

Prioritetskøer

- Prioritetskøer
- Træer og hobe
- Repræsentation af hobe
- Algoritmer på hobe
- Hobkonstruktion
- Hobsortering

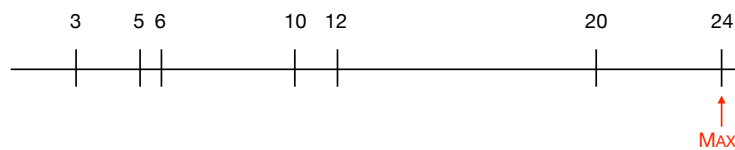
Philip Bille

Prioritetskøer

- Prioritetskøer
- Træer og hobe
- Repræsentation af hobe
- Algoritmer på hobe
- Hobkonstruktion
- Hobsortering

Prioritetskøer

- **Prioritetskøer.** Vedligehold en dynamisk mængde S af elementer. Hver element x er tilknyttet en nøgle $x.key$ og satellitdata $x.data$.
 - $MAX()$: returner element med **største** nøgle.
 - $EXTRACTMAX()$: returner **og fjern** element med **største** nøgle.
 - $INCREASEKEY(x, k)$: sæt $x.key = k$. Vi antager $k \geq x.key$.
 - $INSERT(x)$: sæt $S = S \cup \{x\}$



Prioritetskøer

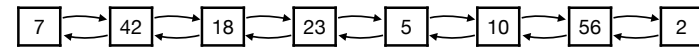
- **Anvendelser.**
 - Skedulering
 - Korteste veje i grafer (Dijkstras algoritme + venner)
 - Mindste udspændende træer i grafer (Prims algoritme)
 - Kompression (Huffmans algoritme)
 - ...

Prioritetskøer

- **Udfordring.** Hvordan kan vi løse problemet med nuværende teknikker?

Prioritetskøer

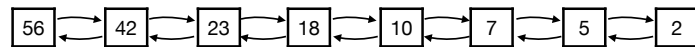
- **Løsning med hægtet liste 1.** Vedligehold S i en dobbelt-hægtet liste.



- MAX(): lineær søgning efter element med største nøgle.
- EXTRACTMAX(): lineær søgning efter element med største nøgle. Returner og fjern element.
- INCREASEKEY(x, k): sæt x.key = k.
- INSERT(x): tilføj element i start af listen.
- **Tid.**
 - MAX og EXTRACTMAX i $O(n)$ tid ($n = |S|$).
 - INCREASEKEY og INSERT i $O(1)$ tid.
- **Plads.**
 - $O(n)$.

Prioritetskøer

- **Løsning med hægtet liste 2.** Vedligehold S i en dobbelt-hægtet **sorteret** liste.



- MAX(): returner første element.
- EXTRACTMAX(): returner og fjern første element.
- INCREASEKEY(x, k): sæt x.key = k. Lineær søgning fremad i listen for at indsætte x på korrekt position.
- INSERT(x): lineær søgning for at indsætte x på korrekt position.
- **Tid.**
 - MAX og EXTRACTMAX i $O(1)$ tid.
 - INCREASEKEY og INSERT i $O(n)$ tid.
- **Plads.**
 - $O(n)$.

Prioritetskøer

Datastruktur	MAX	EXTRACTMAX	INCREASEKEY	INSERT	Plads
hægtet liste	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
sorteret hægtet liste	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$

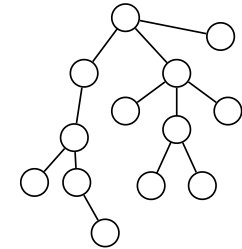
- **Udfordring.** Kan vi gøre det betydeligt bedre?
 - Kræver ny teknologi.

Prioritetskøer

- Prioritetskøer
- Træer og hobe
- Repræsentation af hobe
- Algoritmer på hobe
- Hobkonstruktion
- Hobsortering

Rodfæstede træer

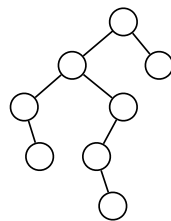
- Rodfæstede træer.
 - Knuder forbundet med kanter.
 - Sammenhængende og uden kredse.
 - En knude udvalgt til at være rod.
 - Speciel type graf.



