Методи оптимізації використання пам’яті для узагальнених суфіксних масивів

Підготувала студентка КН-4
Барабуха Марія
Науковий керівник: Глибовець А.М.

Мета роботи

1. Розробка алгоритму, що дозволить виконувати швидкий пошук по даним у вебсистемі.
2. Досягнення максимальної межі розміру даних, по яким виконуватиметься пошук.
3. Часова оптимізація та зменшення затрат пам’яті на побудову та зберігання структури.

Хід виконання

01 **Дослідження**

наявних рішень

02 **Вибір**

оптимального підходу

03 **Адаптація**

під вебсистему та узагальнені структури

04 **Застосування**

методів компресії

05 **Встановлення**

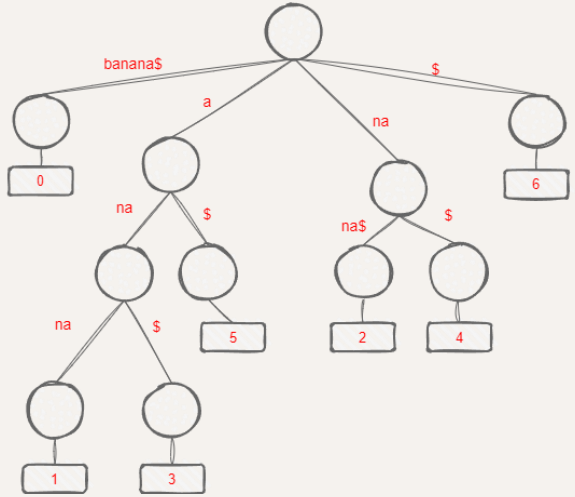
допустимих меж розміру даних

Класичне рішення

Стандартним рішенням для швидкого пошуку є використання *суфіксних дерев*.

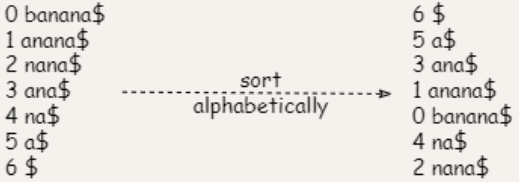
Попри їх перевагу в швидкодії, вони мають недолік – велика *затрата* як *пам'яті*, так і часу на побудову, що ускладнює їхнє застосування у вебсистемі.

Стандартною альтернативою для суфіксних дерев, що використовує менше ресурсів, є *суфіксні масиви*.



Суфіксні масиви

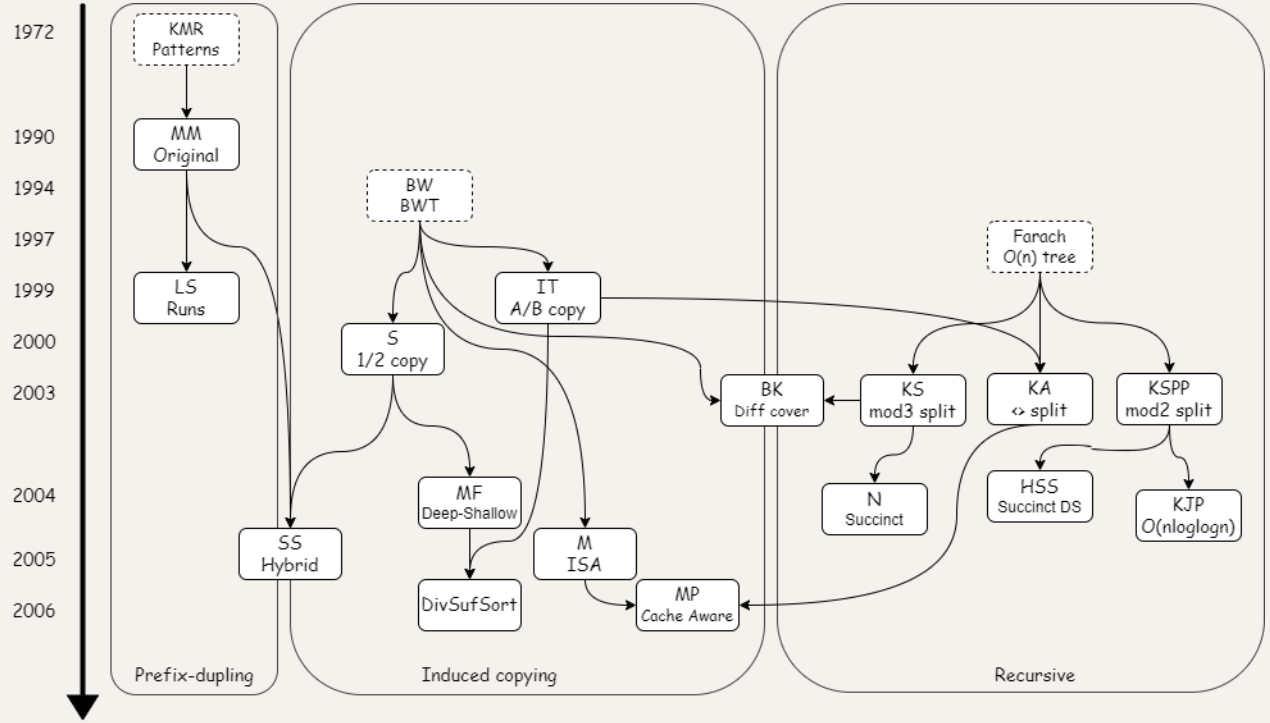
Суфіксний масив (suffix array) – це масив, що складається зі стартових позицій всіх суфіксів певного рядка, впорядкованих в алфавітному порядку



