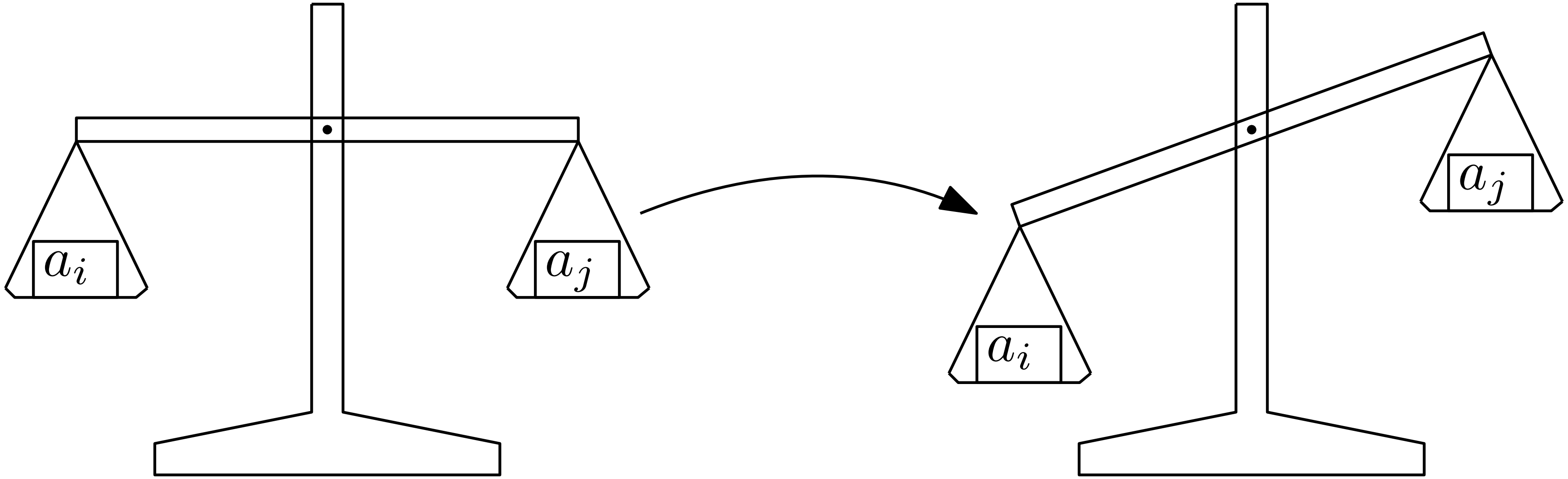
Nedre grænser for sortering og søgning



Mikkel Abrahamsen

Sammenligningssortering

Fokus: Algoritmer til at sortere $[a_1, \ldots, a_n]$, alle tal forskellige.

Begrænsning: Algoritmerne må kun bestemme sorteret rækkefølge vha. sammenligninger. Eksempler: Insertion, merge og heap sort.

Sammenligningssortering

Fokus: Algoritmer til at sortere $[a_1, \ldots, a_n]$, alle tal forskellige.

Begrænsning: Algoritmerne må kun bestemme sorteret rækkefølge vha. sammenligninger. Eksempler: Insertion, merge og heap sort.

Ikke tilladt: Kig på bits, tæl hvor mange af hver, læg tal sammen, osv.

Vi kender ikke *universet* som tallene kommer fra.

Ikke-eksempler: counting, radix og bucket sort.

Sammenligningssortering

Fokus: Algoritmer til at sortere $[a_1, \ldots, a_n]$, alle tal forskellige.

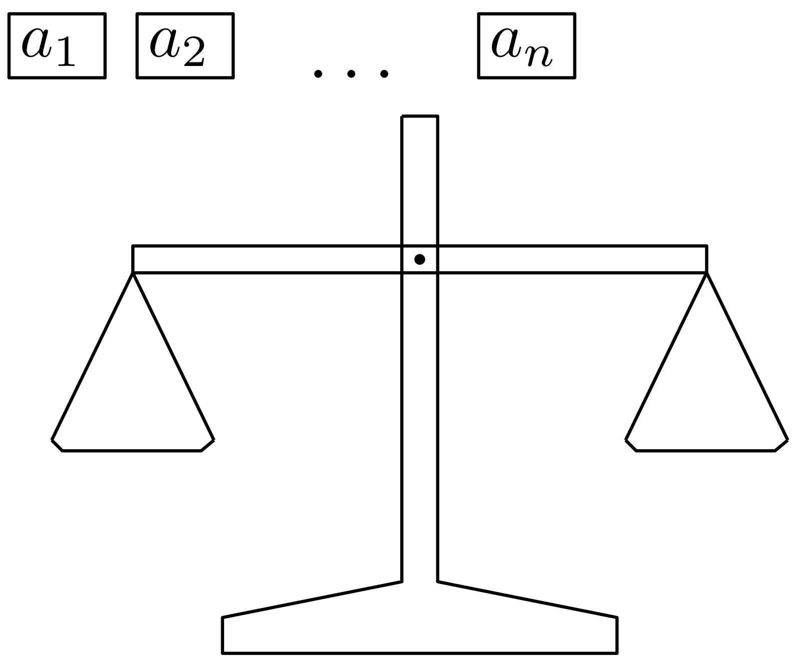
Begrænsning: Algoritmerne må kun bestemme sorteret rækkefølge vha. sammenligninger. Eksempler: Insertion, merge og heap sort.

Ikke tilladt: Kig på bits, tæl hvor mange af hver, læg tal sammen, osv.

Vi kender ikke *universet* som tallene kommer fra.

Ikke-eksempler: counting, radix og bucket sort.

Algoritmen har ikke direkte adgang til $a_1, \ldots, a_n$, men kan spørge: $a_i < a_j$?

