Algoritmen en datastructuren 2: Taak semi-splay

Asma Oualmakran

27 oktober 2019

1 Oplossing theoretische vragen

1.1 Vraag 1

Stel k is een pad in een splayboom dat bestaat uit toppen t. De vorm van het pad wordt bepaald door de boog E die tussen 2 toppen wordt gekozen. In elke top t_1 dat geen blad is kan men steeds kiezen tussen de bogen E(t_1, t_2) en E(t_1, t_3).

Voor de uitdrukking construeren we een functie n waarvoor k een parameter is. Hieruit volgt dat het aantal vormen v = n(k)
In elke top kunnen er zoals boven vermeld 2 keuzes gemaakt worden, maar naar mate dat het pad naar beneden wordt gevolgd resteren er in elke stap k-1 toppen in het pad. Dit geeft ons dan de volgende uitdrukking n(k) = 2 * n(k-1)
Dit is een machtreeks en is gelijk aan n(k) = $2^{(k-1)}$, dit kan men afleiden uit de volgende reeksontwikkeling:
k = 1 $\rightarrow$ $1 = 2^0$
$k = 2 \rightarrow 2 = 2^1$
$k = 3 \rightarrow 4 = 2^2$
$k = 4 \rightarrow 8 = 2^3$
$k = 5 \rightarrow 16 = 2^4$

De uitrukking voor het aantal mogelijke vormen van een pad met k toppen tijdens de splay stap is n(k) = 2 * n(k-1).

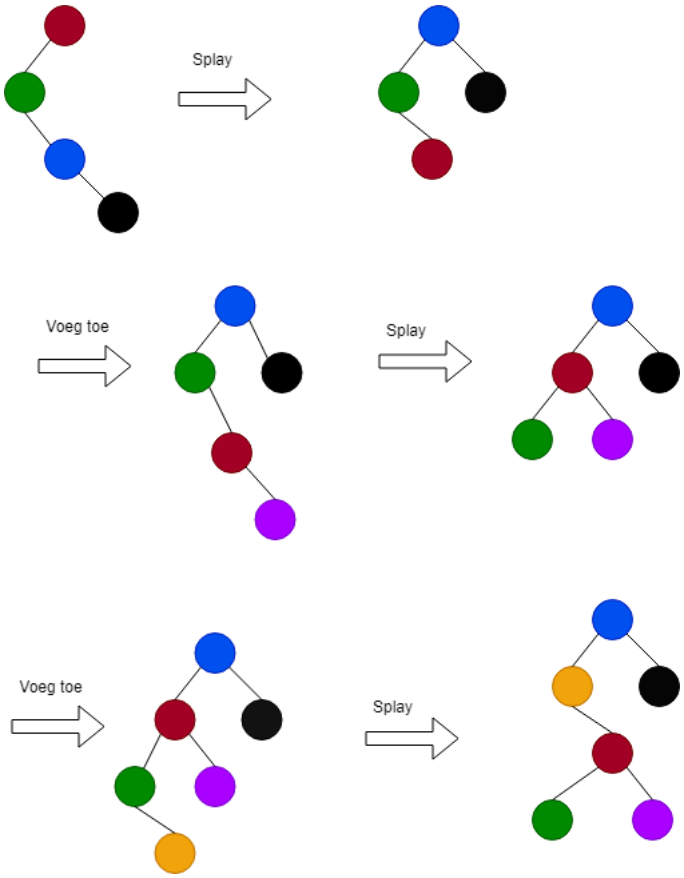
1.2 Vraag 2

Boom met splay grootte 3:
Er bestaat geen splay boom isomorf met deze boom. Dit komt door de eigenschap beschreven in de cursus, wanneer er 3 toppen worden gesplayed dan zal dit steeds een binaire boom zijn met hoogtte 2. Dit zorgt er voor dat er aan de wortel steeds een rechter kind zal hebben bij elke toevoeg en opzoek bewerking. Door de manier van splayen zal er ook nooit enkel een rechter- of linkerkind worden toegewezen worden aan een top. Hierdoor zal er nooit een reeks ontstaan met toppen die enkel een rechter kind hebben zoals in de beschrijving.

