Øvingsforelesning 6

TDT4120 - Algoritmer og datastrukturer

Øving 5

Indeksering i hauger

Oppgave 2: I en haug, hva er indeksen til foreldrenoden til en node med indeks 9?

Indeksering i hauger

Oppgave 2: I en haug, hva er indeksen til foreldrenoden til en node med indeks 9?

$$\lfloor i/2 \rfloor$$

Indeksering i hauger

Oppgave 2: I en haug, hva er indeksen til foreldrenoden til en node med indeks 9?

$$\lfloor i/2 \rfloor = \lfloor 9/2 \rfloor = 4$$

Indeksering i hauger

Oppgave 2: I en haug, hva er indeksen til foreldrenoden til en node med indeks 9?

$$\lfloor i/2 \rfloor = \lfloor 9/2 \rfloor = 4$$

Oppgave 3: Hvis indeksen er 10, hva er indeksene til barnene?

Indeksering i hauger

Oppgave 2: I en haug, hva er indeksen til foreldrenoden til en node med indeks 9?

$$\lfloor i/2 \rfloor = \lfloor 9/2 \rfloor = 4$$

Oppgave 3: Hvis indeksen er 10, hva er indeksene til barnene?

$$2i$$

$$2i + 1$$

Indeksering i hauger

Oppgave 2: I en haug, hva er indeksen til foreldrenoden til en node med indeks 9?

$$\lfloor i/2 \rfloor = \lfloor 9/2 \rfloor = 4$$

Oppgave 3: Hvis indeksen er 10, hva er indeksene til barnene?

$$2i = 2 \cdot 10 = 20$$
$$2i + 1 = 2 \cdot 10 + 1 = 21$$

Søketrær - Programmering

Oppgave 4: Søk etter en streng i et søketre.

Oppgave 5: Bygg et søketre for forekomster av strenger.

Diverse om binære hauger

Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Diverse om binære hauger

Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Min-haug egenskapen: $\mathrm{A}[\textsc{Parent}(i)] \geqslant \mathrm{A}[i]$
Max-haug egenskapen: $\mathrm{A}[\textsc{Parent}(i)] \leqslant \mathrm{A}[i]$
Binært-søketre egenskapen: Hvis y er det venstre barnet til x har vi $y.key \leqslant x.key$ og om y er det høyre barnet til x har vi $y.key \geqslant x.key$.

Diverse om binære hauger

Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Min-haug egenskapen: $\mathrm{A}[\textsc{Parent}(i)] \geqslant \mathrm{A}[i]$
Max-haug egenskapen: $\mathrm{A}[\textsc{Parent}(i)] \leqslant \mathrm{A}[i]$
Binært-søketre egenskapen: Hvis y er det venstre barnet til x har vi $y.key \leqslant x.key$ og om y er det høyre barnet til x har vi $y.key \geqslant x.key$.

Ingen binære hauger er binære søketrær.

Diverse om binære hauger

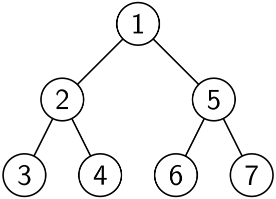
Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Oppgave 7: Ligger de $\lceil n/2 \rceil$ største elementene i en komplett binær min-haug i løvnodene?

Diverse om binære hauger

Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

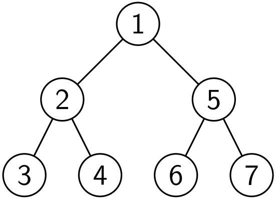
Oppgave 7: Ligger de $\lceil n/2 \rceil$ største elementene i en komplett binær min-haug i løvnodene?

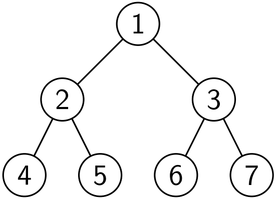


Diverse om binære hauger

Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Oppgave 7: Ligger de $\lceil n/2 \rceil$ største elementene i en komplett binær min-haug i løvnodene?





Diverse om binære hauger

Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Oppgave 7: Ligger de $\lceil n/2 \rceil$ største elementene i en komplett binær min-haug i løvnodene?

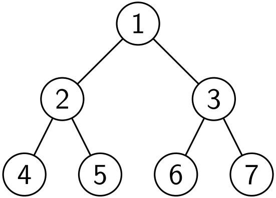
Oppgave 8: Representerer en sortert tabell med tall en haug?

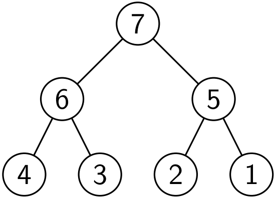
Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Oppgave 7: Ligger de $\lceil n/2 \rceil$ største elementene i en komplett binær min-haug i løvnodene?

Oppgave 8: Representerer en sortert tabell med tall en haug?

$\langle 1, 2, 3, 4, 5, 6, 7 \rangle$ og $\langle 7, 6, 5, 4, 3, 2, 1 \rangle$.





