

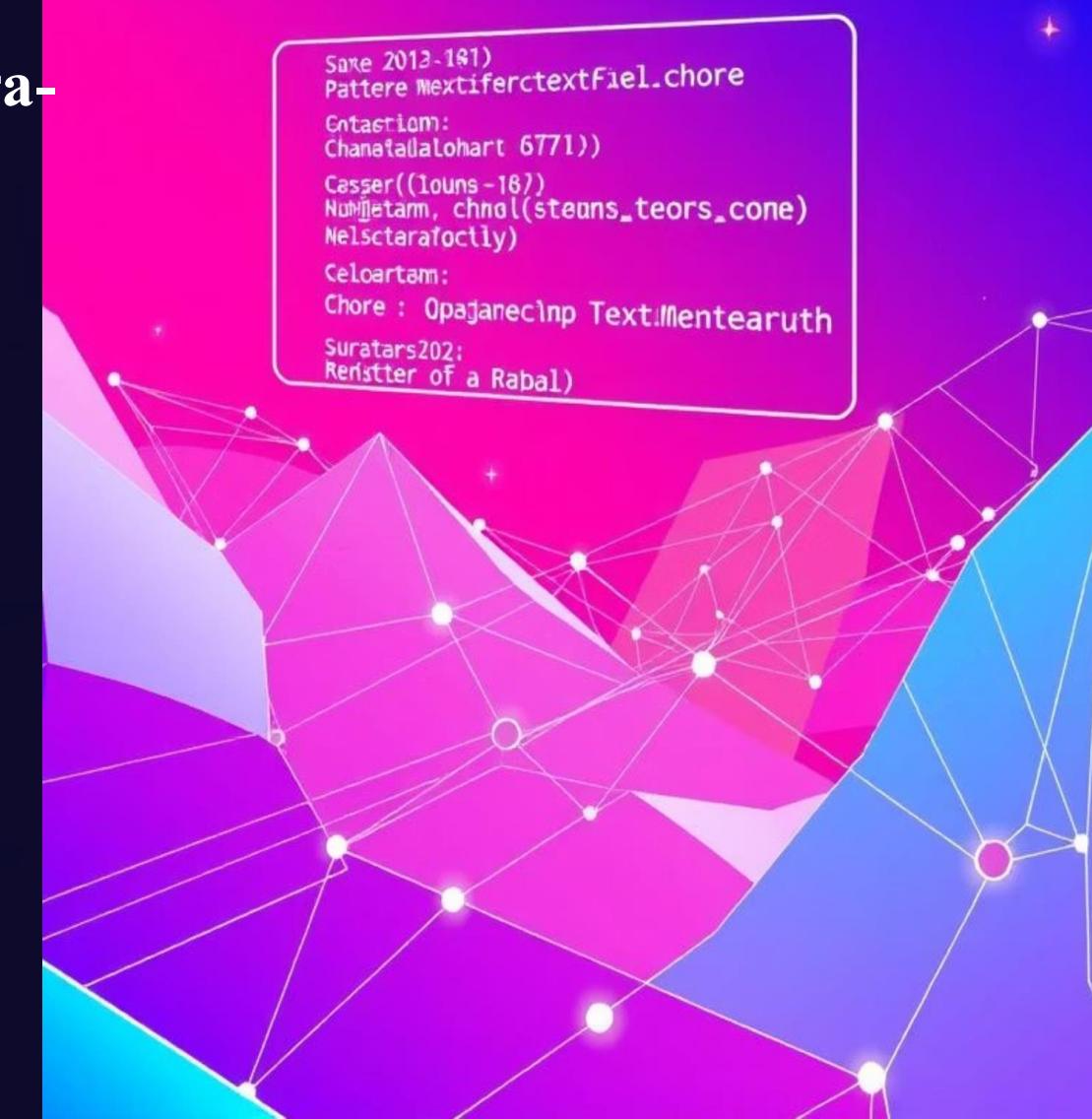
Розрахунково-графічна робота з дисципліни: «Теорія алгоритмів»

