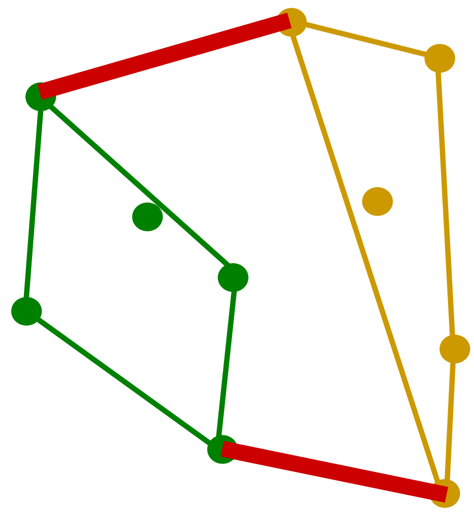
CS 6463: AT Computational Geometry
Fall 2010



Convex Hulls

Carola Wenk

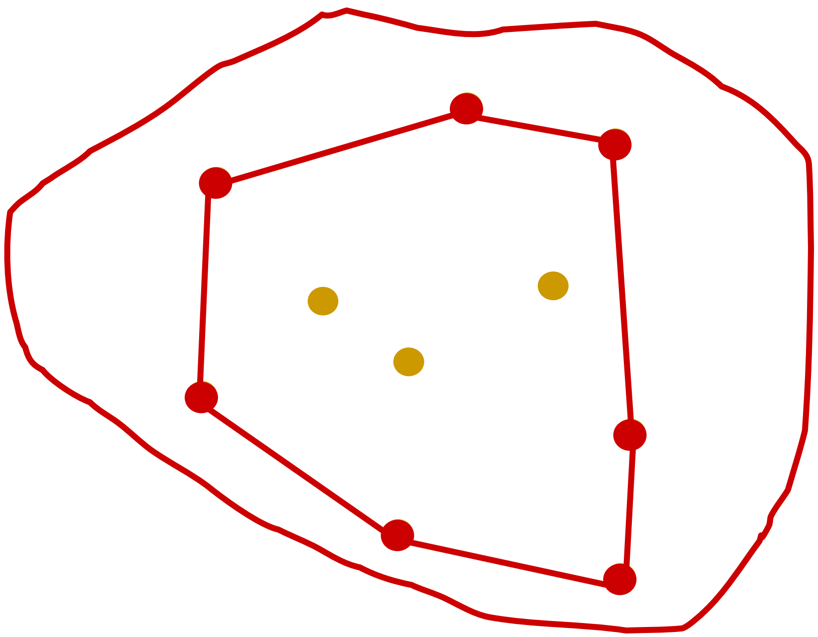
Convex Hull Problem

- Given a set of pins on a pinboard

  and a rubber band around them.

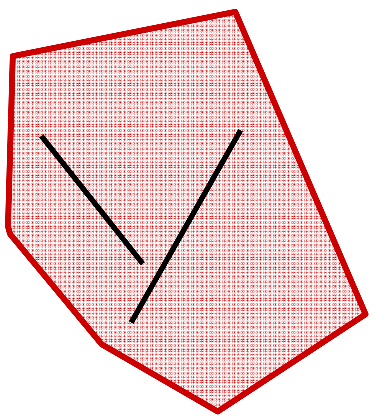
  How does the rubber band look when it snaps tight?