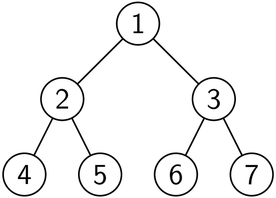
Diverse om binære hauger

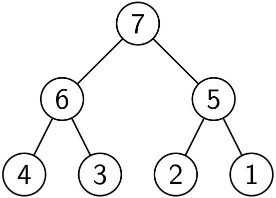
Oppgave 6: Hvordan er overlappet mellom binære hauger og søketrær med kun unike elementer og minst 3 elementer?

Oppgave 7: Ligger de $\lceil n/2 \rceil$ største elementene i en komplett binær min-haug i løvnodene?

Oppgave 8: Representerer en sortert tabell med tall en haug?

$\langle 1, 2, 3, 4, 5, 6, 7 \rangle$ og $\langle 7, 6, 5, 4, 3, 2, 1 \rangle$.

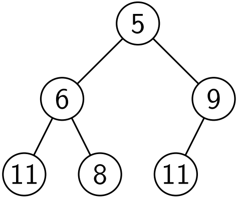




Ja, alltid.

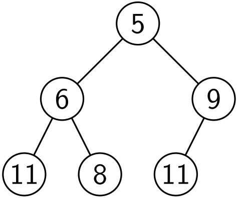
Representasjon av hauger

Oppgave 9: Hvordan er følgende haug representert som en tabell?



Representasjon av hauger

Oppgave 9: Hvordan er følgende haug representert som en tabell?



$\langle 5, 6, 9, 11, 8, 11 \rangle$

Oppgave 10: Hva er sant om HEAPSORT?

- Det er en stabil sorteringsalgoritme.
- Algoritmen har tidskompleksitet $\mathrm{O}(n)$ om alle elementene har samme verdi.
- Algoritmen er *in-place*.

Heapsort

Oppgave 10: Hva er sant om Heapsort?

- Det er en stabil sorteringsalgoritme.
- Algoritmen har tidskompleksitet $\mathrm{O}(n)$ om alle elementene har samme verdi.
- Algoritmen er *in-place*.

```
BUILD-MAX-HEAP(A)
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

Heapsort

Oppgave 10: Hva er sant om Heapsort?

- ~~Det er en stabil sorteringsalgoritme.~~
- Algoritmen har tidskompleksitet $O(n)$ om alle elementene har samme verdi.
- Algoritmen er *in-place*.

```
BUILD-MAX-HEAP(A)
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

Min-prioritetskø

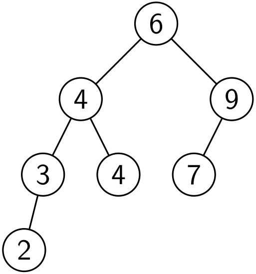
Oppgave 11: Hvilke operasjoner støtter en min-prioritetskø?

Min-prioritetskø

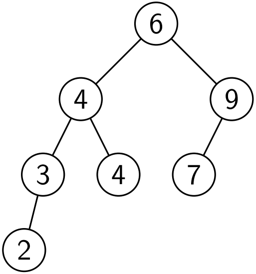
Oppgave 11: Hvilke operasjoner støtter en min-prioritetskø?

INSERT, MINIMUM, DECREASE-KEY og EXTRACT-MIN

Oppgave 12: I hvilke rekkefølge kunne tallene i følgende binære søketre blitt satt inn ved hjelp av TREE-INSERT

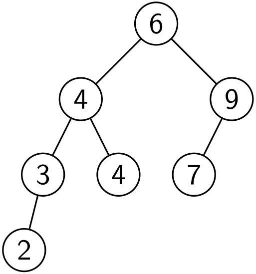


Oppgave 12: I hvilke rekkefølge kunne tallene i følgende binære søketre blitt satt inn ved hjelp av TREE-INSERT



Et barn må være satt inn etter sin foreldrenode.

Oppgave 12: I hvilke rekkefølge kunne tallene i følgende binære søketre blitt satt inn ved hjelp av TREE-INSERT



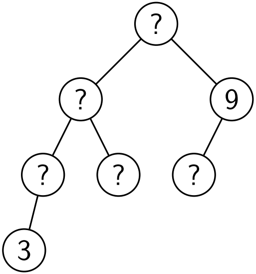
Et barn må være satt inn etter sin foreldrenode.

$\langle 6, 4, 9, 3, 4, 7, 2\rangle, \langle 6, 9, 4, 3, 2, 7, 4\rangle$ og $\langle 6, 4, 3, 9, 2, 7, 4\rangle$.

Binære søketre

Oppgave 12: I hvilke rekkefølge kunne tallene i følgende binære søketre blitt satt inn ved hjelp av TREE-INSERT

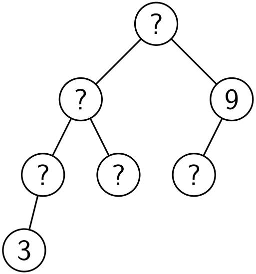
Oppgave 13: Hva er den største og minste summen av tallene man kan ha i følgende binære søketre?