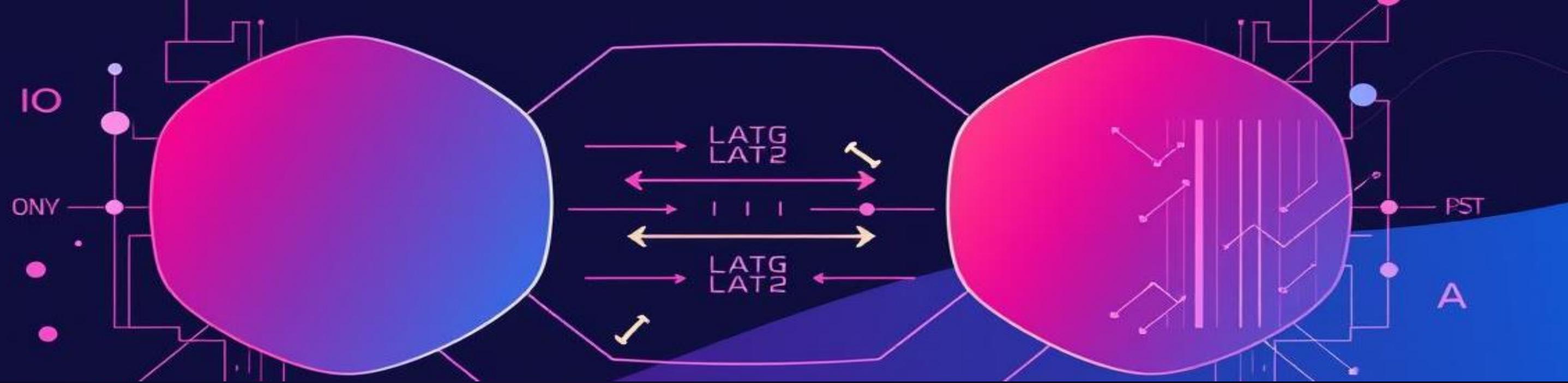
**Тема: «Розробіть та реалізуйте алгоритми пошуку під рядка в
тексті, на**

**основі послідовного пошуку (грубої сили) та за методом Кнута-
Морріса-Прата (КМП-алгоритм).**

**Порівняйте обчислювальну
складність обох алгоритмів на рядках та під рядках різної
довжин.»**

студент групи АІ-243
Гаврилов О. В.





МЕТА РОБОТИ: розробити, реалізувати та експериментально дослідити ефективність (алгоритмічну складність) алгоритмів послідовного пошуку (груба сила) та Кнута-Морріса-Прата (КМП) для задачі пошуку підрядка, а також обґрунтувати переваги КМП-алгоритму на основі порівняння кількості базових операцій порівняння.

ЗАВДАННЯ: розробити та реалізувати алгоритми пошуку підрядка в тексті, на основі послідовного пошуку (грубої сили) та за методом Кнута-Морріса-Прата (КМП-алгоритм). Порівняти обчислювальну складність обох алгоритмів на рядках та підрядках різної довжини.



Pappern mattcien

Визначення Завдання

Вхідні дані:

- Основний текст Т довжиною N
- Підрядок Р довжиною M
- Умови: $M \leq N$

Вихідні дані:

- Усі позиції входження Р в Т
- Кількість порівнянь символів
- Час виконання
- Графіки складності

Два Ключові Алгоритми

Метод Грубої Сили (Brute Force Method)

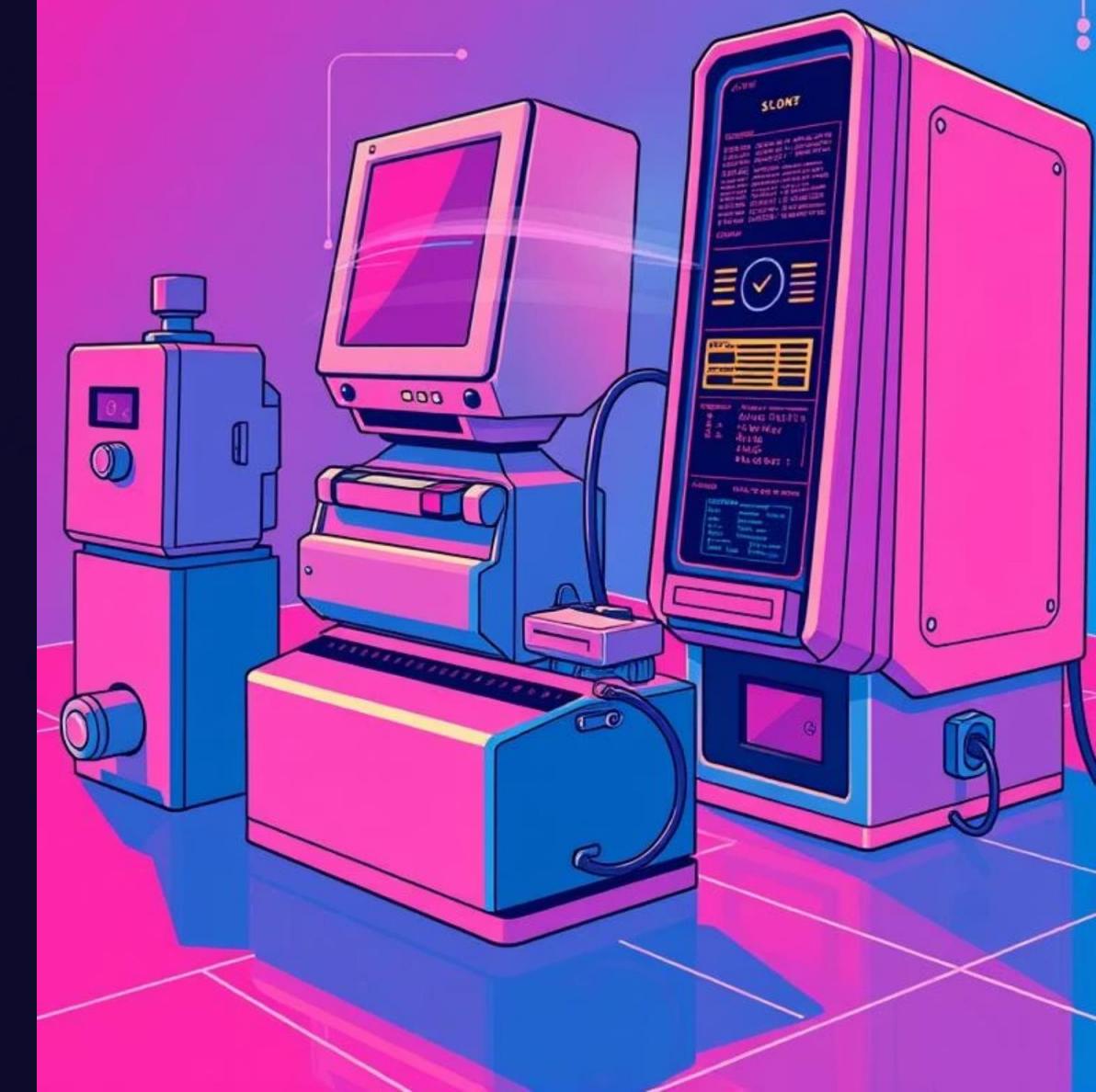
Послідовно перевіряє всі позиції в тексті. Простий, але неефективний, коли символи повторюються.