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

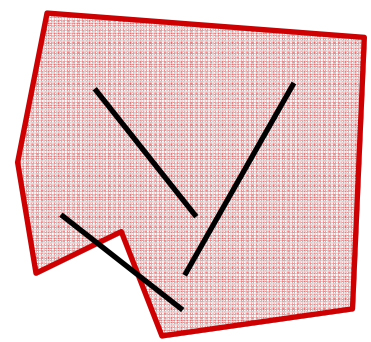


Convexity

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



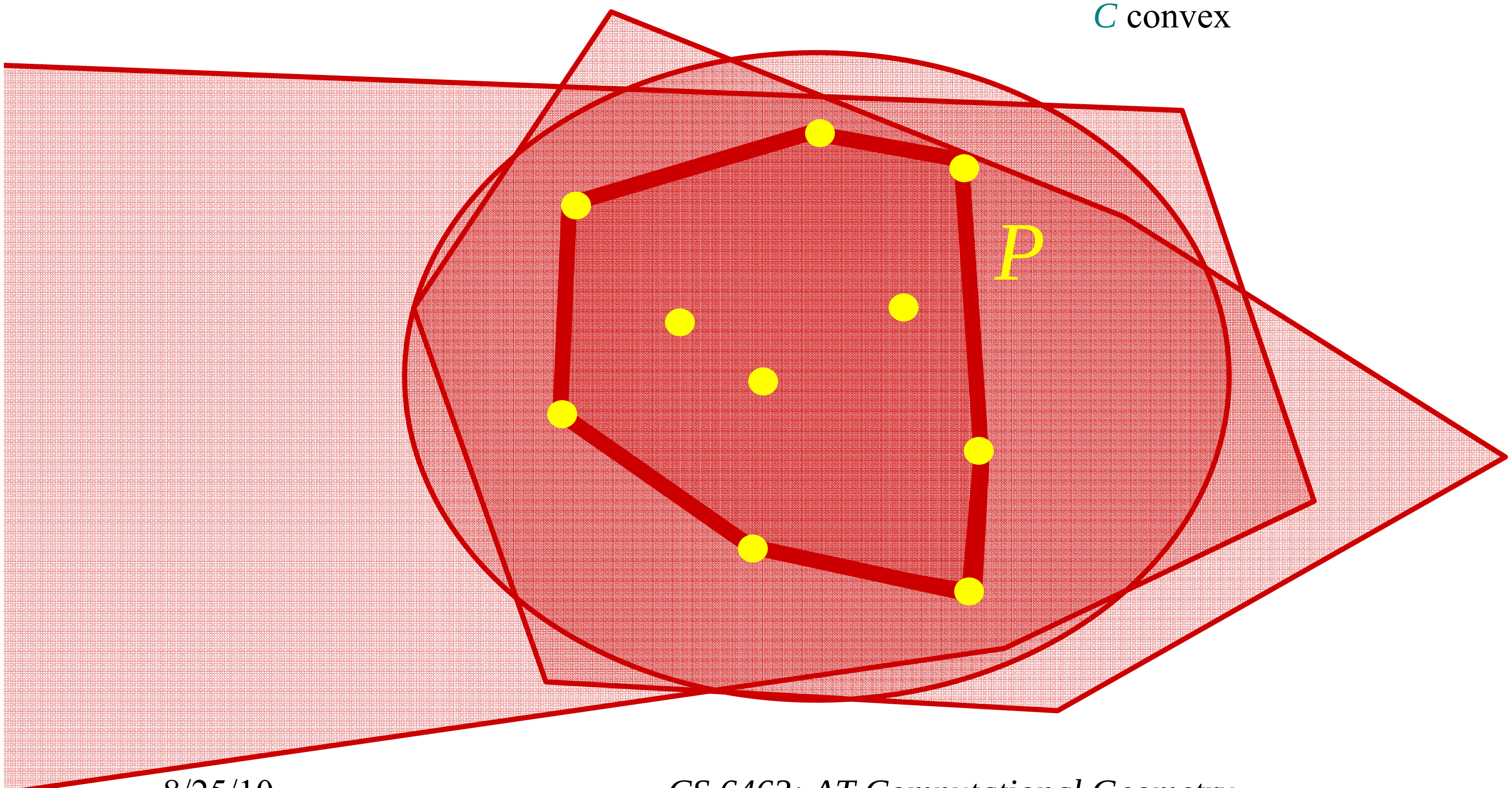
convex



non-convex

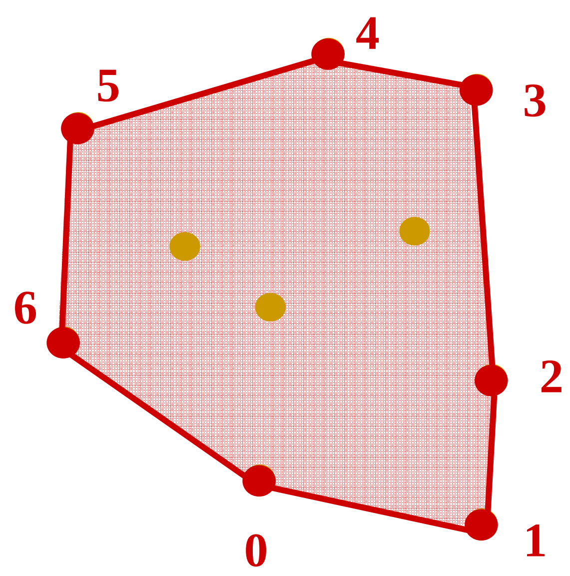
Convex Hull

- 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 $CH(P) = \bigcap_{\substack{C \supseteq P \\ C \text{ convex}}} C$.



Convex Hull

- **Observation:** CH(P) is the unique convex polygon whose vertices are points of P and which contains all points of P.
- We represent the convex hull as the sequence of points on the convex hull polygon (the boundary of the convex hull), in counter-clockwise order.

