Introduction to Algorithms

Divide and Conquer: Finding Root
Closest Pair of Points

Finding the Root of a Function

Finding the Root of a Function

Given a continuous function f and two points a < b such that

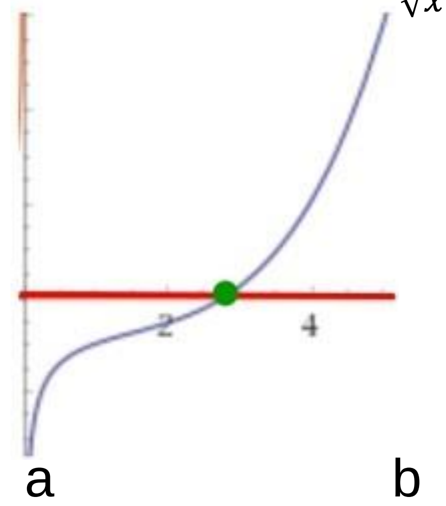
Assumption
$$\begin{cases} f(a) \le 0 \\ f(b) \ge 0 \end{cases}$$
$f(a)\,f(b) \le 0$

Find an approximate root of f (a point c where $f(c) = 0$).

return c s.t. $\exists x$ $|c-x| < \varepsilon$ $f(x)=0$

f has a root in $[a, b]$ by
intermediate value theorem

$f(x) = \sin(x) - \frac{100}{\sqrt{x}} + x^4$



Note that roots of f may be irrational,
So, we want to approximate
the root with an arbitrary precision!

A Naiive Approch

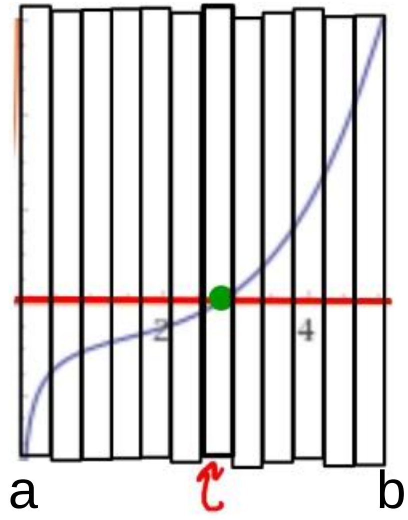
Suppose we want ϵ approximation to a root.

Divide [a,b] into $n = \frac{b-a}{\epsilon}$ intervals. For each interval check

$$f(x) \leq 0, f(x+\epsilon) \geq 0$$

This runs in time $O(n) = O(\frac{b-a}{\epsilon})$

Can we do faster?



D&C Approach (Based on Binary Search)

Bisection(a,b, ε)

  if $(b - a) < \epsilon$ then

    return (a)

  else

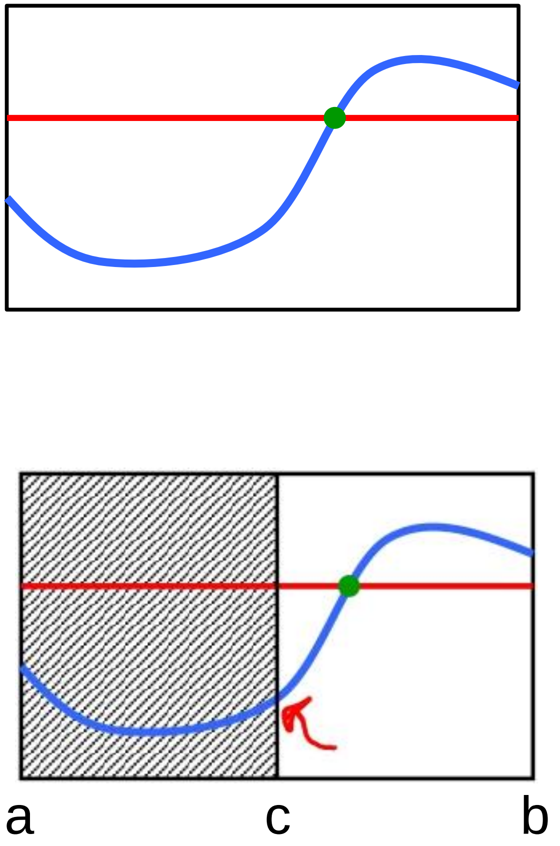
    $c \leftarrow (a + b)/2$

    if $f(c) \leq 0$ then

      return(Bisection(c, b, ε))

    else

      return(Bisection(a, c, ε))