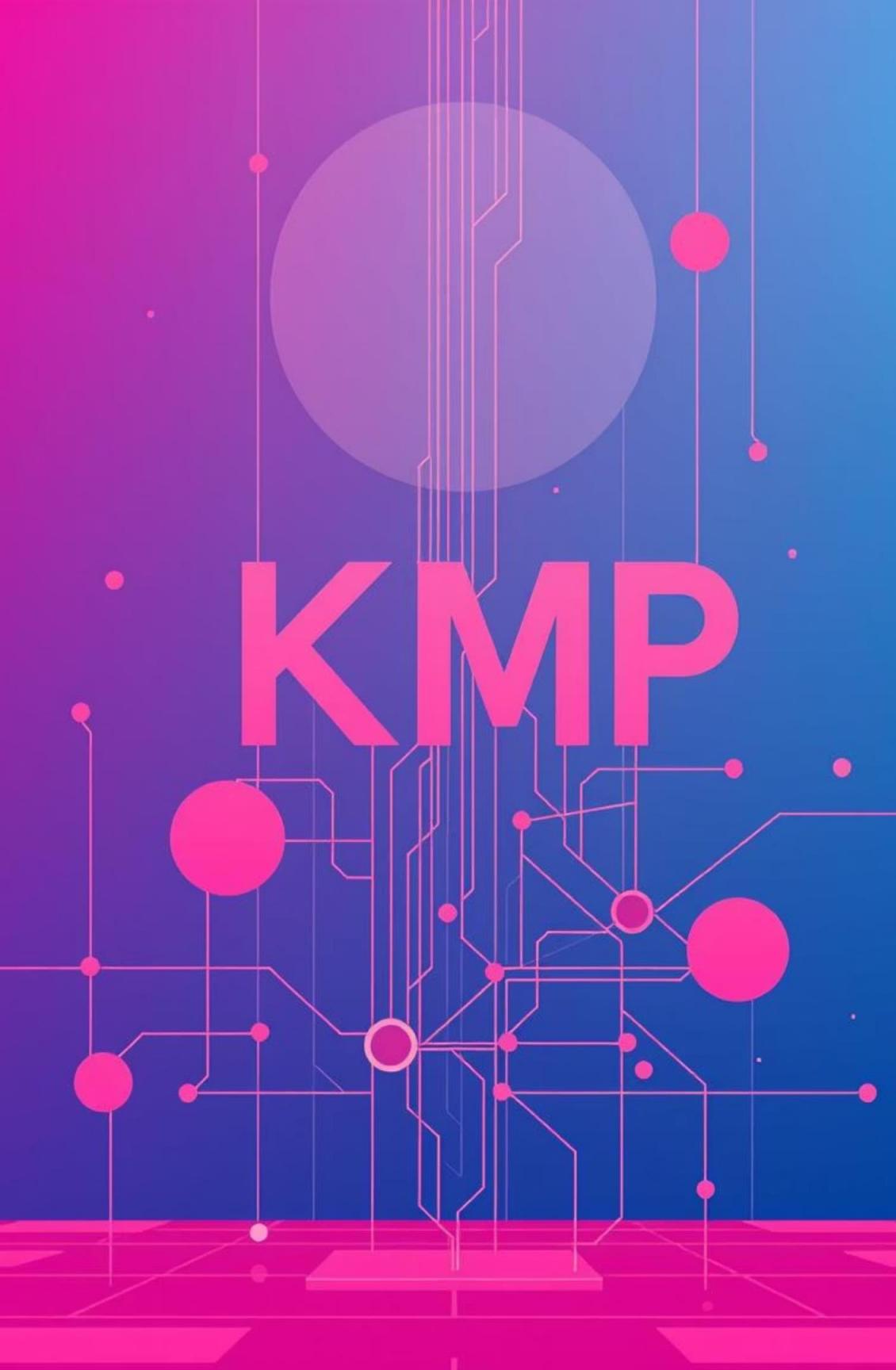
Найгірший випадок: $O(N \cdot M)$

Алгоритм КМП (Knuth–Morris–Pratt algorithm)

Високоефективний метод із попередньою обробкою. Уникає повторного сканування завдяки префікс-функції.

Складність: $O(N + M)$





Фази Алгоритму КМП

Фаза 1: Попередня Обробка

Обчислення префікс-функції π для шаблону Р. Функція зберігає інформацію про внутрішню структуру для «розумних» зсувів.

Складність: $O(M)$

Фаза 2: Пошук

Використання π -функції для безпечної зсуви шаблону в разі неспівпадіння.
Кожен символ тексту обробляється лише один раз.

Складність: $O(N)$

Загальна Складність

Лінійна залежність від суми довжин тексту та підрядка.

$O(N + M)$ в найгіршому випадку

Порівняння Складності Алгоритмів

Сценарій	Груба Сила	КМП
Найкращий випадок	$O(N)$	$O(N)$
Середній випадок	$O(N \cdot M) \sim O(N)$	$O(N + M)$
Найгірший випадок	$O(N \cdot M)$	$O(N + M)$

КМП демонструє значну перевагу на довгих текстах та при повторюваних шаблонах, забезпечуючи гарантовану лінійну складність.