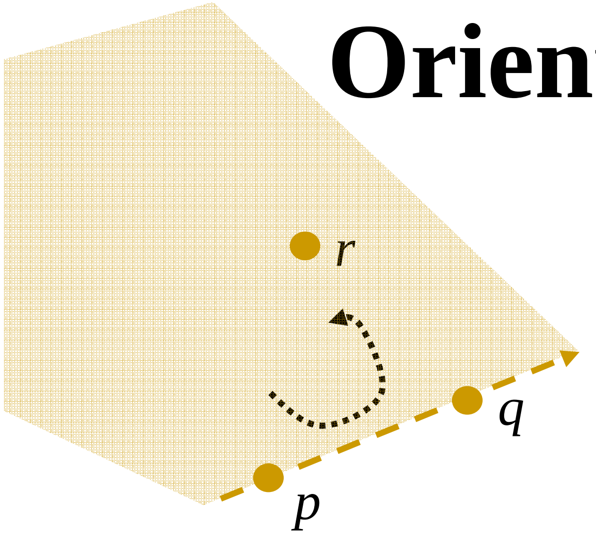


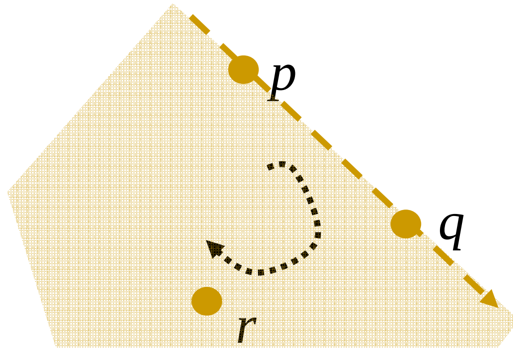
A First Try

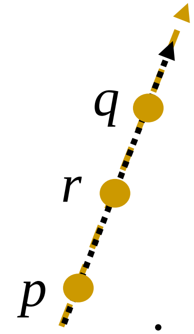
Algorithm SLOW_CH(P):
/* CH(P) = Intersection of all half-planes that are defined by the directed line through ordered pairs of points in P and that have all remaining points of P on their left */
Input: Point set $P \subseteq \mathbf{R}^2$
Output: A list L of vertices describing the CH(P) in counter-clockwise 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 left?

Orientation Test / Halfplane Test







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

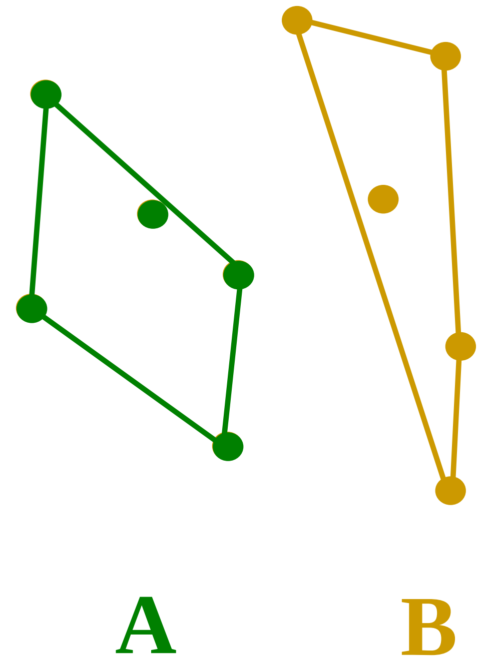
- negative orientation (clockwise)
- r lies to the right of $\overrightarrow{pq}$

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

- 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

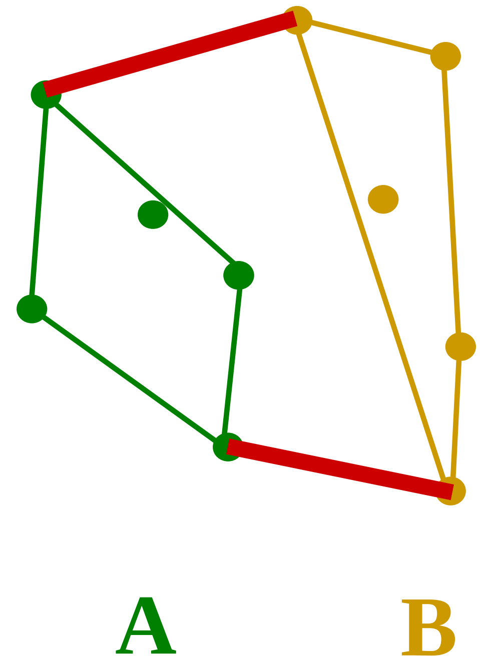
Convex Hull: Divide & Conquer

- 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

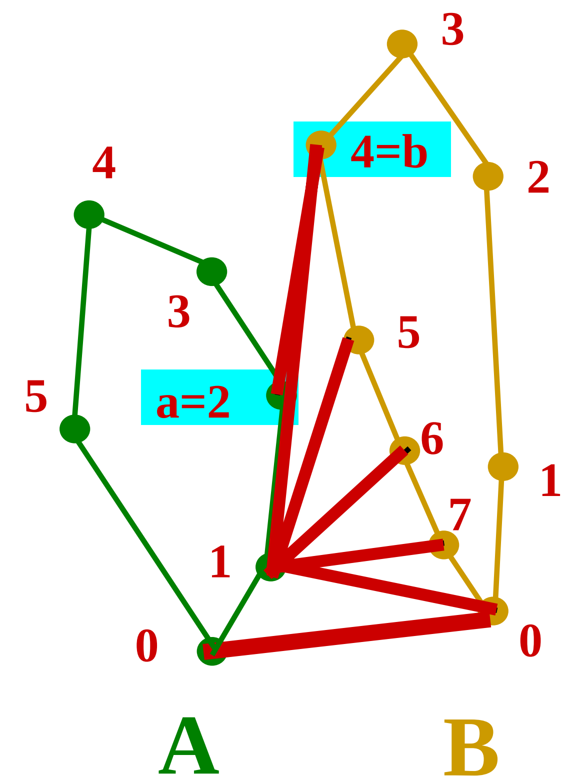
- **Find upper and lower tangent**
- With those tangents the convex hull of A∪B can be computed from the convex hulls of A and the convex hull of B in O(n) linear time

