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)

Креневич 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

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 місять 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/експоненційний?
1.7. Наведіть означення таких понять як найкращий/найгірший час виконання програми. Що таке час виконання програми в середньому.
1.8. Наведіть означення O-оцінки: що означає f=O(g)?
1.9. Наведіть означення Омега-оцінки: що означає f=Омега(g)?
1.10. Наведіть означення Тета-оцінки: що означає f=Тета(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)

```
i = 0
while i < n:
    j = 0
    while j < i * i:
        k += 1
        j += 1
    i += 1
```

h)

```
i = 0
while i < n:
    j = n
    while j != 0:
        k += 1
        j //= 3
    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) $\sqrt{2+\sqrt{2+\cdots+\sqrt{2}}}$, ($n$ коренів);

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

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

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

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

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:

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

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:

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

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 ($0 \le n \le 2000000000$).

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

Вхідні дані #1	*Вихідні дані #1*
5	2
Вхідні дані #2	*Вихідні дані #2*
7	3

Зауваження. Скористайтеся 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. Затятий колекціонер метеликів
https://www.e-olymp.com/uk/problems/3966

2.6. 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`.

2.7. 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`.

2.8. Мутанти
https://www.e-olymp.com/uk/problems/3970

2.9. Реалізуйте алгоритм лінійного пошуку всіх елементів списку, що є степенями двійки та визначте асимптотичну складність отриманого алгоритму.

2.10. 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?

2.11. Реалізуйте алгоритм бінарного пошуку з використанням рекурсії.

ЛАБОРАТОРНА РОБОТА 3.
РОЗВ'ЯЗАННЯ РІВНЯНЬ МЕТОДОМ БІЄКЦІЇ.

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. 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.

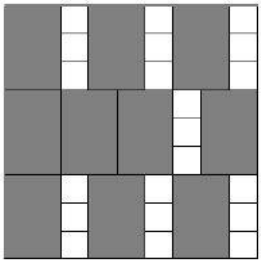
Контрольні запитання

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 виявилося, 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 ($1 \le w, h, n \le 109$).

Вихідні дані
У вихідний файл вивести відповідь до сформульованої задачі.

Вхідні дані #1	*Вихідні дані #1*
2 3 10	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 ($1 \le n \le 2 \cdot 108$, $1 \le x, y \le 10$).

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

Вхідні дані #1	*Вихідні дані #1*
4 1 1	3
Вхідні дані #2	*Вихідні дані #2*
5 1 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 реалзується 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 цією 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):
    """ Перевіряє 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 міститься 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. https://www.e-olymp.com/uk/problems/4842
5.4. https://www.e-olymp.com/uk/problems/1227

Завдання для самостійної роботи підвищеної складності

5.5. https://www.e-olymp.com/uk/problems/131
5.6. https://www.e-olymp.com/uk/problems/2035
5.7. 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 реалізовуючи хеш-таблицю з розв’язанням колізій методом ланцюжків.