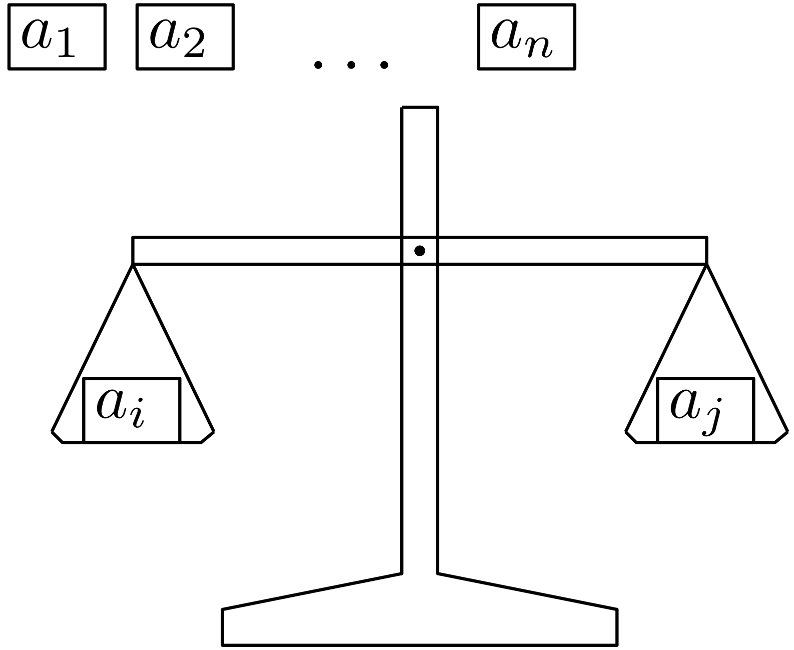


Sammenligningssortering

Fokus: Algoritmer til at sortere $[a_1, \ldots, a_n]$, alle tal forskellige.

Begrænsning: Algoritmerne må kun bestemme sorteret rækkefølge vha. sammenligninger. Eksempler: Insertion, merge og heap sort.
Ikke tilladt: Kig på bits, tæl hvor mange af hver, læg tal sammen, osv.
Vi kender ikke *universet* som tallene kommer fra.
Ikke-eksempler: counting, radix og bucket sort.

Algoritmen har ikke direkte adgang til $a_1, \ldots, a_n$, men kan spørge: $a_i < a_j$?



Sammenligningssortering

Fokus: Algoritmer til at sortere $[a_1, \ldots, a_n]$, alle tal forskellige.

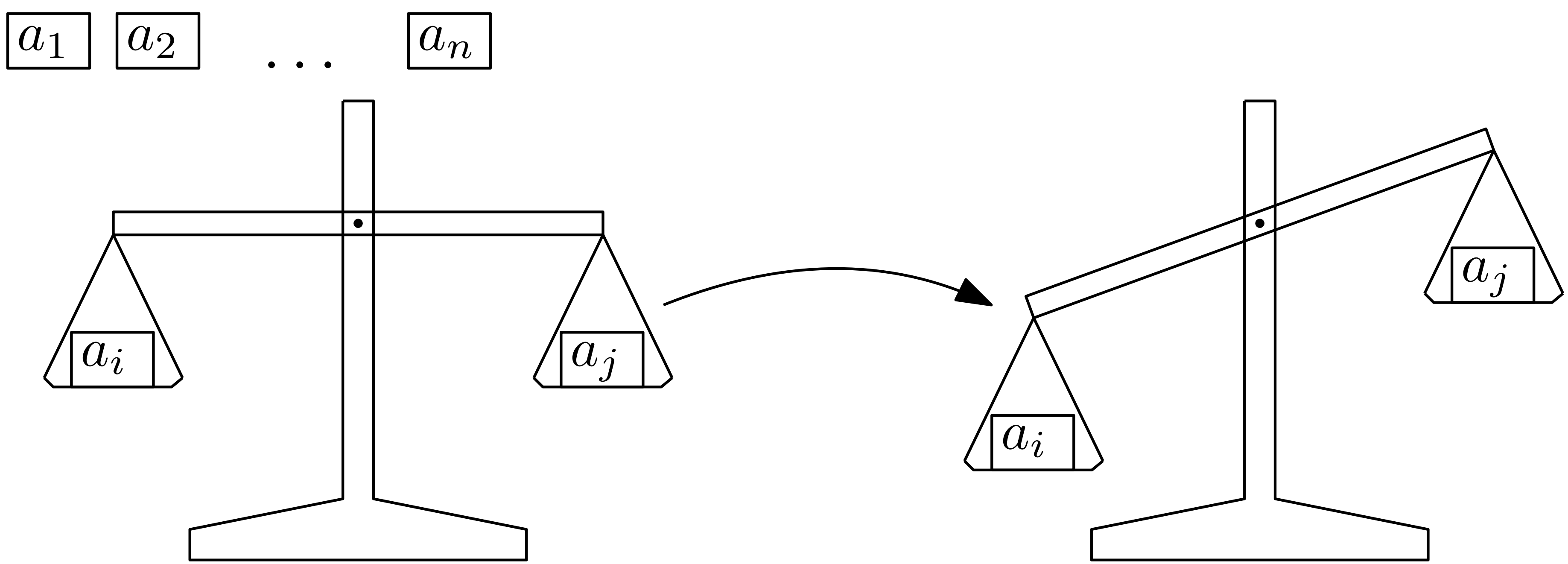
Begrænsning: Algoritmerne må kun bestemme sorteret rækkefølge vha. sammenligninger. Eksempler: Insertion, merge og heap sort.

Ikke tilladt: Kig på bits, tæl hvor mange af hver, læg tal sammen, osv.

Vi kender ikke *universet* som tallene kommer fra.

Ikke-eksempler: counting, radix og bucket sort.

Algoritmen har ikke direkte adgang til $a_1, \ldots, a_n$, men kan spørge: $a_i < a_j$?