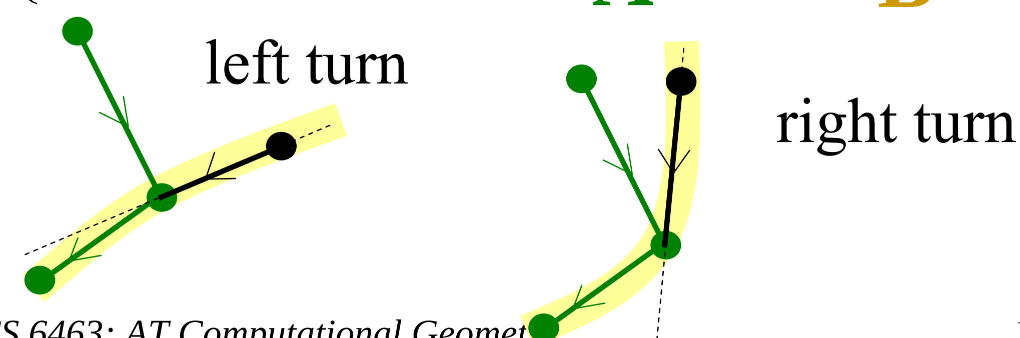


Finding the lower tangent

```
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)

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

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

$$
\begin{array}{ccc}
