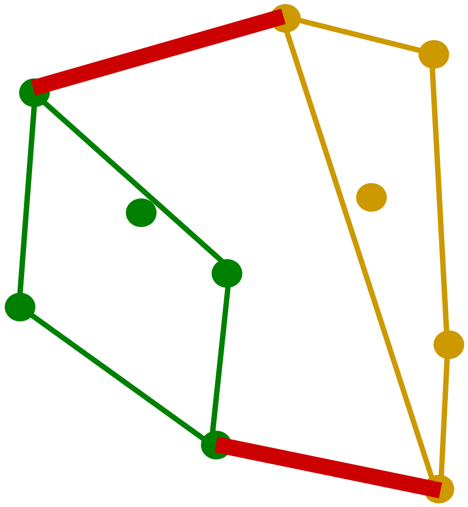
CMPS 3130/6130: Computational Geometry
Spring 2015



Convex Hulls

Carola Wenk

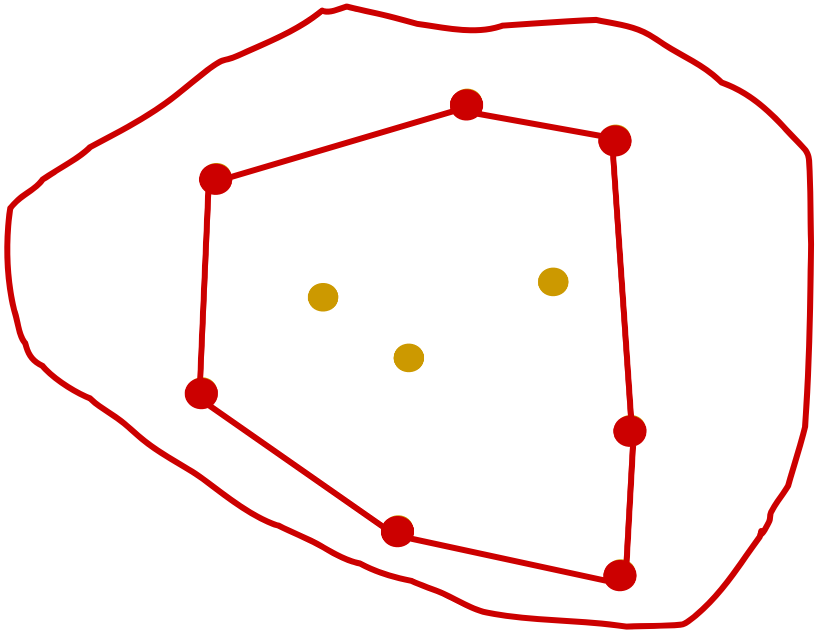
Convex Hull Problem

- Given a set of pins on a pinboard

  and a rubber band around them.

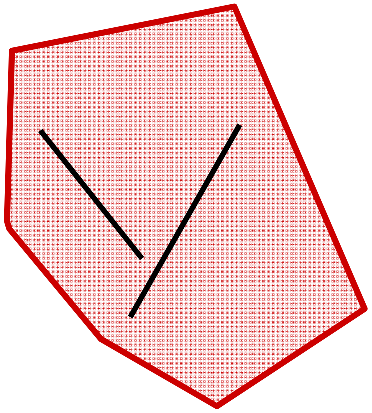
  How does the rubber band look when it snaps tight?

- The convex hull of a point set is one of the simplest shape approximations for a set of points.

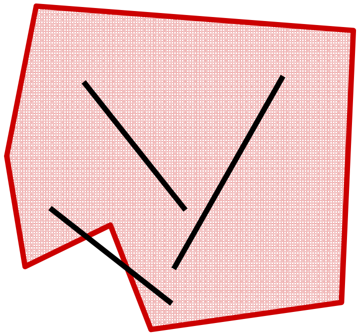


Convexity

- A set $C \subseteq \mathbf{R}^2$ is *convex* if for every two points $p,q \in C$ the line segment $\overline{pq}$ is fully contained in C.



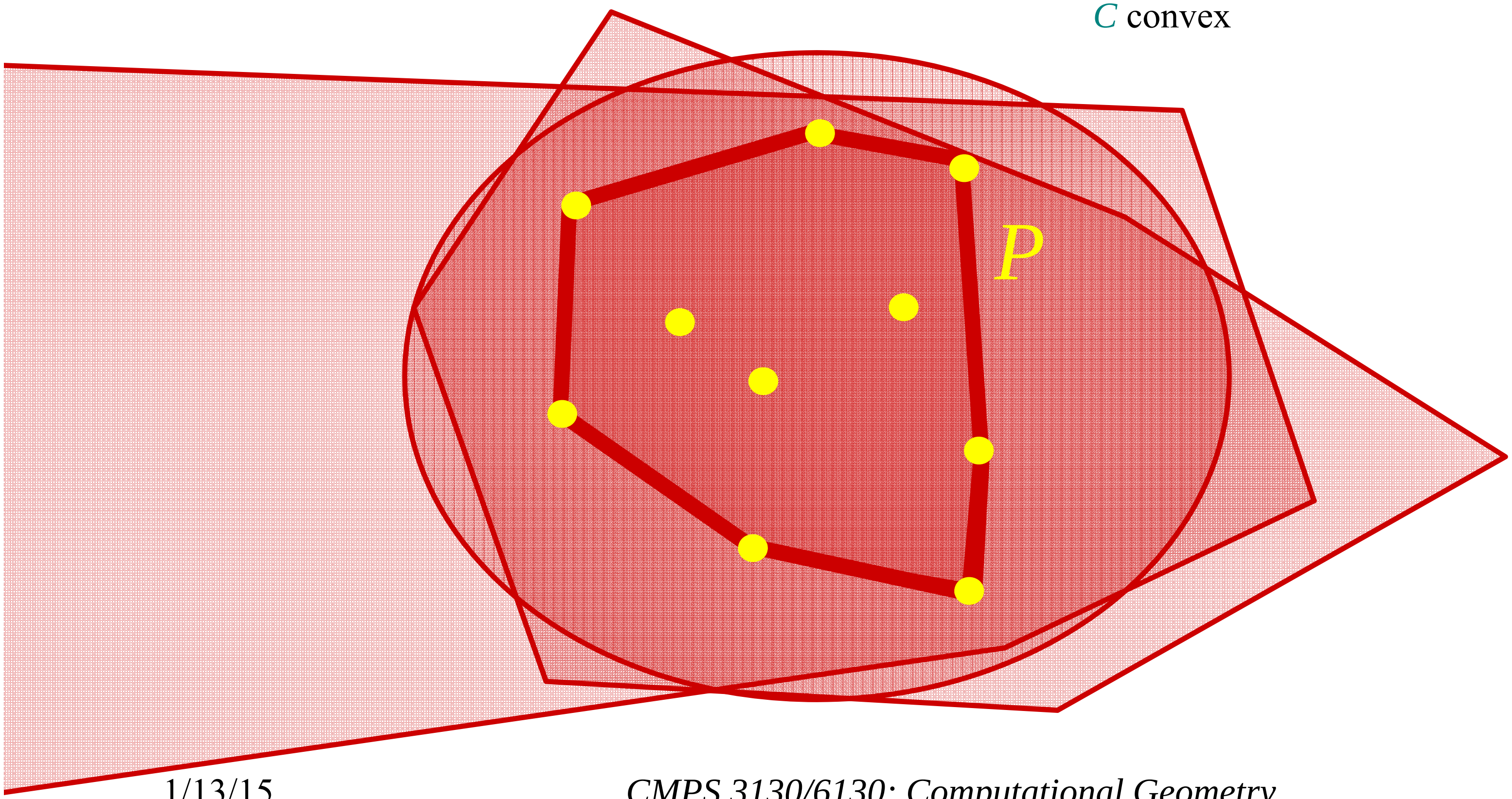
convex



non-convex

Convex Hull

- The convex hull *CH(P)* of a point set $P \subseteq \mathbf{R}^2$ is the smallest convex set $C \supseteq P$. In other words $\mathrm{CH}(P) = \bigcap_{\substack{C \supseteq P \\ C \text{ convex}}} \mathrm{C}$.

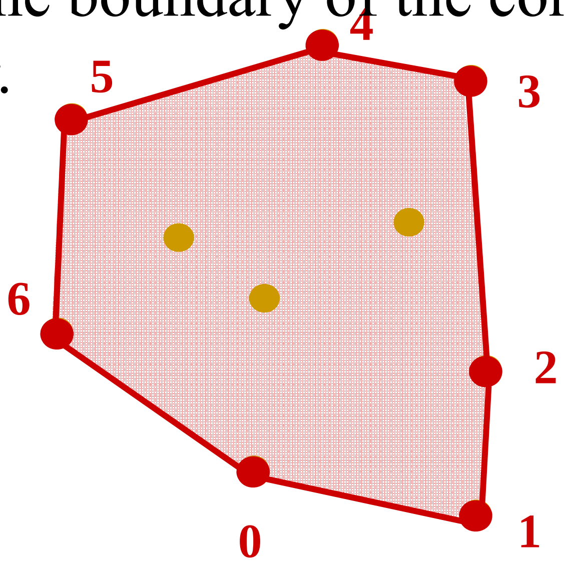


Convex Hull

- **Observation:** CH(P) is the unique convex polygon whose vertices are points of P and which contains all points of P.
- **Goal:** Compute CH(P).

What does that mean? How do we represent/store CH(P)?

⇒ Represent the convex hull as the sequence of points on the convex hull polygon (the boundary of the convex hull), in counter-clockwise order.

