task1>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію у файлі user.py. Для перевірки правильності алгоритму запустіть файл main.py.

Завдання для самостійної роботи

- 7.5. Сортвання бульбашкою
<https://www.e-olymp.com/uk/problems/2663>
- 7.6. Словник
<https://www.e-olymp.com/uk/problems/5089>

Завдання для самостійної роботи підвищеної складності

- 7.7. Бібліотечний метод
<https://www.e-olymp.com/uk/problems/2664>
- 7.8. Сортвання часу
<https://www.e-olymp.com/uk/problems/972>

- 7.9. Градуїований лексикографічний порядок
<https://www.e-olymp.com/uk/problems/1130>
- 7.10. Особові справи
<https://www.e-olymp.com/uk/problems/1344>
- 7.11. Хитре сортування
<https://www.e-olymp.com/uk/problems/1462>
- 7.12. Метод мінімуму
<https://www.e-olymp.com/uk/problems/2662>
- 7.13. Библиотека
<https://www.e-olymp.com/ru/problems/4230>

ЛАБОРАТОРНА РОБОТА 8.

Сортування – швидкі алгоритми.

Контрольні запитання

- 8.1. Наведіть ідею алгоритму сортування злиттям. Яка його асимптотична складність? Які його особливості?
- 8.2. Наведіть ідею алгоритму швидкого сортування. Яка його асимптотична складність? Які його особливості у порівнянні з іншими алгоритмами сортуванням?

Завдання для аудиторної роботи

- 8.1. Реалізуйте алгоритм сортування злиттям. Реалізацію здійсніть у вигляді функції

```
def sort(array):  
    """ Сортування масиву  
    :param array: Вхідний масив даних, що треба відсортувати. """  
    pass # TODO: реалізуйте відповідний алгоритм тут
```

що отримує на вхід масив array та сортує його. Порівняйте час виконання цього алгоритму сортування з алгоритмами реалізованими у попередній лабораторній роботі.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Sort/task1>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію у файлі user.py. Для перевірки правильності алгоритму запустіть файл main.py.

Зауваження: Для визначення розміру масиву вхідних даних, встановіть відповідне значення змінній N., що описана у початку модуля user.py.

```
N = 1000000 # Кількість елементів масиву.  
            # Використовується у головній програмі для генерування  
            # масиву з випадкових чисел  
            # Для швидких алгоритмів сортування з асимптотикою  
            #  $n \log(n)$  встановіть значення 1 000 000
```

- 8.2. Реалізуйте швидкий алгоритм сортування QuickSort. Реалізацію здійсніть у вигляді функції

```
def sort(array):  
    """ Сортування масиву
```

```
:param array: Вхідний масив даних, що треба відсортувати. ""  
pass # TODO: реалізуйте відповідний алгоритм тут
```

що отримує на вхід масив `array` та сортує його. Порівняйте час виконання цього алгоритму сортування з алгоритмом сортування вставкою.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Sort/task1>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадану вище функцію у файлі `user.py`. Для перевірки правильності алгоритму запустіть файл `main.py`.

- 8.3. Проведіть аналіз швидкодії реалізованих алгоритмів сортування для різних типів та розмірів масивів (не відсортований масив згенерований випадковим чином, масив відсортований за зростанням, масив відсортований за спаданням елементів). Для цього опишіть функції

```
def bubble_sort(array):  
    """ Сортування "Бульбашкою"  
    :param array: Масив (список однотипових елементів)  
    """  
    pass # TODO: реалізуйте відповідний алгоритм тут  
  
def bubble_sort_optimized(array):  
    """ Модифікований алгоритм сортування "Бульбашкою"  
    :param array: Масив (список однотипових елементів).  
    """  
    pass # TODO: реалізуйте відповідний алгоритм тут  
  
def selection_sort(array):  
    """ Сортування вибором  
    :param array: Масив (список однотипових елементів)  
    :return: None  
    """  
    pass # TODO: реалізуйте відповідний алгоритм тут  
  
def insertion_sort(array):  
    """ Сортування вставкою  
    :param array: Масив (список однотипових елементів)  
    :return: None  
    """  
    pass # TODO: реалізуйте відповідний алгоритм тут  
  
def merge_sort(array):  
    """ Сортування злиттям  
    :param array: Масив (список однотипових елементів)  
    :return: None
```

```

"""
pass # TODO: реалізуйте відповідний алгоритм тут

def quick_sort(array):
    """ Швидке сортування
        :param array: Масив (список однотипових елементів)
        :return: None
    """
    pass # TODO: реалізуйте відповідний алгоритм тут