 & dn & \\
\diagup & & \diagdown \\
T(n/2) & & T(n/2)
\end{array}
$$

Recursion tree

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

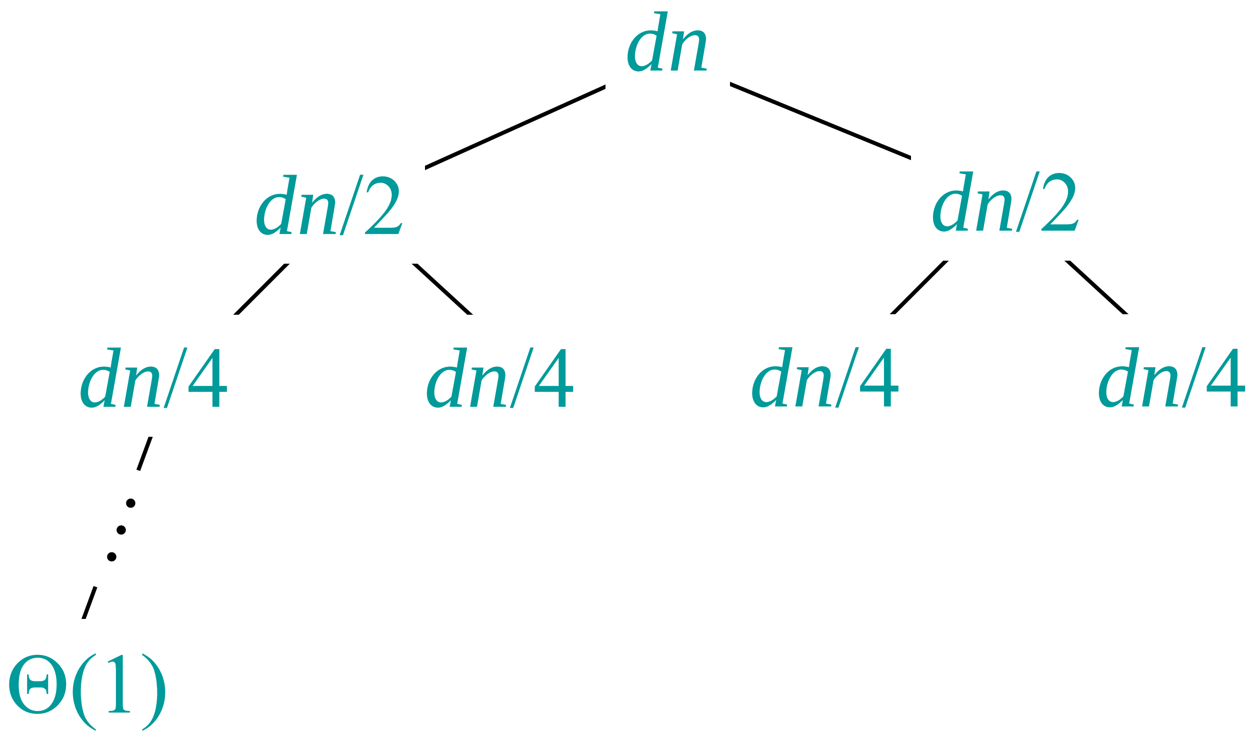
$$dn$$

$$dn/2 \qquad\qquad dn/2$$

