Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Алгоритми та структури даних

Лабораторна робота №4

«Елементарні методи сортування»

Виконала:
студентка групи ІО-64
Бровченко А. В.
Перевірив:
Саверченко В. Г.

Київ
2016 р.

Алгоритм сортування — це алгоритм, що розв'язує задачу сортування, тобто здійснює впорядкування лінійного списку (масиву) елементів.

Для алгоритму сортування (як і для будь-якого іншого сучасного алгоритму) основними характеристиками є: час необхідний на впорядкування n-елементного масиву і додаткова пам'ять необхідна для впорядкування. Крім цих двох характеристик, сортування буває стабільним чи нестабільним, з використанням додаткової інформації про елементи, чи без використання.

Для значної кількості алгоритмів середній і найгірший час впорядкування n-елементного масиву є $O(n^2)$, це пов'язано з тим, що в них передбачені перестановки елементів, що стоять поряд (різниця між індексами елементів не перевищує деякого заданого числа). Такі алгоритми зазвичай є стабільними, хоча і не ефективними для великих масивів.

Інший клас алгоритмів здійснює впорядкування за час $O(n \log n)$. В цих алгоритмах використовується можливість обміну елементів, що знаходяться на будь-якій відстані один від одного.

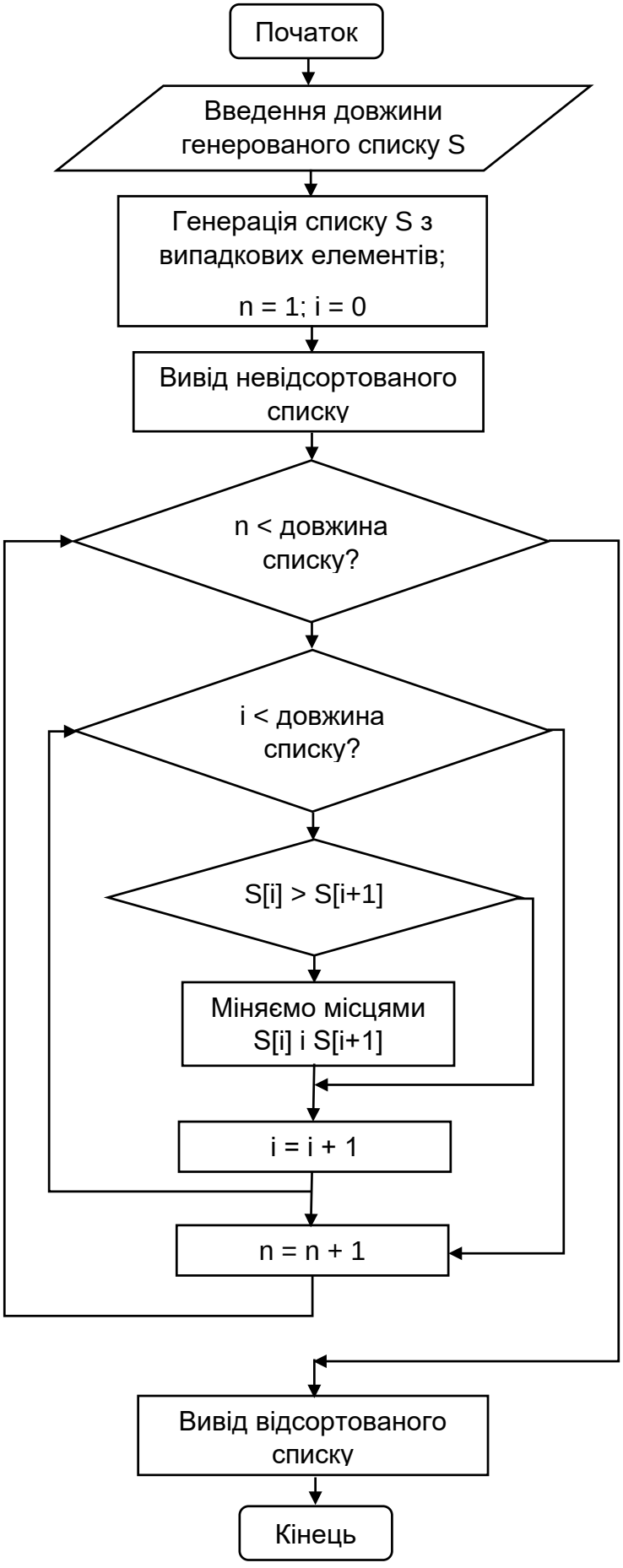
Сортування обміном або **сортування бульбашкою**

```
import random
s = []
length = int(input("Enter length: "))
for i in range(length):
    s.append(random.randint(-50, 50))
print(s)
n = 1
while n < len(s):
    for i in range(len(s)-n):
        if s[i] > s[i+1]:
            s[i], s[i + 1] = s[i + 1], s[i]
    n += 1
print(s)
```