- Terminologi.
 - Børn, forælder, efterkommer, forfader, blade, interne knuder, sti.
- Dybde og højde.
 - Lad v være en knude i træ T.
 - dybden af v = længden af sti fra v til roden.
 - højden af v = længden af længste sti fra v til en bladefterkommer.
 - dybden af T = højden af T = længden af længste sti fra rod til et blad.

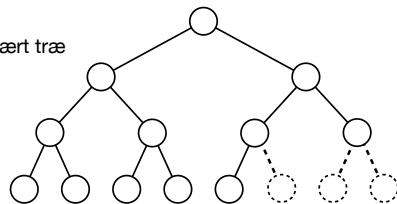
Binære træer

- Binært træ. Binært træ er enten
 - Tomt
 - En knude med et venstre og højre barn der begge er binære træer.



- Komplet binært træ. Binært træ hvor alle interne knuder har præcis to børn.
- Næsten komplet binært træ. Komplet binært træ hvor 0 eller flere blade er fjernet fra højre mod venstre.

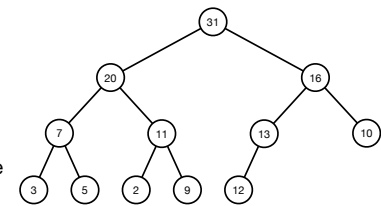
- Lemma. Højden af et (næsten) komplet binært træ med n knuder er $\Theta(\log n)$.
- Bevis. Se ugeseddel.



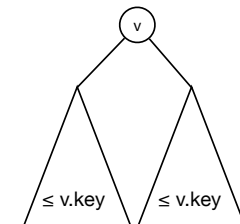
Hob

- Hob (heap). Næsten komplet binært træ der overholder hob-orden.

- Hob-orden (heap-order).
 - Alle knuder indeholder et element.
 - For alle knuder v:
 - alle nøgler i venstre deltræ og højre deltræ er $\leq v.key$.



- Max-hob vs min-hob. Ovenstående er max-hob. Udskiift \leq med \geq for at få min-hob.



Prioritetskøer

- Prioritetskøer
- Træer og hobe
- **Repræsentation af hobe**
- Algoritmer på hobe
- Hobkonstruktion
- Hobsortering

Hob

- **Repræsentation.** Vi skal bruge følgende navigationsoperationer på en hob:
 - PARENT(x): returner forældren af x.
 - LEFT(x) : returner venstre barn af x.
 - RIGHT(x): returner højre barn af x.
- **Udfordring.** Hvordan kan vi repræsentere en hob så vi kan understøtte navigation effektivt?

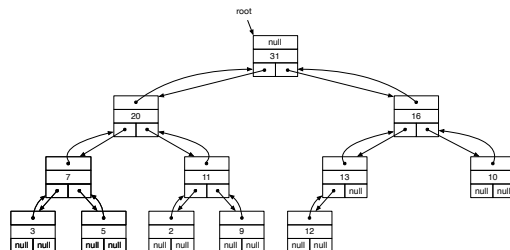
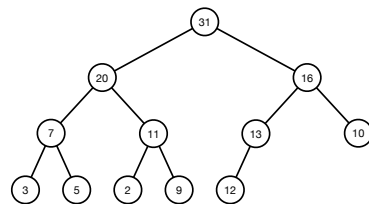
Hob

- **Repræsentation med hægter.** Hver knude v består af

- v.key
- v.parent
- v.left
- v.right

- PARENT, LEFT, RIGHT ved at følge pointer.

- **Tid.** $O(1)$
- **Plads.** $O(n)$



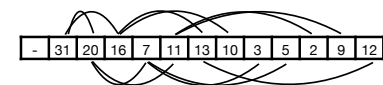
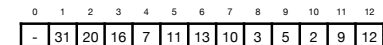
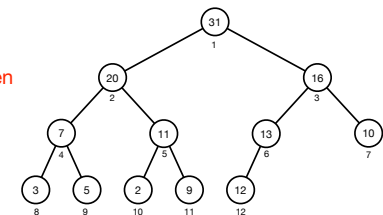
Hob

- **Repræsentation med tabel.**

- Tabel $H[0..n]$
- $H[0]$ bruges ikke.
- $H[1..n]$ indeholder knuder i **niveauorden** (level order).