```

кожна з яких отримує на вхід масив array та сортує його відповідним методом.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T3/Sort/task2>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте задані вище функції у файлі user.py. Для перевірки запустіть файл main.py.

Для визначення розміру масиву вхідних даних, встановіть відповідне значення змінній N., що описана у початку модуля user.py.

```

N = 10000 # Кількість елементів масиву.
          # Використовується у головній програмі для генерування
          # масиву з випадкових чисел

```

Завдання для самостійної роботи

- 8.4. Сортування за зростом (перемістити у лінійний пошук)
<https://www.e-olymp.com/uk/problems/3607>
- 8.5. Сортування
<https://www.e-olymp.com/uk/problems/2321>
- 8.6. Сортування злиттям
<https://www.e-olymp.com/uk/problems/4037>

Завдання для самостійної роботи підвищеної складності

- 8.7. Ультра швидке сортування
<https://www.e-olymp.com/uk/problems/1303>
- 8.8. Станція "Сортувальна"
<https://www.e-olymp.com/uk/problems/1457>

ЛАБОРАТОРНА РОБОТА 9. Рекурсія, повний перебір, Метод «розділяй і володарюй».

Контрольні запитання

- 9.1. Які алгоритми називаються алгоритмами повного перебору?
- 9.2. Чи існують загальні методи побудови алгоритмів повного перебору?
- 9.3. Що можна сказати про асимптотичну складність алгоритмів повного перебору?
- 9.4. У чому полягає метод гілок та меж?
- 9.5. У чому полягає метод декомпозиції («розділяй і володарюй»)?

Завдання для аудиторної роботи

- 9.1. 23 з 5
<https://www.e-olymp.com/uk/problems/1540>
- 9.2. Машина арифметика
<https://www.e-olymp.com/uk/problems/1296>
- 9.3. Задача про призначення
<https://www.e-olymp.com/uk/problems/1781>

Завдання для самостійної роботи

- 9.4. Печеньки
<https://www.e-olymp.com/uk/problems/2633>
- 9.5. Банк
<https://www.e-olymp.com/uk/problems/7470>

Завдання для самостійної роботи підвищеної складності

- 9.6. Відпали мирних ферзів!
<https://www.e-olymp.com/uk/problems/2764>
- 9.7. Торговець
<https://www.e-olymp.com/uk/problems/224>
- 9.8. CD
<https://www.e-olymp.com/uk/problems/1266>
- 9.9. Розподіл оцінок
<https://www.e-olymp.com/uk/problems/1524>

ЛАБОРАТОРНА РОБОТА 10. Рекурсія, повний перебір, Метод «розділяй і володарюй».

Контрольні запитання

- 10.1. У чому полягає метод гілок та меж?
- 10.2. У чому полягає метод декомпозиції («розділяй і володарюй»)?

Завдання для аудиторної роботи

- 10.1. Словник
<https://www.e-olymp.com/uk/problems/364>
- 10.2. Путівки
<https://www.e-olymp.com/uk/problems/6>
- 10.3. Міст
<https://www.e-olymp.com/uk/problems/1592>

Завдання для самостійної роботи

- 10.4. Дуже швидке множення
<https://www.e-olymp.com/uk/problems/317>
Вказівка: Використайте алгоритм [Множення Карацуби](#).
- 10.5. Кафе "Хоботанія"
<https://www.e-olymp.com/uk/problems/4746>

Завдання для самостійної роботи підвищеної складності

- 10.6. Мінне поле
<https://www.e-olymp.com/uk/problems/2000>
- 10.7. А побалакати?
<https://www.e-olymp.com/uk/problems/3606>
- 10.8. Істотні інверсії
<https://www.e-olymp.com/uk/problems/7031>
- 10.9. Шкільний бал
<https://www.e-olymp.com/uk/problems/7227>

ЛАБОРАТОРНА РОБОТА 11. Стек – базові алгоритми.