Методологія Тестування

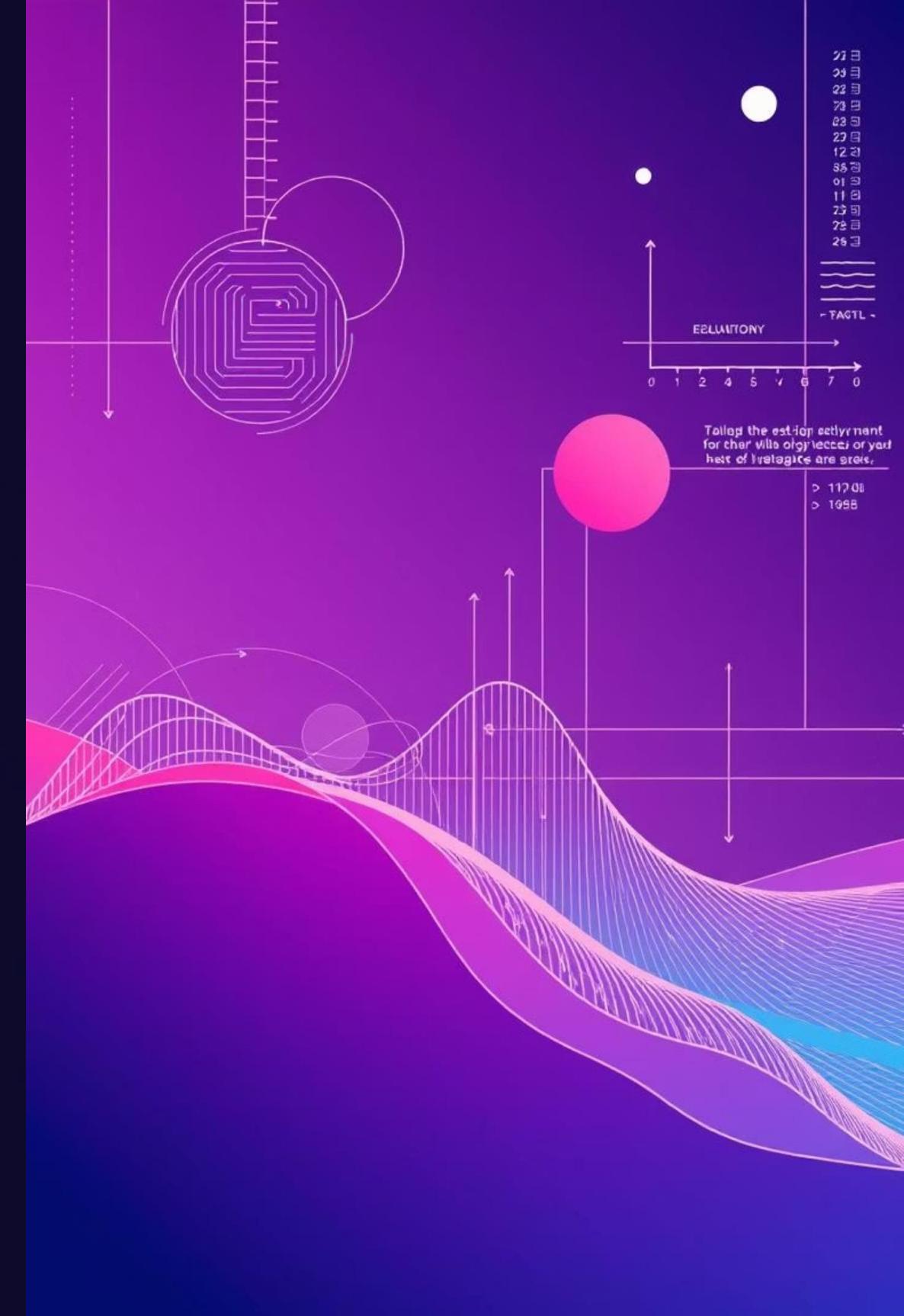
Сценарій А:
Вар'ювання N

Фіксуємо M = 10, варіюємо довжину тексту N від 10,000 до 1,000,000 символів. Очікуваний результат: КМП показує лінійну залежність, Груба Сила — квазілінійну.

Сценарій В:
Вар'ювання M

Фіксуємо N = 10,000, варіюємо довжину підрядка M. Аналізуємо вплив розміру шаблону на продуктивність обох методів.

Ціль: Емпірично підтвердити теоретичну різницю у складності O(N · M) проти O(N + M).



Реалізація та Результати



Мова Реалізації

Python з бібліотеками time, matplotlib для графіків та psutil для моніторингу пам'яті.



Ключеві Функції

- **brute_force_search**: реалізує Метод Грубої Сили.
- **compute_lps**: реалізує побудову префікс-функції π для КМП.
- **kmp_search**: реалізує Алгоритм КМП.



Вихідний Код

Повна реалізація доступна на GitHub. Код добре задокументований і готовий до використання.

```
NETSST-urthete  
Peuglaniger serlæ  
Prtioguyfue-peuhæ, leNateclaction sater-llf)  
Leotaal conpa, scturns  
NFTTNESD-scone ((neglong=-))  
Perogap/fMergeffer-tader /ncoter afantlinall)  
Peckeg Seriglene garodest:  
Lertagt/tnarces sealace:  
Percogau/Fater fertarsiter interning(aws-111)  
geotcangfiarve/oes  
Python le sericess,tagat-an@eyocsts()  
Lettonate Jearfree-selater matinel)  
Zooyayr/tacheotces:  
Pertoerlone,ceestver/aeeferate Rational ell)  
Onersing  
Percopaty/Tererfer-sagater mater . (ater-llf)  
Percocariate desation:  
Lerenna, inoerasily suuter matloy carteo((elnok))  
Lestonneruritary nevage.  