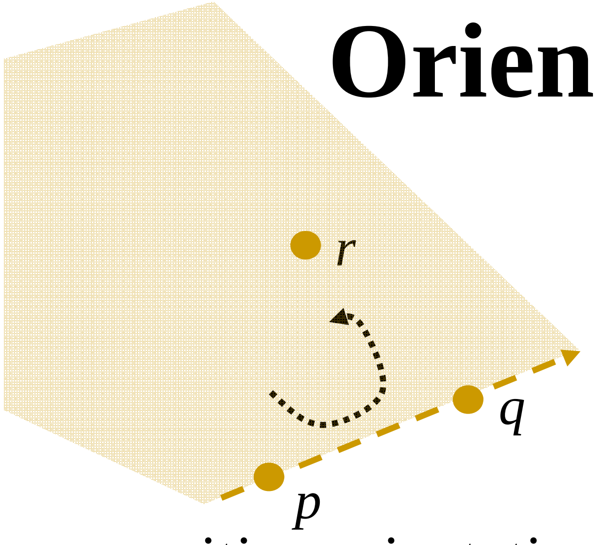


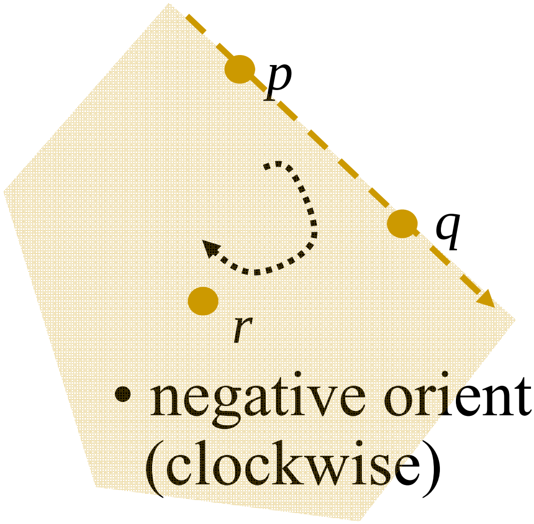
A First Try

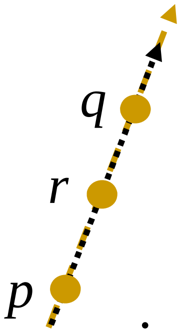
```
Algorithm SLOW_CH($P$):
/* CH(P) = Intersection of all half-planes that are defined by the directed line through
    ordered pairs of points in P and that have all remaining points of P on their left */
Input: Point set $P \subseteq \mathbf{R}^2$
Output: A list $L$ of vertices describing the CH($P$) in counter-clockwise order
$E := \emptyset$
for all $(p,q) \in P \times P$ with $p \neq q$   // ordered pair
    valid := true
    for all $r \in P$, $r \neq p$ and $r \neq q$
        if $r$ lies to the right of directed line through $p$ and $q$   // takes constant time
            valid := false
    if valid then
        $E := E \cup \overrightarrow{pq}$   // directed edge
Construct from $E$ sorted list $L$ of vertices of CH($P$) in counter-clockwise order
```

- Runtime: $O(n^3)$, where $n = |P|$
- How to test that a point lies to the right of a directed line?

Orientation Test / Halfplane Test







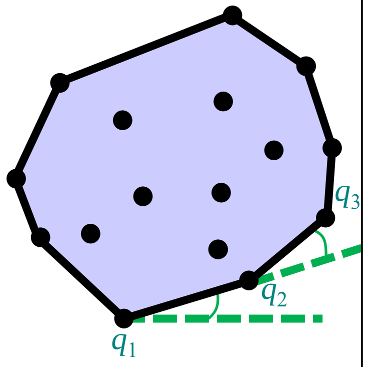
- positive orientation (counter-clockwise)
- *r* lies to the left of *pq*

- negative orientation (clockwise)
- $\overrightarrow{\text{r}}$ lies to the right of $\overrightarrow{\text{pq}}$

- zero orientation
- r lies on the line $\overrightarrow{\text{pq}}$

- Orient(p,q,r) = sign det $\begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix}$,where $p = (p_x,p_y)$
- Can be computed in constant time

Jarvis’ March (Gift Wrapping)



```
**Algorithm** Giftwrapping_CH($P$):
// Compute CH(P) by incrementally inserting points from left to right
**Input:** Point set $P \subseteq \mathbf{R}^2$
**Output:** List $q_1, q_2, \ldots$ of vertices in counter-clockwise order around CH($P$)
$q_1$ = point in $P$ with smallest $y$ (if ties, with smallest $x$)
$q_2$ = point in $P$ with smallest angle to horizontal line through $q_1$
$i = 2$
do {
    $i$++
    $q_i$ = point with smallest angle to line through $q_{i-2}$ and $q_{i-1}$
} while $q_i \neq q_1$
```

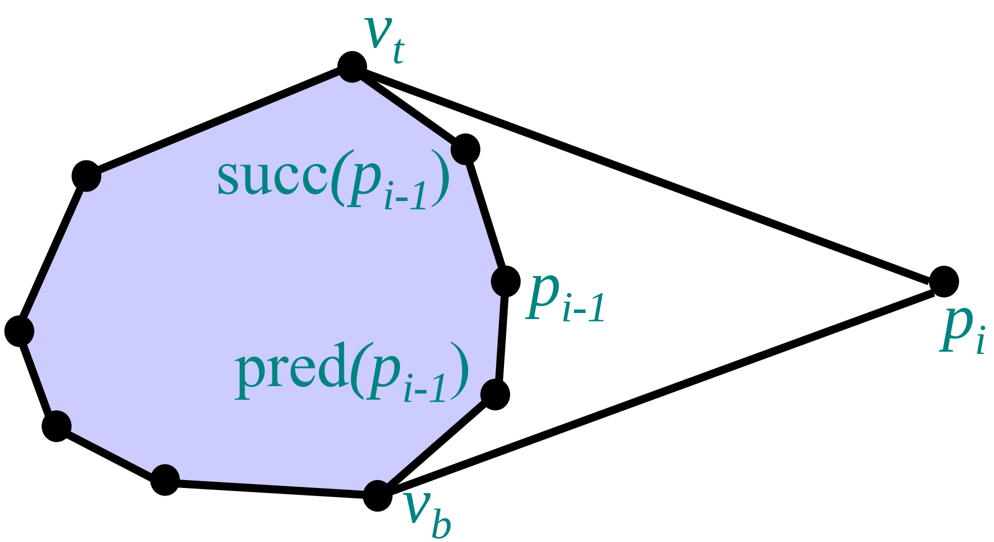
- Runtime: O(hn) , where $n = |P|$ and h = #points on CH(P)
- Output-sensitive algorithm

Incremental Insertion

```
                  Algorithm Incremental_CH($P$):
                  // Compute CH(P) by incrementally inserting points from left to right
                  Input: Point set $P \subseteq \mathbf{R}^2$
                  Output: $C$=CH($P$), described as a list of vertices in counter-clockwise order
$O(n \log n)$     Sort points in $P$ lexicographically (by $x$-coordinate, break ties by $y$-coordinate)
$O(1)$            Remove first three points from $P$ and insert them into $C$ in counter-clockwise
                    order around the triangle described by them.
n-3 times         for all $p \in P$   // Incrementally add p to hull
   $O(i)$             Compute the two tangents to $p$ and $C$
   $O(i)$             Remove enclosed non-hull points from $C$, and insert $p$
```

- Runtime: $O(\sum_{i=3}^{n} i) = O(n^2)$, where $n = |P|$
- Really?

Tangent computation



```
upper_tangent($C$,$p_i$):
// Compute upper tangent to $p_i$ and $C$. Return tangent vertex $v_t$
$v_t = p_{i\text{-}1}$
while succ($v_t$) lies above line through $p_i$ and $v_t$
        $v_t$ = succ($v_t$)
return $v_t$
```

⇒ **Amortization:** Every vertex that is checked during tangent computation is afterwards deleted from the current convex hull C

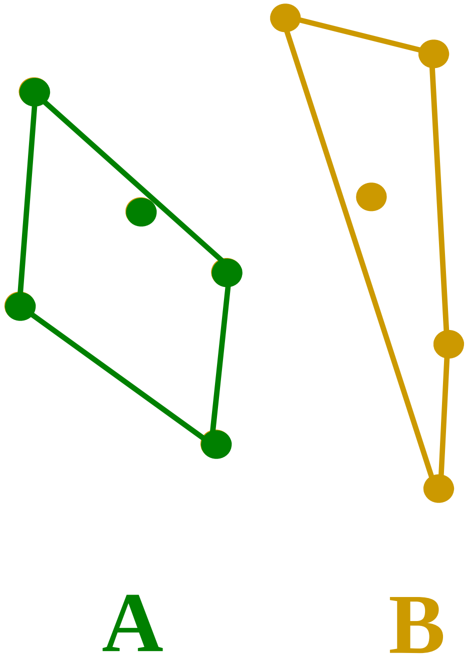
Incremental Insertion

```
              Algorithm Incremental_CH($P$):
              // Compute CH(P) by incrementally inserting points from left to right
              Input: Point set $P \subseteq \mathbf{R}^2$
              Output: $C$=CH($P$), described as a list of vertices in counter-clockwise order
$O(n \log n)$  Sort points in $P$ lexicographically (by x-coordinate, break ties by y-coordinate)
$O(1)$         Remove first three points from $P$ and insert them into $C$ in counter-clockwise
                 order around the triangle described by them.
n-3 times     for all $p \in P$   // Incrementally add p to hull
  $O(1)$ amort.     Compute the two tangents to $p$ and $C$
  $O(1)$ amort.     Remove enclosed non-hull points from $C$, and insert $p$
```

- Runtime: $O(n \log n + n) = O(n \log n)$, where $n = |P|$

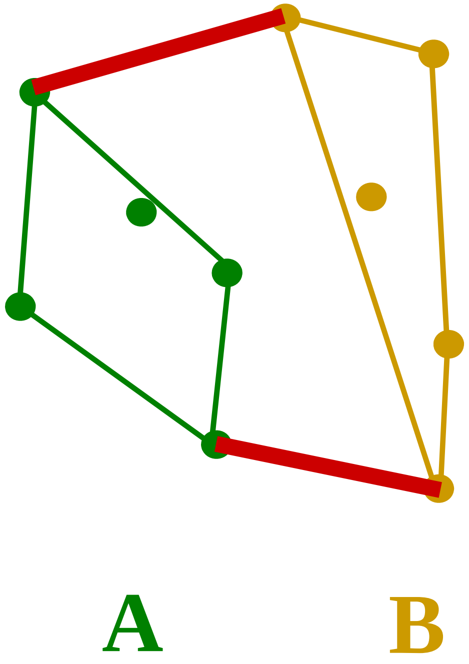
Convex Hull: Divide & Conquer

- Preprocessing: sort the points by x-coordinate
- Divide the set of points into two sets **A** and **B**:
  - **A** contains the left $\lfloor n/2 \rfloor$ points,
  - **B** contains the right $\lceil n/2 \rceil$ points
- Recursively compute the convex hull of **A**
- Recursively compute the convex hull of **B**
- Merge the two convex hulls



Merging

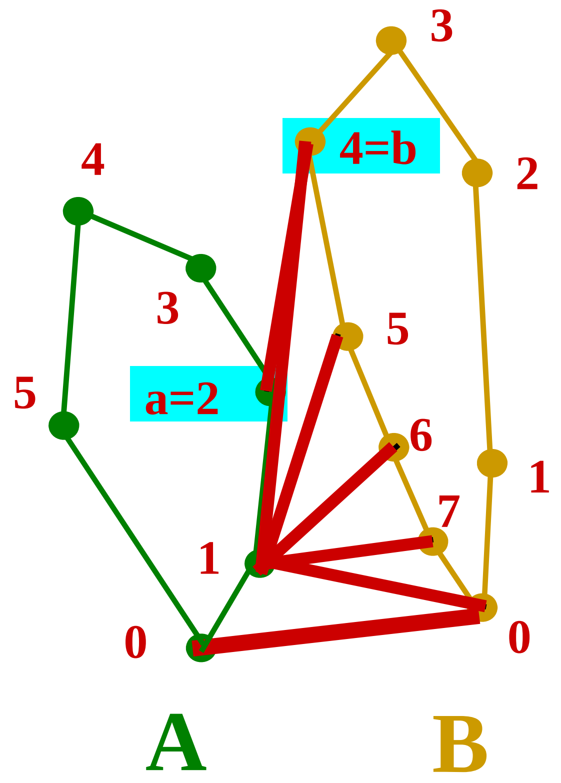
- **Find upper and lower tangent**
- With those tangents the convex hull of A∪B can be computed from the convex hulls of A and the convex hull of B in O(n) linear time

