Tema 3: Träd

Wilhelm Durelius `widu7139`

6 februari 2026

1 Grundläggande begrepp

Ett träd börjar alltid med en rot, det är den enda noden utan en förälder. Alla andra noder har exakt en förälder, alltså en nod direkt ovanför dem.

Nedåt i trädet kan varje nod ha 0 till N antal barn. Barn med samma förälder kallas för syskon. En nod kan vara både förälder, barn och syskon samtidigt.

Djupet på en nod är antalet bågar från noden till roten. Höjden på en nod däremot är det högsta antalet bågar till ett löv.

Ett löv är en nod utan egna barn. Ett subträd är en nod tillsammans med alla dess ättlingar. Både en enskild rot och ett enskilt löv kan vara ett subträd. Poängen är att ett subträd kan starta ifrån vilken typ av nod som helst i ett träd.

2 Typer av träd

Ett träd är en länkad struktur, på samma sätt som en länkad lista, men där varje nod kan ha flera vägar att gå, istället för en enda som i länkade listor. Ett bra exempel på ett träd är en 'path'-struktur i ett filsystem.

Binära träd fungerar som ovan, men har begränsningen att varje förälder får ha max 2 barn. Detta hade fungerat mindre bra för ett filsystem, då fler än 2 filer per mapp är önskvärt.

Binära sökträd har samma begränsning som vanliga binära träd, men med tillägget att innehållet ska vara sorterat. Letar du efter värdet 5 och rotnoden har värdet 10, då vet du att du ska traversera i vänsterledet, då värdet du letar efter är mindre än rotnoden. Applicera sedan samma logik på varje barn du traverserar över. Binära sökträd är i värsta fallet $O(n)$.

AVL-träd löser ett problem som binära sökträd har, att de inte ställer några krav på höjden. AVL-träd kräver att höjdskillnaden mellan vänster och höger subträden är max 1.

Splay-träd löser samma problem som AVL, men med en annan metod. Istället för att balansera jämnt ut, så flyttar Splay-träd upp nyligen åtkomna noder till roten. Generellt när en specifik nod begärts är sannolikheten hög att den kommer att begäras igen, eller en nod i nära anslutning till den. Splay-träd garanterar att m antal operationer tar som mest $O(m\ log\ n)$ lång tid. Över tid blir det alltså väldigt nära $O(log\ n)$, men varje enskild operation är inte garanterad den tidskomplexiteten.

3 Traversering

Traversering innebär i denna kontext att alla noder i trädet besöks.

Inorder traversering för binära träd utgår ifrån det vänstra subträdet, sedan rotnoden och slutligen högra subträdet. I ett binärt sökträd besöker inorder noderna i stigande ordning.

```
1 || Inorder: Left subtree -> Root node -> Right subtree
```

Preorder traversering utgår ifrån roten, sedan vänstra subträdet och slutligen högra subträdet. Preorder traversering används ofta vid kopiering av träd när den ursprungliga strukturen ska behållas.

```
1 || Preorder: Root node -> Left subtree -> Right subtree
```

Postorder traversering utgår ifrån vänstra subträdet, sedan högra subträdet och slutligen rotnoden. Används ofta vid borttagning av noder. Med postorder besöks noders barn innan själva noden, vilket gör att traversering kan fortsätta även efter borttag.

```
1 || Postorder: Left subtree -> Right subtree -> Root node
```

Preorder och postorder fungerar för generella träd, men inte inorder. Detta då inorder är definierat så som att den ska besöka ett syskon per nod, finns det fler syskon är det inte definierat vilken av dem som ska besökas.