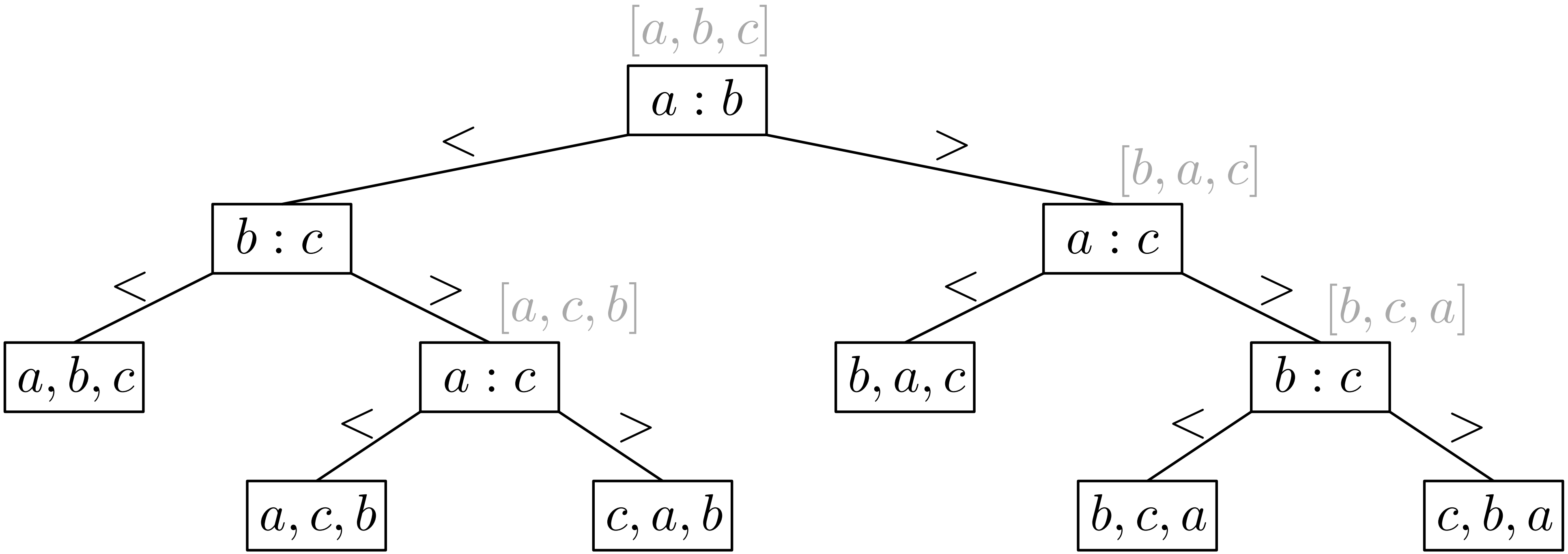


Konklusion: $a_i > a_j$!

Beslutningstræ

Algoritmens opførsel kan beskrives med et *beslutningstræ*.

Eksempel: Insertion-Sort($[a, b, c]$)



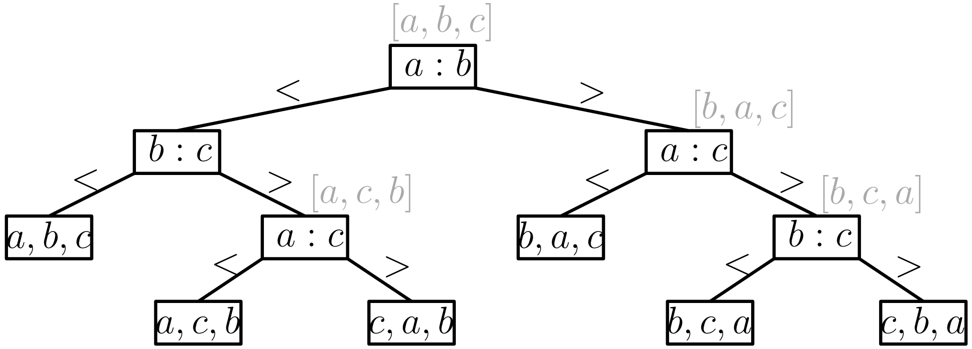
Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.

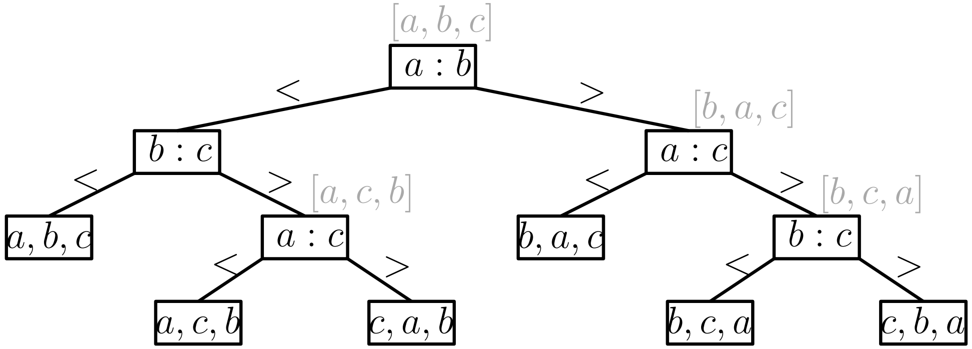


Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.

$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$



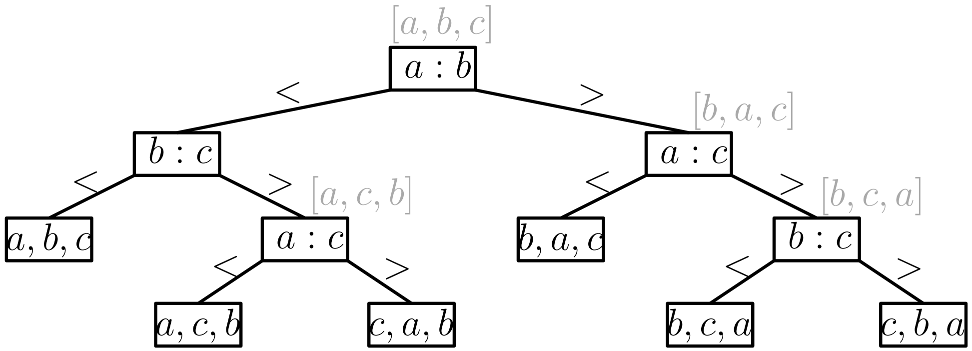
Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.