Контрольні запитання

- 11.1. Яка структура даних називається стеком?
- 11.2. Наведіть основні операції для роботи зі стеком.
- 11.3. Які основні підходи до моделювання структури даних Стек використовуються у програмуванні на практиці?
- 11.4. Які структури даних називаються рекурсивними?

Завдання для аудиторної роботи

- 11.1. Простий стек
<https://www.e-olymp.com/uk/problems/6122>
Вказівка: реалізацію стеку здійсніть на базі вбудованого списку або масиву
- 11.2. Стек із захисом від помилок
<https://www.e-olymp.com/uk/problems/6123>
- 11.3. Стек необмеженого розміру
<https://www.e-olymp.com/uk/problems/6124>
Вказівка: реалізацію стеку здійсніть як рекурсивну структуру.

Завдання для самостійної роботи

- 11.4. Мінімум на стеці
<https://www.e-olymp.com/uk/problems/4259>

Завдання для самостійної роботи підвищеної складності

- 11.5. Рейки
<https://www.e-olymp.com/uk/problems/1776>

ЛАБОРАТОРНА РОБОТА 12. Стек та його застосування.

Контрольні запитання

- 12.1. Наведіть приклади застосування стеку.
- 12.2. Наведіть алгоритм перевірки правильності дужкової послідовності.
- 12.3. Наведіть алгоритм конвертування арифметичного виразу з інфіксованої у постфіксну форму зображення.
- 12.4. Наведіть алгоритм обчислення арифметичного виразу зображеного у інфіксній та постфіксній формах.

Завдання для аудиторної роботи

- 12.1. Системи числення - 1
<https://www.e-olymp.com/uk/problems/5322>
- 12.2. Баланс дужок
<https://www.e-olymp.com/uk/problems/2479>

Завдання для самостійної роботи

- 12.3. Система числення
<https://www.e-olymp.com/uk/problems/7433>
- 12.4. Дужки
<https://www.e-olymp.com/uk/problems/1994>
- 12.5. 3 префіксного у інфіксне
<https://www.e-olymp.com/uk/problems/731>
- 12.6. Дужки (2)
<https://www.e-olymp.com/uk/problems/855>

Додаткові завдання для самостійної роботи

- 12.7. Системи числення
<https://www.e-olymp.com/uk/problems/1008>
- 12.8. Вирази
<https://www.e-olymp.com/uk/problems/3837>

Завдання для самостійної роботи підвищеної складності

- 12.9. Система числення - перезавантаження
<https://www.e-olymp.com/uk/problems/1376>
- 12.10. Дужки
<https://www.e-olymp.com/uk/problems/5205>

- 12.11. Перетворення
<https://www.e-olymp.com/uk/problems/5539>
- 12.12. Магічні дужки
<https://www.e-olymp.com/uk/problems/3013>
- 12.13. Математики та дужки
<https://www.e-olymp.com/uk/problems/4438>
- 12.14. Дужки
<https://www.e-olymp.com/uk/problems/1801>
- 12.15. Дріб у LATEX-i
<https://www.e-olymp.com/uk/problems/2039>
- 12.16. Інопланетяни
<https://www.e-olymp.com/uk/problems/2702>
- 12.17. Яка система числення?
<https://www.e-olymp.com/uk/problems/1377>
- 12.18. Калькулятор
<https://www.e-olymp.com/uk/problems/1394>
- 12.19. Статична складність
<https://www.e-olymp.com/uk/problems/1073>

ЛАБОРАТОРНА РОБОТА 13. Черги та дек. Їхнє застосування.

