Практична робота №3

№1

Алгоритм бульбашкового сортування:

1. Порівнюємо сусідні елементи масиву.
2. Якщо перший елемент більше другого, міняємо їх місцями.

3. Повторюємо цю операцію з кожною парою сусідніх елементів до кінця масиву.

4 .Після першого проходу найбільший елемент "спливає" на останню позицію.

1. Повторюємо кроки 1-4 для решти масиву, виключаючи вже відсортовані елементи.

Код алгоритму на Pyhton:

def bubble\_sort(arr):

n=len(arr):

for in range(n):

for j in range(0, n - i - 1):

if(arr)[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

return arr

Асимптотична оцінка

1. Найгірший випадок: O(n²) – коли масив відсортований у зворотному порядку.
2. Найкращий випадок: O(n) – коли масив вже відсортований. Проте для цього випадку потрібно реалізувати оптимізований варіант бульбашкового сортування, який відстежує, чи були зміни під час проходу.

Порівняння з сортуванням вставлянням

Сортування вставлянням також має асимптотику O(n²) в найгіршому випадку. Однак для частково відсортованих масивів вставки працюють значно швидше і мають асимптотику O(n) в найкращому випадку.

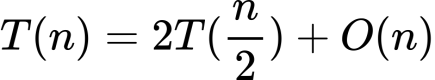
Для бульбашкового сортування кожен елемент переміщується багато разів, навіть якщо масив майже відсортований, що робить його менш ефективним у порівнянні з сортуванням вставлянням.

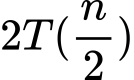
Чому бульбашковий алгоритм менш ефективний, ніж метод злиття?

1. Асимптотика: Бульбашкове сортування має O(n²) в найгіршому випадку, тоді як метод злиття – O(n log n).
2. Кількість операцій: Бульбашковий алгоритм виконує велику кількість перестановок сусідніх елементів, що уповільнює його на великих масивах. Метод злиття, навпаки, поділяє масив і сортує його частини рекурсивно, що значно зменшує кількість операцій.