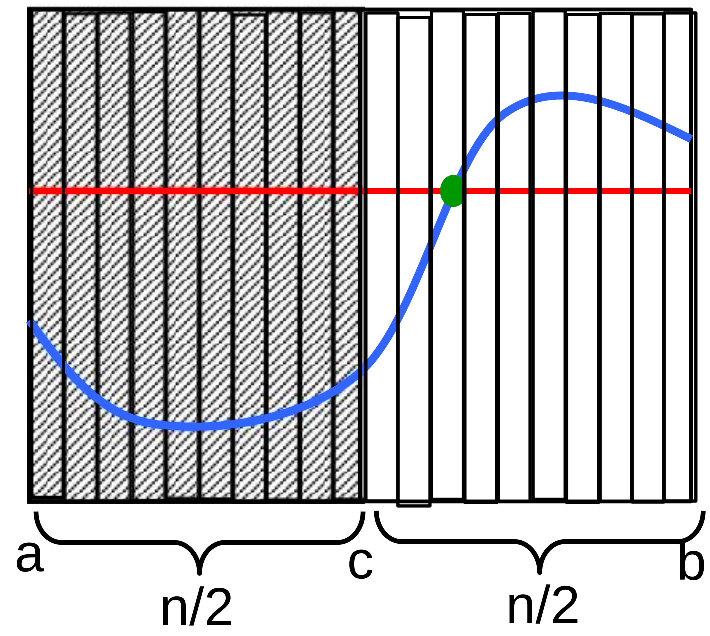


Time Analysis

Let n = $\frac{b-a}{\epsilon}$

And $c = (a + b)/2$

Always half of the intervals lie to the left and half lie to the right of c



So,

eval f(c)

c

$T(n) = T(\frac{n}{2}) + O(1)$

i.e., $T(n) = O(\log n) = O(\log \frac{a-b}{\epsilon})$

$T(n) = T(n/2) + c$

$= T(n/4) + c + c$

$= T(n/8) + c + c + c$

$c \cdot \lg n = \Theta(\lg n)$

Recurrences

Above: Where they come from, how to find them

Next: how to solve them

Master Theorem

Suppose $T(n) = aT(\frac{n}{b}) + cn^k$ for all $n > b$. Then,

- If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$
- If $a < b^k$ then $T(n) = \Theta(n^k)$
- If $a = b^k$ then $T(n) = \Theta(n^k \log n)$

Works even if it is $\left\lceil \frac{n}{b} \right\rceil$ instead of $\frac{n}{b}$.

We also need $a \geq 1, b > 1$, $k \geq 0$ and $T(n) = O(1)$ for $n \leq b$.

Master Theorem

Suppose $T(n) = aT(\frac{n}{b}) + cn^k$ for all $n > b$. Then,

- If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$
- If $a < b^k$ then $T(n) = \Theta(n^k)$
- If $a = b^k$ then $T(n) = \Theta(n^k \log n)$

Example: For mergesort algorithm we have

$$T(n) = 2T(\frac{n}{2}) + O(n).$$

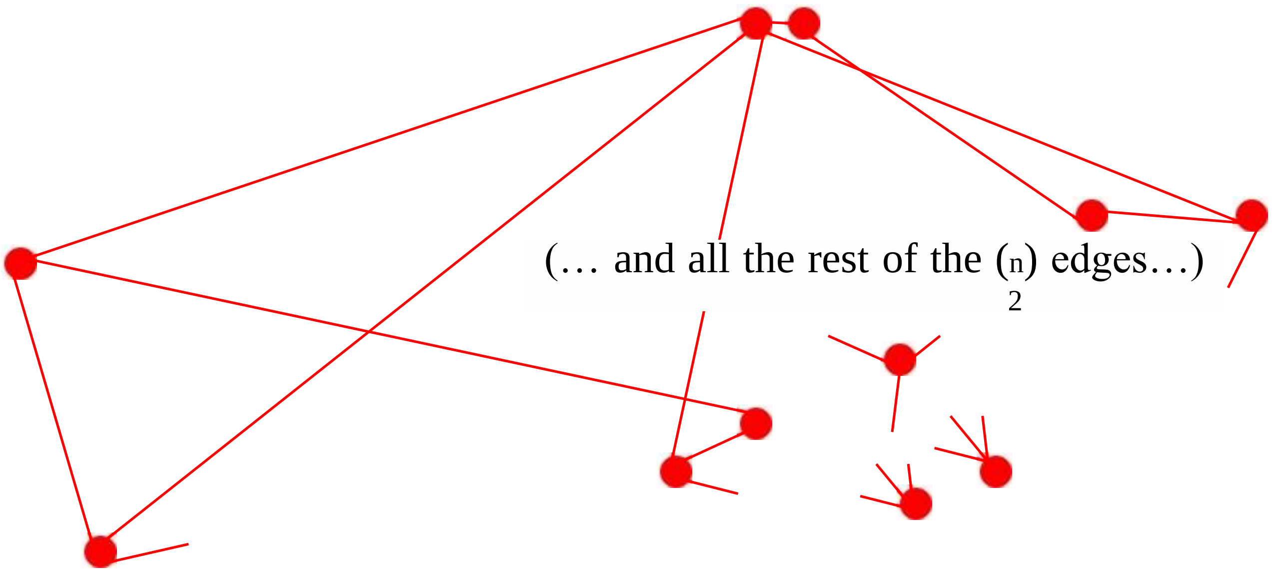
So, k = 1, a = b^k and $T(n) = \Theta(\mathrm{n} \log \mathrm{n})$