Контрольні запитання

- 13.1. Яка структура даних називається чергою?
- 13.2. Наведіть основні операції для роботи з чергою.
- 13.3. Наведіть приклади реалізації черги. Яка додаткова структура даних використовується для реалізації черги як рекурсивної структури даних?
- 13.4. Яка структура даних називається деком (чергою з двома кінцями або двосторонньою чергою)? У чому її відмінність від черги?
- 13.5. Наведіть основні операції для роботи з двосторонньою чергою.
- 13.6. У чому полягає ідея алгоритму реалізації двосторонньої черги як рекурсивної структури? Яка додаткова структура даних при цьому використовується?
- 13.7. Яка структура даних називається пріоритетною чергою?

Завдання для аудиторної роботи

- 13.1. Проста черга
<https://www.e-olymp.com/uk/problems/6125>
Вказівка: реалізацію здійсніть на базі вбудованого списку або масиву
- 13.2. Черга необмеженого розміру
<https://www.e-olymp.com/uk/problems/6127>
Вказівка: реалізацію черги здійсніть як рекурсивну структуру.
- 13.3. Простий дек
<https://www.e-olymp.com/uk/problems/6128>
Вказівка: реалізацію здійсніть на базі вбудованого списку або масиву
- 13.4. Дек необмеженого розміру
<https://www.e-olymp.com/uk/problems/6130>
Вказівка: реалізацію здійсніть як рекурсивну структуру.
- 13.5. Задача Іосифа Флавія
<https://www.e-olymp.com/uk/problems/971>
Вказівка: для розв'язання задачі використовуйте структуру даних черга або двостороння черга реалізовану у попередніх задачах.

Завдання для самостійної роботи

- 13.6. Черга з захистом від помилок
<https://www.e-olymp.com/uk/problems/6126>
- 13.7. Дек з захистом від помилок
<https://www.e-olymp.com/uk/problems/6129>
- 13.8. Гра у п'яницю
<https://www.e-olymp.com/uk/problems/4005>
- 13.9. Варварські племена
<https://www.e-olymp.com/uk/problems/1549>
- 13.10. Деки на 6 мегабайтах
<https://www.e-olymp.com/uk/problems/3161>

Додаткові завдання для самостійної роботи

- 13.11. Іосиф повторюється
<https://www.e-olymp.com/uk/problems/1515>

Завдання для самостійної роботи підвищеної складності

- 13.12. Мінімум в черзі
<https://www.e-olymp.com/uk/problems/694>
- 13.13. Сортуння чергами
<https://www.e-olymp.com/uk/problems/2510>

ЛАБОРАТОРНА РОБОТА 14. Зв'язні списки. Їхнє застосування

Контрольні запитання

- 14.1. Яка структура даних називається однозв'язним списком? Наведіть основні операції з цією структурою даних та ідею алгоритму її реалізації.
- 14.2. Яка структура даних називається двозв'язним списком? Наведіть основні операції з цією структурою даних та ідею алгоритму її реалізації.
- 14.3. Яка структура даних називається кільцевим списком? Наведіть основні операції з цією структурою даних та ідею алгоритму її реалізації.
- 14.4. Яка структура даних називається списком з поточним елементом? Наведіть основні операції з цією структурою даних та ідею алгоритму її реалізації.

Завдання для аудиторної роботи

- 14.1. Реалізуйте структуру даних зв'язний список з поточним елементом. Реалізацію здійсніть у вигляді сукупності функцій описаних нижче.

```
def init()
```

Викликається один раз на початку виконання програми.

```
def empty()
```

Перевіряє чи список порожній.

```
def reset()
```

Переставляє поточний елемент на перший елемент списку.

```
def next()
```

Перейти до наступного елемента – робить поточним елементом списку, елемент, що йде за поточним. Породжує виключення `StopIteration`, якщо поточний елемент є останнім у списку.

```
def current()
```

Повертає навантаження поточного елемента.

```
def insert_after(item):
```

Вставляє новий елемент у список після поточного.

```
def insert_before(item)
```

Вставляє новий елемент у список перед поточним.

```
def delete()
```

Видаляє поточний елемент. Поточним при цьому стає наступний елемент, що йшов у списку після поточного. Якщо елемент, що видаляється був у списку останнім, то поточним стає передостанній елемент цього списку.

```
def damp():
```

Повертає масив у якому записані всі елементи поточного списку.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T5/List/task1>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте задані вище функції, інтерфейси яких містяться у файлі user1.py. Для перевірки правильності алгоритму запустіть файл main.py.