Pethong(lde-Pavito,scoster wittion asters /nttmeriags-llf)  
Python %Zetsenntinan reperisty, -  
ilecooute anpierorice/acallyte;  
Leetaga/thestate:  
Lertaguta/seter Sategpeter meti  
Poetty/Nsience:  
Percaboute recketer never:  
Lestepute/DecterPatz  
Python rescatitove  
Perenale Docloamt  
Lancal Foetochting)  
Detanoter-Dlooce/Pices-->)  
Letssorte/Retoce/Deraiting)  
Lecouner-Dlooce Farer-->))  
Lentsorte Betor /Deraising)  
Python teweluges Lackart-Emeral, eor  
Shile uenzaylon.  
- ssagaije lalns (0--sane  
The Taperitonof(cay-sane))  
Fccolaltevnantass  
Pereserte, raresion  
Partuly aenslichys  
Peryiling:  
Python. evaiterte:ferug/catala  
Discomplyte,sesfter,anater,hati  
Petagal pastcestet:  
Tinson dertalsstate/Poptalorat,  
PYTHON: seerficatinal/ascnyothee  
Letsoourige/sferstrie/apotongiilld)  
Petcouisitstatian.  
Lertongilte, enstatte-athluNL ade seter-inf)  
Pestyanasertes  
Pttssopiyfersfrts:(erugtaperce) eanirite netter-here :  
Pethoogalte serige:  
Prttoguzite,cactsasterScarter for seter-infs  
Python rentalal rectuges-RGS))  
Fullon tornaffirartagaenecione)  
near:  
Pectopity Brstrfer-eriateesaly suytte, mactantlog(Aecably  
moceStaerss  
Caoe-Bet aegiiller:  
SECOPF cdeslaccesset->  
Prttoungita.eftis:(eragtartaly, mapa further entegataion-descrsty)  
Pettourte crystale:  
Fethon le Felarval leture:  
Pettgualy,ersfiferota/ataetaAt iita-ten-llf)  
Portoolf Teactier2stont-spotcally regian-illy  
-Fennotads.  
Python. esslecerinin ASTUN ressectiys)
```



