Tema 5: Prioritetsköer

Wilhelm Durelius widu7139

8 februari 2026

Leftist Heap kontra Binary Heap

En viktig del i prioritetsköer är funktionaliteten att kunna unionisera dem, att slå ihop två heapar till en. När man gör en merge i en vanlig DHeap eller Binary Heap tar man helt enkelt värdena på trädet man vill lägga till och gör insert på varje värde. Detta tar tid, och är onödigt i och med att trädet man vill slå ihop med redan uppfyller heap-ordning på sina värden.

För att lösa detta har man designat flera specialiserade varianter av den vanliga array-baserade binära heapen, en av dem är 'Leftist Heap'. En Leftist Heap använder en länkad datastruktur istället för en array-baserad, då man kan slå ihop två träd genom att styra om $log(n)$ pekare, jämfört med att slå ihop två arrays är detta mycket effektivare, $O(log\ n)$ för Leftist Heap kontra $O(n)$ för Binary Heap.

I binära heapar sker insert av värden i konstant tid, insättning av n värden tar då $O(n)$ tid. Teoretiskt skulle det kunna

ta längre tid, men eftersom värden man stoppar in brukar hamna långt ner i trädet vilket minskar tidskomplexiteten. Hade man stoppat in ett värde mindre än nuvarande minimum på en binär heap varje gång vid insert hade det blivit $O(log\ n)$ istället för konstant tid. En Leftist Heap däremot, använder sin Merge-metod för insättningar, men ser det då som ett subträd med 1 nod. Alla merge-operationer på en Leftist Heap är garanterad $O(log\ n)$ tidskomplexitet. Jämfört med en binär heap är alltså en enskild insert långsammare på en Leftist Heap, men insert av n värden samtidigt (merge) är snabbare.

Anledningen att en Leftist Heap tar logaritmisk tid är inte enbart på grund av pekar-strukturen, utan det som gör en Leftist Heap till en Leftist Heap är Left Heavy NPL-strukturen. NPL (Null Path Length) är längden från en nod till den närmaste noden som har mindre än 2 barn. Left Heavy betyder att NPL är större i vänstra subträdet än högra. Leftist Heap byter ut noder horisontellt med varandra vid merge och ser strukturellt till att det vänstra subträdet har en högre NPL än det högra. Denna 'swap' innebär då tillsammans med pekar-strukturen att ett nodbyte kan flytta ett helt subträd.

RemoveMin i en LeftistHeap använder också merge metoden, man tar bort roten och kör merge metoden på de två subträden som blir kvar under.

Sammanfattningsvis, behöver du slå ihop två träd ofta är en Leftist Heap att föredra, men vid behov av enbart enskilda insättningar är en binär heap ett bättre alternativ generellt.