- 14.2. Реалізуйте структуру даних двобічно зв'язний список з поточним елементом. Реалізацію здійсніть у вигляді сукупності функцій описаних нижче.

```
def init()
```

Викликається один раз на початку виконання програми.

```
def empty()
```

Перевіряє чи список порожній.

```
def set_first()
```

Переставляє поточний елемент на перший елемент списку.

```
def set_last()
```

Переставляє поточний елемент на останній елемент списку.

```
def next()
```

Перейти до наступного елемента – робить поточним елементом списку, елемент, що йде за поточним. Породжує виключення `StopIteration`, якщо поточний елемент є останнім у списку.

```
def prev()
```

Перейти до попереднього елемента – робить поточним елементом елемент списку, що йде перед поточним. Породжує виключення

StopIteration, якщо поточний елемент є першим у списку..

```
def current()
```

Повертає навантаження поточного елементу.

```
def insert_after(item):
```

Вставляє новий елемент у список після поточного.

```
def insert_before(item)
```

Вставляє новий елемент у список перед поточним.

```
def delete()
```

Видаляє поточний елемент. Поточним при цьому стає наступний елемент, що йшов у списку після поточного. Якщо елемент, що видаляється був у списку останнім, то поточним стає передостанній елемент цього списку.

```
def damp()
```

Повертає масив у якому записані всі елементи поточного списку.

```
def len()
```

Повертає кількість елементів у списку

```
def swap_prev()
```

Міняє місцями поточний елемент списку з попереднім. Гарантується, що виклик функції здійснюється лише, якщо поточний елемент не перший у списку. Поточний елемент лишається не змінним

```
def swap_next()
```

Міняє місцями поточний елемент списку з наступним. Гарантується, що виклик функції здійснюється лише, якщо поточний елемент не останній у списку. Поточний елемент лишається не змінним.

Тестова програма розташована за посиланням:

<https://github.com/krenevych/algo/tree/master/labs/T5/List/task2>

Завантажте всі файли, що містяться за цим посиланням, у одну папку. Реалізуйте згадані вище функції, інтерфейси яких містяться у файлі user2.py. Для перевірки правильності алгоритму запустіть файл main.py.

Завдання для самостійної роботи

- 14.3. Виведення зв'язного списку
<https://www.e-olymp.com/uk/problems/7452>
- 14.4. Обертання списку
<https://www.e-olymp.com/uk/problems/7453>
- 14.5. Зміна порядку списку
<https://www.e-olymp.com/uk/problems/7468>
- 14.6. Цикл у зв'язному списку
<https://www.e-olymp.com/uk/problems/7454>

Завдання для самостійної роботи підвищеної складності

- 14.7. Вікна
<https://www.e-olymp.com/uk/problems/2397>

ЛАБОРАТОРНА РОБОТА 15. Дерева. Алгоритми на деревах.

Контрольні запитання

15.1.

Завдання для аудиторної роботи

- 15.1. Манкуніанец і кольорове дерево
<https://www.e-olymp.com/uk/problems/3983>
- 15.2. LCA offline (Easy)
<https://www.e-olymp.com/uk/problems/2317>
Вказівка: LCA - Найменший спільний предок двох вершин

Завдання для самостійної роботи

- 15.3. Дерево
<https://www.e-olymp.com/uk/problems/2923>
- 15.4. Список телефонних номерів
<https://www.e-olymp.com/uk/problems/2303>

Завдання для самостійної роботи підвищеної складності

- 15.5. Рядки у дереві
<https://www.e-olymp.com/uk/problems/2925>
- 15.6. База даних
<https://www.e-olymp.com/uk/problems/3037>
- 15.7. Спільний предок – 2
<https://www.e-olymp.com/uk/problems/3299>
- 15.8. Дерево
<https://www.e-olymp.com/uk/problems/3710>
- 15.9. Дерево
<https://www.e-olymp.com/uk/problems/4173>
- 15.10. Дерево
<https://www.e-olymp.com/uk/problems/4325>

ЛАБОРАТОРНА РОБОТА 16. Бінарні дерева. Бінарні дерева пошуку.