$T(n) = 5T(n/2) + cn^2$. $a=5, b=2, c=2$ $n^{\lg_2 5}$

Finding the Closest Pair of Points

Closest Pair of Points (non geometric)

Given n points and arbitrary distances between them, find the closest pair. (E.g., think of distance as airfare – definitely not Euclidean distance!)



Must look at all n choose 2 pairwise distances, else any one you didn't check might be the shortest.

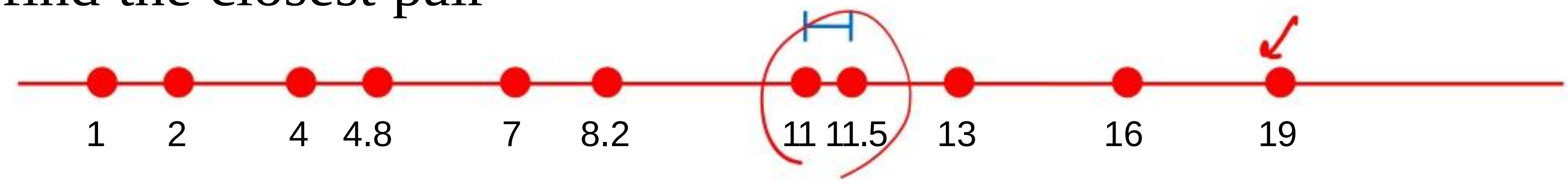
i.e., you have to read the whole input

Closest Pair of Points (1-dimension)

Given n points on the real line, find the closest pair,

e.g., given 11, 2, 4, 19, 4.8, 7, 8.2, 16, 11.5, 13, 1

find the closest pair



Fact: Closest pair is adjacent in ordered list

So, first sort, then scan adjacent pairs.

Time O(n log n) to sort, if needed, Plus O(n) to scan adjacent pairs

Key point: do *not* need to calc distances between all pairs: exploit geometry + ordering

Closest Pair of Points (2-dimensions)

Given n points in the plane, find a pair with smallest Euclidean distance between them.

(1, 2)
(3, 1)
(-1, 5)

Fundamental geometric primitive.

- Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.
- Special case of nearest neighbor, Euclidean MST, Voronoi.

Brute force: Check all pairs of points p and q with $\Theta(n^2)$ time.

Assumption: No two points have same x coordinate.