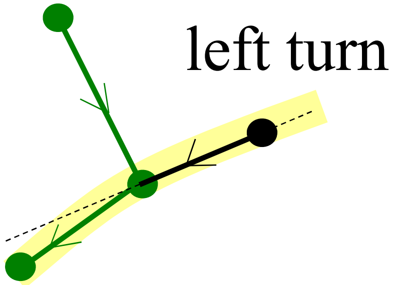


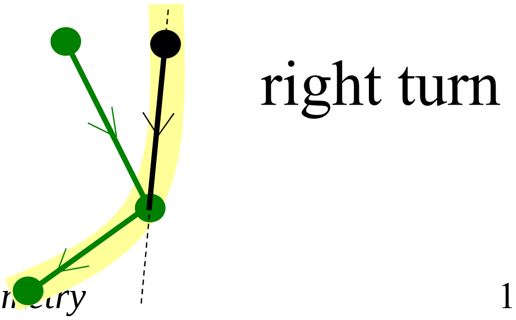
Finding the lower tangent

```
a = rightmost point of A
b = leftmost point of B
while T=ab not lower tangent to both
        convex hulls of A and B do{
    while T not lower tangent to
     convex hull of A do{
        a=a-1
    }
    while T not lower tangent to
     convex hull of B do{
        b=b+1
    }
}
```

check with orientation test







Convex Hull: Runtime

- Preprocessing: sort the points by x-coordinate

  $O(n \log n)$ just once

- Divide the set of points into two sets **A** and **B**:

  $O(1)$

  - **A** contains the left $\lfloor n/2 \rfloor$ points,
  - **B** contains the right $\lceil n/2 \rceil$ points

- Recursively compute the convex hull of **A**

  $T(n/2)$

- Recursively compute the convex hull of **B**

  $T(n/2)$

- Merge the two convex hulls

  $O(n)$

Convex Hull: Runtime

- Runtime Recurrence:

  $T(n) = 2\,T(n/2) + cn$

- Solves to $T(n) = \Theta(n \log n)$

Recurrence
(Just like merge sort recurrence)

1. ***Divide:*** Divide set of points in half.
2. ***Conquer:*** Recursively compute convex hulls of 2 halves.
3. ***Combine:*** Linear-time merge.

$$T(n) = 2\,T(n/2) + O(n)$$

# subproblems

subproblem size

work dividing and combining

Recurrence (cont'd)

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1; \\ 2T(n/2) + \Theta(\mathrm{n}) & \text{if } n > 1. \end{cases}$$

- How do we solve $T(n)$? I.e., how do we find out if it is $O(\mathrm{n})$ or $O(\mathrm{n}^2)$ or …?

Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

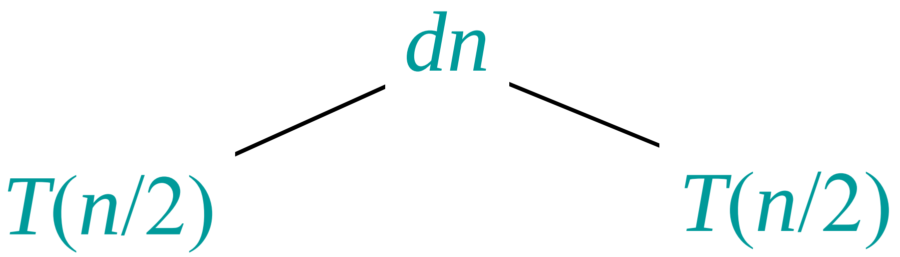
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

$$T(n)$$

Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.