Контрольні запитання

16.1.

Завдання для аудиторної роботи

- 16.1. Висота дерева
<https://www.e-olymp.com/uk/problems/2312>
- 16.2. Кількість елементів
<https://www.e-olymp.com/uk/problems/2313>
- 16.3. Другий максимум
<https://www.e-olymp.com/uk/problems/2314>
- 16.4. Вывод листьев
<https://www.e-olymp.com/uk/problems/2316>

Завдання для аудиторної роботи

16.5.	https://www.e-olymp.com/uk/problems/1513
-------	---

Варіанти індивідуальних завдань

16.1.	https://www.e-olymp.com/uk/problems/1846
16.2.	https://www.e-olymp.com/uk/problems/1516
16.3.	https://www.e-olymp.com/uk/problems/468
16.4.	https://www.e-olymp.com/uk/problems/2950
16.5.	https://www.e-olymp.com/uk/problems/3326

16.6.	https://www.e-olymp.com/uk/problems/2242
-------	---

ЛАБОРАТОРНА РОБОТА 17. Збалансовані бінарні дерева пошуку.

Контрольні запитання

17.1.

Завдання для аудиторної роботи

17.1. Збалансоване бінарне дерево
<https://www.e-olymp.com/uk/problems/7466>

17.2.

Завдання для самостійної роботи

Завдання для самостійної роботи підвищеної складності

ЛАБОРАТОРНА РОБОТА 18. Двійкова купа та пріоритетна черга.

Контрольні запитання

18.1.

Завдання для аудиторної роботи

18.1.	https://www.e-olymp.com/uk/problems/4847
-------	---

18.2.	https://www.e-olymp.com/uk/problems/3809
-------	---

18.3.	https://www.e-olymp.com/uk/problems/1228
-------	---

18.1.	https://www.e-olymp.com/uk/problems/4481
18.2.	https://www.e-olymp.com/uk/problems/4482
18.3.	https://www.e-olymp.com/uk/problems/2919
18.4.	https://www.e-olymp.com/uk/problems/3737
18.5.	https://www.e-olymp.com/uk/problems/4039

Варіанти індивідуальних завдань

18.4.	https://www.e-olymp.com/uk/problems/5952
18.5.	https://www.e-olymp.com/uk/problems/695
18.6.	https://www.e-olymp.com/uk/problems/4496
18.7.	https://www.e-olymp.com/uk/problems/3318
18.8.	https://www.e-olymp.com/uk/problems/3329
18.9.	https://www.e-olymp.com/uk/problems/402

18.10.	https://www.e-olymp.com/uk/problems/4487
18.11.	https://www.e-olymp.com/uk/problems/4488
18.12.	https://www.e-olymp.com/uk/problems/4491
18.13.	https://www.e-olymp.com/uk/problems/4492
18.14.	https://www.e-olymp.com/uk/problems/4082
18.15.	https://www.e-olymp.com/uk/problems/3358
18.16.	https://www.e-olymp.com/uk/problems/5274

ЛАБОРАТОРНА РОБОТА 19. Дерево відрізків.

Контрольні запитання

19.1.

19.1.	https://www.e-olymp.com/uk/problems/3809
-------	---

19.2.	https://www.e-olymp.com/uk/problems/3615
-------	---

19.3.	https://www.e-olymp.com/uk/problems/3615
-------	---

Завдання для аудиторної роботи

19.1.	https://www.e-olymp.com/uk/problems/4481
19.2.	https://www.e-olymp.com/uk/problems/4482
19.3.	https://www.e-olymp.com/uk/problems/2919
19.4.	https://www.e-olymp.com/uk/problems/3737
19.5.	https://www.e-olymp.com/uk/problems/4039