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

Завдання для самостійної роботи

6.3. https://www.e-olymp.com/uk/problems/4842

6.4. https://www.e-olymp.com/uk/problems/1227

6.5. https://www.e-olymp.com/uk/problems/131

Завдання для самостійної роботи підвищеної складності

6.6. https://www.e-olymp.com/uk/problems/2035

6.7. 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):
    """ 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 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):
    """ 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 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. https://www.e-olymp.com/uk/problems/2663
7.6. https://www.e-olymp.com/uk/problems/5089

Завдання для самостійної роботи підвищеної складності

7.7. https://www.e-olymp.com/uk/problems/2664
7.8. https://www.e-olymp.com/uk/problems/972
7.9. https://www.e-olymp.com/uk/problems/1130
7.10. https://www.e-olymp.com/uk/problems/1344
7.11. https://www.e-olymp.com/uk/problems/1462
7.12. https://www.e-olymp.com/uk/problems/2662
7.13. 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
              # nlog(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 згадані 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. https://www.e-olymp.com/ru/problems/3607
8.5. https://www.e-olymp.com/uk/problems/2321
8.6. https://www.e-olymp.com/uk/problems/4037

Завдання для самостійної роботи підвищеної складності

8.7. https://www.e-olymp.com/uk/problems/1303
8.8. 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.	https://www.e-olymp.com/uk/problems/364
9.2.	https://www.e-olymp.com/uk/problems/1296
9.3.	https://www.e-olymp.com/uk/problems/1540
9.4.	https://www.e-olymp.com/uk/problems/1592
9.5.	https://www.e-olymp.com/uk/problems/1781

Варіанти індивідуальних завдань

9.1.	https://www.e-olymp.com/uk/problems/224
9.2.	https://www.e-olymp.com/uk/problems/1266
9.3.	https://www.e-olymp.com/uk/problems/1524
9.4.	https://www.e-olymp.com/uk/problems/2000
9.5.	https://www.e-olymp.com/uk/problems/2633
9.6.	https://www.e-olymp.com/uk/problems/2764
9.7.	https://www.e-olymp.com/uk/problems/3606
9.8.	https://www.e-olymp.com/uk/problems/4746
9.9.	https://www.e-olymp.com/uk/problems/7031
9.10.	https://www.e-olymp.com/uk/problems/7227
9.11.	https://www.e-olymp.com/ru/problems/6

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

Контрольні запитання

10.1.

Завдання для аудиторної роботи

10.1.	
10.2.	
10.3.	
10.4.	
10.5.	

Варіанти індивідуальних завдань

10.1.	https://www.e-olymp.com/uk/problems/317 – Множення Карацуби
10.2.	

ЛАБОРАТОРНА РОБОТА 11. Стек - базові алгорими..

Контрольні запитання

11.1.

Завдання для аудиторної роботи

11.1.	Простий стек https://www.e-olymp.com/uk/problems/6122
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
11.4.	
11.5.	
11.6.	
11.7.	
11.8.	

Варіанти індивідуальних завдань

11.1.	
11.2.	
11.3.	
11.4.	
11.5.	
11.6.	
11.7.	
11.8.	
11.9.	
11.10.	
11.11.	
11.12.	
11.13.	
11.14.	
11.15.	

ЛАБОРАТОРНА РОБОТА 12. Стек та його застосування.

Контрольні запитання

12.1.

Завдання для аудиторної роботи

12.1.	https://www.e-olymp.com/uk/problems/1994
12.2.	https://www.e-olymp.com/uk/problems/4259
12.3.	https://www.e-olymp.com/uk/problems/2479
12.4.	https://www.e-olymp.com/uk/problems/7433
12.5.	https://www.e-olymp.com/uk/problems/855
12.6.	https://www.e-olymp.com/uk/problems/731
12.7.	https://www.e-olymp.com/uk/problems/5322
12.8.	

Варіанти індивідуальних завдань

12.1.	https://www.e-olymp.com/uk/problems/1376
12.2.	https://www.e-olymp.com/uk/problems/5205
12.3.	https://www.e-olymp.com/uk/problems/3013
12.4.	https://www.e-olymp.com/uk/problems/4438
12.5.	https://www.e-olymp.com/uk/problems/1801
12.6.	https://www.e-olymp.com/uk/problems/2039
12.7.	https://www.e-olymp.com/uk/problems/5539
12.8.	https://www.e-olymp.com/uk/problems/2702
12.9.	https://www.e-olymp.com/uk/problems/1008
12.10.	https://www.e-olymp.com/uk/problems/1377
12.11.	https://www.e-olymp.com/uk/problems/1394
12.12.	https://www.e-olymp.com/uk/problems/1073
12.13.	https://www.e-olymp.com/uk/problems/1515
12.14.	https://www.e-olymp.com/uk/problems/1549
12.15.	https://www.e-olymp.com/uk/problems/3837
	https://www.e-olymp.com/ru/problems/1776

Допоміжна література по темі

12.1.

ЛАБОРАТОРНА РОБОТА 13. Черги та зв’язні списки. Їхнє застосування

Контрольні запитання

13.1.