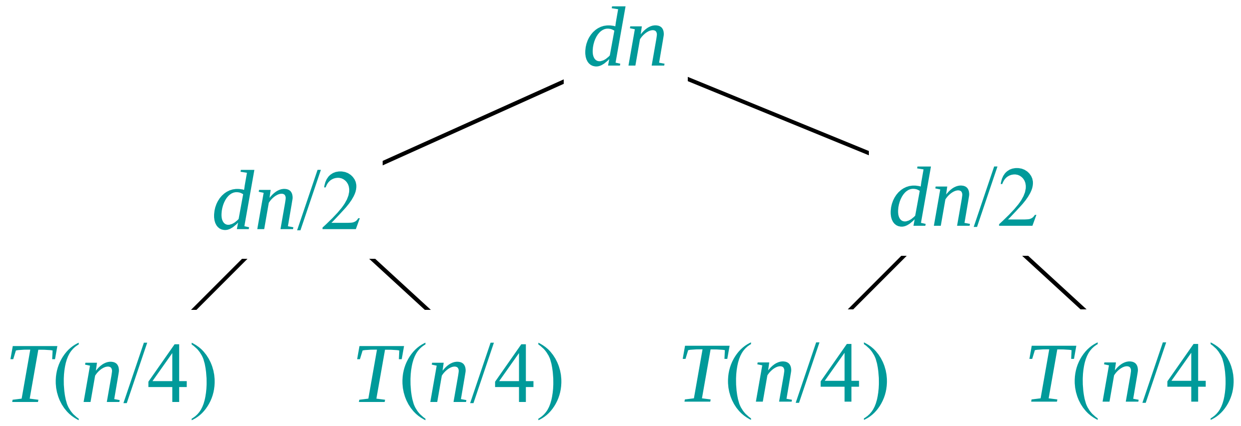


dn

$T(n/2)$ $T(n/2)$

Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



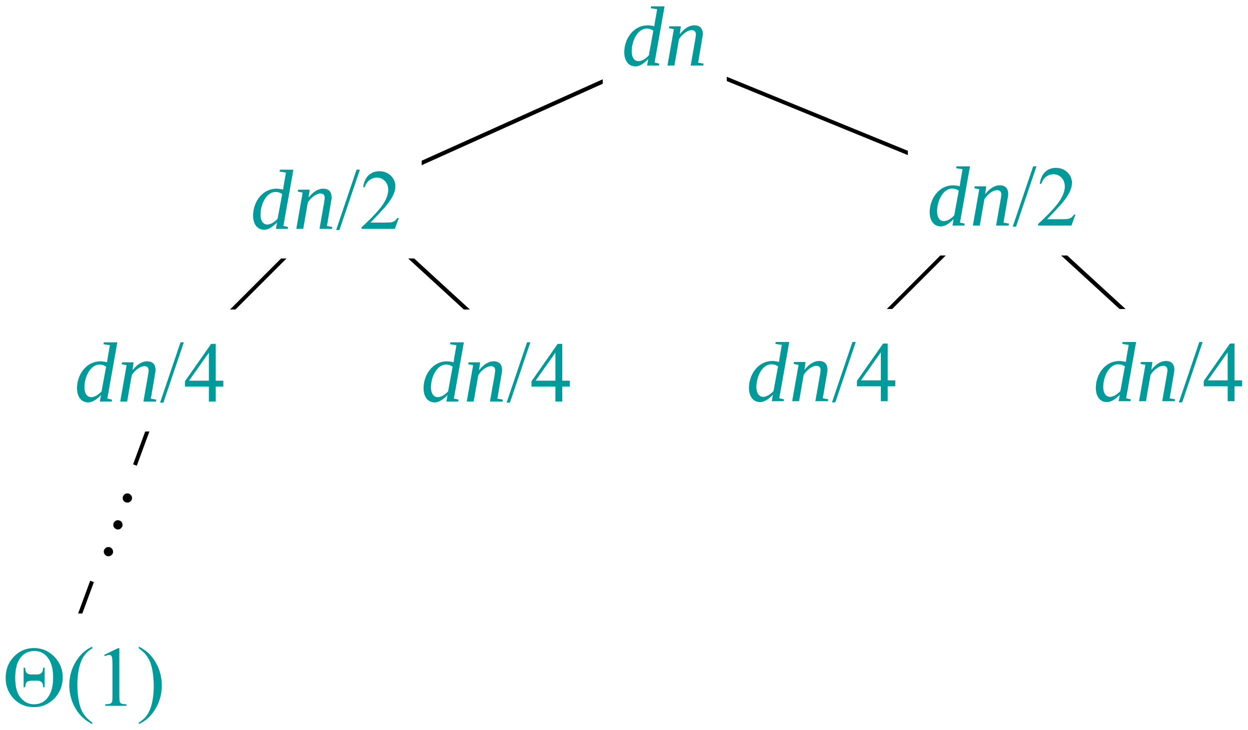
dn

$dn/2$ $dn/2$

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$

Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



dn

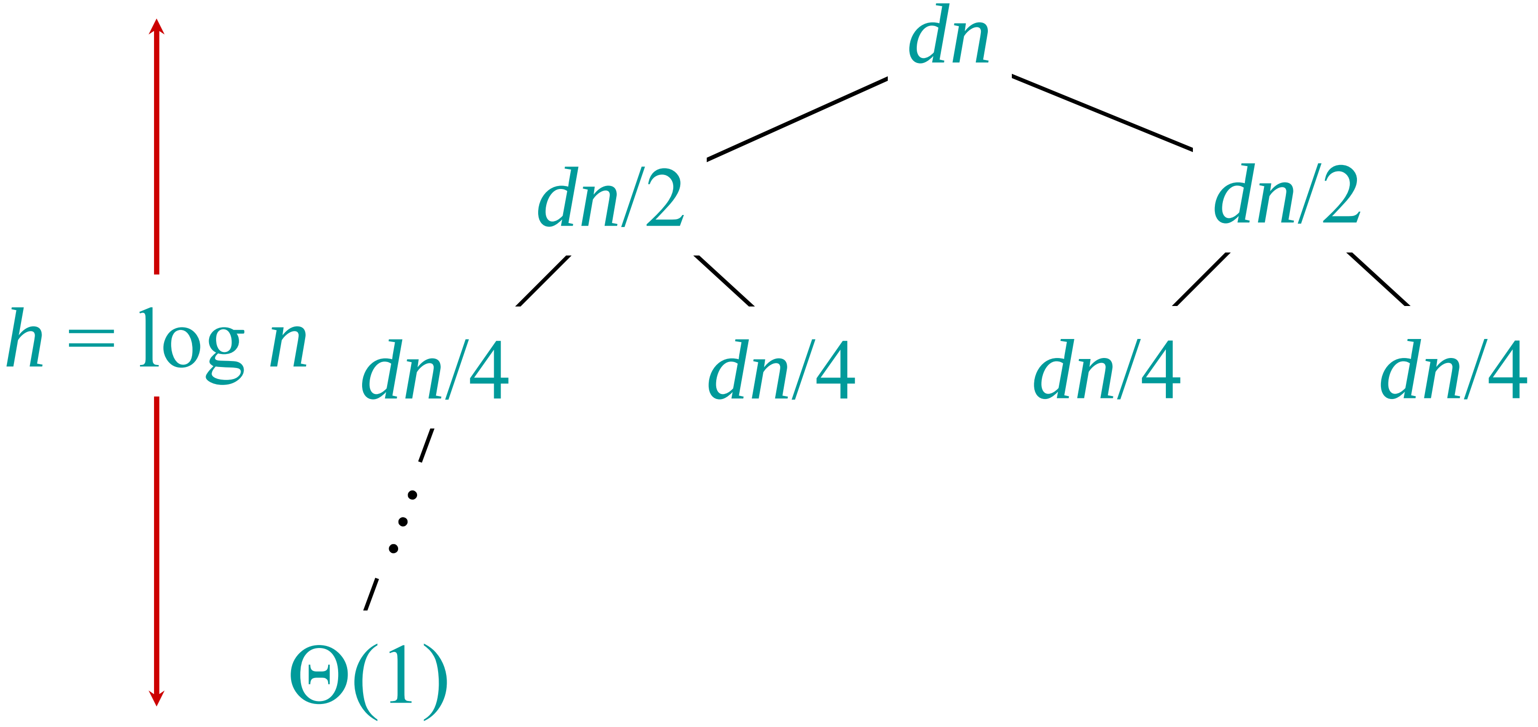
$dn/2$ $dn/2$

$dn/4$ $dn/4$ $dn/4$ $dn/4$

$\Theta(1)$

Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



dn

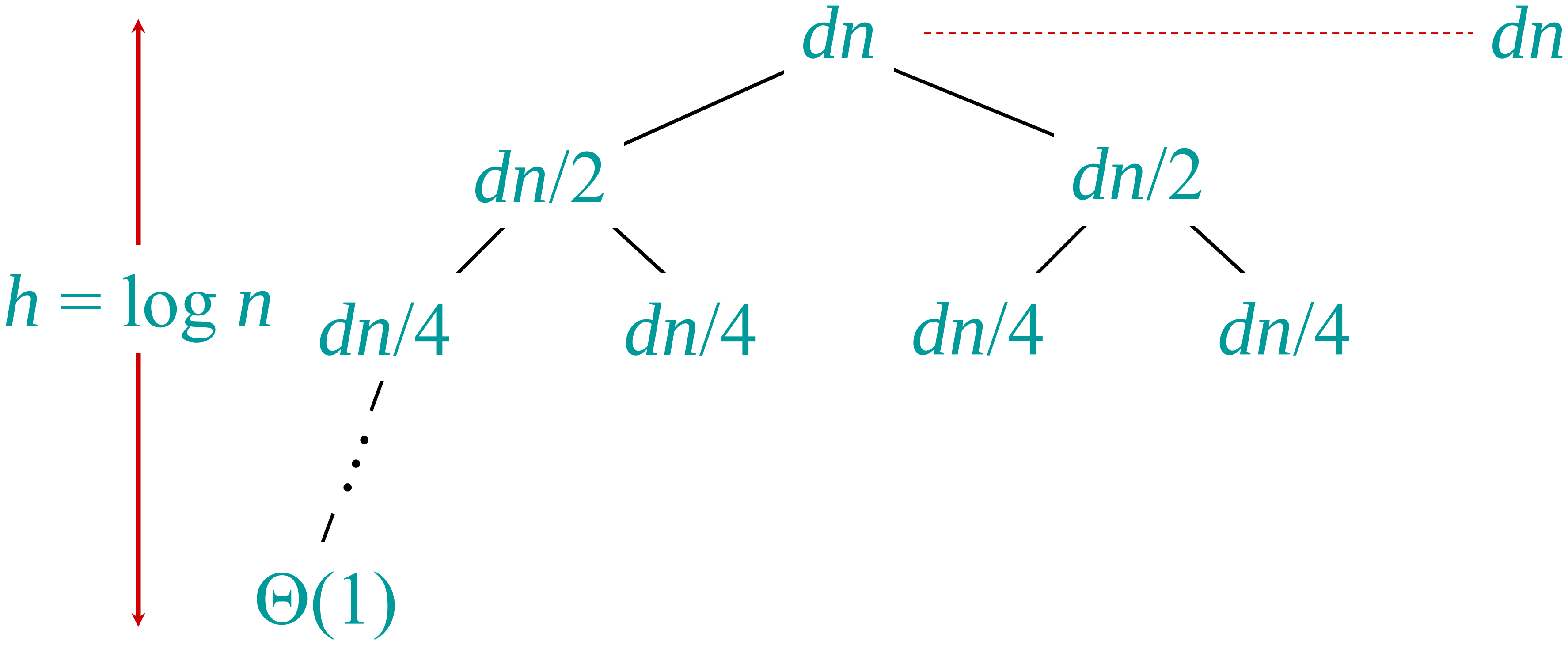
$dn/2$ $dn/2$

$h = \log n$ $dn/4$ $dn/4$ $dn/4$ $dn/4$

$\Theta(1)$

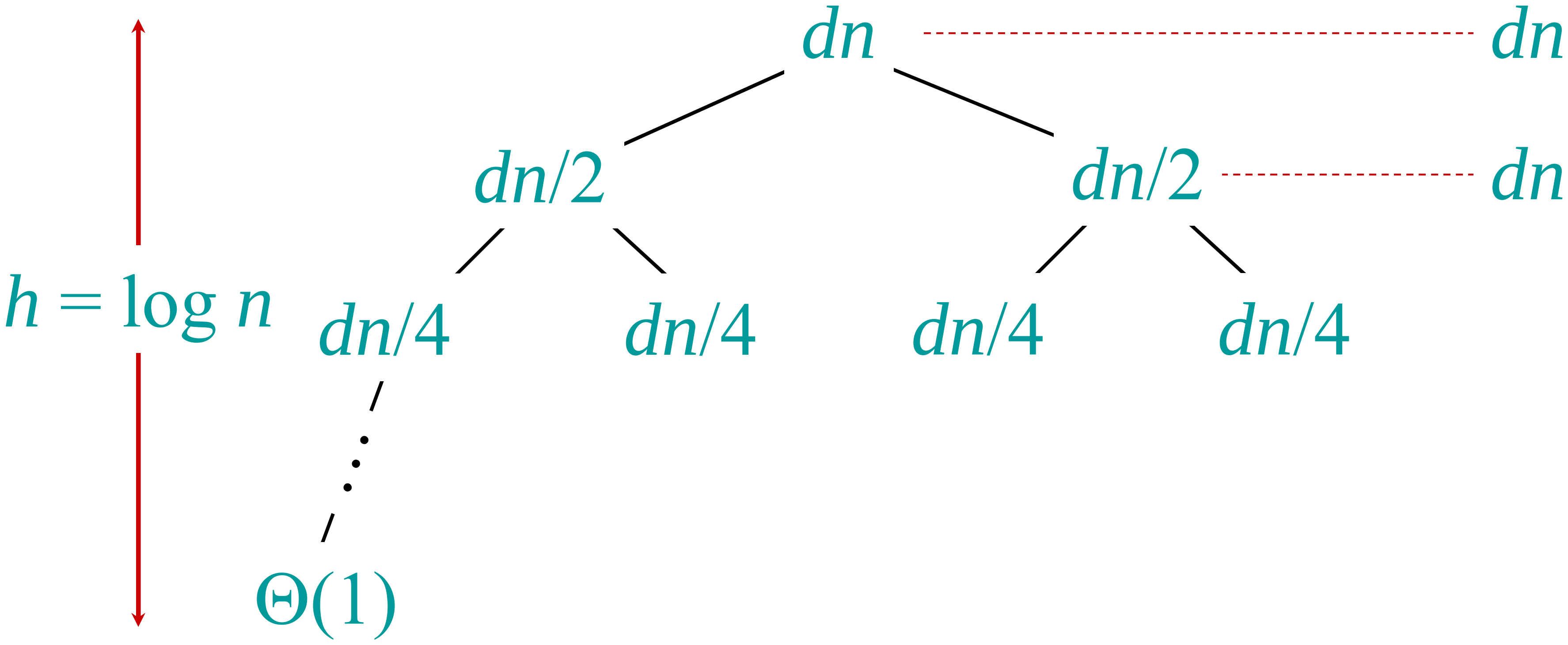
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



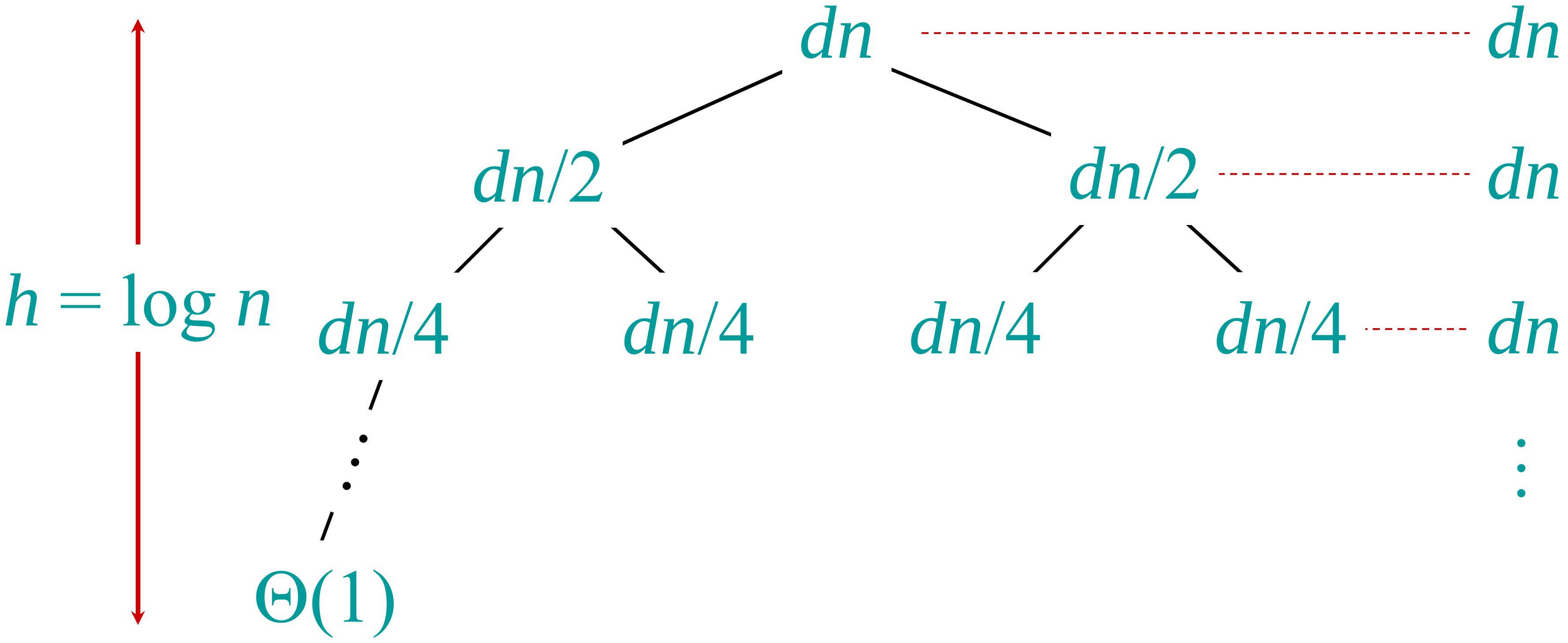
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