- PARENT(x): returner $\lfloor x/2 \rfloor$
- LEFT(x) : returner $2x$.
- RIGHT(x): returner $2x + 1$

- **Tid.** $O(1)$
- **Plads.** $O(n)$

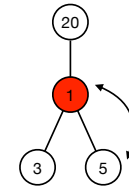
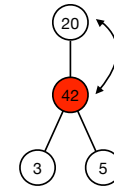


Prioritetskøer

- Prioritetskøer
- Træer og hobe
- Repræsentation af hobe
- Algoritmer på hobe
- Hobkonstruktion
- Hobsortering

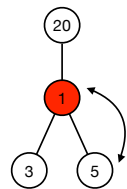
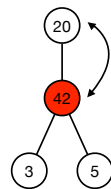
Algoritmer på hobe

- BUBBLEUP(x):
 - Hvis hoborden er overtrådt i knude x fordi nøgle i x er > nøgle i PARENT(x).
 - Ombyt x og PARENT(x).
 - Gentag med PARENT(x) indtil hoborden er opfyldt.
- BUBBLEDOWN(x):
 - Hvis hoborden er overtrådt i knude x fordi nøgle i x er < nøgle i LEFT(x) eller RIGHT(x).
 - Ombyt x og barn b med største nøgle.
 - Gentag med b indtil hoborden er opfyldt.



Algoritmer på hobe

- BUBBLEUP(x):
 - Hvis hoborden er overtrådt i knude x fordi nøgle i x er > nøgle i PARENT(x).
 - Ombyt x og PARENT(x).
 - Gentag med PARENT(x) indtil hoborden er opfyldt.
- BUBBLEDOWN(x):
 - Hvis hoborden er overtrådt i knude x fordi nøgle i x er < nøgle i LEFT(x) eller RIGHT(x).
 - Ombyt x og barn b med største nøgle.
 - Gentag med b indtil hoborden er opfyldt.
- Tid. Hvor hurtigt kører de?
 - BUBBLEUP og BUBBLEDOWN i $\Theta(\log n)$ tid.
- Hvordan kan vi bruge dem til implementation af prioritetskøoperationer?



Prioritetskøoperationer

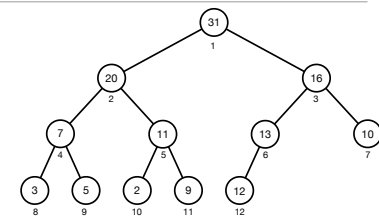
```

MAX()
return H[1]

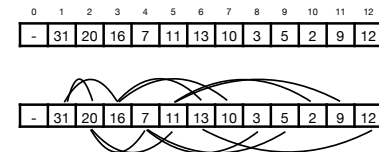
INSERT(x)
n = n + 1
H[n] = x
BUBBLEUP(n)

EXTRACTMAX()
r = H[1]
H[1] = H[n]
n = n - 1
BUBBLEDOWN(1)
return r

INCREASEKEY(x, k)
H[x] = k
BUBBLEUP(x)
    
```



- Opgave. Udfør følgende sekvens i initielt tom hob: 2, 5, 7, 6, 4, E, E
- Tal betyder INSERT og E betyder EXTRACTMAX.



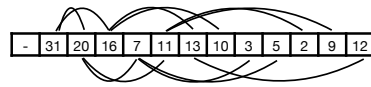
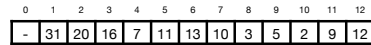
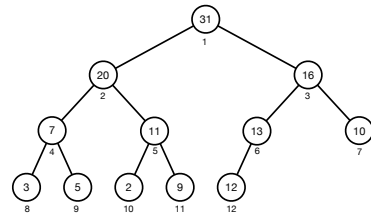
Prioritetskøoperationer

MAX()
return H[1]

INSERT(x)
n = n + 1
H[n] = x
BUBBLEUP(n)

EXTRACTMAX()
r = H[1]
H[1] = H[n]
n = n - 1
BUBBLEDOWN(1)
return r

INCREASEKEY(x, k)
H[x] = k
BUBBLEUP(x)



• Tid. Hvor hurtigt kører de?

- Max i $\Theta(1)$ tid.
- EXTRACTMAX, INCREASEKEY og INSERT i $\Theta(\log n)$ tid.

Prioritetskøer

Datastruktur	MAX	EXTRACTMAX	INCREASEKEY	INSERT	Plads
hægtet liste	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(n)$
sorteret hægtet liste	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$
hob	$O(1)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(n)$

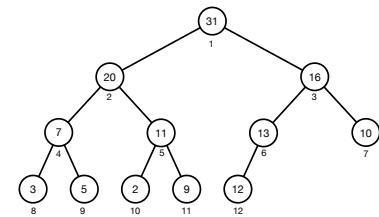
- Hob (med tabelrepræsentation) er eksempel på en **implicit datastruktur**.