Binære søketre

Oppgave 12: I hvilke rekkefølge kunne tallene i følgende binære søketre blitt satt inn ved hjelp av TREE-INSERT

Oppgave 13: Hva er den største og minste summen av tallene man kan ha i følgende binære søketre?



Binært-søketre egenskapen: Hvis y er det venstre barnet til x har vi $y.key \leqslant x.key$ og om y er det høyre barnet til x har vi $y.key \geqslant x.key$.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

INORDER-TREE-WALK har kjøretid $\mathrm{O}(n)$.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

INORDER-TREE-WALK har kjøretid $\mathrm{O}(n)$.

Verste tilfelle: Sortert liste, gir et tree på høyde n.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

INORDER-TREE-WALK har kjøretid $\mathrm{O}(n)$.

Verste tilfelle: Sortert liste, gir et tree på høyde n.

Beste tilfelle: Komplett binært søketre har maks dybde på $\lg n$. Gir $O(n \lg n)$ tid.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

INORDER-TREE-WALK har kjøretid $\mathrm{O}(n)$.

Verste tilfelle: Sortert liste, gir et tree på høyde n.

Beste tilfelle: Komplett binært søketre har maks dybde på $\lg n$. Gir $O(n \lg n)$ tid.
Minst $n/2$ tall på dybde $\geqslant \lg n$. Gir $\Omega(n/2 \lg n) = \Omega(n \lg n)$ tid.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre $\textsc{Inorder-Tree-Walk}$ på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

$\textsc{Inorder-Tree-Walk}$ har kjøretid $\mathrm{O}(n)$.

Verste tilfelle: Sortert liste, gir et tree på høyde n.

Beste tilfelle: Komplett binært søketre har maks dybde på $\lg n$. Gir $O(n \lg n)$ tid.
Minst $n/2$ tall på dybde $\geqslant \lg n$. Gir $\Omega(n/2 \lg n) = \Omega(n \lg n)$ tid.

Gjennomsnittlig kjøretid: Forventet høyde på et tilfeldig bygget binært søketre, $\Theta(\lg n)$.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

INORDER-TREE-WALK har kjøretid $\mathrm{O}(n)$.

Verste tilfelle: Sortert liste, gir et tree på høyde n.

Beste tilfelle: Komplett binært søketre har maks dybde på $\lg n$. Gir $O(n \lg n)$ tid.
Minst $n/2$ tall på dybde $\geqslant \lg n$. Gir $\Omega(n/2 \lg n) = \Omega(n \lg n)$ tid.

Gjennomsnittlig kjøretid: Forventet høyde på et tilfeldig bygget binært søketre, $\Theta(\lg n)$. Forventet kjøretid $\Theta(n \lg n)$.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

Oppgave 15: Hva stemmer om denne sorteringsalgoritmen?

- Sorteringsalgoritmen er asymptotisk optimal i verste tilfelle for sammenligningsbasert sortering.
- Hvis alle elementene i tabellen du skal sortere er like, vil algoritmen ta $\mathrm{O}(n)$ tid, siden tabellen allerede er sortert.
- Sorteringsalgoritmen er stabil dersom du setter inn tallene i søketreet i samme rekkefølge som de var i den opprinnelige tabellen.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

Oppgave 15: Hva stemmer om denne sorteringsalgoritmen?

- ~~Sorteringsalgoritmen er asymptotisk optimal i verste tilfelle for sammenligningsbasert sortering.~~
- Hvis alle elementene i tabellen du skal sortere er like, vil algoritmen ta $\mathrm{O}(n)$ tid, siden tabellen allerede er sortert.
- Sorteringsalgoritmen er stabil dersom du setter inn tallene i søketreet i samme rekkefølge som de var i den opprinnelige tabellen.

Sortering i binære søketrær

Man kan sortere et sett med tall ved å sette dem inn i et binært søketre og kjøre INORDER-TREE-WALK på dette treet.

Oppgave 14: Hva er kjøretiden til denne sorteringsalgoritmen?

Oppgave 15: Hva stemmer om denne sorteringsalgoritmen?

- ~~Sorteringsalgoritmen er asymptotisk optimal i verste tilfelle for sammenligningsbasert sortering.~~
- ~~Hvis alle elementene i tabellen du skal sortere er like, vil algoritmen ta $\mathrm{O}(n)$ tid, siden tabellen allerede er sortert.~~
- Sorteringsalgoritmen er stabil dersom du setter inn tallene i søketreet i samme rekkefølge som de var i den opprinnelige tabellen.

Konstruksjon av binære søketrær

Oppgave 16: Kan vi finne en algoritme som bygger et binært søketre i lineær tid i verstetilfelle?

