18
Multivejstræer og B-træer.

Multivejs søgetræer.

Søgning i multivejssøgetræer.

Pragmatisk lagring af data i multivejstræer.

B-træer.

Indsættelse i B-træer.

Eksempel på indsættelse i B-træ.

Facts om B-træer.

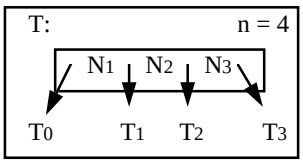
B+ træer.

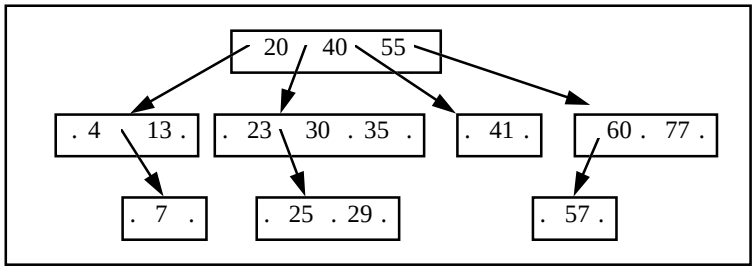
PS1 -- Multivejstræer og B-træer

Noter

Multivejstræer og multivejs-søgetræer.

- Et *multivejstræ* er et ordnet træ af grad større end 2.
- Et *multivejs-søgetræ af grad n* er et multivejstræ T af grad n, som opfylder:
  - Nøglerne N_i i roden af T er ordnede, $1 <= i <= (n - 1)$.
  - For alle nøgler L i T_{i-1} og for alle nøgler R i T_i : $L < N_i < R$ $(1 <= i < n)$.
  - T_0 ... T_{n-1} er multivejs-søgetræer.





PS1 -- Multivejstræer og B-træer

Noter

Motivationen for at se på multivejstræer er at kunne arbejde med søgetræer, der egner sig til at blive lagret på pladelager (disk). Emnet er altså **ekstern søgning.**

Det er ikke attraktivt at benytte binære søgetræer på et sekundært og langsomt lagermedium. Årsagen er, at i et binært træ skal relativ mange, små knuder aflæses/påvirkes for at gennemføre en given operation (søgning, indsættelse, sletning).

I et multivejstræ kan der repræsenteres mange dataobjekter i én knude i træet. Knudestørrelsen kan afpasses efter blokstørrelsen på pladelageret.

Ligesom vi kender det fra binære søgetræer, vil vi her antage, at en nøgle højst er tilstede én gang i et multivejs-søgetræ.

Hvis vi vil arbejde med balancerede multivejstræer, giver vi køb på udnyttelsesgraden af træets knuder. Mao. vi accepterer, at der i visse situationer er uudnyttet plads i knuderne.

Ovenstående viser et eksempel på et multivejs-søgetræ af graden 4. En prik i en knude betyder en null-pointer (void). Bemærk at der højst kan være 3 nøgler (data) i en knude i dette multivejstræ.

Søgning i multivejs-søgetræer.

Multivejs-søgning(T: Multivejs-søgetræ, N: Nøgle): Søgeresultat:

Søg efter N i roden af T ;
If *N er fundet*
then Returner *det dataobjekt hvori N er nøgle*
elseif *find nøglerne N0 og N1 i roden af T så N0 < N < N1* ;
If *der findes et ikke-tomt undertræ U mellem N0 og N1*
then Returner Multivejs-søgning(U, N)
else Returner *roden af T*

Søg efter N i roden af T :

