DV2: Algoritmer och problemlösning

OU2

Theodor Jonsson

2022-01-31

ens18trn thjo0102

Innehållsförteckning

1. Inledning

Vad är de bästa sättet att representera ett kalkylblad? I denna rapport har tre stycken kriterier tagits fram för att svara på denna fråga. Samt har tre förslag på representationer tagits fram. Dessa kriterier används sedan för att utvärdera dessa representationer. Till sist ges ett omdöme om vilken representation rekommenderas att användas baserat på dessa kriterier.

2. Kriterier

Dessa kriterier är framtagna genom ett seminarium med en grupp av fem datavetenskap studenter. Detta seminarium hölls för att diskutera gemensamt vilka kriterier som bör användas när man ska utvärdera representationer till detta problem.

2.1 Tidseffektiv insättning.

En bra representation har en låg tid för insättning, på grund av att denna operation kommer användas flitigt av användaren

Tidseffektivitet har mätts genom stora ordo detta är ett mått för att se operationens tidseffektivitet genom att bara kolla på dess tillväxthastighet. Mer information om Stora ordo kan hittas här[1] . I denna kriterium kommer vi kolla efter medeltidskomplexiteten samt värstafallstidskomplexiteten detta används för till exempel har sökning inom ett icke komplett binärt sökträd en värstafallstidskomplexitet av O(n) och medeltidskomplexiteten är O(log n).

2.2 Tidseffektiv sökning av värden.

Att söka värden blir en stor del av programmet vilket menar att denna operationer bör vara tidseffektivitet. Stora ordo används här igen för att mäta tidseffektiviteten. Samt kommer också medeltids och värstafallstidskomplexiteten mätas.

2.3 Tidseffektiv Borttagning av värden.

Borttagning av värden i ett kalkylblad är en del som är mycket använt vilket gör att denna operationen bör vara tidseffektiv. I detta kriterium kommer också kolla på både medeltids och värstafallstidskomplexiteten med stora ordo. Till exempel i en Stack kan de element som vi vill ta bort vara längst nere i stacken vilket gör att vi måste gå igenom alla element för att nå detta element vilket gör att borttagningstiden blir O(n)

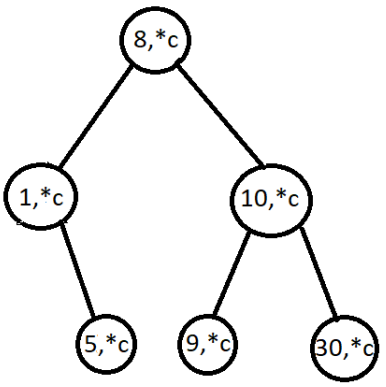
3. Representationer

Dessa representationer kommer exemplet av [A: {(r, c, v)}] där A är alla element, r är rader, c är kolumner och v är värdet. För att enklare förklara dessa kommer vi använda oss av mängden B: {(8, 3, e), (10, 4, 8), (1, 2, 6), (30, 15, a), (5, 3, 10), (10, 6, a), (1, 3, 7), (9, 3, 3}

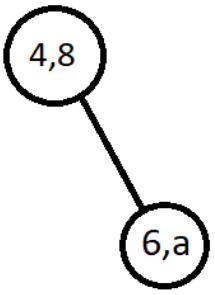
3.1 Binära sökträd nästlad i ett binärt sökträd.

Ett binärt sökträd är en typ av binärt träd där varje nod följer en viss ordning. Denna ordning är alla barn till vänster av noden är mindre än noden och alla barn åt höger av noden är större.

I denna representation kommer första trädets noder innehålla rader samt pekare till de nästlade träden. Dessa innehåller kolumnerna samt värdet i cellen. Om mängden B antas vara ordnad som skrivet ovan då (8,3,e) blir insatt först och sist (9,3,3) blir de binära sökträdet uppbyggt som Figur 1. Om alla värden i rad 10 ska hittas så kommer de binära sökträdet för rad 10 visas som i Figur 2.



Figur 1. Binärt sökträd som innehåller noder som i sig innehåller rad samt pekare till binärt sökträd som innehåller kolumner.

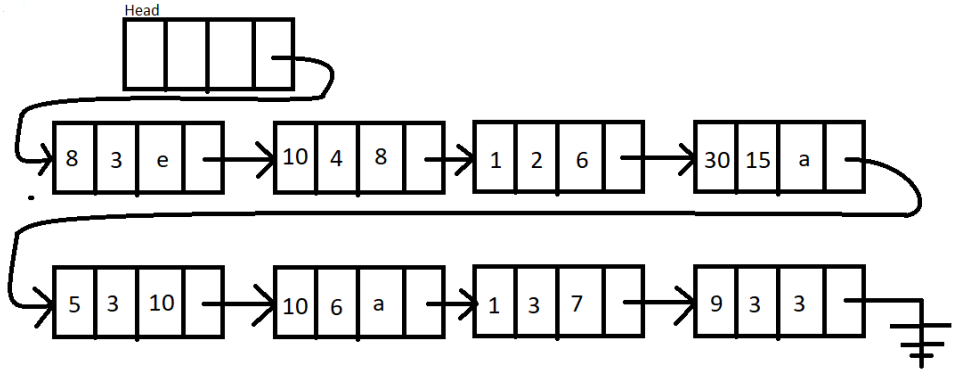


Figur 2. Binärt sökträd som innehåller noder som i sig kolumner samt värdet av cellen. Figur 1 nod 10 pekar på detta träd.