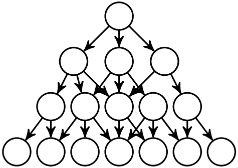
Konstruksjon av binære søketrær

Oppgave 16: Kan vi finne en algoritme som bygger et binært søketre i lineær tid i verstetilfelle?

Nei. Bryter med den nedre grensen for sammenligningsbasert sortering.

Trær med flere foreldre

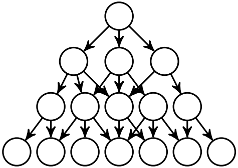
Har følgende datastruktur:



Oppgave 17: Kan vi finne antall stier fra rotnoden til hver av etterkommernodene i lineær tid i verste tilfelle, som funksjon av antall kanter?

Trær med flere foreldre

Har følgende datastruktur:

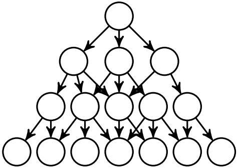


Oppgave 17: Kan vi finne antall stier fra rotnoden til hver av etterkommernodene i lineær tid i verste tilfelle, som funksjon av antall kanter?

Antall stier til en etterkommernode er summen av antall stier til foreldrenodene.

Trær med flere foreldre

Har følgende datastruktur:



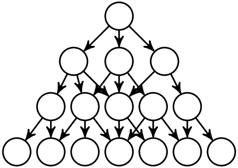
Oppgave 17: Kan vi finne antall stier fra rotnoden til hver av etterkommernodene i lineær tid i verste tilfelle, som funksjon av antall kanter?

Antall stier til en etterkommernode er summen av antall stier til foreldrenodene.

1. La alle rotnoden ha 1 som antall stier og resten 0.

Trær med flere foreldre

Har følgende datastruktur:

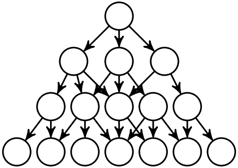


Oppgave 17: Kan vi finne antall stier fra rotnoden til hver av etterkommernodene i lineær tid i verste tilfelle, som funksjon av antall kanter?

Antall stier til en etterkommernode er summen av antall stier til foreldrenodene.

1. La alle rotnoden ha 1 som antall stier og resten 0.
2. Gå igjennom hver av nodene, fra toppen til bunnen. For hvert barn av en gitt node

Har følgende datastruktur:



Oppgave 17: Kan vi finne antall stier fra rotnoden til hver av etterkommernodene i lineær tid i verste tilfelle, som funksjon av antall kanter?

Antall stier til en etterkommernode er summen av antall stier til foreldrenodene.

1. La alle rotnoden ha 1 som antall stier og resten 0.
2. Gå igjennom hver av nodene, fra toppen til bunnen. For hvert barn av en gitt node
   a. Øk antall stier til barnet med antall stier til noden.

Oppgave 18: Har en DNA streng, og et sett med segmenter. Ønsker å finne ut hvor mange ganger disse segmentene totalt opptrer i DNA strengen. Vil ha $\mathrm{O}(d(n+k))$ i tidskompleksitet.

Strengmatching

Oppgave 18: Har en DNA streng, og et sett med segmenter. Ønsker å finne ut hvor mange ganger disse segmentene totalt opptrer i DNA strengen. Vil ha $\mathrm{O}(d(n+k))$ i tidskompleksitet.

```
STRING-MATCH(dna, segments)
root = BUILD-TREE(segments)
count = 0
for i = 1 to dna.length
        count += COUNT-SEGMENTS(dna, i, root)
return count
```

```
COUNT-SEGMENTS(dna, i, node)
if i > dna.length
        return 0
if dna[i] ∉ node.children
        return 0
node = node.children[dna[i]]
return node.count + COUNT-SEGMENTS(dna, i + 1, node)
```

Binærtrær og binærhauger - høyde

Oppgave 19: Hva vet vi om høyden til et tilfeldig binært søketre med n noder?

Binærtrær og binærhauger - høyde

Oppgave 19: Hva vet vi om høyden til et tilfeldig binært søketre med n noder?

Verste tilfelle: Høyden er $\mathrm{O}(n)$, (f.eks. TREE-INSERT på en liste av like tall)

Binærtrær og binærhauger - høyde

Oppgave 19: Hva vet vi om høyden til et tilfeldig binært søketre med n noder?

Verste tilfelle: Høyden er $\mathrm{O}(n)$, (f.eks. TREE-INSERT på en liste av like tall)

Beste tilfelle: Må ha høyde $\Omega(\lg n)$.

Binærtrær og binærhauger - høyde

Oppgave 19: Hva vet vi om høyden til et tilfeldig binært søketre med n noder?

Oppgave 20: Hva vet vi om høyden til en tilfeldig binærhaug med $n!$ elementer.

Binærtrær og binærhauger - høyde

Oppgave 19: Hva vet vi om høyden til et tilfeldig binært søketre med n noder?

Oppgave 20: Hva vet vi om høyden til en tilfeldig binærhaug med n! elementer.

Vet at høyden er $\Theta(\lg(n!))$.