$$T(n/4) \qquad T(n/4) \qquad T(n/4) \qquad T(n/4)$$

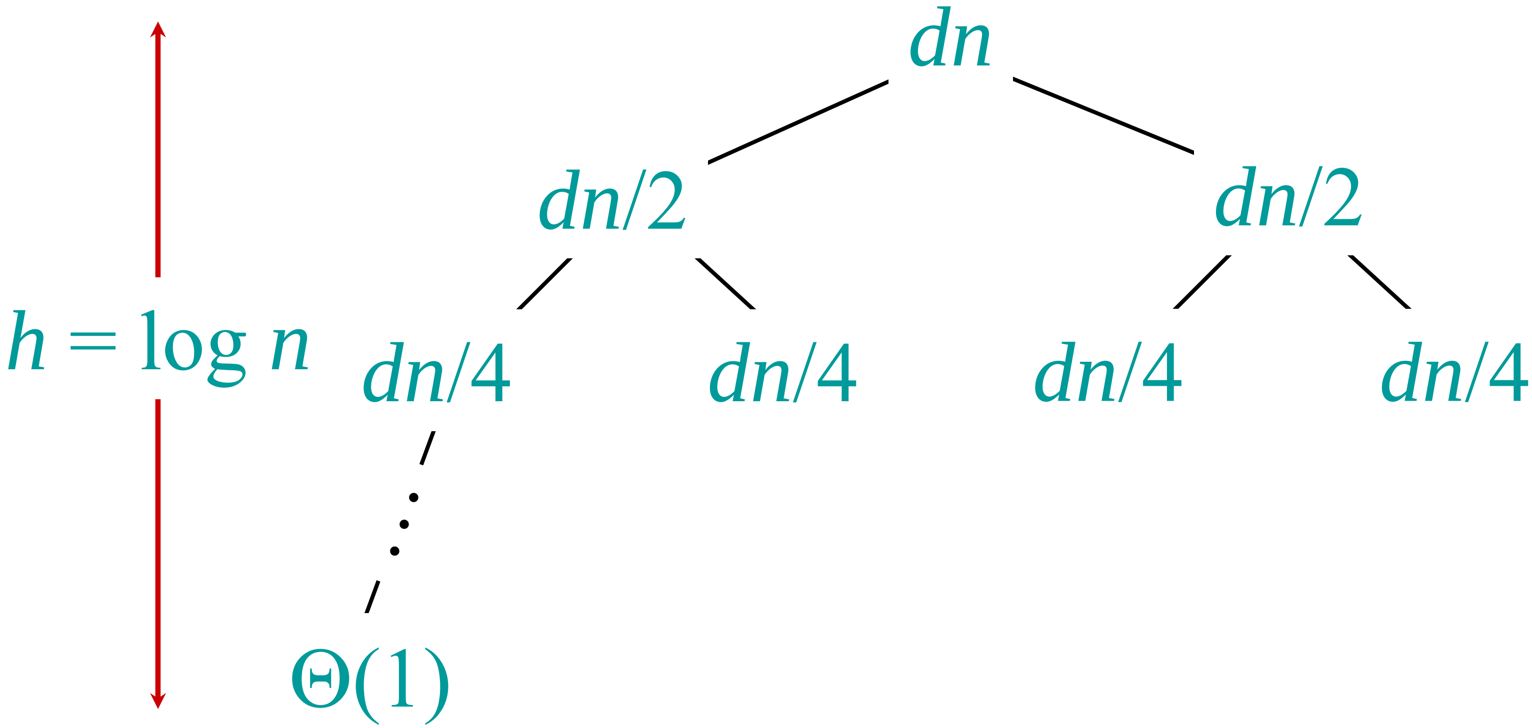
Recursion tree

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



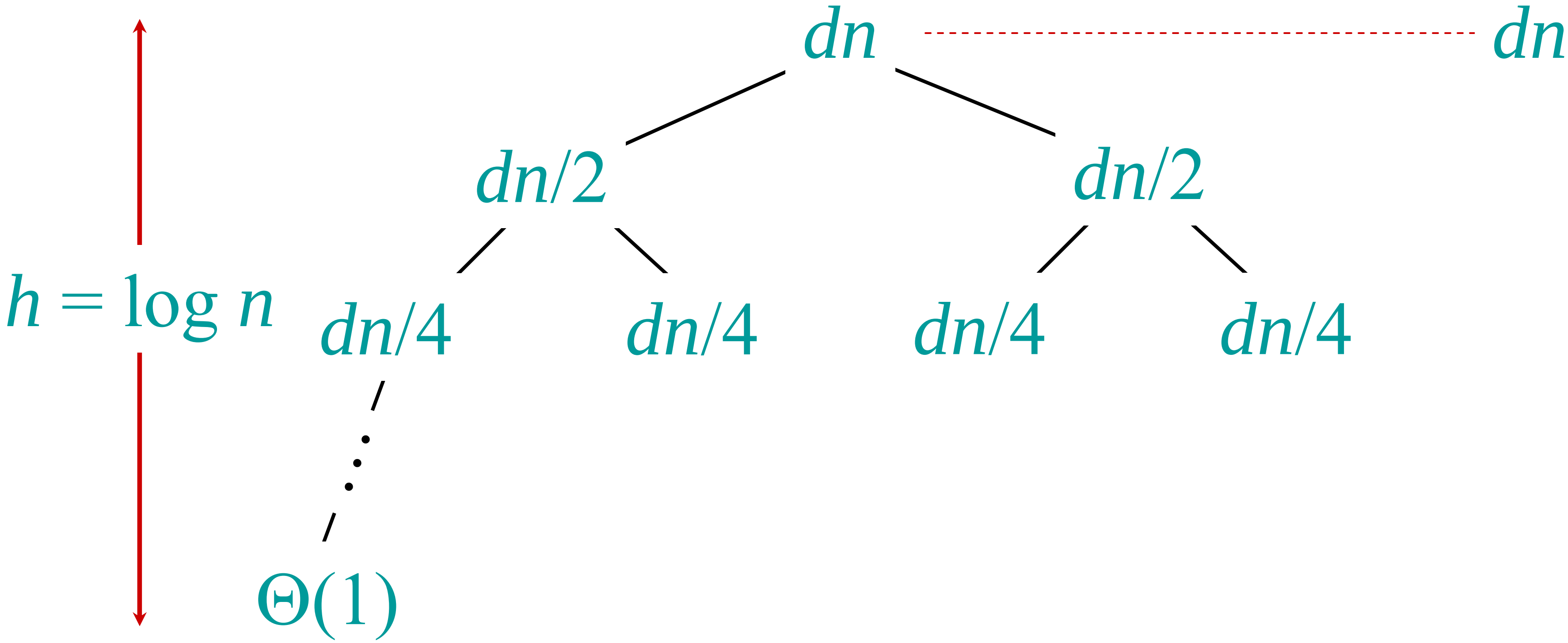
Recursion tree

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