Варіанти індивідуальних завдань

19.4.	https://www.e-olymp.com/uk/problems/5952
19.5.	https://www.e-olymp.com/uk/problems/695
19.6.	https://www.e-olymp.com/uk/problems/4496
19.7.	https://www.e-olymp.com/uk/problems/3318
19.8.	https://www.e-olymp.com/uk/problems/3329
19.9.	https://www.e-olymp.com/uk/problems/402
19.10.	https://www.e-olymp.com/uk/problems/4487
19.11.	https://www.e-olymp.com/uk/problems/4488
19.12.	https://www.e-olymp.com/uk/problems/4491
19.13.	https://www.e-olymp.com/uk/problems/4492
19.14.	https://www.e-olymp.com/uk/problems/4082
19.15.	https://www.e-olymp.com/uk/problems/3358

19.16.	https://www.e-olymp.com/uk/problems/5274
--------	---

	Діма та масив https://www.e-olymp.com/uk/problems/2941
	Діма та великий масив https://www.e-olymp.com/uk/problems/2940
	Діма та масив https://www.e-olymp.com/uk/problems/2939

ЛАБОРАТОРНА РОБОТА 20. Алгоритми на не зважених графах.

Контрольні запитання

20.1.

Завдання для аудиторної роботи

20.1.	Зв'язність - https://www.e-olymp.com/uk/problems/982
20.2.	Обход в ширину - https://www.e-olymp.com/uk/problems/2401
20.3.	Перевірка на неорієнтовність – https://www.e-olymp.com/uk/problems/2470
20.4.	Від матриці суміжності до списку ребер – https://www.e-olymp.com/uk/problems/2471
20.5.	Витоки та стоки - https://www.e-olymp.com/uk/problems/3986
20.6.	Компоненти зв'язності - 2 – https://www.e-olymp.com/uk/problems/4816
20.7.	Найкоротша відстань - https://www.e-olymp.com/uk/problems/4852
20.8.	Найкоротший шлях - https://www.e-olymp.com/uk/problems/4853
20.9.	Підрахунок кількості ребер – https://www.e-olymp.com/uk/problems/5072
20.10.	Мультиребра - https://www.e-olymp.com/uk/problems/5073
20.11.	Степені вершин за списками ребер – https://www.e-olymp.com/uk/problems/5074
20.12.	Півстепені вершин за списками ребер – https://www.e-olymp.com/uk/problems/5075
20.13.	Кількість висячих вершин 1 – https://www.e-olymp.com/uk/problems/5080
20.14.	Степені вершин - https://www.e-olymp.com/uk/problems/5082

Варіанти індивідуальних завдань

20.1.	Маршрути в горах - https://www.e-olymp.com/uk/problems/122
20.2.	Стародавній рукопис - https://www.e-olymp.com/uk/problems/610
20.3.	Отримай дерево - https://www.e-olymp.com/uk/problems/978
20.4.	Топологічне сортування – https://www.e-olymp.com/uk/problems/1948
20.5.	Повний граф - https://www.e-olymp.com/uk/problems/3987

20.6.	Обхід у глибину - https://www.e-olymp.com/uk/problems/4000
20.7.	Геть списування! - https://www.e-olymp.com/uk/problems/4002
20.8.	Чи є цикл? - https://www.e-olymp.com/uk/problems/4004
20.9.	Числа - https://www.e-olymp.com/uk/problems/4007
20.10.	Підпал - https://www.e-olymp.com/uk/problems/4369
20.11.	Зв'язність графа - https://www.e-olymp.com/uk/problems/4374
20.12.	Магічна машинка - https://www.e-olymp.com/uk/problems/4850
20.13.	Регулярний граф - https://www.e-olymp.com/uk/problems/5076
20.14.	Напівповний граф - https://www.e-olymp.com/uk/problems/5077
20.15.	Транзитивність орієнтовного графа – https://www.e-olymp.com/uk/problems/5079
20.16.	Покриття шляхами - https://www.e-olymp.com/uk/problems/5299

Варіанти для командних завдань

20.1.	Рекурсія - https://www.e-olymp.com/uk/problems/553
20.2.	Ходи ферзем! - https://www.e-olymp.com/uk/problems/1098
20.3.	Сильная связность - https://www.e-olymp.com/uk/problems/2403
20.4.	Забавна гра - https://www.e-olymp.com/uk/problems/4050
20.5.	Хрещений батько - https://www.e-olymp.com/uk/problems/5366

Допоміжна література по темі

20.1.