Binærtrær og binærhauger - høyde

Oppgave 19: Hva vet vi om høyden til et tilfeldig binært søketre med n noder?

Oppgave 20: Hva vet vi om høyden til en tilfeldig binærhaug med $n!$ elementer.

Vet at høyden er $\Theta(\lg(n!))$.

$\Theta(\lg(n!)) = \Theta(n \lg n)$ (3.19, s. 58).

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis det er flere løvnoder enn det er interne noder i et binærtre er høyden på treet $\Theta(\lg n)$.
- Hvis antall interne noder med to barn er k og antall løvnoder er ℓ, så er $k + 1 = \ell$.
- Hvis antall løvnoder er $\mathrm{O}(1)$ er høyden på treet $\Theta(n)$.
- Hvis antall interne noder med kun én barnenode er $\Theta(n)$ er høyden på treet $\Theta(n)$.
- Et komplett binærtre har $\lfloor n/2 \rfloor$ løvnoder.
- Et komplett binærtre har $\lfloor n/2 \rfloor$ interne noder.
- Et komplett binærtre har høyde $\Theta(\lg n)$.

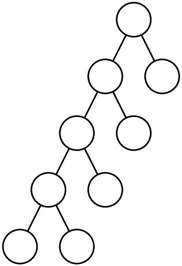
Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis det er flere løvnoder enn det er interne noder i et binærtre er høyden på treet $\Theta(\lg n)$.

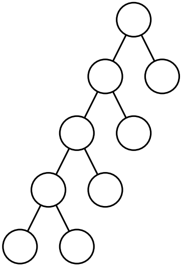
Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis det er flere løvnoder enn det er interne noder i et binærtre er høyden på treet $\Theta(\lg n)$.



Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- ~~Hvis det er flere løvnoder enn det er interne noder i et binærtre er høyden på treet $\Theta(\lg n)$.~~



Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall interne noder med to barn er k og antall løvnoder er ℓ, så er $k + 1 = \ell$.

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall interne noder med to barn er k og antall løvnoder er ℓ, så er $k + 1 = \ell$.

Stemmer for $n = 1$

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall interne noder med to barn er k og antall løvnoder er ℓ, så er $k + 1 = \ell$.

Stemmer for $n = 1$

Antar dette stemmer for en gitt n, med k_n og ℓ_n:
Til løvnode: $k_{n+1} = k_n$, $\ell_{n+1} = \ell_n - 1 + 1 = \ell_n$
Til internnode: $k_{n+1} = k_n + 1$, $\ell_{n+1} = \ell_n + 1$

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall løvnoder er $\mathrm{O}(1)$ er høyden på treet $\Theta(n)$.

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall løvnoder er $\mathrm{O}(1)$ er høyden på treet $\Theta(n)$.

Siden $k + 1 = \ell$, har vi $\mathrm{O}(1)$ interne noder med to barn.

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall løvnoder er $\mathrm{O}(1)$ er høyden på treet $\Theta(n)$.

Siden $k + 1 = \ell$, har vi $\mathrm{O}(1)$ interne noder med to barn.

Får ℓ kjeder med lengde på minst $\frac{n}{\ell} = \Theta(n)$.

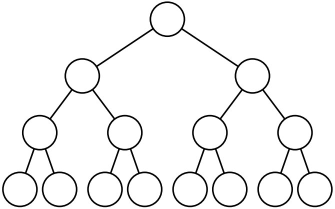
Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Et komplett binærtre har høyde $\Theta(\lg n)$.

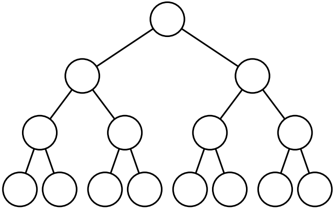
Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Et komplett binærtre har høyde $\Theta(\lg n)$.



Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Et komplett binærtre har høyde $\Theta(\lg n)$.

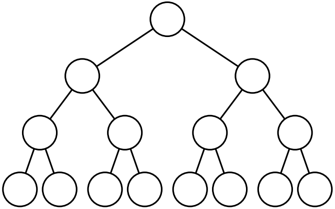


$$\mathrm{H}(n) = \mathrm{H}(n/2) + 1$$

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Et komplett binærtre har høyde $\Theta(\lg n)$.



$$\mathrm{H}(n) = \mathrm{H}(n/2) + 1 = \Theta(\lg n)$$

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

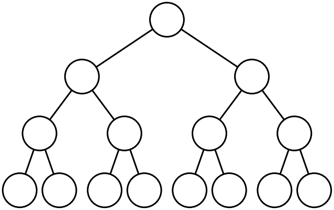
- Hvis antall interne noder med kun én barnenode er $\Theta(n)$ er høyden på treet $\Theta(n)$.

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall interne noder med kun én barnenode er $\Theta(n)$ er høyden på treet $\Theta(n)$.

Høyden på et komplett binærtre er $\Theta(\lg n)$.