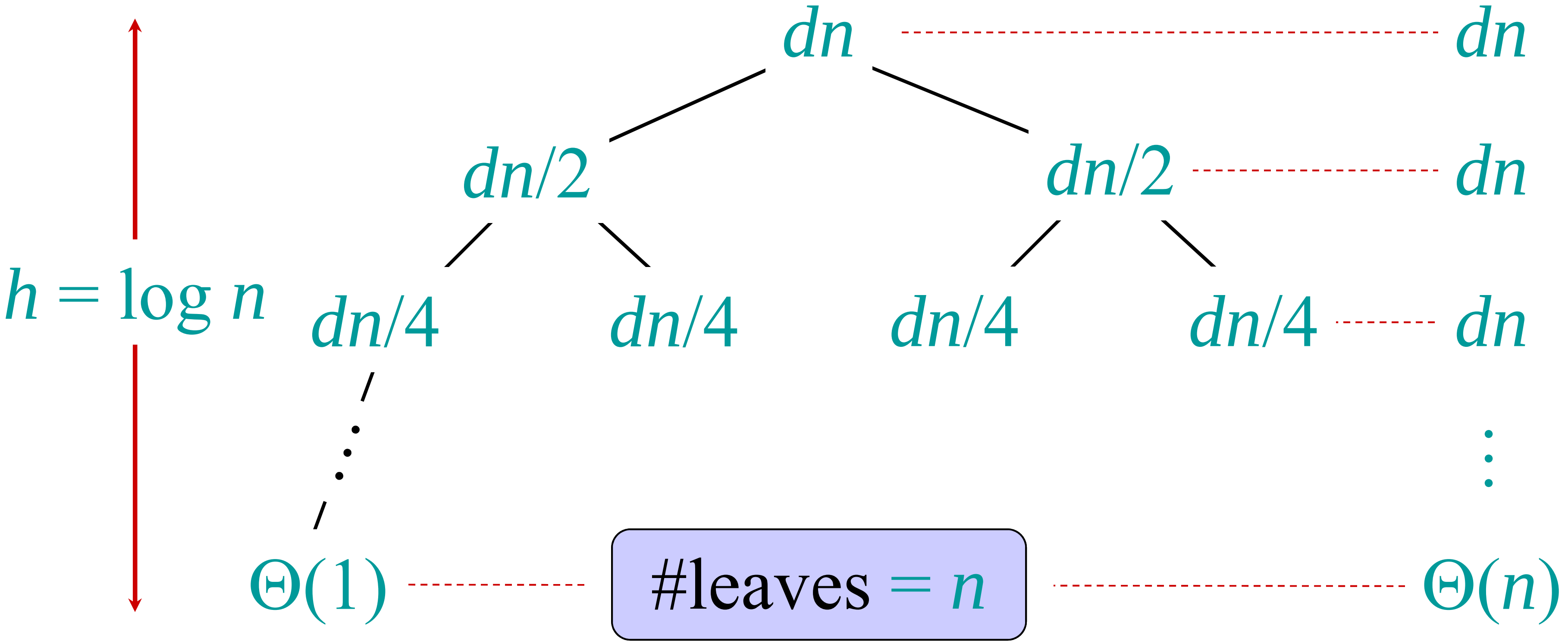
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



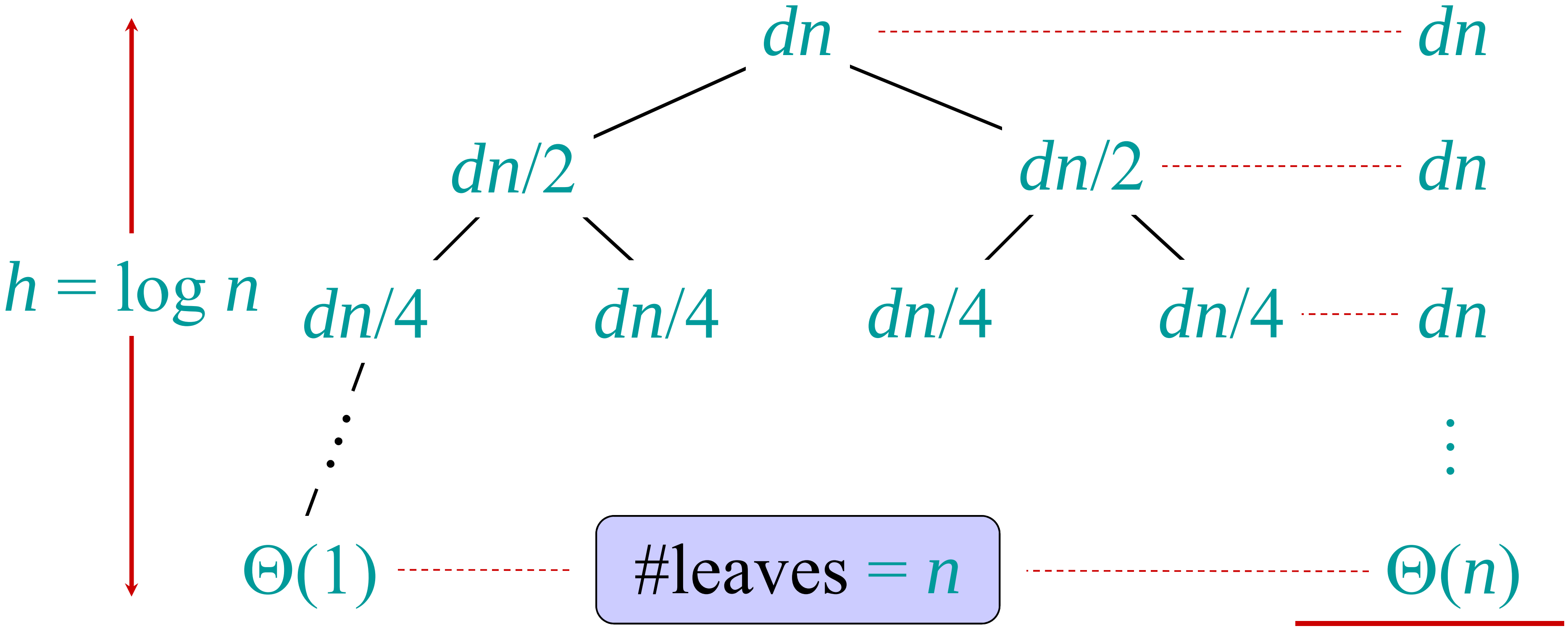
Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



Recursion tree

Solve $T(n) = 2T(n/2) + dn$, where $d > 0$ is constant.



Total $\Theta(n \log n)$

The divide-and-conquer design paradigm

1. ***Divide*** the problem (instance) into subproblems.

   a subproblems, **each** of size *n/b*

2. ***Conquer*** the subproblems by solving them recursively.

3. ***Combine*** subproblem solutions.

   Runtime is *f(n)*

Master theorem

$$T(n) = a\,T(n/b) + f(n)\ ,$$

where $a \geq 1$, $b > 1$, and f is asymptotically positive.

Case 1: $f(n) = O(n^{\log_b a - \varepsilon})$
$\Rightarrow T(n) = \Theta(n^{\log_b a})$.

Case 2: $f(n) = \Theta(n^{\log_b a} \log^k n)$
$\Rightarrow T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.

Case 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$ and $a\,f(n/b) \leq c\,f(n)$
$\Rightarrow T(n) = \Theta(f(n))$.

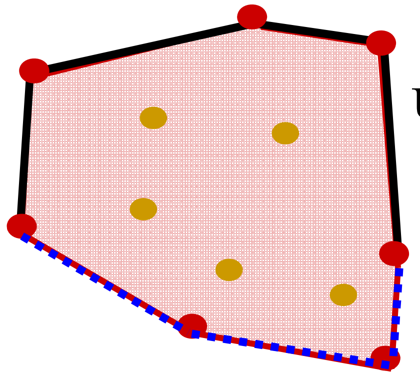
Convex hull: $a = 2$, $b = 2$ $\Rightarrow$ $n^{\log_b a} = n$
$\Rightarrow$ CASE 2 ($k = 0$) $\Rightarrow$ $T(n) = \Theta(n \log n)$.

Graham’s Scan

Another incremental algorithm

- Compute solution by incrementally adding points
- Add points in which order?
  - Sorted by *x*-coordinate
  - But convex hulls are cyclically ordered

  → Split convex hull into **upper** and **lower** part



upper convex hull UCH(P)

lower convex hull LCH(P)

Graham’s LCH

```
Algorithm Grahams_LCH($P$):
// Incrementally compute the lower convex hull of P
Input: Point set $P \subseteq \mathbf{R}^2$
Output: A list $L$ of vertices describing LCH($P$) in counter-clockwise order

Sort $P$ in increasing order by $x$-coordinate → $P = \{p_1,\ldots,p_n\}$
$L = \{p_2,p_1\}$
for $i$=3 to $n$
    while $|L|$>=2 and orientation(L.second(), L.first(), $p_i$,) <= 0 // no left turn
            delete first element from L
    Append $p_i$ to the front of $L$
```

O(n log n)

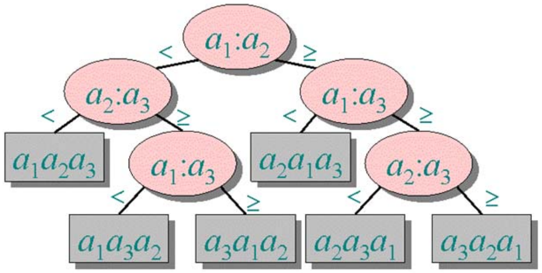
O(n)

- Each element is appended only once, and hence only deleted at most once ⇒ the for-loop takes O(n) time
- O(n log n) time total

Lower Bound

- Comparison-based sorting of n elements takes $\Omega(n \log n)$ time.
- How can we use this lower bound to show a lower bound for the computation of the convex hull of n points in $\mathbf{R}^2$?