y

A Divide and Conquer Alg

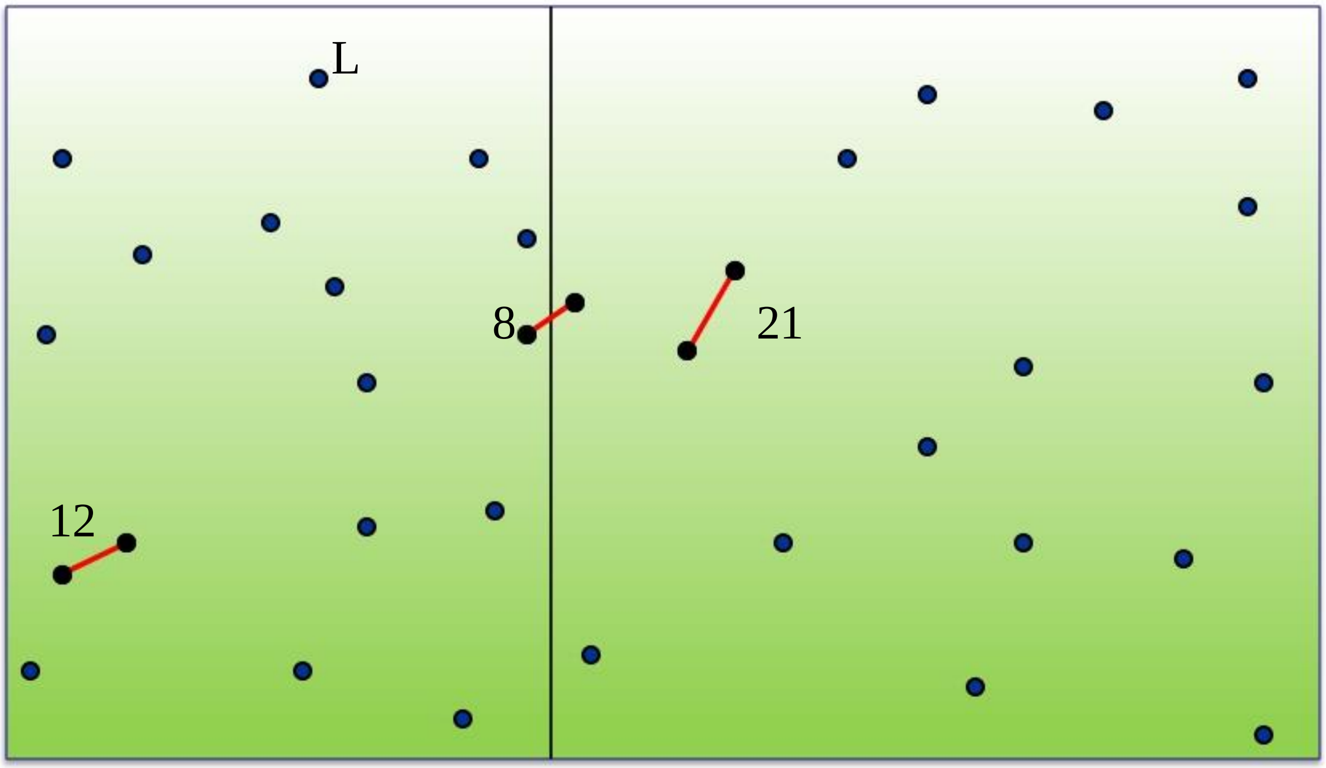
Divide: draw vertical line L with ≈ **n/2 points on each** side.

Conquer: find closest pair on each side, recursively.

Combine to find closest pair overall

seems like $\Theta(n^2)$?