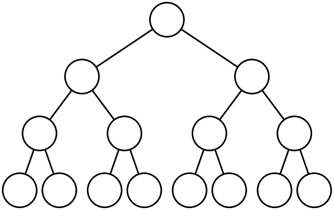


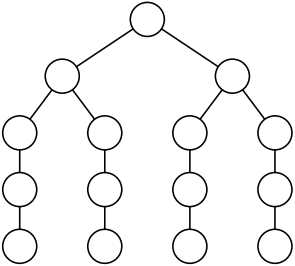
Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Hvis antall interne noder med kun én barnenode er $\Theta(n)$ er høyden på treet $\Theta(n)$.

Høyden på et komplett binærtre er $\Theta(\lg n)$.

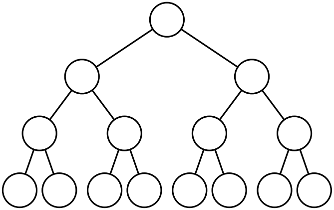


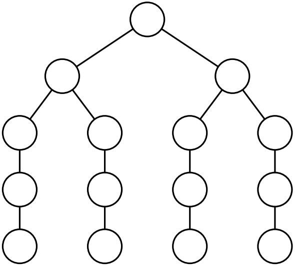


Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- ~~Hvis antall interne noder med kun én barnenode er $\Theta(n)$ er høyden på treet $\Theta(n)$.~~

Høyden på et komplett binærtre er $\Theta(\lg n)$.





Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Et komplett binærtre har $\lfloor n/2 \rfloor$ løvnoder.
- Et komplett binærtre har $\lfloor n/2 \rfloor$ interne noder.

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Et komplett binærtre har $\lfloor n/2 \rfloor$ løvnoder.
- Et komplett binærtre har $\lfloor n/2 \rfloor$ interne noder.

I et komplett binærtre har alle interne noder 2 barn.

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- Et komplett binærtre har $\lfloor n/2 \rfloor$ løvnoder.
- Et komplett binærtre har $\lfloor n/2 \rfloor$ interne noder.

I et komplett binærtre har alle interne noder 2 barn.

Har $k + 1 = \ell$. Det vil si $k = \lfloor n/2 \rfloor$ og $\ell = \lceil n/2 \rceil$.

Forhold i binærtrær

Oppgave 21: Hvilke av følgende påstander stemmer for alle binærtrær med n noder?

- ~~Et komplett binærtre har $\lfloor n/2 \rfloor$ løvnoder.~~
- Et komplett binærtre har $\lfloor n/2 \rfloor$ interne noder.

I et komplett binærtre har alle interne noder 2 barn.

Har $k + 1 = \ell$. Det vil si $k = \lfloor n/2 \rfloor$ og $\ell = \lceil n/2 \rceil$.

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Hvert nivå kan ha 2048 ganger så mange noder som forrige nivå.

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Hvert nivå kan ha 2048 ganger så mange noder som forrige nivå.

$$\mathrm{H}(n) = \mathrm{H}(n/2048) + 1$$

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Hvert nivå kan ha 2048 ganger så mange noder som forrige nivå.

$$\mathrm{H}(n) = \mathrm{H}(n/2048) + 1$$

Ved tilfelle (2) i masterteoremet: $\mathrm{H}(n) = \Theta(\lg n)$.

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Oppgave 23: Hva er den minste mulige høyden på et tre hvor en node med k etterkommernoder kan ha opptil $\lfloor\sqrt{k}\rfloor$ barnenoder?

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Oppgave 23: Hva er den minste mulige høyden på et tre hvor en node med k etterkommernoder kan ha opptil $\lfloor\sqrt{k}\rfloor$ barnenoder?

Ønsker flest mulige barn per nivå.

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Oppgave 23: Hva er den minste mulige høyden på et tre hvor en node med k etterkommernoder kan ha opptil $\lfloor\sqrt{k}\rfloor$ barnenoder?

Ønsker flest mulige barn per nivå.

Rotnoden kan ha $\lfloor\sqrt{n}\rfloor$ barn.

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Oppgave 23: Hva er den minste mulige høyden på et tre hvor en node med k etterkommernoder kan ha opptil $\lfloor\sqrt{k}\rfloor$ barnenoder?

Ønsker flest mulige barn per nivå.

Rotnoden kan ha $\lfloor\sqrt{n}\rfloor$ barn.

Neste nivå $\left\lfloor\sqrt{\frac{n}{\lfloor\sqrt{n}\rfloor}}\right\rfloor \approx \left\lfloor\sqrt{\sqrt{n}}\right\rfloor$.

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Oppgave 23: Hva er den minste mulige høyden på et tre hvor en node med k etterkommernoder kan ha opptil $\lfloor\sqrt{k}\rfloor$ barnenoder?

Ønsker flest mulige barn per nivå.

Rotnoden kan ha $\lfloor\sqrt{n}\rfloor$ barn.