Decision-tree model

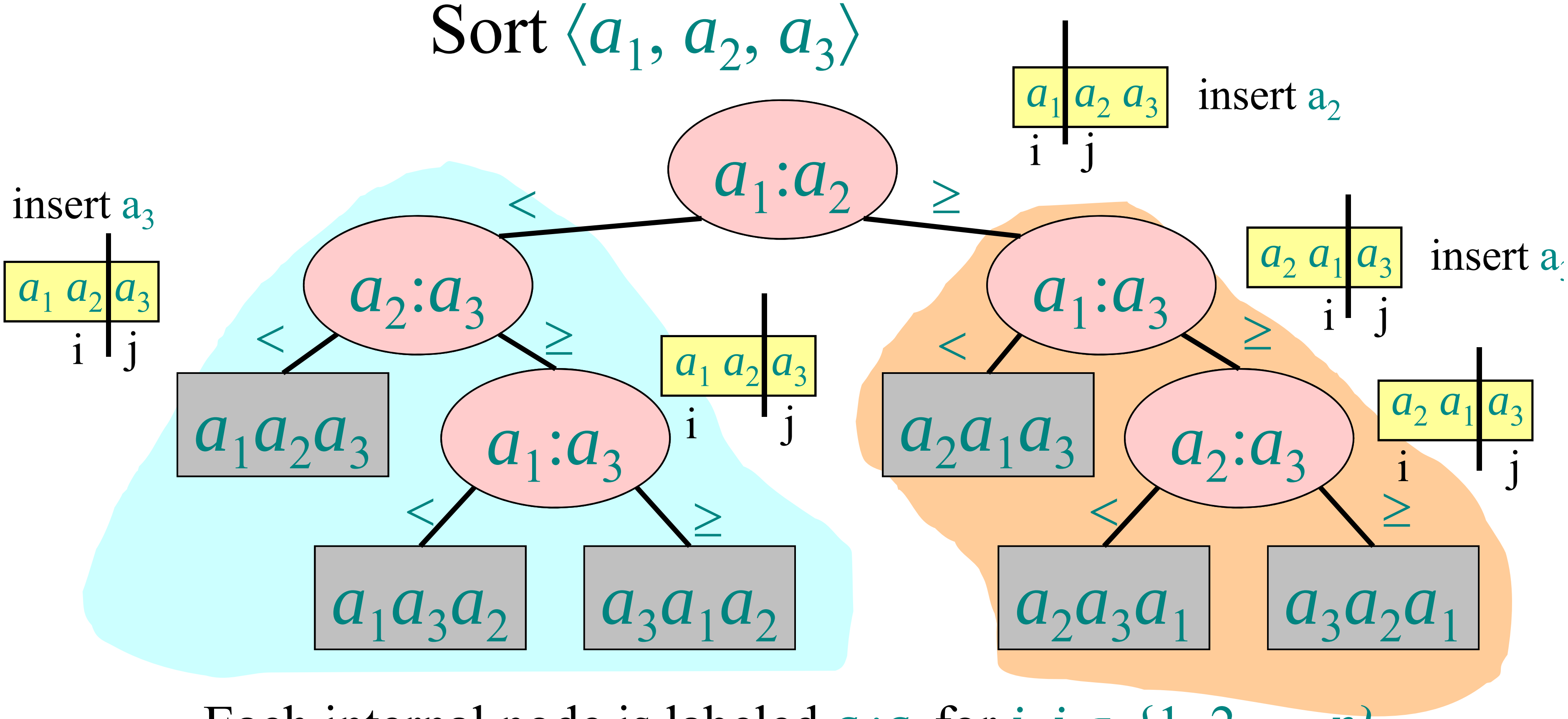


A decision tree models the execution of any comparison sorting algorithm:

- One tree per input size n.
- The tree contains **all** possible comparisons (= if-branches) that could be executed for **any** input of size n.
- The tree contains **all** comparisons along **all** possible instruction traces (= control flows) for **all** inputs of size n.
- For one input, only one path to a leaf is executed.
- Running time = length of the path taken.
- Worst-case running time = height of tree.

Decision-tree for insertion sort

Sort $\langle a_1, a_2, a_3 \rangle$

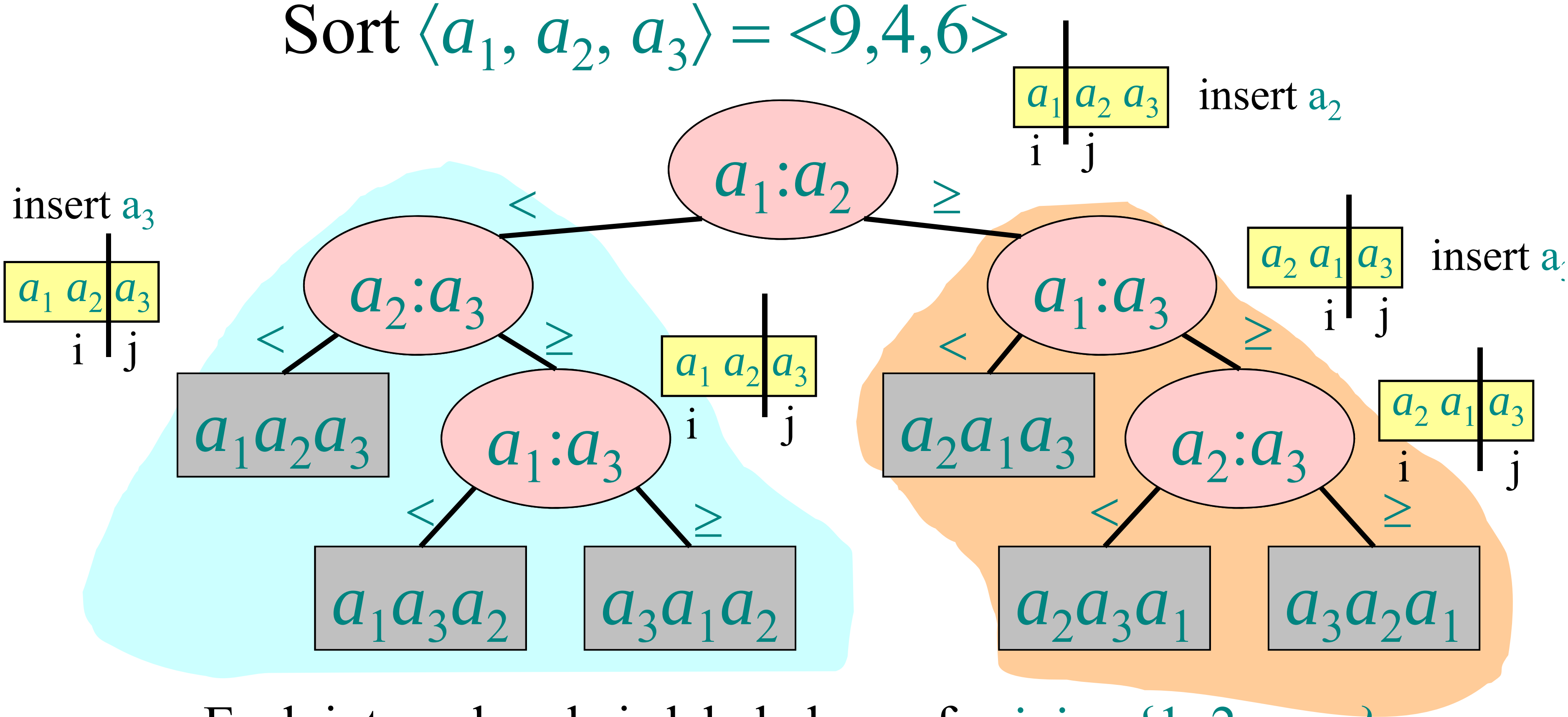


Each internal node is labeled $a_i{:}a_j$ for $i, j \in \{1, 2, \ldots, n\}$.

- The left subtree shows subsequent comparisons if $a_i < a_j$.
- The right subtree shows subsequent comparisons if $a_i \geq a_j$.

Decision-tree for insertion sort

Sort $\langle a_1, a_2, a_3 \rangle$ = <9,4,6>

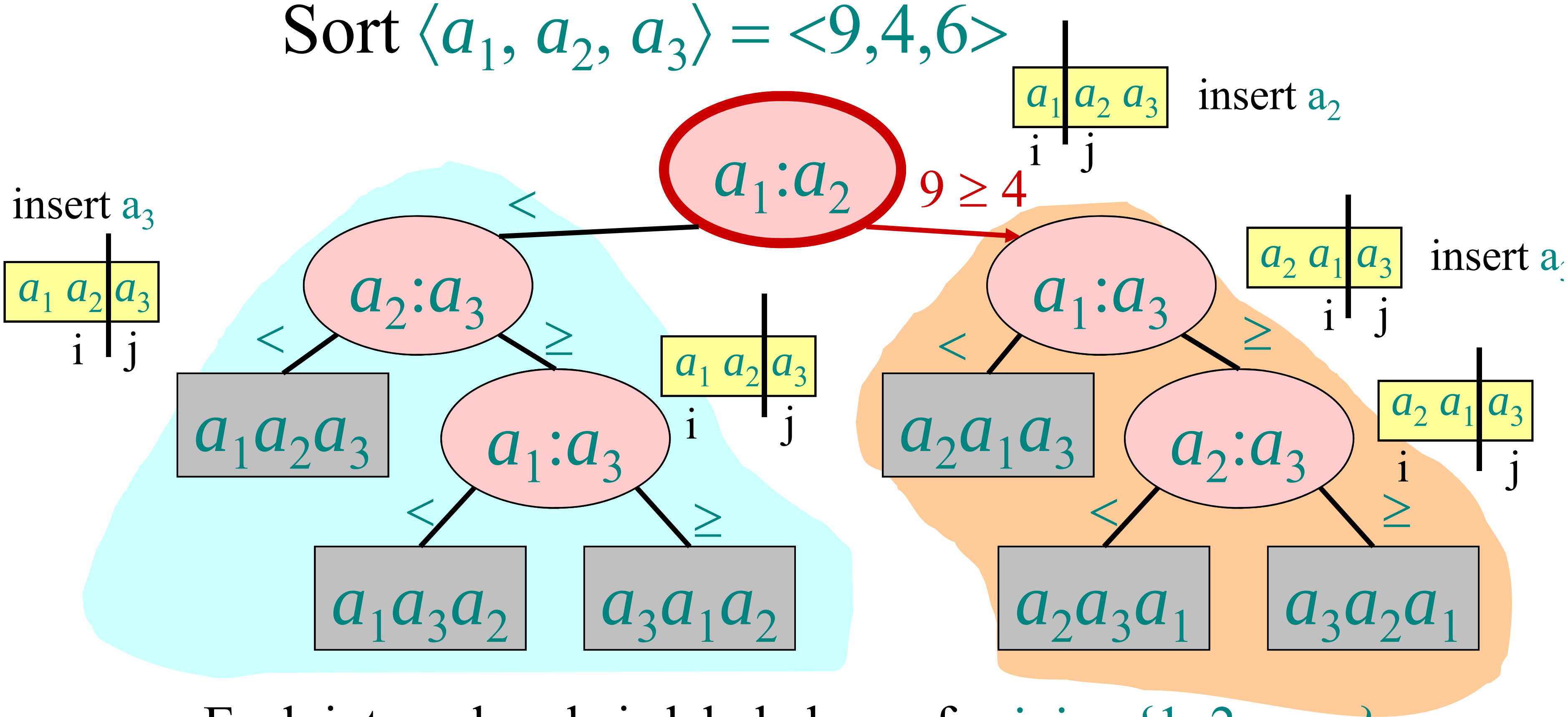


Each internal node is labeled $a_i{:}a_j$ for $i, j \in \{1, 2, \ldots, n\}$.