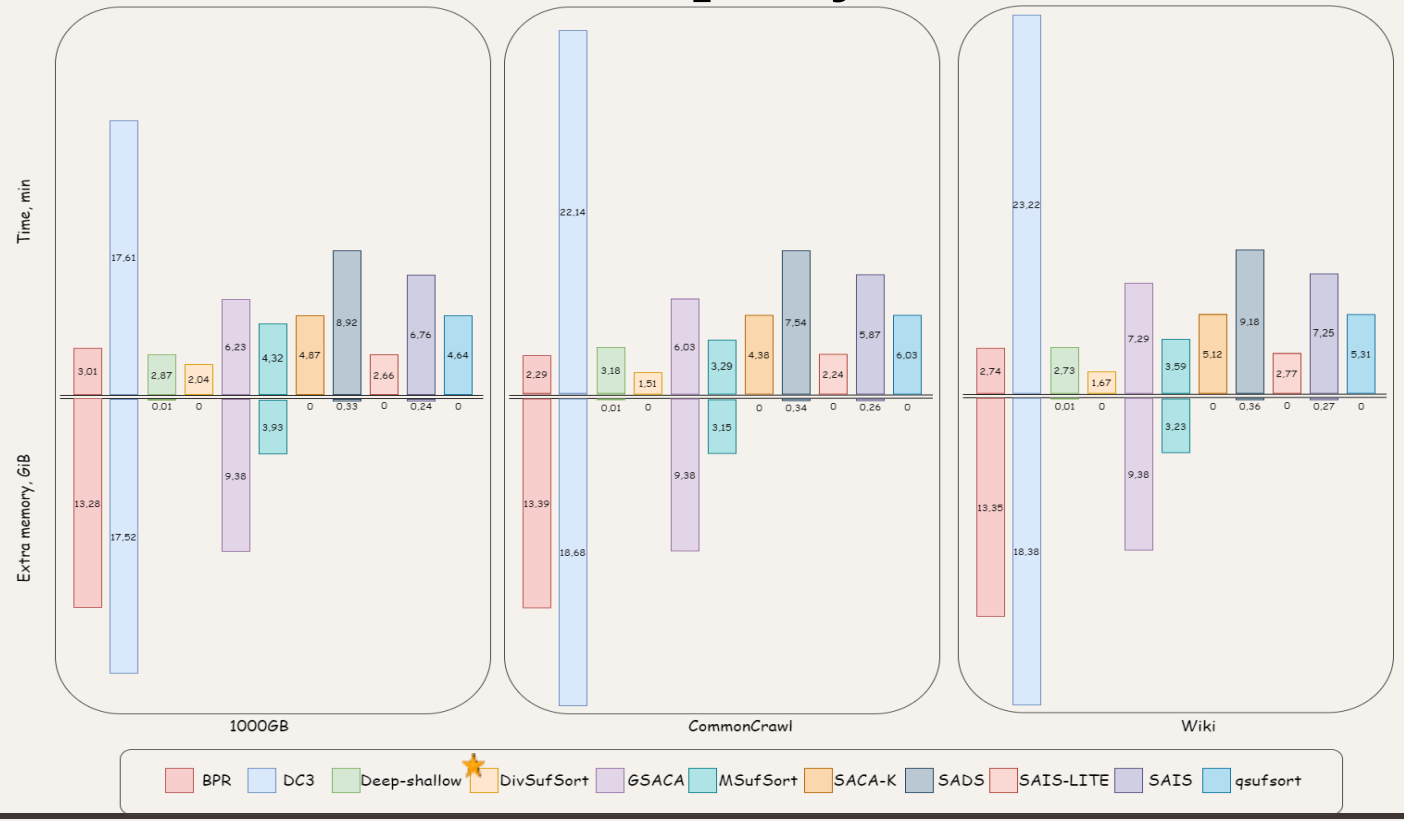
Для суфіксних масивів існує *узагальнений* варіант реалізації, що призначений для зберігання та обробки колекції рядків.