- Binær søgning
- Lineær søgning

Noter

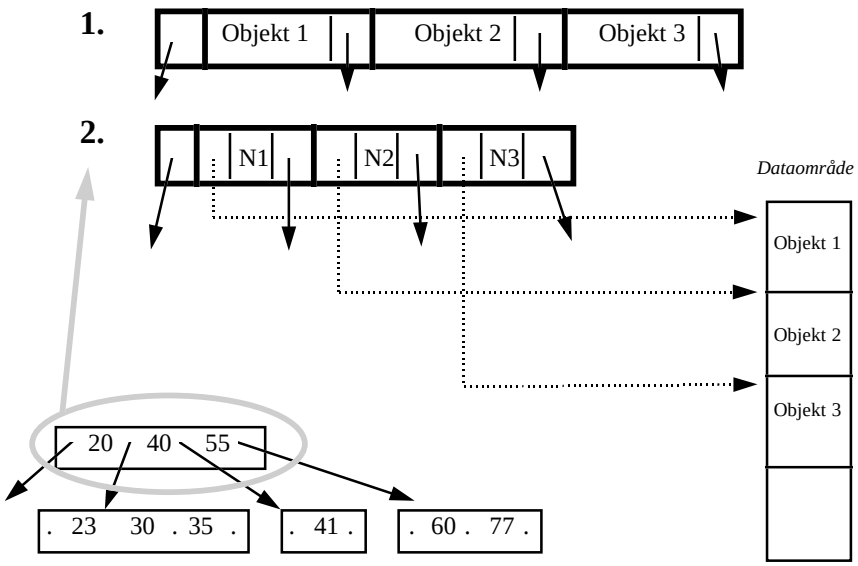
Ovenstående algoritmeskitse for søgning i et multivejstræ returnerer enten det dataobjekt, hvori N er nøgle, eller den knude i multivejstræet hvori et evt. objekt med N som nøgle kan indsættes, hvis der senere bliver behov for dette.

Når vi skriver "dataobjekt" mener vi nøglen og satellit-informationen tilsammen.

Hvis N i ovenstående algoritmeskitse er mindre end den mindste nøgle i en knude eksiterer N0 naturligvis ikke. Tilsvarende hvis N er større end den største nøgle i knuden eksisterer N1 ikke. I disse tilfælde refererer U til undertræet mest til venstre, hhv. undertræet mest til højre i knuden.

Som antydet, kan man enten benytte lineær søgning eller binær søgning for at finde nøglen i knuden.

Praktisk lagring af data i multivejstræer.



PS1 -- Multivejstræer og B-træer

Noter

Det vises på denne slide, hvordan en udvalgt knude fra træet forneden til venstre kan repræsenteres.

I tilfælde 1 er selve de tre dataobjekter (inden i hvilke nøglerne og andre data befinder sig) lagret inden i knuden. Dette giver en stor knude. Dette kan være uhensigtsmæssigt, idet man under søgning (og dermed også indsættelse og sletning) vil være tvunget til at læse dataobjekterne af en række irrelevante knuder ind fra sekundært lager til primært lager, i selve søgeprocessen.

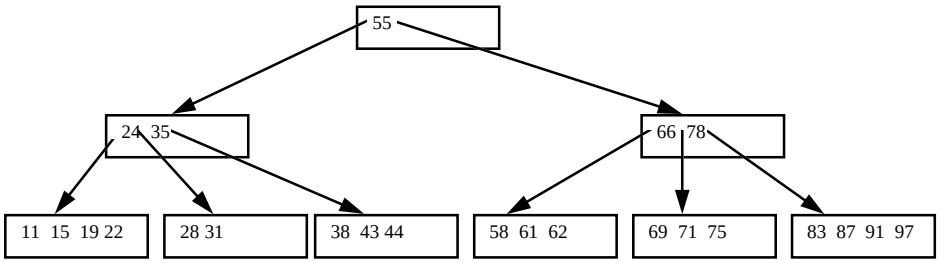
I tilfælde 2 er kun nøglen af objekterne lagret i knuderne i træet. Selve dataobjekterne er lagret i et særskilt dataområde. De stiplede pile angiver referencer til (adresser på) disse objekter.

I kapitlet om filer og indekser vil vi vende tilbage til problematikken om dataområder (som en abstraktion over filer hvorpå der kan lagres objekter).

B-træer.

Et B-træ af grad n er et multivejs-søgetræ af grad n hvor:

1. Knuder, som ikke er blade eller roden, har x sønner, hvor n/2 <= x <= n.
2. Roden har mindst 2 sønner (medmindre den er et blad).
3. Knuder, som ikke er roden, har y dataobjekter, hvor n/2-1 <= y <= n-1.
4. Indre knuder med k sønner indeholder k-1 dataobjekter.
5. Alle blade har samme dybde.



PS1 -- Multivejstræer og B-træer

Noter

Punkt 3 følger af punkt 4 pånær i tilfælde af bladene. Den egentlige rolle af punkt 3 er altså at begrænse antallet af dataobjekter i bladene.