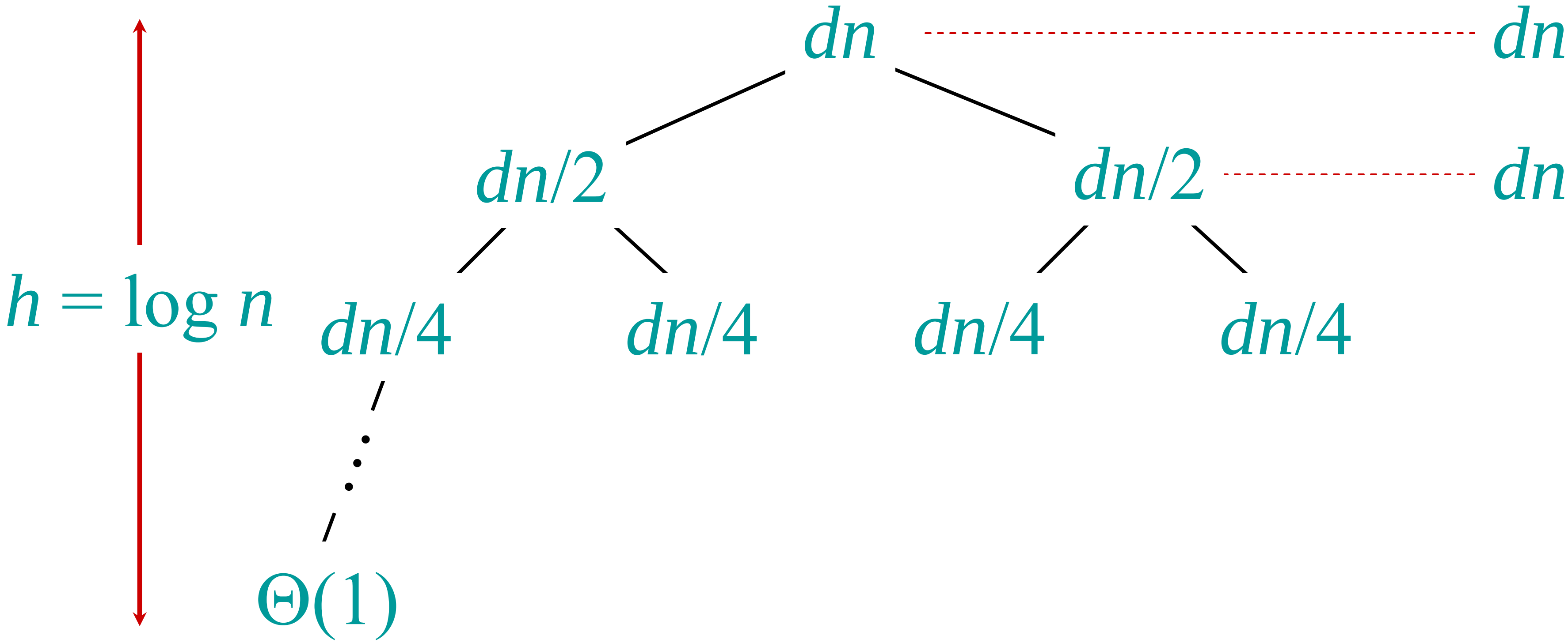
Recursion tree

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



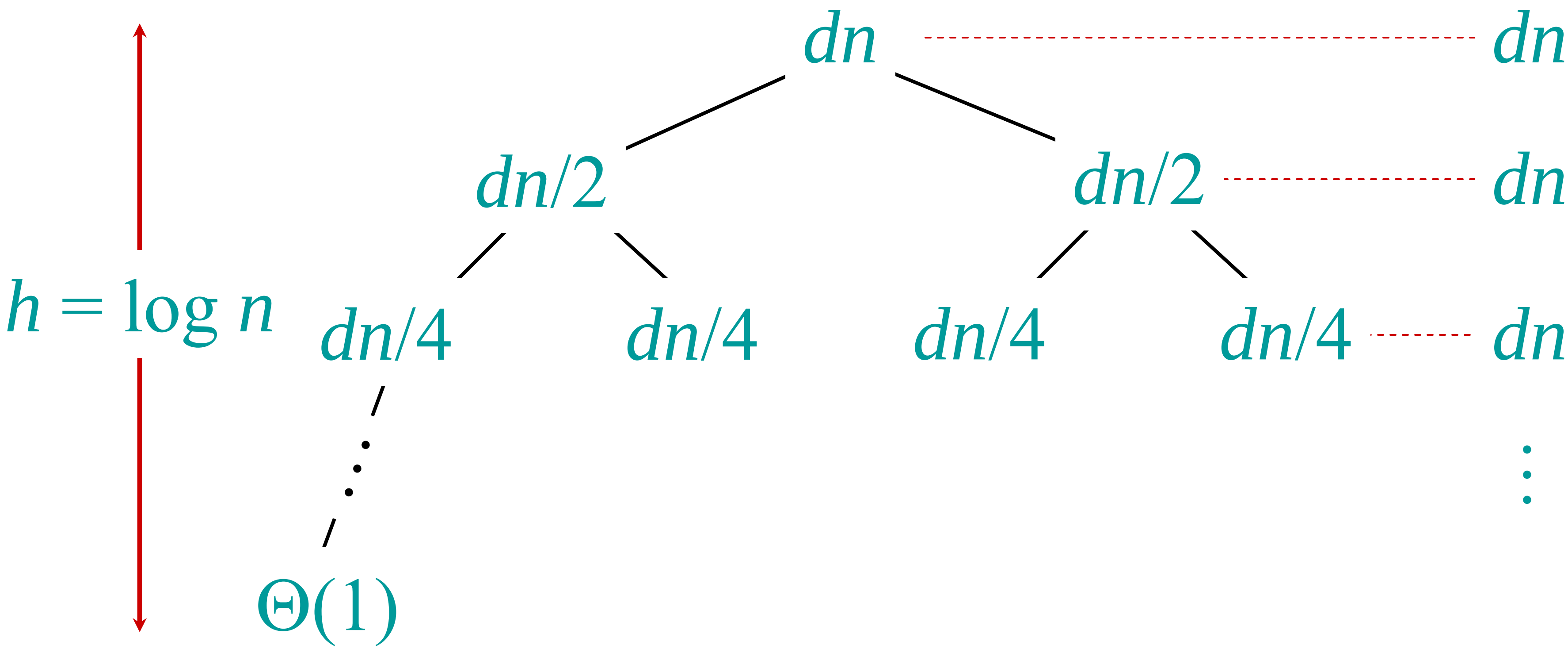
Recursion tree

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



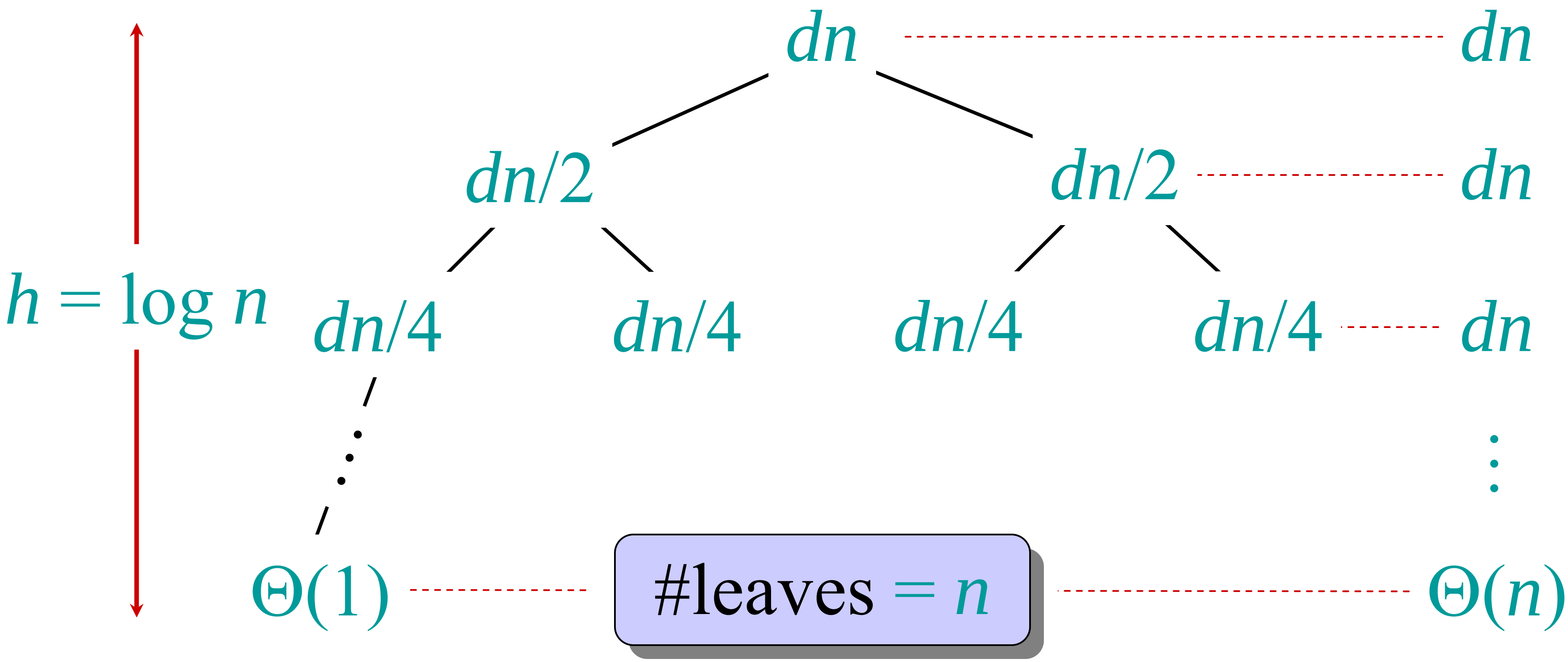