$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$

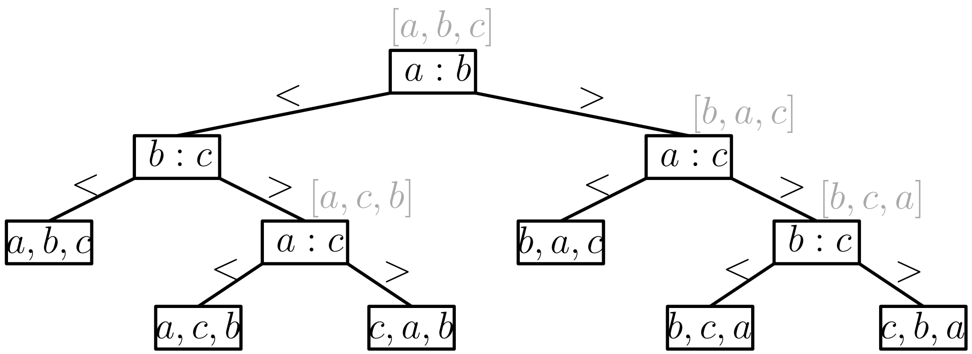
$\#$blade $\geq \#$perm. $= n!$



Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

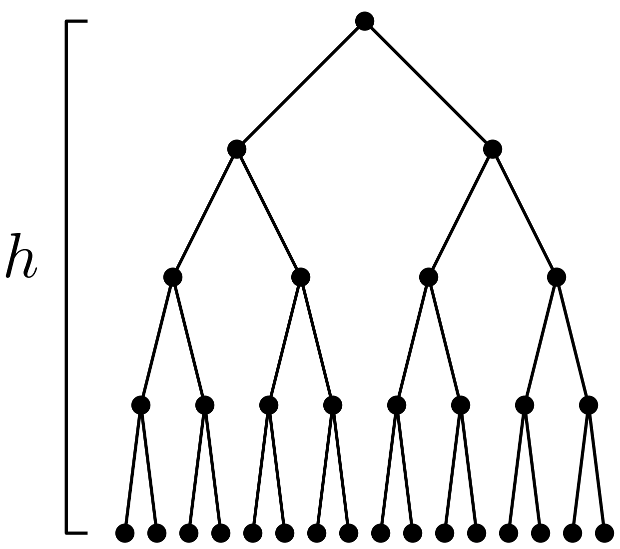
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#$blade $\geq$ $\#$perm. $= n!$

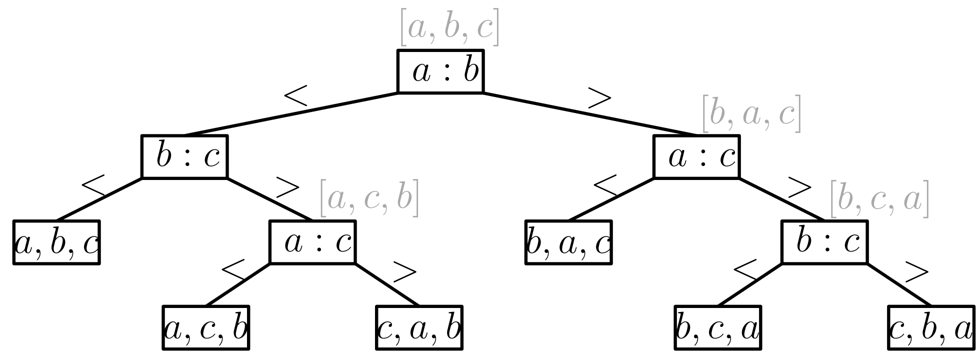
$T(n) \geq h$, h højden af beslutningstræ.



Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

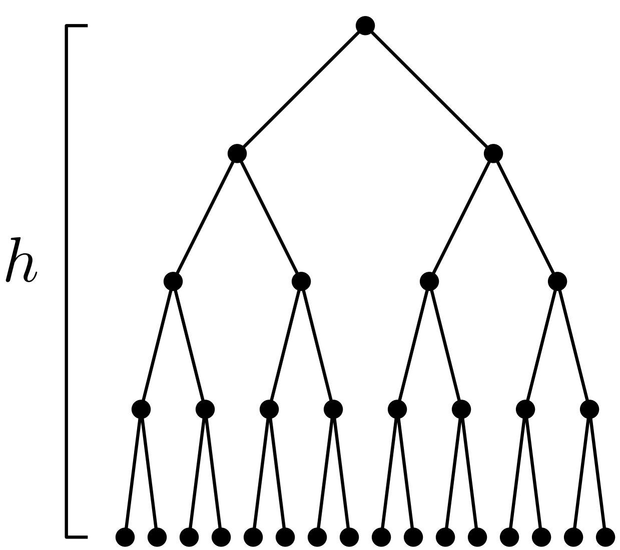
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#\text{perm.} = n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#\text{blade} \geq \#\text{perm.} = n!$

$T(n) \geq h$, h højden af beslutningstræ.

