Algoritmen en datastructuren 3
Verslag

Janne Cools

UNIVERSITEIT
GENT

FACULTEIT
WETENSCHAPPEN

Array trie

Voor de eerst trie heb ik de patricia trie geïmplementeerd. Hierbij hou ik voor elke top enkele variabelen bij. De kinderen van de top worden bijgehouden in een lijst die in de heap wordt gealloceerd en het aantal kinderen wordt in een aparte variabele opgeslagen. Als de top een blad is, wordt de string in dat blad bijgehouden. Zo niet, dan bevat de variabele gewoon een null-waarde. Daarnaast wordt het karakter bijgehouden waarmee de top vertakt is vanuit zijn oudertop en wordt een skip bijgehouden. Deze variabele skip is een string die de karakters voorstelt die overgeslagen worden in de boom. Hiermee worden de paden van de boom gecomprimeerd.

Bij deze trie worden 3 variabelen gealloceerd in de heap, namelijk de kinderen van een top, de eventuele string en de skip. Deze worden allemaal terug vrijgegeven wanneer de top verwijderd wordt of wanneer de methode arraytrie_free wordt opgeroepen.

Ik heb zelf een extra methode toegevoegd om een nieuwe ArrayTrie aan te maken. Hierbij worden het karakter, de string, de lengte van de skip en de string van de skip meegegeven. De parameter string kan ook een null-waarde zijn. Deze methode wordt gebruikt in de methode om een string toe te voegen aan de boom (arraytrie_add).

De methode om een string te verwijderen uit de boom werd nog niet voldoende getest met de basistesten. Daarom heb ik zelf vooral testen toegevoegd voor deze methode en minder voor de methode om een string toe te voegen. Ik heb hierbij naar de verschillende mogelijkheden gekeken voor de ouder van het blad dat verwijderd werd. De ouder kan immers 2 of meer kinderen hebben, waarbij beide gevallen anders behandeld moeten worden. Het geval van 2 kinderen heb ik nog gesplitst in de situatie waarbij beide kinderen bladeren zijn, of het ene kind een blad is en het andere een interne top. Ten slotte heb ik gecontroleerd dat de methode correct wordt uitgevoerd als de ouder de wortel is.

Ternary trie

Bij de ternary trie hou ik zoals bij de patricia trie variabelen bij voor het karakter en voor de string als het een blad is. Verder bevat elke top pointers naar zijn kinderen. Elke top heeft ten hoogste 3 kinderen naar toppen met een kleiner, groter of hetzelfde karakter als de huidige top. Als laatste wordt de ouder van de top nog bijgehouden. Aangezien de paden niet gecomprimeerd zijn, kan het dat veel toppen verwijderd moeten worden wanneer een string verwijderd wordt. Deze toppen worden dan op een bottom-up manier verwijderd, waardoor je de ouders van de toppen nodig hebt en je deze dus moet bijhouden in een variabele.

Custom trie

Aangezien er op voorhand geen alfabet vastgelegd wordt, worden de tries aangevuld naarmate er meer strings worden toegevoegd. Dit zorgt ervoor dat een vertakking volgen bij de patricia trie minder efficiënt is. Je moet immers de karakters van alle kinderen vergelijken met het karakter van de nieuwe string om te weten welk kind je moet volgen. Op dat vlak is de ternary trie efficiënter, aangezien je in elke top slechts één vergelijking maakt en dan doorschuift naar een top met een groter, gelijk of kleiner karakter.

Aan de andere kant is het nadeel bij de ternary trie dat de boom niet gecomprimeerd is en er dus een lang pad met weinig vertakkingen kan ontstaan naar een blad. Dit kan de boom onnodig groot maken.

Daarom heb ik voor de custom trie een combinatie van de beide tries gebruikt. Ik heb namelijk de ternary trie geïmplementeerd waarbij de boom wel gecomprimeerde paden bevat. Elke top heeft dus een extra parameter, namelijk de string die geskipt wordt (zoals bij de patricia trie).