Boom met splay grootte 4:

Er bestaat een splay boom isomorf met deze boom. Doordat er altijd een even aantal toppen zijn die worden gesplayed, zijn de vervang bomen nooit gebalanceerd. Hierdoor veranderen niet altijd alle posities van alle toppen. In onderstaande afbeelding worden de stappen getoond om de boom te bekomen. In de eerste boom, wordt er eerst toegevoegd tot er een diepte wordt bereik van 4 zodat er kan worden gesplayed. Vervolgens worden er enkel toevoegbewerkingen uitgevoerd om het nodige resultaat te bekomen.

Boom met splay groote 7:
Er bestaat een splay boom isomorf met deze boom. Aangezien dat de splay diepte 7 is kan men deze boom verkrijgen door steeds de toppen toe te voegen in een volgorde dat ervoor zorgt dat de diepte nooit 7 wordt. In dit geval betekend het de boom zo gebalanceerd mogelijk houden en voorkomen dat er wordt gesplayed.
De wortel is dan de middelste waarde uit de reeks, deze moet als eerste toegevoegd worden vervolgens kunnen de opeen na grootste en kleinste ten op zichte van de wortel worden toegevoegd. Dit kan dan verder recursief worden uitgevoerd door de bladeren van de tussen resultaten te zien als de wortel van een nieuw te construeren deelboom.



2 Implementatie

Er is ervoor gekozen om een Klasse en interface aan te maken om te toppen van de splay boom voor te stellen. Hierin zijn ook de methoden toegevoegd om de waarden en de referenties van de toppen op te vragen en aan te passen. Daarom is er ook voor gekozen om de methode iterator van de Iterator interface niet te implementeren en elke methode in splay boom toe te laten een eigen iteratie techniek te laten gebruiken. Momenteel wordt er bij elke methode gebruik gemaakt van de breath-first methode, aangezien deze de backtracking stap overslaat van de depth-first methode en daardoor een fractie sneller zal zijn.

Momenteel wordt er bij elke operatie van op de splay boom het pad geconstrueerd die nodig was om de nodige top te bereiken. Dit ter voorbereiding van de splay stap in de volgende implementatie. Dit wordt gerealiseerd door de methode find(). Deze volgt een pad dat wordt aangestuurd van de gegeven parameter. Dit zal aan de hand van vergelijkingen de locatie bepalen waar deze waarde zich in de boom moet bevinden. Dit garandeerd niet dat de waarde wordt gevonden, maar de stack die als pad wordt geconstrueerd zal als top el-

ement de gezochte top vinden of de ouder van de gezochte top als de boom de top niet bevat. Deze informatie kan dan gebruikt worden in de andere operaties door na de pad constructie de boom eventueel te manipuleren.

Er zijn momenteel geen experimenten gebeurd. Er werd geopteerd dat voor deze tussentijdse deadline de unit tests voldoende zijn maar experimenten nog niet veel toegevoegde waarden zouden hebben. Zoals eerder vermeld wordt er gebruik gemaakt van de breath-first search vanwegen het wegvallen van de backtracking. Er is ook gekozen om gebruik te maken van een ArrayList als datastructuur vermits dit toe laat om deze te laten groeien. Het laten groeien gebeurd in het geval wanneer er een top moet worden toegevoegd op een index die nog niet is bereik door de ArrayList. De oplossing hiervoor is om de tussenliggende locaties op te vullen met de waarde "null" dit gebeurd in de methode addToTree(). Dit is een O(n) operatie, maar een kleinere n waarde dan wanneer men de elmenten zou moeten kopieëren naar een grotere Array. Ook groeit hierdoor het geheugen niet exponentieel. Maar de O(1) access snelheid van de elementen blijft wel behouden.