- The left subtree shows subsequent comparisons if $a_i < a_j$.
- The right subtree shows subsequent comparisons if $a_i \geq a_j$.

Decision-tree for insertion sort

Sort $\langle a_1, a_2, a_3 \rangle$ = <9,4,6>

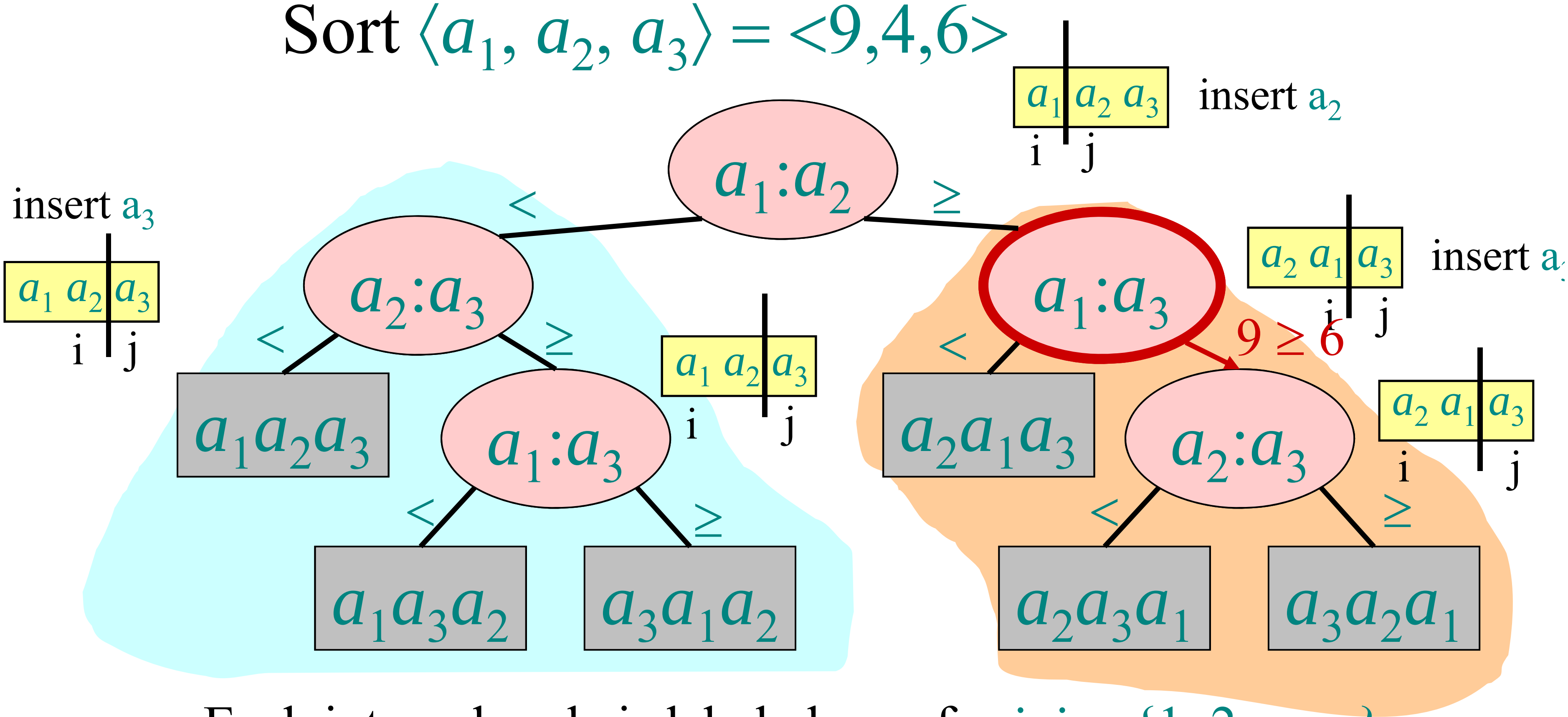


Each internal node is labeled $a_i{:}a_j$ for $i, j \in \{1, 2, \ldots, n\}$.

- The left subtree shows subsequent comparisons if $a_i < a_j$.
- The right subtree shows subsequent comparisons if $a_i \geq a_j$.

Decision-tree for insertion sort

Sort $\langle a_1, a_2, a_3 \rangle = <9,4,6>$

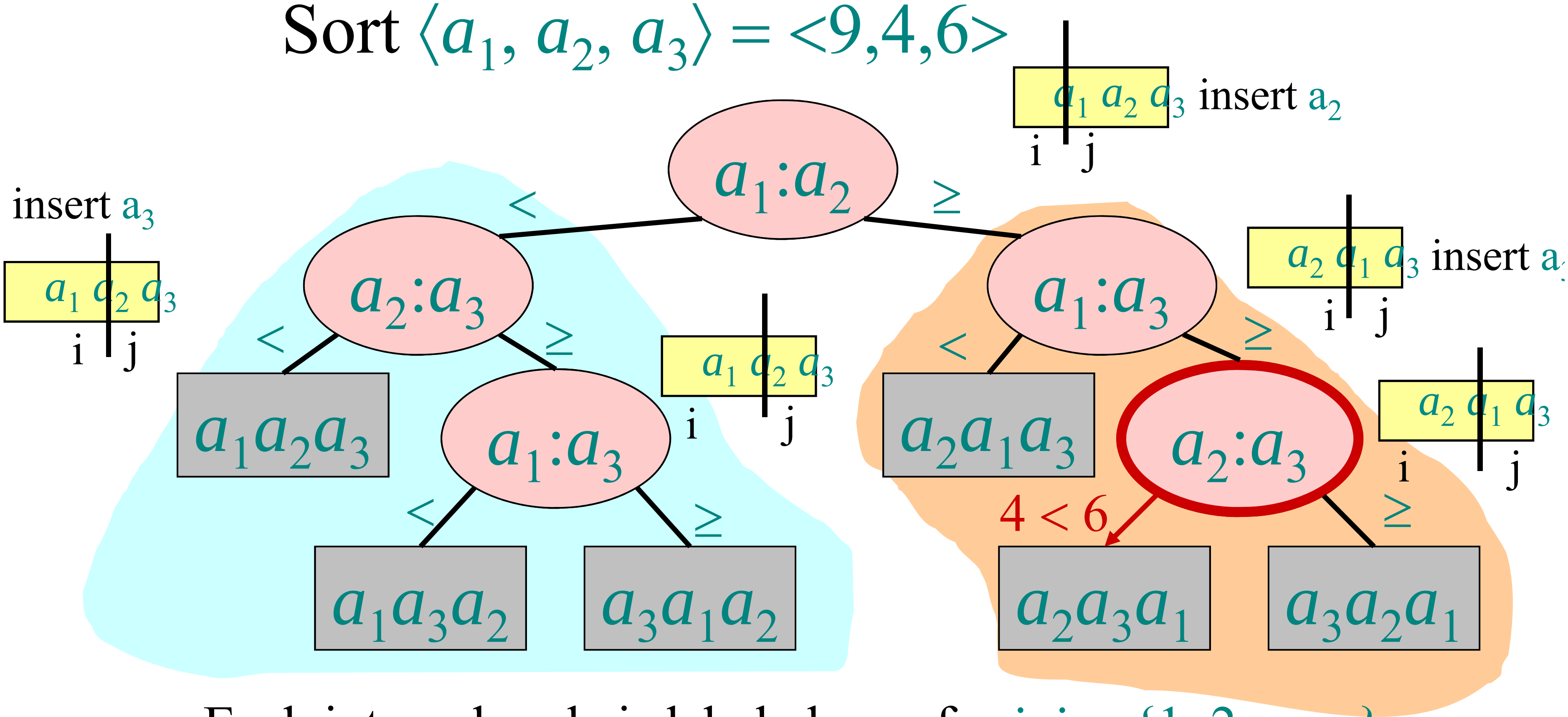


Each internal node is labeled $a_i{:}a_j$ for $i, j \in \{1, 2, \ldots, n\}$.

- The left subtree shows subsequent comparisons if $a_i < a_j$.
- The right subtree shows subsequent comparisons if $a_i \geq a_j$.