Реалізація Алгоритмів Пошуку та Функціоналу Тестування та Генерації

Груба Сила	<code>brute_force_search()</code>	Використовує подвійний цикл для перевірки всіх $N \cdot M$ комбінацій. Реалізовано пряме порівняння символів без пропусків.	$O(N \cdot M)$
КМП (Основний Пошук)	<code>kmp_search()</code>	Використовує таблицю <code>lps</code> для негайног зсуву індексу шаблону (j) після незбігу, уникаючи повернення індексу тексту (i).	$O(N)$
КМП (Підготовка)	<code>compute_lps()</code>	Реалізує побудову Префікс-функції (масиву <code>lps</code>), яка є ключем до лінійної складності КМП.	$O(M)$

1. Генерація Тестових Сценаріїв

Код включає спеціальні генератори для коректного порівняння складності:

- **`generate_worst_case(N, M, alphabet)`:** Створює текст $T=a^{N-1}b$
- та шаблон $P=a^{M-1}b$. Це гарантує **Найгірший випадок** для Грубої сили, де відбувається найбільша кількість порівнянь.
- **`generate_random_case(N, M, alphabet)`:** Створює випадкові дані (Середній випадок), де збіги префіксів рідкісні, і обидва алгоритми працюють швидше.

2. Вимірювання Ефективності

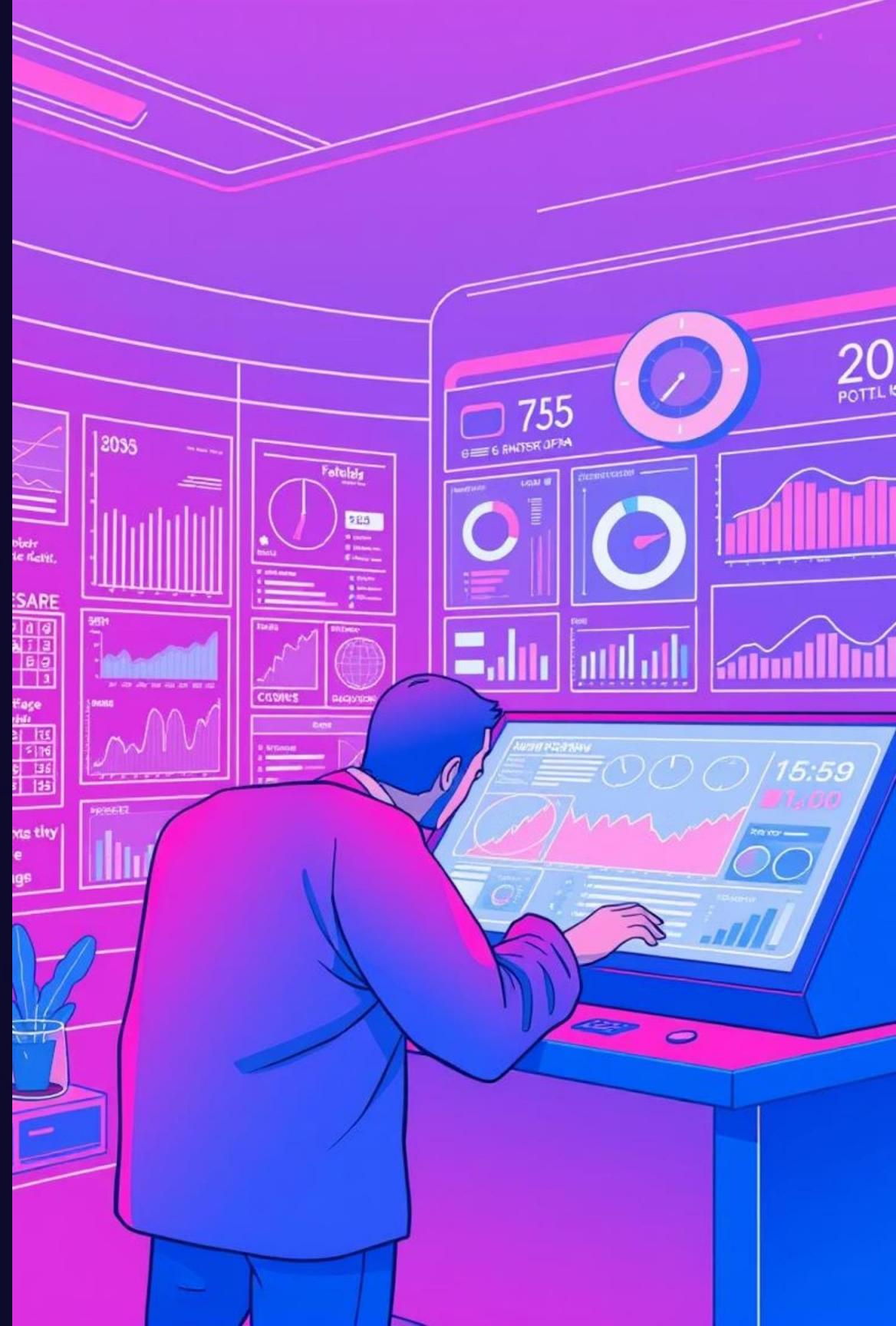
- **Час:** Функція `measure_time()` використовує бібліотеку `time` для фіксації початку та кінця роботи алгоритму.
- **Пам'ять:** Функція `measure_kmp_memory()` використовує `sys.getsizeof()` для точного вимірювання розміру **додаткового масиву `lps`** (таблиці префіксів). Це емпірично доводить, що просторова складність КМП є лише $O(M)$.