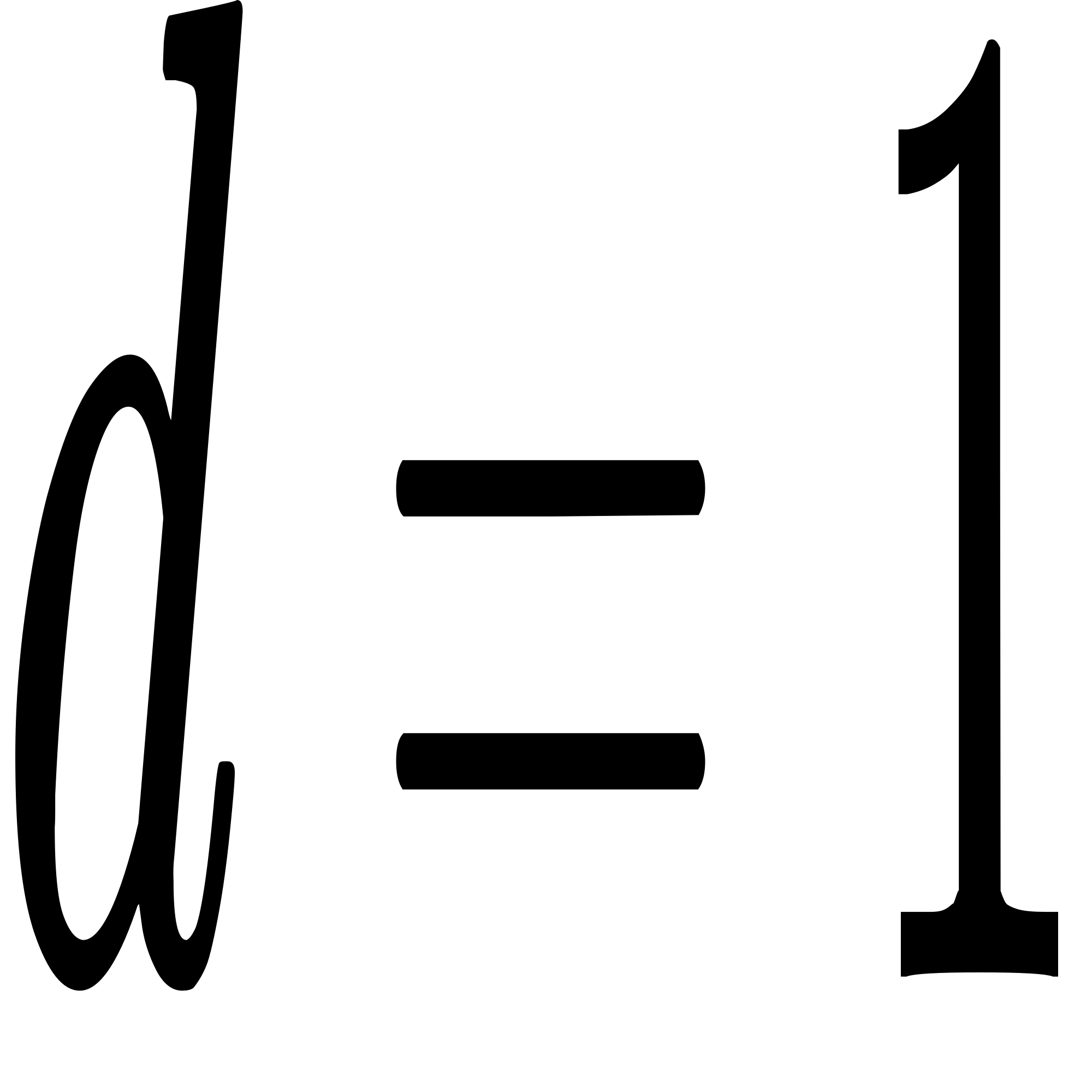
№2

Для сортування злиттям використовуємо рекурсивний вираз:



де  - час сортування двох половин, wps - час злиття.

За основною теоремою рекурсії:

1. wps
2. Порівняємо з (з wps, і оскільки вони рівін, складність - wps

Отже, для сортування злиттям асимптотична складність — O(n log n), що робить його ефективнішим за бульбашкове сортування wps.

№3

Алгоритм швидкого сортування працює так:

1. Вибирається опорний елемент (pivot).
2. Масив розділяється на дві частини: елементи, менші за опорний, і елементи, більші за нього.
3. Кожна з частин рекурсивно сортується.
4. Після сортування об'єднуються ліві, опорний, і праві частини.

Код алгоритму швидкого сортування на Python:

def quicksort(arr):

if len(arr) <= 1:

return arr

pivot = arr[len(arr) // 2] # Вибір опорного елемента

left = [x for x in arr if x < pivot]

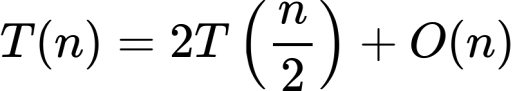
middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

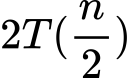
return quicksort(left) + middle + quicksort(right)

Оцінка асимптотичної складності

Рекурсивний вираз для швидкого сортування:

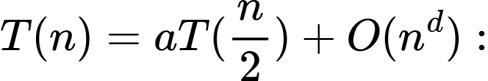


де:

 - час для сортування двох половин,

wps - час для сортування розподілу елементів відносно опорного.