Et B-træ af graden n kan maksimalt have n-1 dataobjekter (nøgler) pr. knude. Hvis n er ulige, f.eks. 7, siger ovenstående definition, at 3.5 <= antal sønner <= 7. I praksis betyder dette, at antallet af sønner er 4, 5, 6 eller 7. Ligeledes siger ovenstående definition, at 2.5 <= antal dataobjekter <= 6. Dette betyder, at antallet af dataobjekter pr. knuder er 3, 4, 5 eller 6.

Sidste punkt i definitionen af B-træ afspejler, at træet er vel afbalanceret.

Der er vist et konkret eksempel på et B-træ, som vi antager har grad 5. Nøglerne repræsenterer her dataobjekterne, og nøglerne er heltal. I dette træ har alle knuder (på nær roden) 3, 4 eller 5 sønner; og alle knuder (igen pånær roden) har 2, 3 eller 4 dataobjekter.

Det er vigtigt at bemærke, at B-træer, som defineret i disse noter, er anderledes end B-træerne der diskuteres i Weiss.

Vi kalder Weiss' B-træer for B+ træer. Vi diskuterer kort B+ træer i slutningen af dette kapitel.

Indsættelses-algoritme for B-træer.

```
Indsæt( T: B-træ, Obj: Data-objekt):

    Resultat := Multivejs-søgning( T, Obj.nøgle);
    If  Obj ikke er fundet i  Resultat
    then   -- Resultat er knuden, hvori Obj skal indsættes
           Indsæt-i-knude(Resultat, Obj)
```

```
Indsæt-i-knude(K: B-træes-knude;  Obj: Data-objekt)
    if  plads til endnu et objekt i K
    then  indsæt Obj på den rigtige plads i K
    else  del K i to knuder L og M, samt et opadstigende data-objekt Op ;
          If K.far eksisterer
          then  F := K.far
          else F := En ny rod ;
          Indsæt-i-knude(F, Op );
          Indsæt L og M som undertræer af  F
```


Noter

Proceduren anvender 'multivejs-søgning', som er vist tidligere i denne forelæsning. Det er let at indse, at hvis dataobjektet ikke findes i forvejen i B-træet, da bliver 'insæt-i-knude' kaldt på et blad i 'indsæt'.

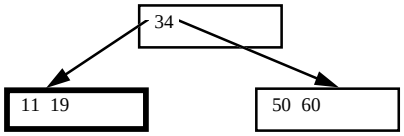
På ét punkt er proceduren 'indsæt-i-knude' ikke dækkende : Hvis et element indsættes i forældre knuden til K, og hvis denne forældre knude igen skal deles, hvordan indsættes L og M i "de to forældreknuder"?

Hvis der ikke er plads til et dataobjekt i en knude i B-træet kunne det være en god ide at undersøge, om der er god plads i søskende-knuderne. Hvis dette er tilfældet, er det forholdsvis let at foretage en *omfordeling* af objekter mellem to søskende knuder. Bemærk, at en sådan omfordeling også involverer et dataobjekt i de to søskendeknuders fælles forældreknude. I indsættelse-scenariet herefter, er der diskuteret et tilfælde af en sådan omfordeling.

Indsættelses-scenario i et B-træ (1).

Indsætter
50:

Indsætter
11, 19, 34, 60:



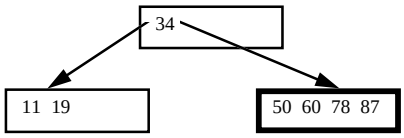
Noter

På denne og de følgende fire slides vises, hvordan et B-træ af grad 5 udvikler sig ved indsættelse af en række tal. De indsatte tal skal opfattes som nøgler for de egentlige objekter i B-træet.

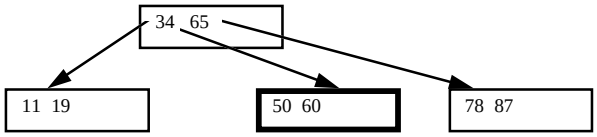
Knuder, som er indrammet med fed streg (eller lignende), tjener kun som identifikation af en knude på tværs af to øjebliksbilleder på samme slide.

Indsættelses-scenario i et B-træ (2).