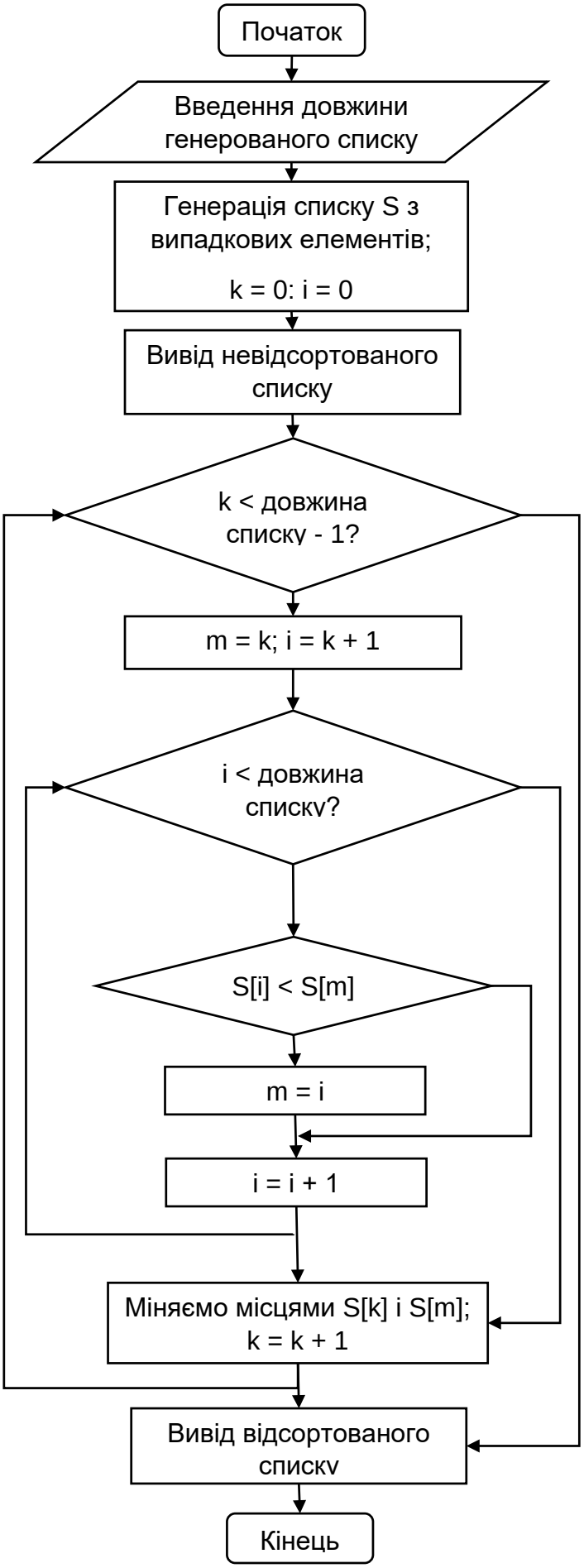
Сортування вибором

```
import random
s = []
length = int(input("Enter length: "))
for i in range(length):
    s.append(random.randint(-50, 50))
print(s)
k = 0
while k < len(s) - 1:
    m = k
    i = k + 1
    while i < len(s):
        if s[i] < s[m]:
            m = i
        i += 1
    s[k], s[m] = s[m], s[k]
    k += 1
print(s)
```