Neste nivå $\left\lfloor\sqrt{\frac{n}{\lfloor\sqrt{n}\rfloor}}\right\rfloor \approx \left\lfloor\sqrt{\sqrt{n}}\right\rfloor$.

Kan generaliseres til rekurrensen $\mathrm{H}(n) = \mathrm{H}(\sqrt{n}) + 1$.

Høyde på ikke-binære trær

Oppgave 22: Hva er den minste høyden et binærtre med n noder kan ha, hvis hver node kan ha 2048 barnenoder?

Oppgave 23: Hva er den minste mulige høyden på et tre hvor en node med k etterkommernoder kan ha opptil $\lfloor\sqrt{k}\rfloor$ barnenoder?

Ønsker flest mulige barn per nivå.

Rotnoden kan ha $\lfloor\sqrt{n}\rfloor$ barn.

Neste nivå $\left\lfloor\sqrt{\frac{n}{\lfloor\sqrt{n}\rfloor}}\right\rfloor \approx \left\lfloor\sqrt{\sqrt{n}}\right\rfloor$.

Kan generaliseres til rekurrensen $\mathrm{H}(n) = \mathrm{H}(\sqrt{n}) + 1$.

$$\mathrm{H}(n) = \Theta(\lg \lg n)$$

Oppgave 24: Har 5 datastrukturer for oppbevaring av og søking i varer med unike identifikasjonsnummer.

1. Binært søketre
2. Balansert binært søketre
3. Hashtabell
4. Usortert dynamisk tabell
5. Dynamisk tabell som sorteres etter hver innsettelse

Hvilke av følgende påstander stemmer?

- Innsetting i 5 må ha en kjøretid på $\Omega(n \lg n)$ i verste tilfelle hvis man bruker sammenligningsbasert sortering.
- 1, 2, 3 og 5 støtter alle søk med kjøretid på $O(\lg n)$ i gjennomsnitt.
- 1, 3 og 4 har alle kjøretid $O(1)$ i beste tilfelle ved innsetting.
- Tre av algoritmene støtter søk med kjøretid på $o(n)$ i verste tilfelle.

Oppgave 24: Har 5 datastrukturer for oppbevaring av og søking i varer med unike identifikasjonsnummer.

1. Binært søketre
2. Balansert binært søketre
3. Hashtabell
4. Usortert dynamisk tabell
5. Dynamisk tabell som sorteres etter hver innsettelse

Hvilke av følgende påstander stemmer?

- ~~Innsetting i 5 må ha en kjøretid på $\Omega(n \lg n)$ i verste tilfelle hvis man bruker sammenligningsbasert sortering.~~
- 1, 2, 3 og 5 støtter alle søk med kjøretid på $O(\lg n)$ i gjennomsnitt.
- 1, 3 og 4 har alle kjøretid $O(1)$ i beste tilfelle ved innsetting.
- Tre av algoritmene støtter søk med kjøretid på $o(n)$ i verste tilfelle.

Oppgave 24: Har 5 datastrukturer for oppbevaring av og søking i varer med unike identifikasjonsnummer.

1. Binært søketre
2. Balansert binært søketre
3. Hashtabell
4. Usortert dynamisk tabell
5. Dynamisk tabell som sorteres etter hver innsettelse

Hvilke av følgende påstander stemmer?

- ~~Innsetting i 5 må ha en kjøretid på $\Omega(n \lg n)$ i verste tilfelle hvis man bruker sammenligningsbasert sortering.~~
- 1, 2, 3 og 5 støtter alle søk med kjøretid på $\mathrm{O}(\lg n)$ i gjennomsnitt.
- 1, 3 og 4 har alle kjøretid $\mathrm{O}(1)$ i beste tilfelle ved innsetting.
- ~~Tre av algoritmene støtter søk med kjøretid på $o(n)$ i verste tilfelle.~~

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

Hvilke påstander stemmer?

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- MERGE-SORT passer bedre enn INSERTION-SORT i denne situasjonen, siden MERGE-SORT har bedre gjennomsnittlig kjøretid.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- ~~MERGE-SORT passer bedre enn INSERTION-SORT i denne situasjonen, siden MERGE-SORT har bedre gjennomsnittlig kjøretid.~~

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Tre av de fire algoritmene bruker $\mathrm{O}(1)$ tid på en oppdatering i beste tilfelle.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- ~~Tre av de fire algoritmene bruker $O(1)$ tid på en oppdatering i beste tilfelle.~~

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med Merge-Sort.
2. Bruk en tabell og sorter med Insertion-Sort.
3. Bruk en maks-haug og kjør Heap-Increase-Key.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Alle algoritmene bruker $\mathrm{O}(n)$ tid på en oppdatering hvis $k = \mathrm{O}(n)$.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- ~~Alle algoritmene bruker $\mathrm{O}(n)$ tid på en oppdatering hvis $k = \mathrm{O}(n)$.~~

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med Merge-Sort.
2. Bruk en tabell og sorter med Insertion-Sort.
3. Bruk en maks-haug og kjør Heap-Increase-Key.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Alle algoritmene bruker $\mathrm{O}(n)$ minne uansett hva k er.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med Merge-Sort.
2. Bruk en tabell og sorter med Insertion-Sort.
3. Bruk en maks-haug og kjør Heap-Increase-Key.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Alle algoritmene bruker $\mathrm{O}(n)$ minne uansett hva k er.