3.2 Riktad Lista som innehåller koordinater och värde.

En Riktad lista är en specialisering av Lista. En Riktad lista har riktning man kan bara flytta sig åt ett håll. Listan använder sig av ett 'Head' som kommer va samma typ som elementen i listan men dess värden är odefinierade förutom pekaren på nästa element vilket kommer va första elementet i listan.

Denna representation kommer ej vara i en förutbestämd ordning. Listan kommer gå efter första elementet som blir insatt blir de första element i listan samt sista element som blir insatt blir de sista elementet i vektorn. Dessa element kommer innehålla en egendefinierad struktur med fyra variabler rad, kolumn, värde och sist en pekare till nästa element. Om mängden B antas vara ordnad att första element är (8, 3, e) samt sista element (9, 3, 3) kan denna representation visas som figur 3.

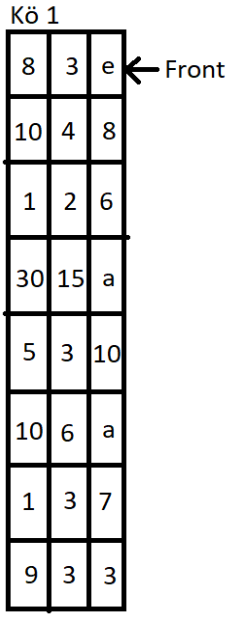


Figur 3. En lista med 8 element där varje element innehåller rad, kolumn, värde samt pekare till nästa element.

3.3 Kö av tripplar.

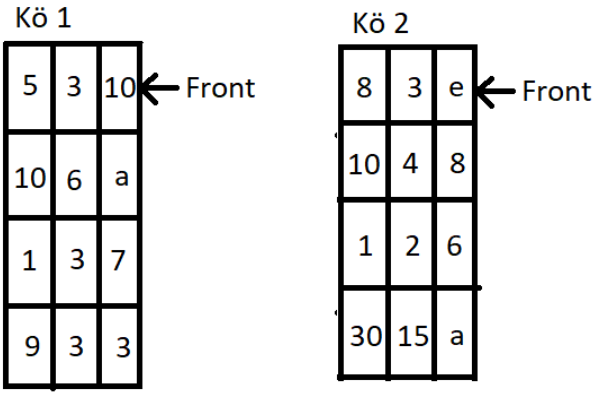
En kö är datatyp vilket sätter in element på en ordning av först in först ut. Denna datatyp kommer bli konstruerad med hjälp av en riktad lista. Vanligtvist i en kö om man vill komma åt ett element om de inte är längst fram i kön så måste man ta bort elementet längst fram tills man kommer till de önskade elementet.

För att inte ta bort elementen helt så kommer vi använda oss av två stycken köar i denna representation en för att hålla alla element och en för att hålla element som blir borttagna under traversering av kön. Elementen i kön kommer innehålla tre variabler rad, kolumn och värde. I figur 4 ser vi en kö med mängden B



Figur 4. En kö med element som innehåller rad, kolumn och värde.

Exempel om vi skulle nå elementet med värde 10 på rad 5 kolumn 3 kan representation se ut som i figur 5.



Figur 5. Kö från figur 4 som traverserat till elementet med rad 5 kolumn 3 och värde 10. Den har sparat de borttagna värden i kö 2.

4. Utvärdering

I Tabell 1 ser vi de olika representationerna samt deras resultat i de olika kriterierna. Tidskomplexiteten anges i stora ordo alltså O(n) där n är antal element i representationen.

	Insättning (Ordo)		Sökning(Ordo)		Borttagning (Ordo)	
	Värsta	Medel	Värsta	Medel	Värsta	Medel
Binära sökträd nästlad i ett binärt sökträd	O(n)	O(log n)	O(n)	O(log n)	O(n)	O(log n)
Riktad lista	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)
Kö av tripplar	O(1)	O(1)	O(n)	O(n)	O(n)	O(n)

Tabell 1. Sammanfattning av utvärdering av representationerna

5. Slutsats

Den bästa representation av representationerna genom dessa tre kriterier är Binära sökträd nästlad i ett binärt sökträd. Detta är för att till exempel medeltidskomplexiteten för alla kriterier för binära sökträd är O(log n). Samt så är värstafallstidskomplexiteten för ett binärt sökträd så osannolikt när antalet element växer. Kriteriet som gjorde skillnaden för binära sökträdet är sökning där de andra har båda O(n) för medel och värstafallstidskomplexitet medans binära sökträdet har O(n) och O(log n).

Vet man dock att kalkylbladet kommer användas med bara en liten mängd element kan riktad lista vara ett bättre alternativ med en snabbare insättning och borttagning.

6. Källförteckning

1. Sehgal, Karuna. A simplified explanation of the Big O notation. Medium. 2017-11-27

https://medium.com/karuna-sehgal/a-simplified-explanation-of-the-big-o-notation-82523585e835

(Hämtad 2022-01-27)