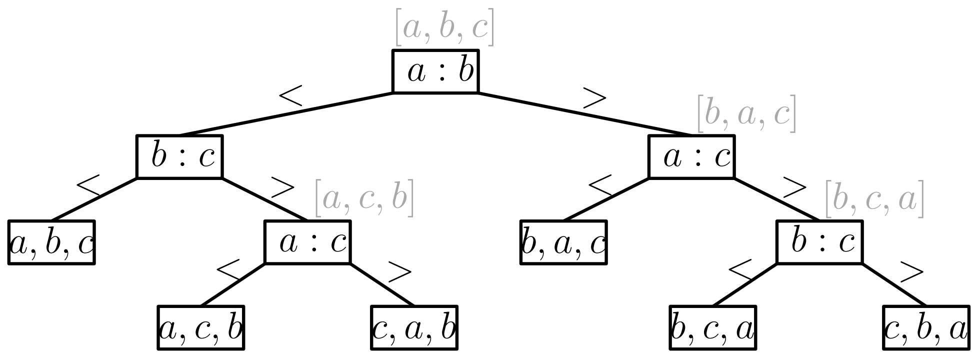


$2^h \geq \#\text{blade}$

Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

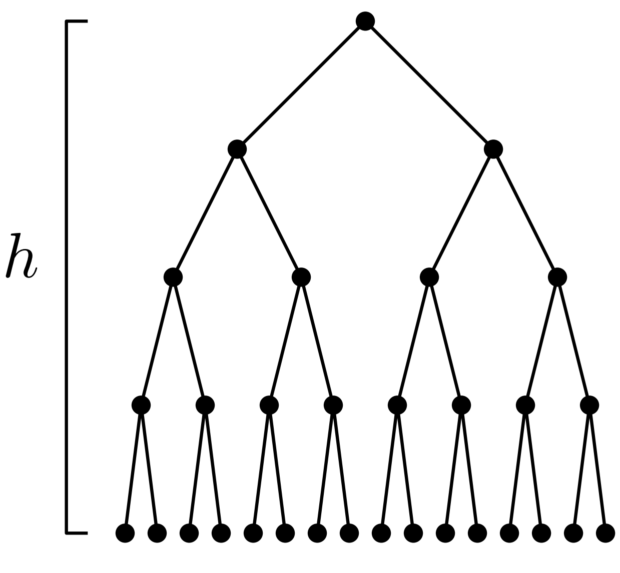
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#$blade $\geq \#$perm. $= n!$

$T(n) \geq h$, h højden af beslutningstræ.



$2^h \geq \#$blade

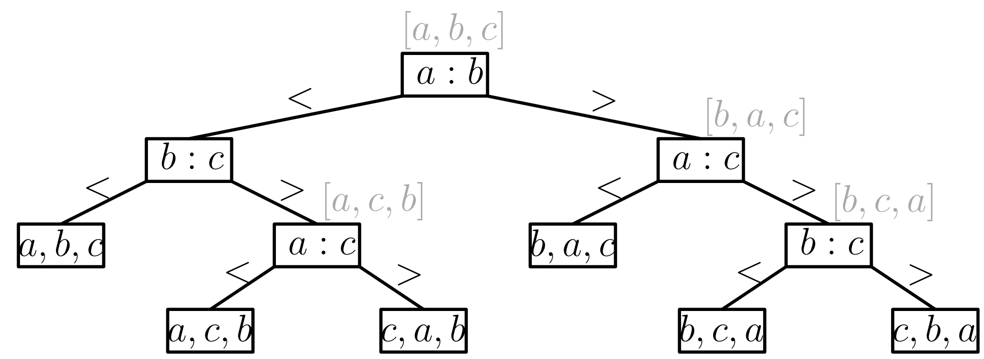
$2^h \geq \#$blade $\geq n! \Longrightarrow$

$h \geq \lg(n!) = \lg n + \lg(n-1) + \ldots + \lg 2 + \lg 1$

Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

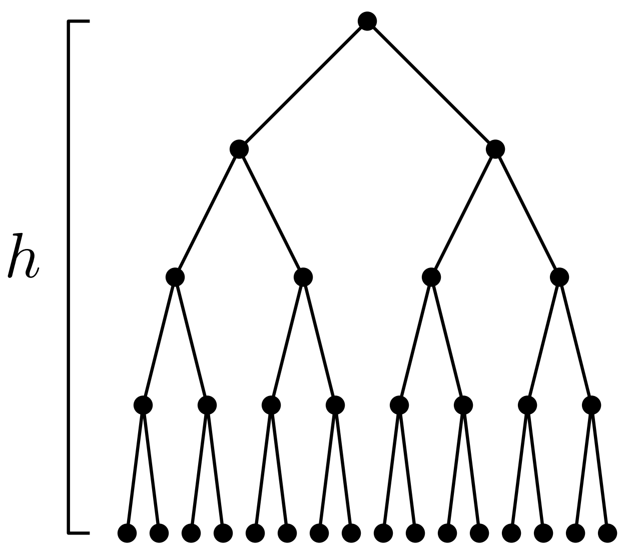
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#$blade $\geq \#$perm. $= n!$

$T(n) \geq h$, h højden af beslutningstræ.



$2^h \geq \#$blade

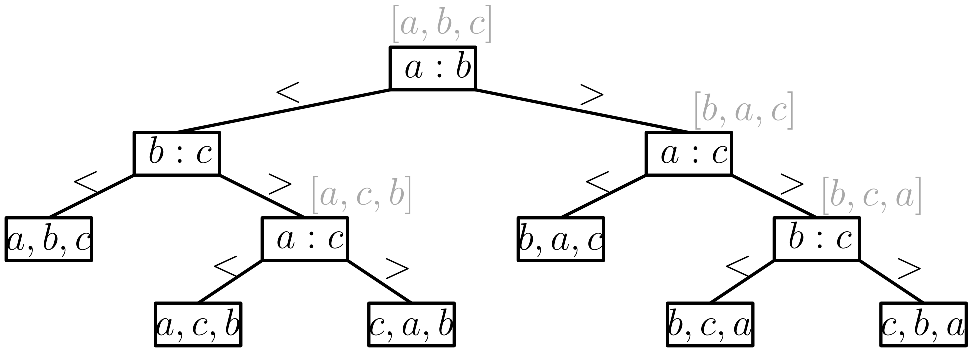
$2^h \geq \#$blade $\geq n! \Longrightarrow$

$$h \geq \lg(n!) = \lg n + \lg(n-1) + \ldots + \lg 2 + \lg 1$$
$$\geq \lg n + \lg(n-1) + \ldots + \lg \tfrac{n}{2}$$

Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

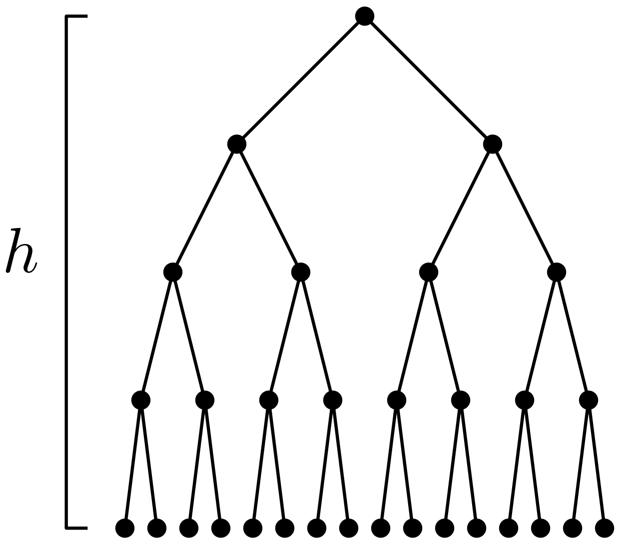
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#$blade $\geq \#$perm. $= n!$

$T(n) \geq h$, h højden af beslutningstræ.



$2^h \geq \#$blade

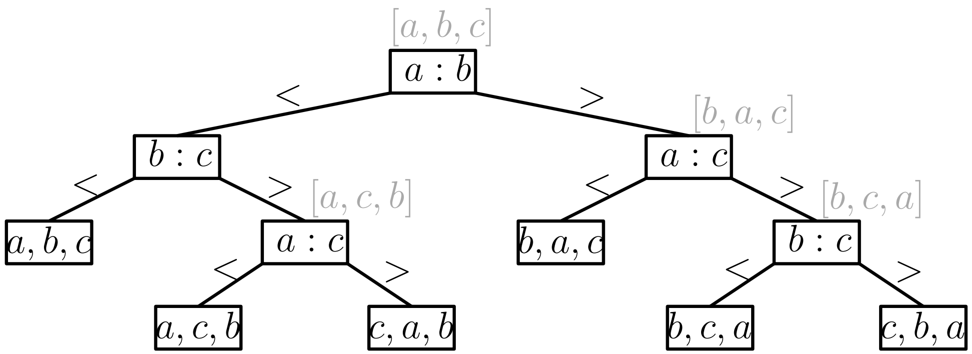
$2^h \geq \#$blade $\geq n! \Longrightarrow$

$$h \geq \lg(n!) = \lg n + \lg(n-1) + \ldots + \lg 2 + \lg 1$$
$$\geq \lg n + \lg(n-1) + \ldots + \lg \tfrac{n}{2}$$
$$\geq \lg \tfrac{n}{2} + \lg \tfrac{n}{2} + \ldots + \lg \tfrac{n}{2}$$
$$= \tfrac{n}{2} \lg \tfrac{n}{2} = \tfrac{n}{2}(\lg n - 1) = \tfrac{n}{2} \lg n - \tfrac{n}{2}$$

Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

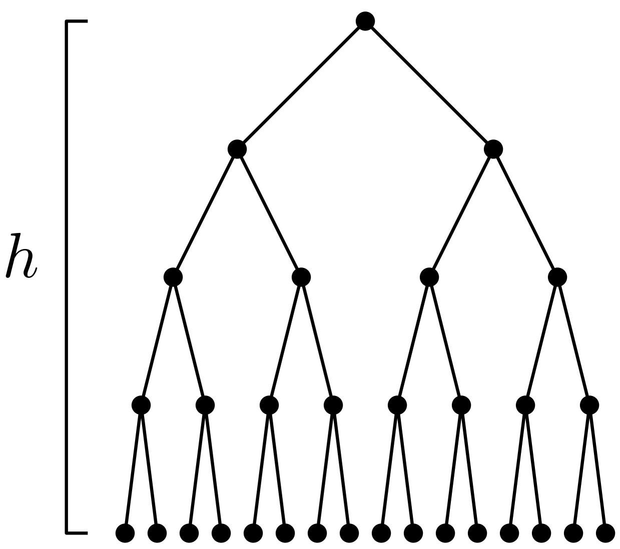
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#$blade $\geq \#$perm. $= n!$

$T(n) \geq h$, h højden af beslutningstræ.



$$2^h \geq \#\text{blade}$$

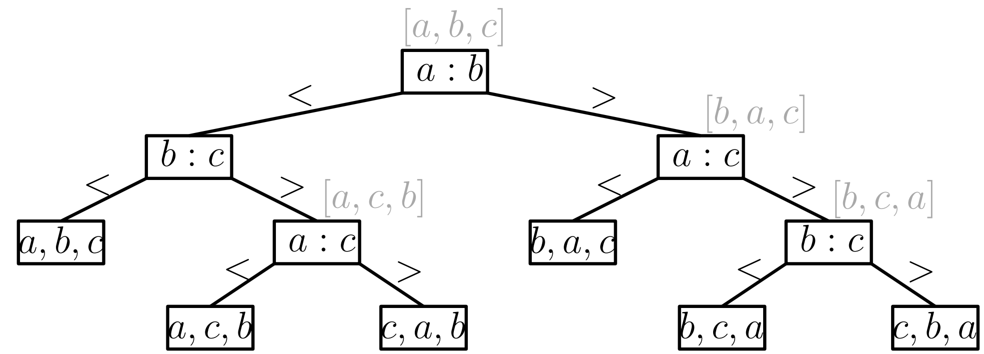
$$2^h \geq \#\text{blade} \geq n! \Longrightarrow$$

$$\begin{aligned}
h &\geq \lg(n!) = \lg n + \lg(n-1) + \ldots + \lg 2 + \lg 1 \\
&\geq \lg n + \lg(n-1) + \ldots + \lg \tfrac{n}{2} \\
&\geq \lg \tfrac{n}{2} + \lg \tfrac{n}{2} + \ldots + \lg \tfrac{n}{2} \\
&= \tfrac{n}{2} \lg \tfrac{n}{2} = \tfrac{n}{2}(\lg n - 1) = \tfrac{n}{2} \lg n - \tfrac{n}{2} \\
&\geq \tfrac{n}{2} \lg n - \tfrac{n}{2} \tfrac{\lg n}{2} = \tfrac{n}{2} \lg n - \tfrac{n}{4} \lg n = \tfrac{1}{4} n \ln n
\end{aligned}$$

Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

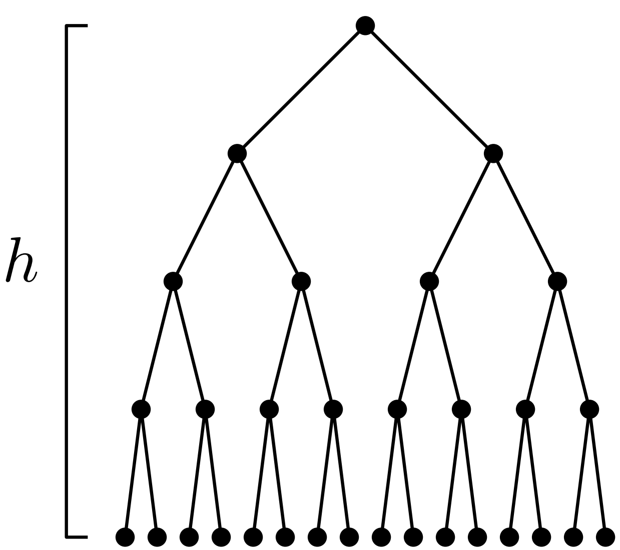
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#$perm. $= n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#$blade $\geq \#$perm. $= n!$

$T(n) \geq h$, h højden af beslutningstræ.



$2^h \geq \#$blade

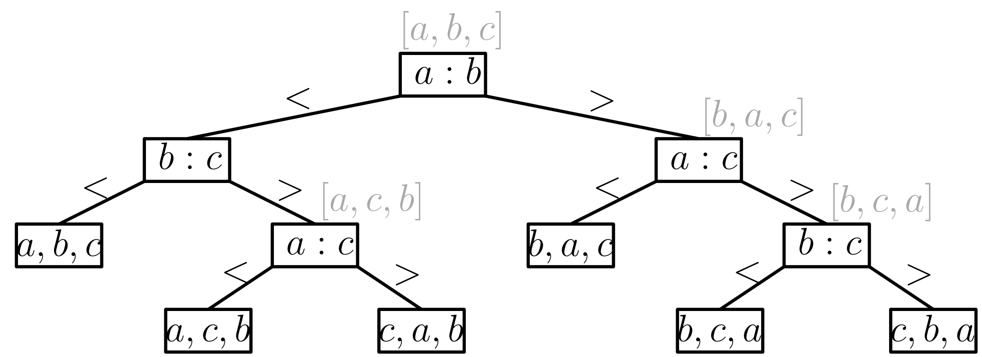
$2^h \geq \#$blade $\geq n! \Longrightarrow$

$$\begin{aligned}
h &\geq \lg(n!) = \lg n + \lg(n-1) + \ldots + \lg 2 + \lg 1 \\
&\geq \lg n + \lg(n-1) + \ldots + \lg \tfrac{n}{2} \\
&\geq \lg \tfrac{n}{2} + \lg \tfrac{n}{2} + \ldots + \lg \tfrac{n}{2} \\
&= \tfrac{n}{2} \lg \tfrac{n}{2} = \tfrac{n}{2}(\lg n - 1) = \tfrac{n}{2} \lg n - \tfrac{n}{2} \\
&\geq \tfrac{n}{2} \lg n - \tfrac{n}{2} \tfrac{\lg n}{2} = \tfrac{n}{2} \lg n - \tfrac{n}{4} \lg n = \tfrac{1}{4} n \ln n \\
&= \Omega(n \log n).
\end{aligned}$$

Resultat

Sætning: Enhver sammenligningsbaseret algoritme til sortering bruger $\Omega(n \log n)$ tid i værste fald. Dvs. $T(n) \geq c \cdot n \log n$ for konstant $c > 0$.

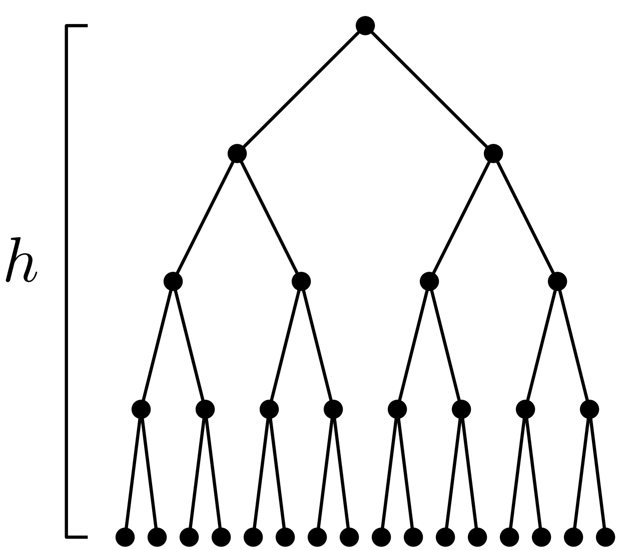
Bevis: Enhver permutation (rækkefølge) af $a_1, \ldots, a_n$ *skal* forekomme i et blad.