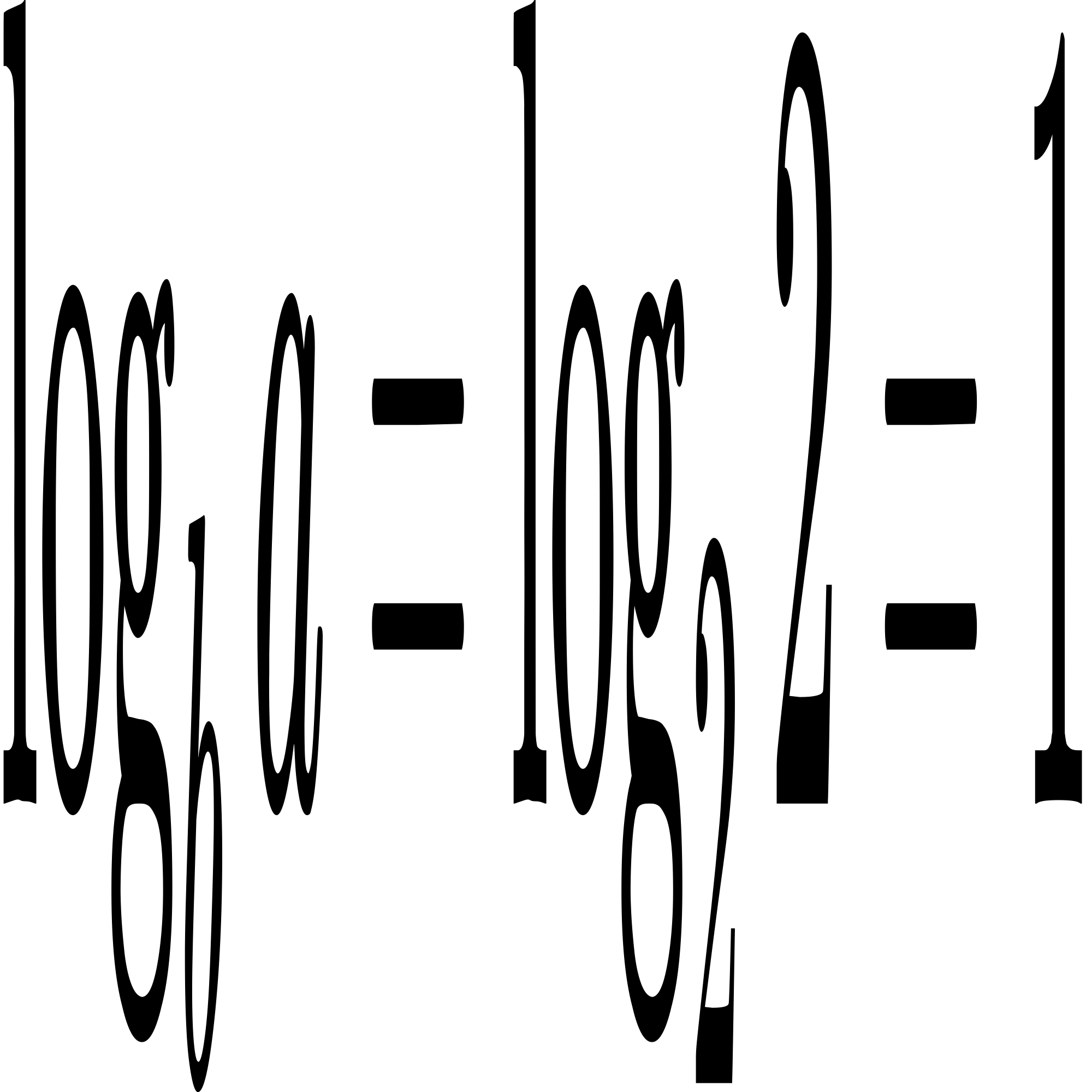
Асимптотика за основною теоремою рекурсії

Для 

a=2(дві частини)

b=2(масив ділиться навпіл),

d=1(лінійний час розподілу),

Рахуємо .

Оскільки wps, асимпотитчна складність:

wps

Висновок: У середньому та найкращому випадках швидке сортування має складність O(n log n), але в найгіршому (коли масив вже відсортований) складність зростає до wps.