Основна проблема використання суфіксних масивів для пошуку полягає в побудові самої структури, а зокрема – в сортуванні суфіксів.

Наявні алгоритми



Показники продуктивності



DivSufSort

01

O(nlogn)

Часова складність для побудови

02

5n + O(1)

Просторова складність

03

O(mlogn)

Часова складність для пошуку

04

C

Мова оригінального алгоритму

05

Не підтримує

роботу з узагальненою структурою

06

Найшвидший

наявний алгоритм

Модифікація **DivSufSort**

01

C → TypeScript

Перенесення коду алгоритму на TypeScript

02

Узагальнена структура

Алгоритм було модифіковано для роботи з колекцією рядків

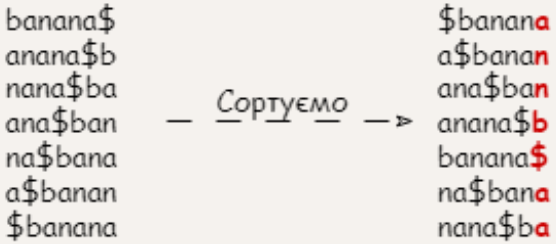
03

Компресія

До отриманої структури були застосовані алгоритми компресії.

Компресія

Для оптимізації використання ресурсів було застосовано *LF-відображення*, що встановлює зв'язок між впорядкованими суфіксами та рядком, утвореним внаслідок *перетворення Берроуза-Вілера*.



Перетворення Берроуза-Вілера: створюємо n ротацій рядка S, де n – довжина рядка S, впорядковуємо їх в лексикографічному порядку та складаємо останні літери кожної ротації в новий рядок.

Компресія

01 F-стовчик

Представлений у вигляді кількості входжень кожного символа алфавіту у тексти.

02 L-стовпчик

Рядок, до якого було застосовано *BWT*.

03 Збереження частин масивів

Зберігаємо частини масивів рангів та суфіксів, де решту елементів можна відновити за O(1) часу.

F	L	ranks			SA
		a	b	n	
1	a	1	0	0	6
3	n	-	-	-	-
1	n	-	-	-	3
2	b	1	1	2	-
	$	-	-	-	0
	a	-	-	-	-
	a	3	1	2	-

Результати

	Назва	M_{orig}, Мб	| Σ |	T, с	M, Мб	Опис
1	sudoku	164	10	75	986	Один мільйон партій в судоку
2	foodprices	87	42	52	407	Ціни на продукти харчування у різних країнах
3	sales	50	39	37	453	Інформація про продажі, близько 740 тис. записів

Висновки

01

Дослідження

було досліджено різні підходи до реалізації швидкого пошуку, а зокрема використання суфіксних дерев та їх аналогу – суфіксних масивів.

02

Модифікація

Перенесення на TypeScript, робота з колекцією рядків, компресія. Хоч алгоритм і поступається в швидкості, проте в результаті було отримано відносно компактну структуру.

03

Розмір даних

Визначення максимальної допустимої межі розміру даних