$\#\text{perm.} = n \cdot (n-1) \cdots 2 \cdot 1 = n!$

$\#\text{blade} \geq \#\text{perm.} = n!$

$T(n) \geq h$, h højden af beslutningstræ.



$$2^h \geq \#\text{blade}$$

$$2^h \geq \#\text{blade} \geq n! \Longrightarrow$$

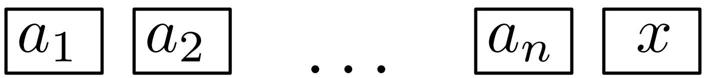
$$\begin{aligned}
h &\geq \lg(n!) = \lg n + \lg(n-1) + \ldots + \lg 2 + \lg 1 \\
&\geq \lg n + \lg(n-1) + \ldots + \lg \tfrac{n}{2} \\
&\geq \lg \tfrac{n}{2} + \lg \tfrac{n}{2} + \ldots + \lg \tfrac{n}{2} \\
&= \tfrac{n}{2} \lg \tfrac{n}{2} = \tfrac{n}{2}(\lg n - 1) = \tfrac{n}{2} \lg n - \tfrac{n}{2} \\
&\geq \tfrac{n}{2} \lg n - \tfrac{n}{2} \tfrac{\lg n}{2} = \tfrac{n}{2} \lg n - \tfrac{n}{4} \lg n = \tfrac{1}{4} n \ln n \\
&= \Omega(n \log n).
\end{aligned}$$

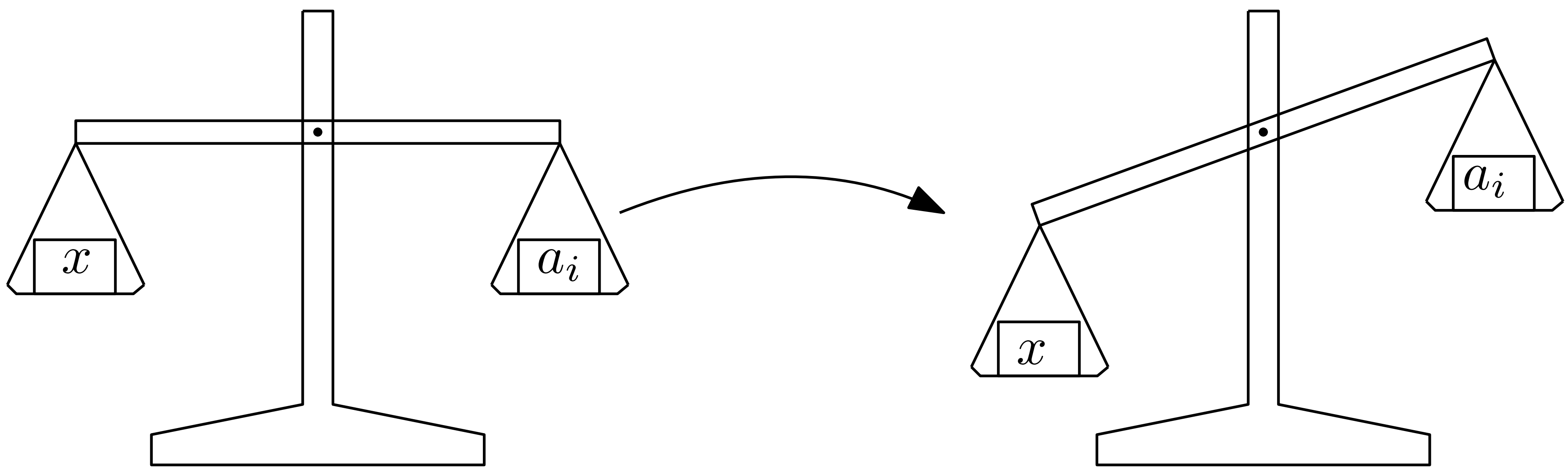
Konsekvens: Merge og heap sort er asymptotisk optimale.

Søgning efter efterfølger

Input: Sorteret array $[a_1, \ldots, a_n]$ og x.
Output: Mindste a_i så $x \leq a_i$ (eller ∞ hvis $x > a_n$).

Algoritmen har ikke direkte adgang til $a_1, \ldots, a_n$ og x, men kan spørge: $x \leq a_i$?



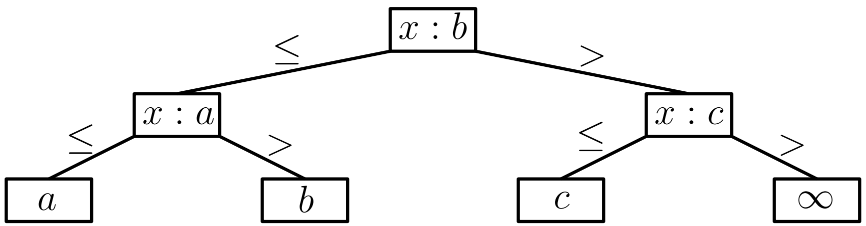


Konklusion: $x > a_i$!

Nedre grænse for søgning

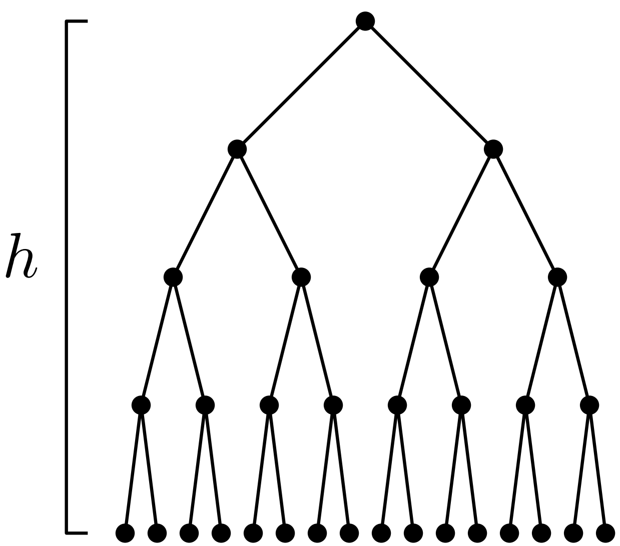
Sætning: Enhver sammenligningsbaseret algoritme til søgning laver mindst $\lg n$ sammenligninger i værste fald, så $T(n) = \Omega(\log n)$.

Bevis: Alle tal $\{a_1, a_2, \ldots, a_n\}$ *skal* forekomme som blade.



#blade $\geq n$

$T(n) \geq h$, h højden af beslutningstræ.



$2^h \geq$ #blade $\geq n \Longrightarrow h \geq \lg n$

Konsekvens: Binær søgning er asymptotisk optimal.