ЛАБОРАТОРНА РОБОТА 21. Алгоритми на зважених графах

Контрольні запитання

21.1.

Завдання для аудиторної роботи

21.1.	Алгоритм Дейкстри - https://www.e-olymp.com/uk/problems/1365
21.2.	Найкоротший шлях - https://www.e-olymp.com/uk/problems/4856
21.3.	

Варіанти індивідуальних завдань

21.1.	Відстань між вершинами – https://www.e-olymp.com/uk/problems/625
21.2.	Флойд - 1 - https://www.e-olymp.com/uk/problems/974
21.3.	Флойд - https://www.e-olymp.com/uk/problems/975
21.4.	Флойд - існування - https://www.e-olymp.com/uk/problems/976
21.5.	Мінімальний каркас - https://www.e-olymp.com/uk/problems/981
21.6.	Червякові діри - https://www.e-olymp.com/uk/problems/1108
21.7.	Заправки - https://www.e-olymp.com/uk/problems/1388
21.8.	Автобуси - https://www.e-olymp.com/uk/problems/1389
21.9.	Форд-Беллман - https://www.e-olymp.com/uk/problems/1453
21.10.	Лабіринт знань - https://www.e-olymp.com/uk/problems/1454
21.11.	Цикл - https://www.e-olymp.com/uk/problems/1455
21.12.	Ліфти - https://www.e-olymp.com/uk/problems/2209
21.13.	Мінімальний каркас - https://www.e-olymp.com/uk/problems/3835
21.14.	Доставка кефірчика - https://www.e-olymp.com/uk/problems/4006
21.15.	

Варіанти для командних завдань

21.1.	Складний тест - https://www.e-olymp.com/uk/problems/626
21.2.	Мандри - https://www.e-olymp.com/uk/problems/2267
21.3.	Альтернативні шляхи - https://www.e-olymp.com/uk/problems/4637
21.4.	

ЛАБОРАТОРНА РОБОТА 22. Лабіринти.

Контрольні запитання

22.1.

Завдання для аудиторної роботи

22.1.	Площа кімнати - https://www.e-olymp.com/uk/problems/4001
22.2.	

Варіанти індивідуальних завдань

22.1.	Помста Лі Чака - https://www.e-olymp.com/uk/problems/88
22.2.	Підземелля - https://www.e-olymp.com/uk/problems/432
22.3.	Лінії - https://www.e-olymp.com/uk/problems/1060
22.4.	Фарбування лабіринту - https://www.e-olymp.com/uk/problems/1061
22.5.	Lines - https://www.e-olymp.com/uk/problems/1062
22.6.	Видалення клітинок - https://www.e-olymp.com/uk/problems/1063
22.7.	Шлях коня - https://www.e-olymp.com/uk/problems/1064
22.8.	Грядки - https://www.e-olymp.com/uk/problems/1065
22.9.	Переміщення коня - https://www.e-olymp.com/uk/problems/2820
22.10.	Іграшковий лабіринт - https://www.e-olymp.com/uk/problems/4452
22.11.	Вихід з лабіринту - https://www.e-olymp.com/uk/problems/4820
22.12.	Місце зустрічі змінити неможна – https://www.e-olymp.com/uk/problems/5069
22.13.	Лабіринт - https://www.e-olymp.com/uk/problems/5622
22.14.	Лабіринт - https://www.e-olymp.com/uk/problems/7215

Варіанти для командних завдань

22.1.	Виведи хомячків - https://www.e-olymp.com/uk/problems/213
22.2.	Сон - https://www.e-olymp.com/uk/problems/499
22.3.	Золота рибка – друге бажання Петрика – https://www.e-olymp.com/uk/problems/4634
22.4.	Гном і монети - https://www.e-olymp.com/uk/problems/7229

Допоміжна література по темі

22.1.

Список літератури та додаткових джерел

1. **Кренивч А.П.** Алгоритми та структури даних. Підручник. – К.: ВПЦ "Київський Університет", 2020. – 777 с.