Завдання для аудиторної роботи

13.1.	https://www.e-olymp.com/uk/problems/971
13.2.	https://www.e-olymp.com/uk/problems/6125
13.3.	https://www.e-olymp.com/uk/problems/6126
13.4.	https://www.e-olymp.com/uk/problems/6127
13.5.	https://www.e-olymp.com/uk/problems/6128
13.6.	https://www.e-olymp.com/uk/problems/6129
13.7.	https://www.e-olymp.com/uk/problems/6130
13.8.	https://www.e-olymp.com/uk/problems/4847

Варіанти індивідуальних завдань

13.1.	https://www.e-olymp.com/uk/problems/3615
13.2.	https://www.e-olymp.com/uk/problems/1549
13.3.	https://www.e-olymp.com/uk/problems/1515
13.4.	https://www.e-olymp.com/uk/problems/2510
13.5.	https://www.e-olymp.com/uk/problems/3809
13.6.	https://www.e-olymp.com/uk/problems/4847
13.7.	https://www.e-olymp.com/uk/problems/694
13.8.	https://www.e-olymp.com/ru/problems/1228
13.9.	https://www.e-olymp.com/ru/problems/4005
13.10.	https://informatics.msk.ru/mod/statements/view.php?id=206
13.11.	https://informatics.msk.ru/mod/statements/view3.php?id=206&chapterid=112984
13.12.	https://informatics.msk.ru/mod/statements/view3.php?id=206&chapterid=53
13.13.	https://informatics.msk.ru/mod/statements/view3.php?id=601&chapterid=746

ЛАБОРАТОРНА РОБОТА 14. Дерева. Алгоритми на деревах.

Контрольні запитання

14.1.

Завдання для аудиторної роботи

14.1.	https://www.e-olymp.com/uk/problems/2317
14.2.	

Варіанти індивідуальних завдань

14.1.	https://www.e-olymp.com/uk/problems/2925
14.2.	https://www.e-olymp.com/uk/problems/4325
14.3.	https://www.e-olymp.com/uk/problems/4173
14.4.	https://www.e-olymp.com/uk/problems/5067
14.5.	https://www.e-olymp.com/uk/problems/2242
14.6.	https://www.e-olymp.com/uk/problems/2923
14.7.	https://www.e-olymp.com/uk/problems/3710
14.8.	https://www.e-olymp.com/uk/problems/3983
14.9.	https://www.e-olymp.com/uk/problems/3299
14.3.	https://www.e-olymp.com/ru/problems/3037

Допоміжна література по темі

14.1.

ЛАБОРАТОРНА РОБОТА 15. Бінарні дерева. Бінарні дерева пошуку.

Контрольні запитання

15.1.

Завдання для аудиторної роботи

15.1.	https://www.e-olymp.com/uk/problems/2312
15.2.	https://www.e-olymp.com/uk/problems/2316
15.3.	https://www.e-olymp.com/uk/problems/1513
15.4.	https://www.e-olymp.com/uk/problems/7466
15.5.	https://www.e-olymp.com/uk/problems/2314

Варіанти індивідуальних завдань

15.1.	Дерево синтаксичного розбору – http://aliev.me/runestone/Trees/ParseTree.html
15.2.	https://www.e-olymp.com/uk/problems/1846
15.3.	https://www.e-olymp.com/uk/problems/1516
15.4.	https://www.e-olymp.com/uk/problems/468
15.5.	https://www.e-olymp.com/uk/problems/2950
15.6.	https://www.e-olymp.com/uk/problems/3326

Допоміжна література по темі

15.1.

ЛАБОРАТОРНА РОБОТА 16. Бінарна купа та дерево відрізків.

Контрольні запитання

16.1.