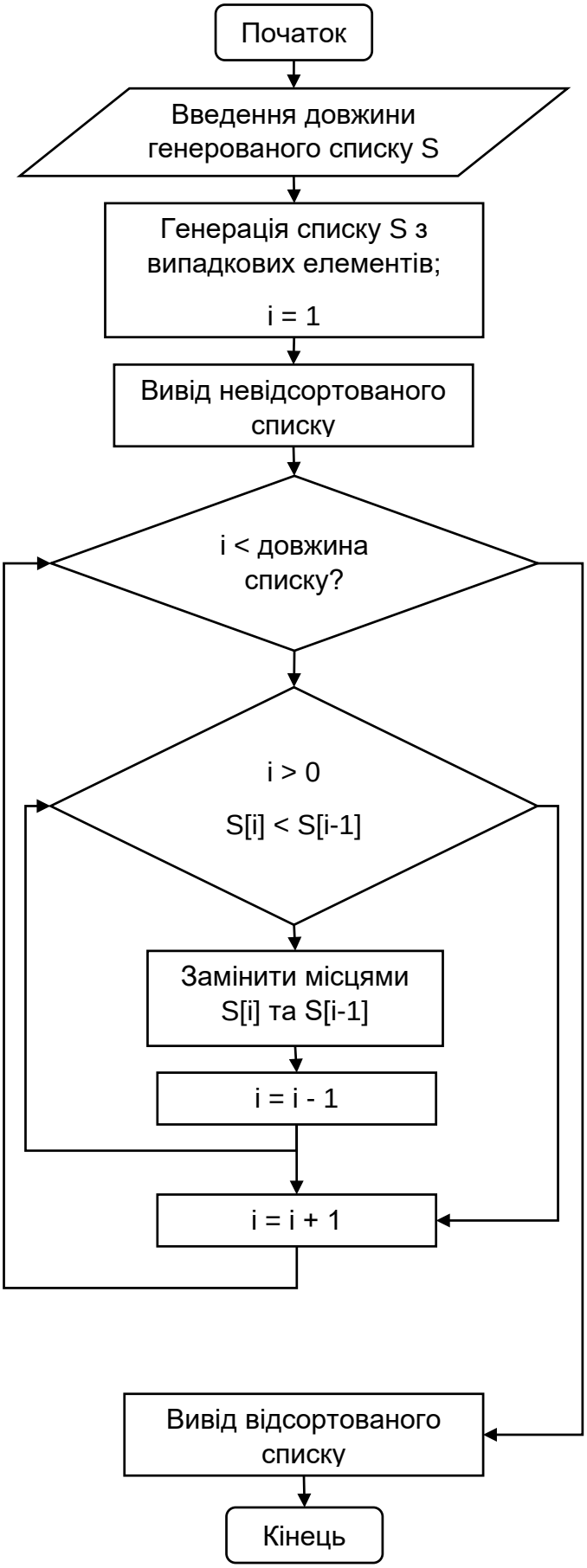
Сортування бульбашкою



Сортування вибором



Сортування включенням




```
import random
s = []
length = int(input("Enter length: "))
for i in range(length):
    s.append(random.randint(-50, 50))
print(s)
for i in range(1, len(s)):
    while i > 0 and s[i] < s[i - 1]:
            s[i], s[i-1] = s[i-1], s[i]
            i -= 1
print(s)
```

Висновок:

Складність алгоритму **сортування бульбашкою** у найгіршому у середньостатистичному випадку рівна $O(n^2)$, де n — кількість елементів для сортування. Існує чимало значно ефективніших алгоритмів, наприклад, з найгіршою ефективністю рівною $O(n \log n)$. Тому даний алгоритм має низьку ефективність у випадках, коли N є досить великим, за винятком рідкісних конкретних випадків, коли заздалегідь відомо, що масив з самого початку буде добре відсортований.

Сортування вибором — простий алгоритм сортування лінійного масиву, на основі вставок. Має ефективність n^2, що робить його неефективним при сортування великих масивів, і в цілому, менш ефективним за подібний алгоритм сортування включенням. Сортування вибором вирізняється більшою простотою, ніж сортування включенням, і в деяких випадках, вищою продуктивністю.

Сортування включенням — простий алгоритм сортування на основі порівнянь. На великих масивах є не дуже ефективним. Однак, має цілу низку переваг:

- простота у реалізації
- ефективний (зазвичай) на маленьких масивах
- ефективний при сортуванні масивів, дані в яких вже непогано відсортовані: продуктивність рівна $O(n + d)$, де d — кількість інверсій
- на практиці ефективніший за більшість інших квадратичних алгоритмів ($O(n^2)$), як то сортування вибором та сортування бульбашкою: його швидкодія рівна $n^2/4$, і в найкращому випадку є лінійною
- є стабільним алгоритмом