Indsætter
78, 87:



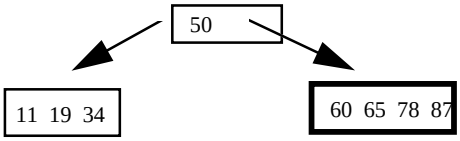
Indsætter
65:



PS1 -- Multivejstræer og B-træer

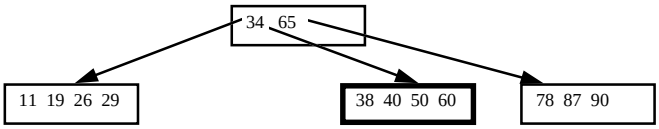
Noter

Når 65 indsættes i det øverste B-træ har vi illustreret, at knuden med de fire dataobjekter opdeles i to knuder. Hvis vi i stedet for foretog en *omfordeling* af dataobjekter i de to blade (som jo er søskende) kunne vi i dette tilfælde have undgået at dele knuden. Dette giver en bedre udnyttelse af knuderne i B-træet. Det resulterende B-træ er vist nedenfor:

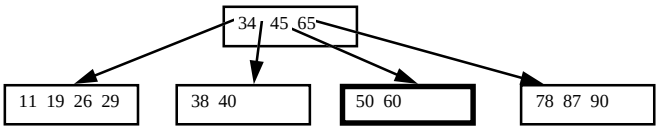


Indsættelses-scenario i et B-træ (3).

Indsætter:
38, 40, 90, 26, 29:



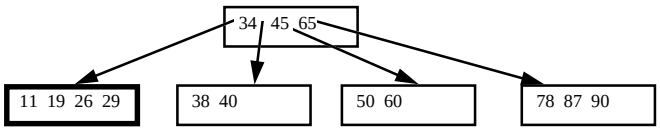
Indsætter:
45:



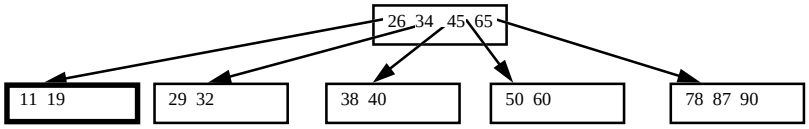
Noter

Tallet 45 skal indsættes mellem 40 og 50 i knuden med den fede ramme i træet øverst på sliden. Da træet er af grad 5, er denne knude fyldt op. Indsættelsesalgoritmen siger, at vi skal dele knuden i to, samt at vi skal indsætte “et opadstigende objekt” (45) i faderen, hvor der er i vort tilfælde er god plads til den.

Indsættelses-scenario i et B-træ (4).



Indsætter
32:



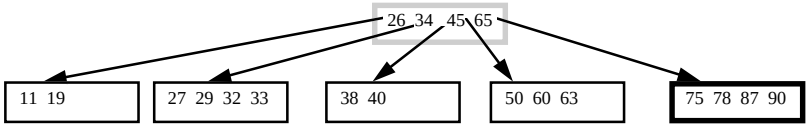
PS1 -- Multivejstræer og B-træer

Noter

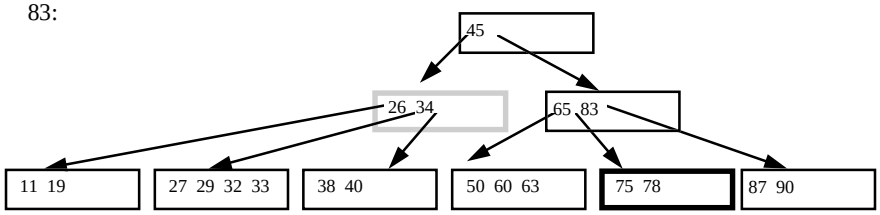
Træet foroven er identisk med træet forneden på forrige slide. Når vi indsætter 32 ser vi, at det hører hjemme i bladet til venstre; her er ikke plads, og derfor deles dette blad i to knuder samt et opadstigende objekt (26). Der er stadig plads til dette objekt i roden i B-træet.

Indsættelses-scenario i et B-træ (5).

Indsætter
27, 33, 63, 75:



Indsætter
83:



PS1 -- Multivejstræer og B-træer

Noter

Efter at have indsat 27, 33, 63 og 75 opstår der en interessant situation når vi indsætter 83. Der er ikke plads til objektet med nøgle 83 i bladet længst til højre, hvor det jo hører hjemme. Vi deler derfor knuden i to samt et opadstigende objekt: 83. Nu er der imidlertid ikke plads til 83 i roden, hvorfor vi også må dele roden i to knuder, samt et nyt opadstigende objekt (45), der nu placeres i en ny rod. Det resulterede træ er vist nederst.

