Uppgift 4 - Strängdatastruktur

Avancerade metoder för text- och bildbehandling DA357A

Therése Larsson

Tids- och utrymmeskomplexitet - Trie

Att söka efter en nyckel i en trie går på O(n) där n = längden på nyckeln som söks. För att söka jämför man tecknena i nyckeln respektive tecknena i trien, och man rör sig nedåt i trien i sökningen.

Utrymmeskomplexiteten för en trie som nyttjar e hashmap är O(Σn), där n = längden på en nyckel.
För att bygga en trie skapar man en nod för varje nytt tecken. Min implementation använder sig av en hashtabell för att hålla reda på alla noder. Utrymmeskomplexiteten för en hashmap är O(n), där n = antalet lagrade nycklar.

Tids- och utrymmeskomplexitet - Suffixarray

Tidskomplexiteten för en suffixarray är långsammare för t.ex. en suffixtrie, men istället har en suffixarray bättre utrymmeskomplexitet än en suffixtrie.

Tidskomplexiteten för att utföra en binärsökning i en suffixarray är O(log n), eftersom vi behöver göra log n stycken jämförelser (när vi jämför strängar) och där en jämförelse i värsta fall tar O(m), där m = längden av söksträngen (P pattern).
Sorteringen av en suffixarray tar minst O(n log n) stycken jämförelser. Sorteringen är nödvändig, annars går det inte att göra en binär sökning.
Som helhet får suffixvektorn tidskomplexiteten $O(n^2 \log n)$.

Utrymmeskomplexiteten för en suffixarray är lineär O(n), då arrayen kommer ha samma storlek (/längd) som det finns antal tecken i strängen man bygger suffixarrayen utifrån.
Om vi t.ex. ska bygga en suffixarray utifrån ordet “choklad” kommer varje delsträng av ordet att ta upp en plats i arrayen: [choklad, hoklad, oklad, klad, lad, ad, d]. Ordet består av 7 tecken, och arrayens storlek kommer också vara 7.