Завдання для аудиторної роботи

16.1.	https://www.e-olymp.com/uk/problems/4481
16.2.	https://www.e-olymp.com/uk/problems/4482
16.3.	https://www.e-olymp.com/uk/problems/2919
16.4.	https://www.e-olymp.com/uk/problems/3737
16.5.	https://www.e-olymp.com/uk/problems/4039

Варіанти індивідуальних завдань

16.1.	https://www.e-olymp.com/uk/problems/5952
16.2.	https://www.e-olymp.com/uk/problems/695
16.3.	https://www.e-olymp.com/uk/problems/4496
16.4.	https://www.e-olymp.com/uk/problems/3318
16.5.	https://www.e-olymp.com/uk/problems/3329
16.6.	https://www.e-olymp.com/uk/problems/402
16.7.	https://www.e-olymp.com/uk/problems/4487
16.8.	https://www.e-olymp.com/uk/problems/4488
16.9.	https://www.e-olymp.com/uk/problems/4491
16.10.	https://www.e-olymp.com/uk/problems/4492
16.11.	https://www.e-olymp.com/uk/problems/4082
16.12.	https://www.e-olymp.com/uk/problems/3358
16.13.	https://www.e-olymp.com/uk/problems/5274

ЛАБОРАТОРНА РОБОТА 17. Алгоритми на не зважених графах.

Контрольні запитання

17.1.

Завдання для аудиторної роботи

17.1.	Зв'язність - https://www.e-olymp.com/uk/problems/982
17.2.	Обход в ширину - https://www.e-olymp.com/uk/problems/2401
17.3.	Перевірка на неорієнтовність – https://www.e-olymp.com/uk/problems/2470
17.4.	Від матриці суміжності до списку ребер – https://www.e-olymp.com/uk/problems/2471
17.5.	Витоки та стоки - https://www.e-olymp.com/uk/problems/3986
17.6.	Компоненти зв`язності - 2 – https://www.e-olymp.com/uk/problems/4816
17.7.	Найкоротша відстань - https://www.e-olymp.com/uk/problems/4852
17.8.	Найкоротший шлях - https://www.e-olymp.com/uk/problems/4853
17.9.	Підрахунок кількості ребер – https://www.e-olymp.com/uk/problems/5072
17.10.	Мультиребра - https://www.e-olymp.com/uk/problems/5073
17.11.	Степені вершин за списками ребер – https://www.e-olymp.com/uk/problems/5074
17.12.	Півстепені вершин за списками ребер – https://www.e-olymp.com/uk/problems/5075
17.13.	Кількість висячих вершин 1 – https://www.e-olymp.com/uk/problems/5080
17.14.	Степені вершин - https://www.e-olymp.com/uk/problems/5082

Варіанти індивідуальних завдань

17.1.	Маршрути в горах - https://www.e-olymp.com/uk/problems/122
17.2.	Стародавній рукопис - https://www.e-olymp.com/uk/problems/610
17.3.	Отримай дерево - https://www.e-olymp.com/uk/problems/978
17.4.	Топологічне сортування – https://www.e-olymp.com/uk/problems/1948
17.5.	Повний граф - https://www.e-olymp.com/uk/problems/3987

17.6.	Обхід у глибину - https://www.e-olymp.com/uk/problems/4000
17.7.	Геть списування! - https://www.e-olymp.com/uk/problems/4002
17.8.	Чи є цикл? - https://www.e-olymp.com/uk/problems/4004
17.9.	Числа - https://www.e-olymp.com/uk/problems/4007
17.10.	Підпал - https://www.e-olymp.com/uk/problems/4369
17.11.	Зв`язність графа - https://www.e-olymp.com/uk/problems/4374
17.12.	Магічна машинка - https://www.e-olymp.com/uk/problems/4850
17.13.	Регулярний граф - https://www.e-olymp.com/uk/problems/5076
17.14.	Напівповний граф - https://www.e-olymp.com/uk/problems/5077
17.15.	Транзитивність орієнтовного графа – https://www.e-olymp.com/uk/problems/5079
17.16.	Покриття шляхами - https://www.e-olymp.com/uk/problems/5299

Варіанти для командних завдань

17.1.	Рекурсія - https://www.e-olymp.com/uk/problems/553
17.2.	Ходи ферзем! - https://www.e-olymp.com/uk/problems/1098
17.3.	Сильная связность - https://www.e-olymp.com/uk/problems/2403
17.4.	Забавна гра - https://www.e-olymp.com/uk/problems/4050
17.5.	Хрещений батько - https://www.e-olymp.com/uk/problems/5366

Допоміжна література по темі

17.1.