Dette afslutter vores indsættelses-scenario.

Sletning af dataobjekter fra B-træer.

Fremgangsmåde.

Givet: Nøglen N og B-træet T.

Ønskes: Slet dataobjektet D med nøglen N i B-træet T.

Metode:

1. Find dataobjektet D med nøglen N i T **;**
2. Hvis D er i en indre knude:

    2a. Erstat D med det største dataobjekt X, der er mindre end D **;**
    2b. Slet dataobjektet X, som altid vil være i et blad.

3. Hvis D er i et blad b:

    3a. Hvis b indeholder mere end det min. antal dataobjekter: Slet D umiddelbart.
    3b. Ellers: Hvis b har minimalt dataindhold:

        3ba. Lån om muligt et dataobjekt fra en af b's søskende knuder.
        3bb. Ellers: Slå b sammen med en af b's søskendeknuder,
        og træk objektet mellem disse to knuder ned i den ny knude.
        Dette kan forårsage, at indre knuder slåes sammen,
        og ultimativt, at træets højde reduceres.

PS1 -- Multivejstræer og B-træer

Noter

Bemærk, at tilfælde 2b kan realiseres ved et rekursivt kald af den slette-operation, vi er ved at redegøre for.

Opgave: Tag udgangspunkt i det resulterede B-træ i indsættelses-scenariet på de foregående slides, og slet alle de indsatte objekter

(a) først i samme rækkefølge, som de blev indsat,

(b) dernæst i modsat rækkefølge, som de blev indsat.

Tegn passende øjebliksbilleder af B-træerne undervejs i slette-processen.

Facts om B-træer.

T er B-træ af grad g og højde h:

- Maximum antal knuder: $g^0 + g^1 + g^2 + ... + g^h = \frac{g^{h+1} - 1}{g - 1}$
- Maximum antal dataobjekter: $g^{h+1} - 1$
- Minimum antal dataobjekter: $2\left(\frac{g+1}{2}\right)^h - 1$

T er et B-træ af grad g med N dataobjekter:

- Højde af T <= $\log_x((N + 1)/2)$ hvor x = (g+1)/2.

Noter

På denne slide antager vi af bekvemmelighedshensyn (for at undgå at skulle angive afrundinger), at B-træer har ulige grad.

Bemærk, at vi skelner mellem *knuder* i B-træer og *dataobjekter*, hvoraf der jo kan være op til g-1 pr. knude, hvor g betegner B-træets grad.

Lad a være det minimale antal dataobjekter pr. knude og lad b være det minimale antal sønner pr. knude. Hvis g er lige er a = g/2 -1, og b = g/2. Hvis g er ulige er a = (g-1)/2 og b = (g+1)/2. Det minimale antal dataobjekter i B-træet er nu 1 + 2a +2ba + 2bba+ .... Når man regner lidt på denne sum får man ovenstående resultat.

Vurderingen af højden opnåes ud fra en observation om, at hvis et B-træ af højde h indeholder N dataobjekter vil der være N+1 "stopklodsknuder" (se nedenfor) på højde-niveau h+1 i træet. Endvidere gælder, at der minimum vil være 2 b^h knuder på dette "stopklodsniveau", hvor b har samme betydning som ovenfor. Dette giver ovenstående vurdering.

"Stopklodsknuder" er tænkte knuder på et niveau dybere end bladknuderne i et B-træ. I stedet for stopklodsknuder kan vi tale om nil-pointere til undertræer. Der er stopklodsknuder før, mellem og efter alle dataobjekter i de egentlige blad-knuder i træet.

Køretid af søgning, indsættelse og sletning i B-træer.

T er et B-træ af grad g med N dataobjekter:

Søgning efter dataobjekt i T:

Køretid = arbejdet pr. knude · højden af træet <=
$k_1 \cdot \log(g) \cdot \log_x((N + 1)/2) =$
$O(\log(g) \cdot \log(N))$

Indsættelse eller sletning af et dataobjekt i T:

Køretid = arbejdet pr. knude · højden af træet <=
$k_2 \cdot g \cdot \log_x((N + 1)/2) =$
$O(g \cdot \log(N))$

Noter

I ovenstående anvender vi vurderingen af højden af T fra forrige slides.