Prioritetskøer

- Prioritetskøer
- Træer og hobe
- Repræsentation af hobe
- Algoritmer på hobe
- Hobkonstruktion
- Hobsortering

Hobkonstruktion

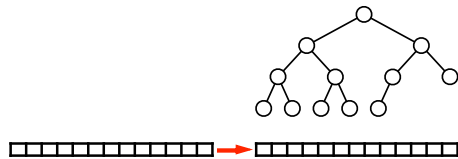
- **Hobkonstruktion**. Givet n heltal i en tabel H[0..n], byg tabel om til en hob.



Hobkonstruktion

- **Algoritme 1.**

- Etabler hoborden oppefra og ned.
- Rod er hob af størrelse 1.
- For alle andre knuder fra venstre til højre brug BUBBLEUP.



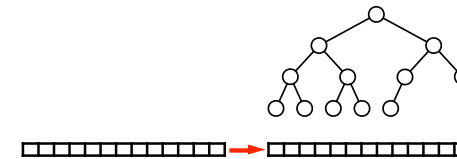
- **Tid.**

- For hver knude af dybde d bruger vi $O(d)$ tid.
- $\sim n/2$ knuder af dybde $\log n$, $\sim n/4$ knuder af dybde $\log n - 1$, $\sim n/8$ knuder af dybde $\log n - 2$, ..., 2 knuder af dybde 1.
- $n/2 \cdot \log n + n/4 \cdot (\log n - 1) + n/8 \cdot (\log n - 2) + \dots + 2 \cdot 1 = \Theta(n \log n)$
- $\Rightarrow O(n \log n)$ tid.
- **Udfordring.** Kan vi gøre det bedre?

Hobkonstruktion

- **Algoritme 2.**

- Etabler hoborden nedefra og op.
- Alle blade er allerede hobe af størrelse 1.
- For hver intern knude fra højre til venstre brug BUBBLEDOWN.



- **Tid.**

- For knude af højde h bruger vi $O(h)$ tid.
- $\sim n/4$ knuder af højde 1, $n/8$ knuder af højde 2, $n/16$ knuder af højde 3, ..., 1 knude af højde h .
- $n/4 \cdot 1 + n/8 \cdot 2 + n/16 \cdot 3 + \dots + 1 \cdot h = O(n)$
- $\Rightarrow O(n)$ tid

Prioritetskøer

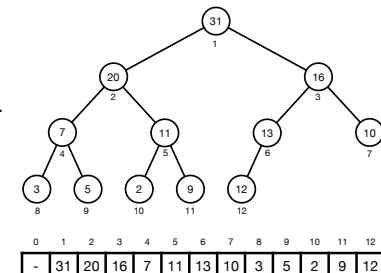
- Prioritetskøer
- Træer og hobe
- Repræsentation af hobe
- Algoritmer på hobe
- Hobkonstruktion
- **Hobsortering**

Hobsortering

- **Sortering.** Hvordan man bruge en hob og prioritetskøoperationer til at sortere en tabel $H[1..n]$ af n tal?

- **Algoritme.**

- Konstruer en hob for H .
- Lav n EXTRACTMAX.
 - Indsæt resultater i slutning af tabel.
- Returner tabel H .



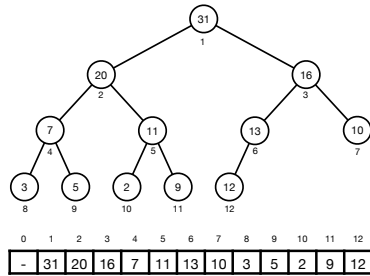
- **Tid.**

- Hobkonstruktion i $\Theta(n)$ tid
- n EXTRACTMAX i $\Theta(n \log n)$ tid.
- i alt $\Theta(n \log n)$ tid.

Hobsortering

• **Theorem.** Vi kan sortere en tabel i $\Theta(n \log n)$ tid.

- Bruger kun $O(1)$ **ekstra plads.**
- **Ækivalens** af sortering og prioritetskøer.



Prioritetskøer

- Prioritetskøer
- Træer og hobe
- Repræsentation af hobe
- Algoritmer på hobe
- Hobkonstruktion
- Hobsortering