ЛАБОРАТОРНА РОБОТА 18. Алгоритми на зважених графах

Контрольні запитання

18.1.

Завдання для аудиторної роботи

18.1.	Алгоритм Дейкстри - https://www.e-olymp.com/uk/problems/1365
18.2.	Найкоротший шлях - https://www.e-olymp.com/uk/problems/4856
18.3.	

Варіанти індивідуальних завдань

18.1.	Відстань між вершинами – https://www.e-olymp.com/uk/problems/625
18.2.	Флойд - 1 - https://www.e-olymp.com/uk/problems/974
18.3.	Флойд - https://www.e-olymp.com/uk/problems/975
18.4.	Флойд - існування - https://www.e-olymp.com/uk/problems/976
18.5.	Мінімальний каркас - https://www.e-olymp.com/uk/problems/981
18.6.	Червякові діри - https://www.e-olymp.com/uk/problems/1108
18.7.	Заправки - https://www.e-olymp.com/uk/problems/1388
18.8.	Автобуси - https://www.e-olymp.com/uk/problems/1389
18.9.	Форд-Беллман - https://www.e-olymp.com/uk/problems/1453
18.10.	Лабіринт знань - https://www.e-olymp.com/uk/problems/1454
18.11.	Цикл - https://www.e-olymp.com/uk/problems/1455
18.12.	Ліфти - https://www.e-olymp.com/uk/problems/2209
18.13.	Мінімальний каркас - https://www.e-olymp.com/uk/problems/3835
18.14.	Доставка кефірчика - https://www.e-olymp.com/uk/problems/4006
18.15.	

Варіанти для командних завдань

18.1.	Складний тест - https://www.e-olymp.com/uk/problems/626
18.2.	Мандри - https://www.e-olymp.com/uk/problems/2267
18.3.	Альтернативні шляхи - https://www.e-olymp.com/uk/problems/4637
18.4.	

ЛАБОРАТОРНА РОБОТА 19. Лабіринти.

Контрольні запитання

19.1.

Завдання для аудиторної роботи

19.1.	Площа кімнати - https://www.e-olymp.com/uk/problems/4001
19.2.	

Варіанти індивідуальних завдань

19.1.	Помста Лі Чака - https://www.e-olymp.com/uk/problems/88
19.2.	Підземелля - https://www.e-olymp.com/uk/problems/432
19.3.	Лінії - https://www.e-olymp.com/uk/problems/1060
19.4.	Фарбування лабіринту - https://www.e-olymp.com/uk/problems/1061
19.5.	Lines - https://www.e-olymp.com/uk/problems/1062
19.6.	Видалення клітинок - https://www.e-olymp.com/uk/problems/1063
19.7.	Шлях коня - https://www.e-olymp.com/uk/problems/1064
19.8.	Грядки - https://www.e-olymp.com/uk/problems/1065
19.9.	Переміщення коня - https://www.e-olymp.com/uk/problems/2820
19.10.	Іграшковий лабіринт - https://www.e-olymp.com/uk/problems/4452
19.11.	Вихід з лабіринту - https://www.e-olymp.com/uk/problems/4820
19.12.	Місце зустічі змінити неможна – https://www.e-olymp.com/uk/problems/5069
19.13.	Лабіринт - https://www.e-olymp.com/uk/problems/5622
19.14.	Лабіринт - https://www.e-olymp.com/uk/problems/7215

Варіанти для командних завдань

19.1.	Виведи хомячків - https://www.e-olymp.com/uk/problems/213
19.2.	Сон - https://www.e-olymp.com/uk/problems/499
19.3.	Золота рибка – друге бажання Петрика – https://www.e-olymp.com/uk/problems/4634
19.4.	Гном і монети - https://www.e-olymp.com/uk/problems/7229

Допоміжна література по темі

19.1.

Список літератури та додаткових джерел

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