Ret beset er det ikke højden af T, som indgår ovenfor, men antallet af niveauer i træet (som er højden + 1). Denne lille forskel berører dog ikke O(...) vurderingen af køretidsresultaterne, idet vi jo ser bort fra lavere ordens led i O(...).

Log(x) betegner ovenfor logaritmefunktionen med grundtal 2.

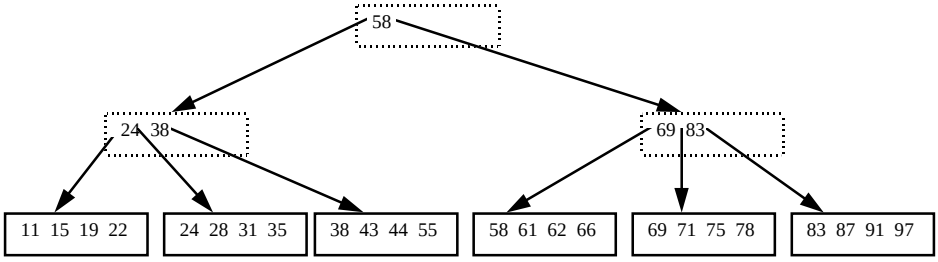
B+ træer (1).

Et B+ træ af grad n er et ordnet træ af grad n hvor

- Alle dataobjekter findes i bladene.
- De indre knuder udgør et *indeks* til dataobjekterne

og hvor

- Knuder, som ikke er blade eller roden, har mellem n/2 og n sønner.
- Roden har mindst 2 sønner.
- Blade har mellem n/2 og n dataobjekter.
- Alle blade har samme dybde.



PS1 -- Multivejstræer og B-træer

Noter

Denne slide giver Weiss' definition af et B+ træ på en form, som er sammenlignelig med vores tidligere definition af et B-træ.

Hvis n er et ulige tal, skal størrelsen n/2 ovenfor *rundes op.*

De understregede dele af definitionen ovenfor angiver afvigelserne fra vores tidligere definition af B-træer.

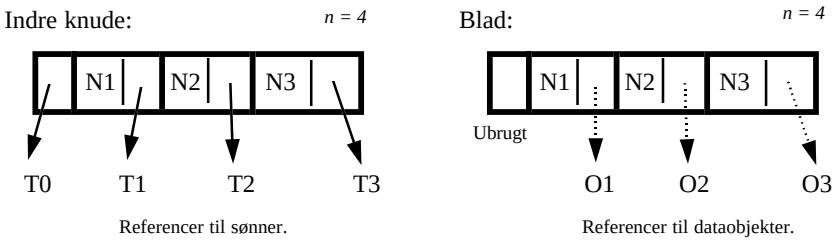
Et B+ træ er ikke et søgetræ (men det minder en del om et søgetræ). Forskellen består i, at et søgetræ har dataobjekter i indre knuder. I et B+ træ udgør de indre knuder et indeks, og elementerne i dette indeks findes også i bladene.

Et element i en indre knude udvælges som den mindste nøgle i det højre undertræ. Efter indsættelse og sletning justeres indekset efter denne regel. Der er altså ikke i B+ træer noget “opadstigende” eller “nedaddalende” dataobjekt at tage vare på under indsættelse hhv. sletning. Dette berøres også på næste slide.

Nederst er vist B+ træet, som svarer til B-træet fra sliden med navn "B-træer".

B+ træer (2).

- I en indre knude af et B+ træ af grad n:
  For alle nøgler L i blade af T_{i-1} og for alle nøgler R i blade af T_i :
  $$L < N_i <= R \quad (1 <= i < n).$$
- Søgning: Analog til søgning i multivejs-søgetræ.
- Indsættelse:
  - Ved knude-delning må strukturen af indekset tilpasses.
  - Intet "opad-stigende" element.
  - B+ træer vokser på samme måde som B-træer.



Noter

Bemærk den strukturmæssige lighed mellem indre knuder og blade. I blade er der et ubrugt pointer-plads, som passende kan bruges til at kæde alle blade sammen. Dette vil gøre det meget effektivt at løbe alle data igennem i ordnet rækkefølge.

I B+ træer er der begrebsmæssig forskel på blade og indre knuder. Derfor kan maximumsantallet af nøgler i bladene gøres helt uafhængig af maximumsantallet af nøgler i indre knuder.