Recursion tree

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



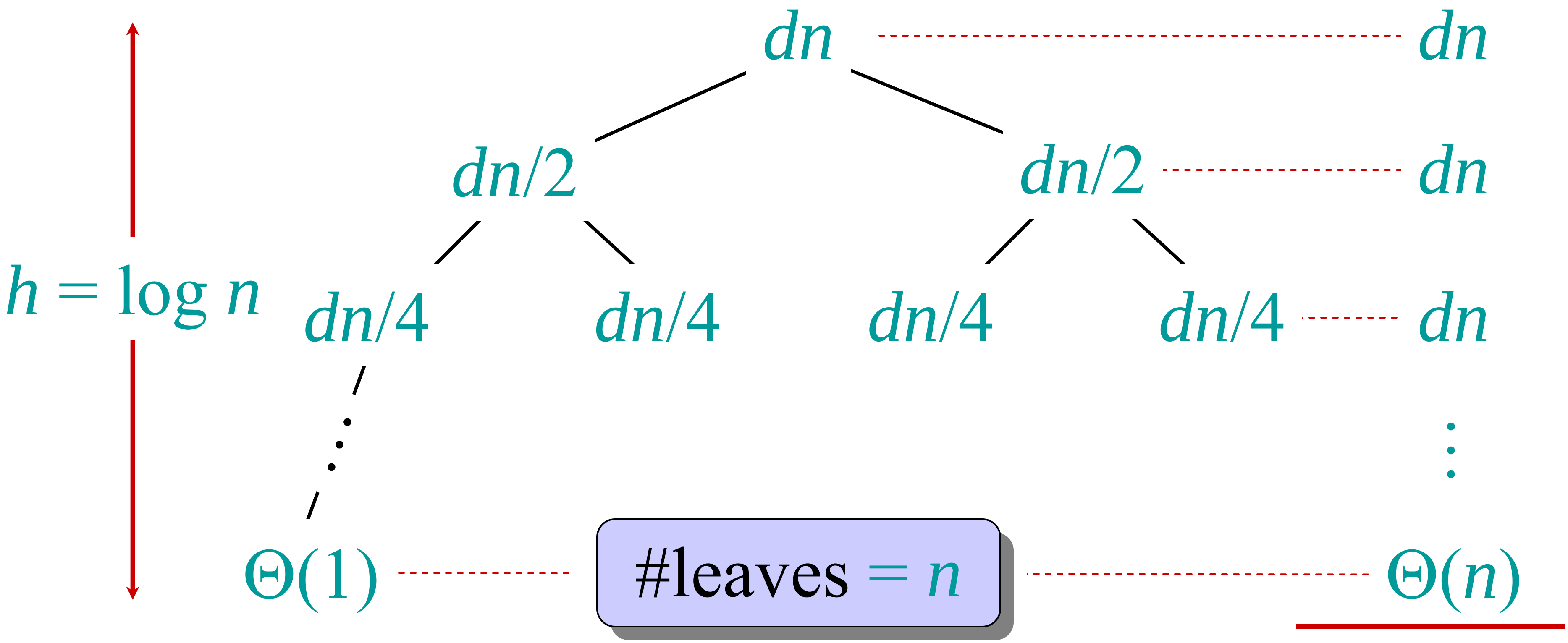
Recursion tree

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 $(k = 0)$ $\Rightarrow$ $T(n) = \Theta(n \log n)$.