Decision-tree for insertion sort

Sort $\langle a_1, a_2, a_3 \rangle$ = <9,4,6>

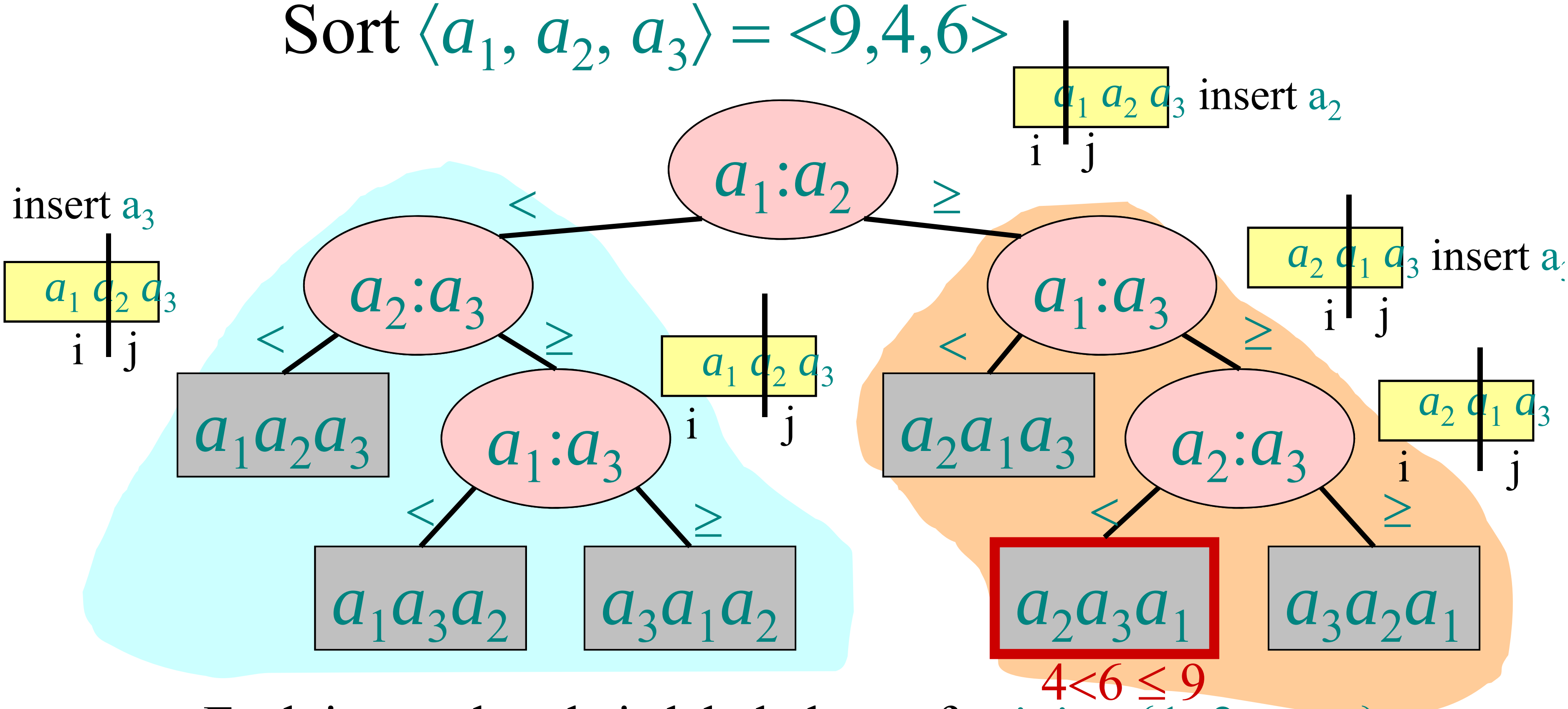


Each internal node is labeled $a_i{:}a_j$ for $i, j \in \{1, 2, \ldots, n\}$.

- The left subtree shows subsequent comparisons if $a_i < a_j$.
- The right subtree shows subsequent comparisons if $a_i \geq a_j$.

Decision-tree for insertion sort

Sort $\langle a_1, a_2, a_3 \rangle$ = <9,4,6>

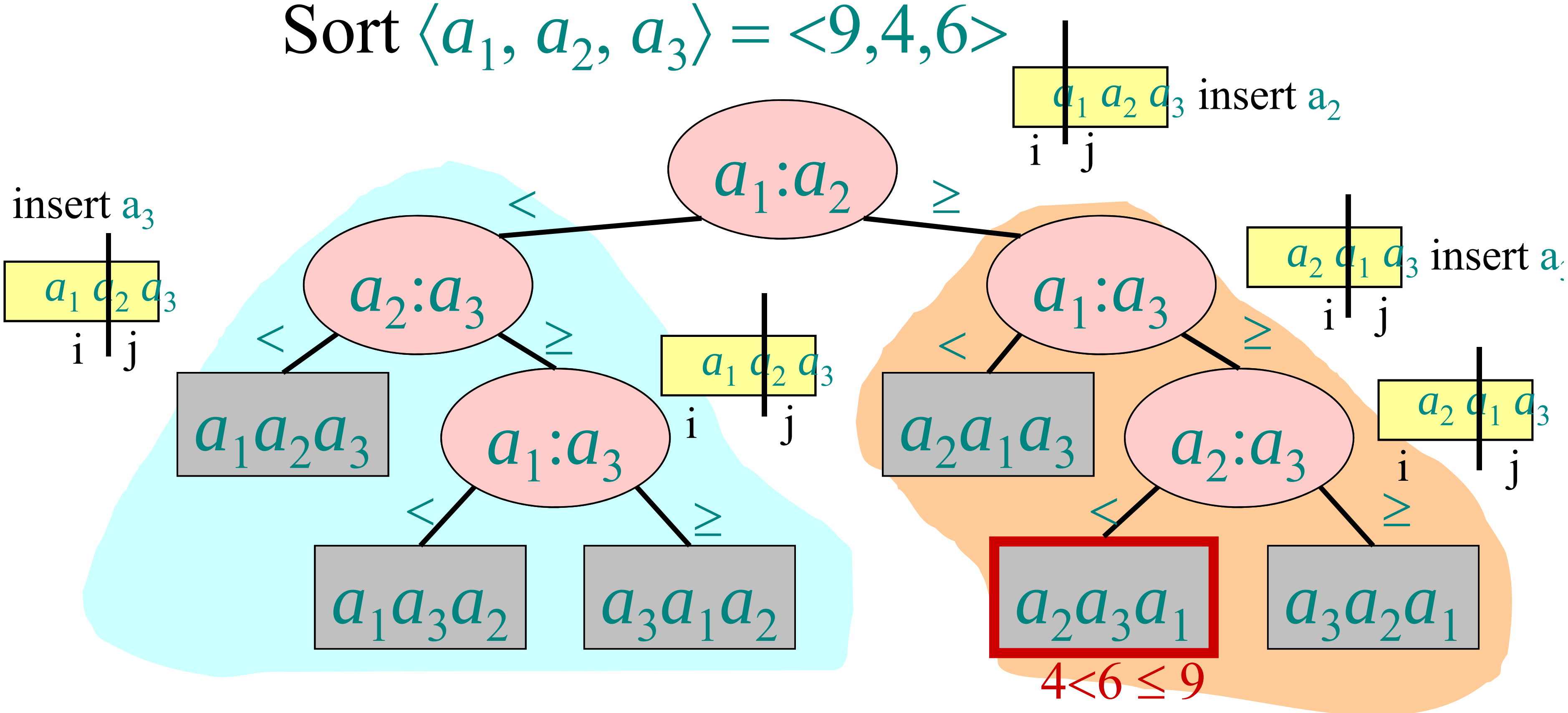


Each internal node is labeled $a_i{:}a_j$ for $i, j \in \{1, 2, \ldots, n\}$.

- The left subtree shows subsequent comparisons if $a_i < a_j$.
- The right subtree shows subsequent comparisons if $a_i \geq a_j$.

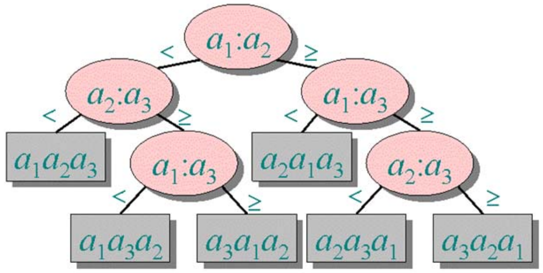
Decision-tree for insertion sort

Sort $\langle a_1, a_2, a_3 \rangle = <9,4,6>$



Each leaf contains a permutation $\langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$ to indicate that the ordering $a_{\pi(1)} \leq a_{\pi(2)} \leq \ldots \leq a_{\pi(n)}$ has been established.

Lower bound for comparison sorting



Theorem. Any decision tree that can sort n elements must have height $\Omega(n \log n)$.

Proof. The tree must contain $\geq n!$ leaves, since there are $n!$ possible permutations. A height-h binary tree has $\leq 2^h$ leaves. Thus, $n! \leq 2^h$.

$$\begin{aligned} \therefore\ h &\geq \log(n!) && \text{(log is mono. increasing)} \\ &\geq \log\left((n/2)^{n/2}\right) \\ &= n/2 \log n/2 \\ \Rightarrow h &\in \Omega(n \log n). \end{aligned}$$

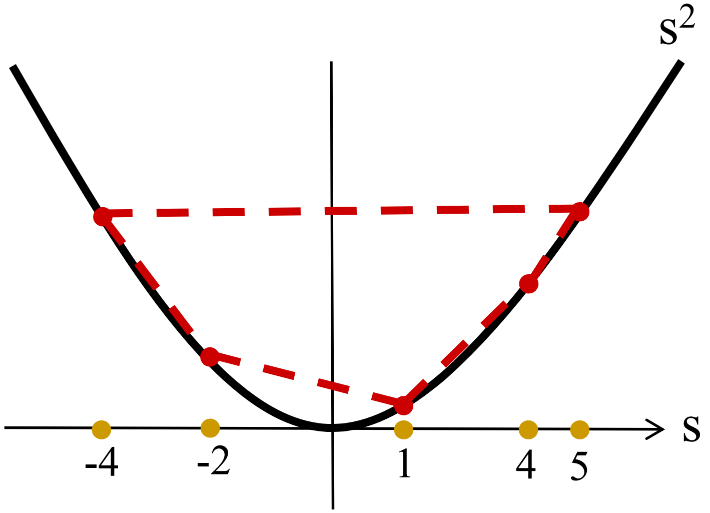
□

Lower Bound

- Comparison-based sorting of n elements takes $\Omega(n \log n)$ time.
- How can we use this lower bound to show a lower bound for the computation of the convex hull of n points in $\mathbf{R}^2$?
- Devise a sorting algorithm which uses the convex hull and otherwise only linear-time operations
  - ⇒ Since this is a comparison-based sorting algorithm, the lower bound $\Omega(n \log n)$ applies
  - ⇒ Since all other operations need linear time, the convex hull algorithm has to take $\Omega(n \log n)$ time

CH_Sort

```
Algorithm CH_Sort($S$):
/* Sorts a set of numbers using a convex hull
   algorithm.
   Converts numbers to points, runs CH,
   converts back to sorted sequence. */
Input: Set of numbers $S \subseteq \mathbf{R}$
Output: A list $L$ of of numbers in $S$ sorted in
           increasing order
P=∅
for each $s \in S$ insert $(s,s^2)$ into $P$
$L'$ = CH($P$) // compute convex hull
Find point $p' \in P$ with minimum x-coordinate
for each $p=(p_x,p_y) \in L'$, starting with $p'$,
   add $p_x$ into $L$
return $L$
```



Convex Hull Summary

• Brute force algorithm:	$O(n^3)$
• Jarvis’ march (gift wrapping):	$O(nh)$
• Incremental insertion:	$O(n \log n)$
• Divide-and-conquer:	$O(n \log n)$
• Graham’s scan:	$O(n \log n)$
• Lower bound:	$\Omega(n \log n)$