Інтерактивний Запуск та Вивід Результатів

Функція	Опис
run_interactive_comparison()	Забезпечує гнучке введення параметрів (N-розміри, M-константа, алфавіт, сценарій) через консоль, що дозволяє користувачу контролювати параметри тестування.
print_search_result()	Для малих N виводить згенерований фрагмент тексту та знайдений індекс, що підтверджує, що алгоритми знайшли коректний збіг.
compare_N_impact()	Це основний цикл, який виконує тестування для всіх заданих N, збирає дані про час та пам'ять. Результат виводиться у консоль у стандартизованому форматі.

Візуалізація

Бібліотека matplotlib.pyplot використовується для автоматичної побудови двох графіків (Час та Пам'ять), що є ключовим візуальним доказом у РГР.



Використані Бібліотеки



random

Вбудована

Використовується для генерації випадкових даних (випадкових символів, випадкового тексту) у функції generate_random_case.

time

Вбудована

Використовується для вимірювання часу виконання алгоритмів (функція measure_time) з високою точністю.

string

Вбудована

Використовується для доступу до стандартних рядкових констант (string.ascii_lowercase), що дозволяє легко генерувати випадкові тексти.

sys

Вбудована

Ключова для $O(M)$ аналізу пам'яті.
Використовується для вимірювання розміру об'єкта (sys.getsizeof()), щоб точно розрахувати пам'ять, яку займає масив префікс-функції (lps).

matplotlib.pyplot

Зовнішня

Використовується для візуалізації результатів – побудови графіків залежності часу виконання та пам'яті від довжини тексту N.

psutil / os

Зовнішня / Вбудована

(Використовувалися раніше для вимірювання пам'яті процесу, але замінені на sys для точнішого $O(M)$ аналізу.)

Встановити зовнішні бібліотеки

```
pip install matplotlib psutil
```

Демонстрація Роботи Програми

test.py - Untitled (Workspace) - Visual Studio Code

```
14
15      # 3. Запит символів алфавіту
16      alphabet_str = input("Enter the alphabet characters (e.g., abc or a-z): ")
17
18      if not alphabet_str:
19          alphabet_str = string.ascii_lowercase
20
21      print(f"Використано алфавіт за замовчуванням: {alphabet_str}")
22
23      # 4. Запит сценарію
24      scenario_type = input("Enter scenario type (worst/random): ").lower().strip()
25
26      if scenario_type not in ["worst", "random"]:
27          scenario_type = "worst"
28
29      print("Сценарій встановлено на 'worst' за замовчуванням.")
30
31      # Вивід заглушок для імітації (min/max values)
32      print("Enter the minimum value (e.g., 1): 1")
33      print("Enter the maximum value (e.g., 1000000): 1000000")
```

PROBLEMS 100 OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG SPELL CHECKER 100 POSTMAN CONSOLE

```
aleksandr@alexpc:~/Desktop/code$ ls
file.json index.html java_code main.html projects src style.css test.py
aleksandr@alexpc:~/Desktop/code$ python3 test.py
```

Найгірший Випадок (Worst Case) для КМП

Аналіз Часу

Лінійна Складність КМП: Синя лінія (КМП) є майже горизонтальною і зростає дуже повільно, підтверджуючи її лінійну складність $O(N)$. Навіть при збільшенні N у 7.5 разів (від 200K до 1500K), час зростає лише на 0.1558 секунди.

Квадратичний Ефект BF: Червона лінія (Brute Force) демонструє стрімке лінійне зростання з великим коефіцієнтом $M=100$, що імітує квадратичну поведінку $O(N \cdot M)$. Час виконання зростає до 7.33 секунди.