Algoritme 4 har til slutt $n(k+1)$ noder.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med $\textsc{Merge-Sort}$.
2. Bruk en tabell og sorter med $\textsc{Insertion-Sort}$.
3. Bruk en maks-haug og kjør $\textsc{Heap-Increase-Key}$.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- ~~Alle algoritmene bruker $\mathrm{O}(n)$ minne uansett hva k er.~~

Algoritme 4 har til slutt $n(k+1)$ noder.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Kjøretiden til en oppdatering med algoritme 4 er $\Theta(\lg(n^k))$ i verste tilfelle.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Kjøretiden til en oppdatering med algoritme 4 er $\Theta(\lg(n^k))$ i verste tilfelle.

Innsetting i en maks-haug tar $\mathrm{O}(\mathrm{n})$, når n er antall noder.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med $\textsc{Merge-Sort}$.
2. Bruk en tabell og sorter med $\textsc{Insertion-Sort}$.
3. Bruk en maks-haug og kjør $\textsc{Heap-Increase-Key}$.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Kjøretiden til en oppdatering med algoritme 4 er $\Theta(\lg(n^k))$ i verste tilfelle.

Innsetting i en maks-haug tar $\mathrm{O}(\mathrm{n})$, når n er antall noder.

Det vil si $\Theta(\lg(n(k+1)))) = \Theta(\lg nk)$ i verste tilfelle.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres *k* ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- ~~Kjøretiden til en oppdatering med algoritme 4 er $\Theta(\lg(n^k))$ i verste tilfelle.~~

Innsetting i en maks-haug tar $\mathrm{O}(\mathrm{n})$, når *n* er antall noder.

Det vil si $\Theta(\lg(n(k+1))) = \Theta(\lg nk)$ i verste tilfelle.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med Merge-Sort.
2. Bruk en tabell og sorter med Insertion-Sort.
3. Bruk en maks-haug og kjør Heap-Increase-Key.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Algoritme 4 bruker like lang tid i verste tilfelle for en oppdatering enten $k = n!$ eller $k = n^n$.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med Merge-Sort.
2. Bruk en tabell og sorter med Insertion-Sort.
3. Bruk en maks-haug og kjør Heap-Increase-Key.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Algoritme 4 bruker like lang tid i verste tilfelle for en oppdatering enten $k = n!$ eller $k = n^n$.

$$\Theta(\lg(n \cdot n^n)) = \Theta(\lg n^{n+1}) = \Theta((n+1)\lg n) = \Theta(n \lg n).$$

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med Merge-Sort.
2. Bruk en tabell og sorter med Insertion-Sort.
3. Bruk en maks-haug og kjør Heap-Increase-Key.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Algoritme 4 bruker like lang tid i verste tilfelle for en oppdatering enten $k = n!$ eller $k = n^n$.

$\Theta(\lg(n \cdot n^n)) = \Theta(\lg n^{n+1}) = \Theta((n+1)\lg n) = \Theta(n \lg n)$.
$\Theta(\lg(n \cdot n!)) = \Theta(\lg n!) = \Theta(n \lg n)$.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres *k* ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Algoritme 3 er like rask som algoritme 4 på en oppdatering uansett hva *k* er.

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres *k* ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- ~~Algoritme 3 er like rask som algoritme 4 på en oppdatering uansett hva *k* er.~~

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Hvis $k = \Theta(n^3)$ bruker algoritme 3 og 4 like lang tid på en oppdatering i verste tilfelle.

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Hvis $k = \Theta(n^3)$ bruker algoritme 3 og 4 like lang tid på en oppdatering i verste tilfelle.

Algoritme 3: $\Theta(\lg n)$

Turnering

Oppgave 25: Skal holde styr på poengsummen til deltagere. Poengsummen kan kun øke, og vi ønsker å kunne hente ut den beste poengsummen. Hver poengsum oppdateres k ganger.

1. Bruk en tabell og sorter med MERGE-SORT.
2. Bruk en tabell og sorter med INSERTION-SORT.
3. Bruk en maks-haug og kjør HEAP-INCREASE-KEY.
4. Bruk en maks-haug og sett inn en ny node med poengsummen etter oppdatering.

- Hvis $k = \Theta(n^3)$ bruker algoritme 3 og 4 like lang tid på en oppdatering i verste tilfelle.

Algoritme 3: $\Theta(\lg n)$
Algoritme 4: $\Theta(\lg(n \cdot n^3)) = \Theta(\lg n^4) = \Theta(4 \lg n) = \Theta(\lg n)$