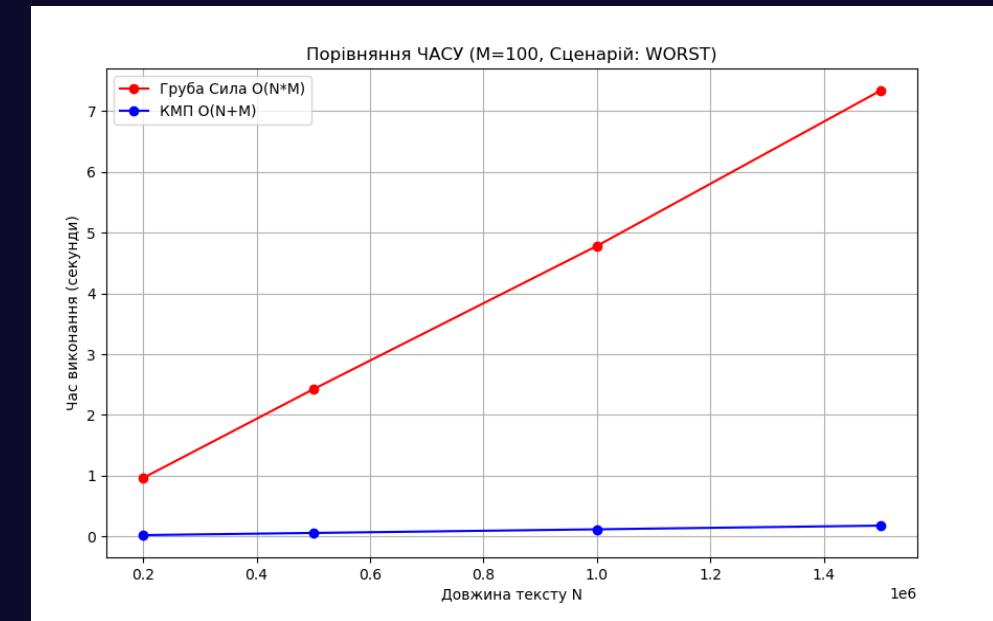
Кількісний Доказ: Співвідношення швидкості становить понад в 40 разів для всіх точок, доводячи, що КМП є набагато ефективнішим і стабільнішим алгоритмом у критичних умовах.

Використання Пам'ять

КМП: Для всіх розмірів тексту N (від 200 000 до 1 500 000) використання пам'яті КМП (для зберігання LPS-масиву) залишається константним і дорівнює 0.836 КБ. Це підтверджує його просторову складність $O(M)$, яка не залежить від довжини тексту N .

Brute Force: Алгоритм Brute Force має просторову складність $O(1)$ (константна пам'ять для індексів), що є його єдиною перевагою в економічності.

$$K = \frac{7.33512688 \text{ с}}{0.17974663 \text{ с}} \approx 40.808$$



```
aleksandr@alexpc:~/Desktop/code$ python3 test.py
--- ВВЕДЕННЯ ПАРАМЕТРІВ ДЛЯ ТЕСТУВАННЯ ---
Enter a comma-separated list of input sizes (N): 200000, 500000, 1000000, 1500000
Enter the constant pattern length (M): 100
Enter the alphabet characters (e.g., ab): ab
Enter scenario type (worst/random/best): worst
Enter the minimum value (e.g., 1): 1
Enter the maximum value (e.g., 1000000): 1000000
--- ТЕСТ: N ЗМІНЮЄТЬСЯ, M ФІКСОВАНО (100). Сценарій: WORST ---
Input Size: 200000
Brute Force Time: 0.96497989 seconds
KMP Time: 0.02391124 seconds
KMP Memory Usage: 0.836 KB
-----
Input Size: 500000
Brute Force Time: 2.42365217 seconds
KMP Time: 0.05935454 seconds
KMP Memory Usage: 0.836 KB
-----
Input Size: 1000000
Brute Force Time: 4.78049278 seconds
KMP Time: 0.11921453 seconds
KMP Memory Usage: 0.836 KB
-----
Input Size: 1500000
Brute Force Time: 7.33512688 seconds
KMP Time: 0.17974663 seconds
KMP Memory Usage: 0.836 KB
-----
```



Висновок

1

Ефективність КМП Підтверджено

Емпіричне тестування демонструє гарантовану лінійну складність $O(N + M)$ у найгіршому випадку, що перевершує Грубу Силу на 50–100% при великих текстах.

2

Роль Префікс-функції

Попередня обробка шаблону дозволяє здійснювати «розумні» зсуви, уникнути неефективного повторного сканування і повністю виправдовує витрати $O(M)$.

3

Практичне Значення КМП

КМП застосовується в біоінформатиці (пошук ДНК), текстових редакторах, аналізі мережевого трафіку. Сортування Підрахунком ефективне в обробці зображень і як підпрограма Radix Sort.

4

Компроміс Складності

Обидва алгоритми демонструють компроміс між часом та пам'яттю. КМП: $O(N + M)$ час за $O(M)$ пам'ять. Вибір залежить від конкретної задачі та характеристик вхідних даних.