Return best solutions



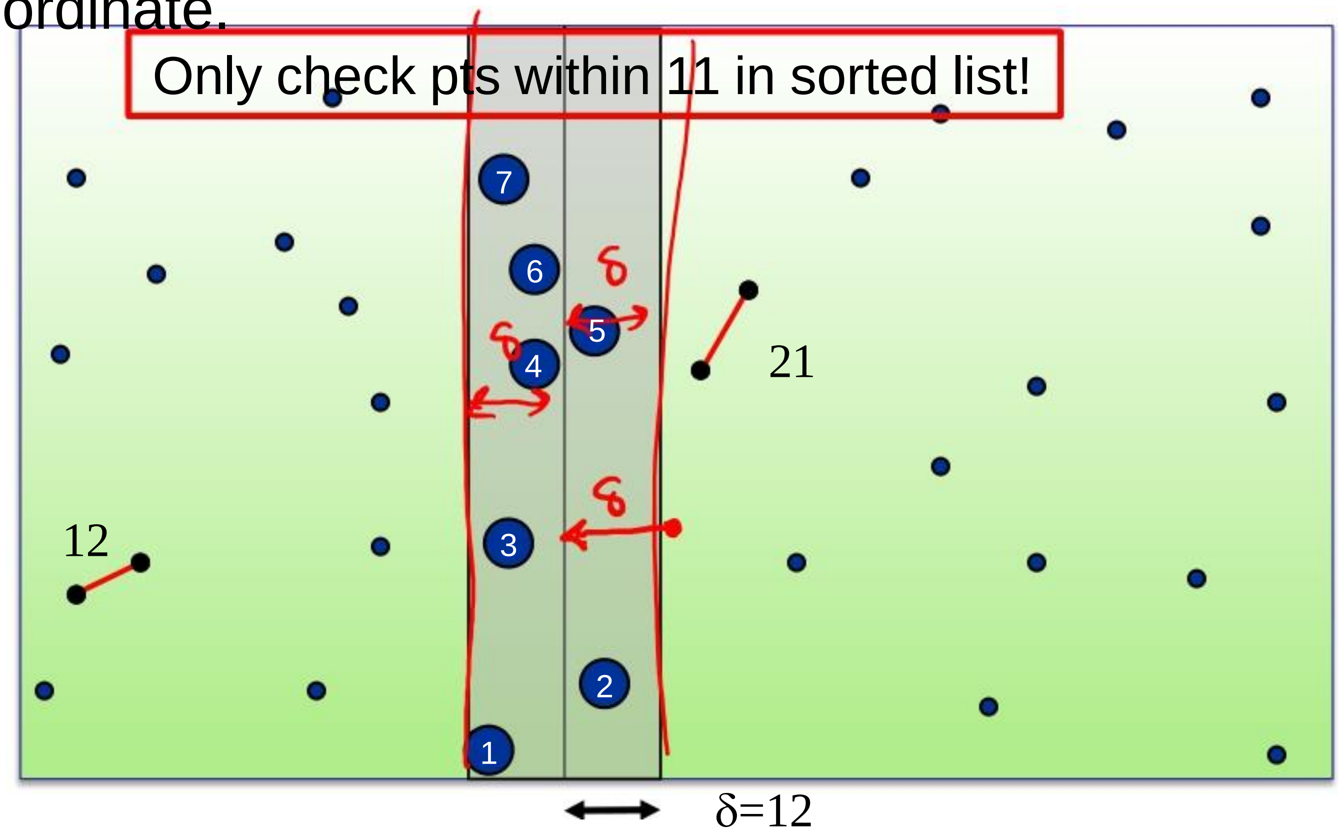
Key Observation

Suppose δ is the minimum distance of all pairs in left/right of L.

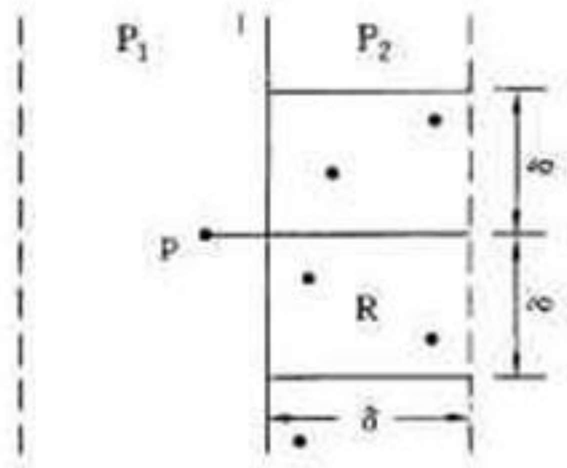
$$\delta = \min(12, 21) = 12.$$

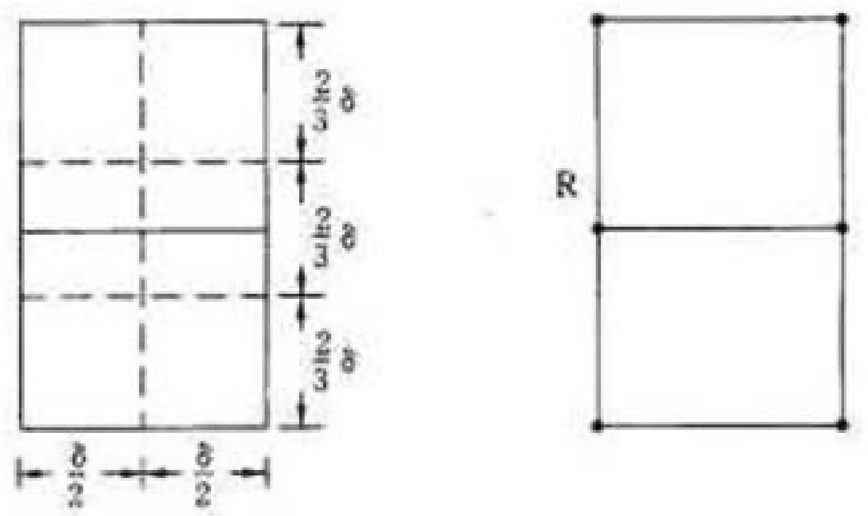
Key Observation: suffices to consider points within δ of line L.

Almost the one-D problem again: Sort points in 2δ-strip by their y coordinate.



Key Observation





(a) (b)

Closest Pair (2Dim Algorithm)

```
Closest-Pair(p1, …, pn) {
   if(n <= ??) return ??

   Compute separation line L such that half the points
   are on one side and half on the other side.

   δ1 = Closest-Pair(left half)
   δ2 = Closest-Pair(right half)
   δ  = min(δ1, δ2)

   Delete all points further than δ from separation line L

   Sort remaining points p[1]…p[m] by y-coordinate.

   for i = 1..m
      Check nearest 6 nodes on the opposite size of p[i];
      Determine a shortest distance δ’ ;
      δ = min(δ, δ’ );

   return δ.
}
```

Closest Pair Analysis I

Let D(n) be the number of pairwise distance calculations in the Closest-Pair Algorithm when run on n ≥ 1 points

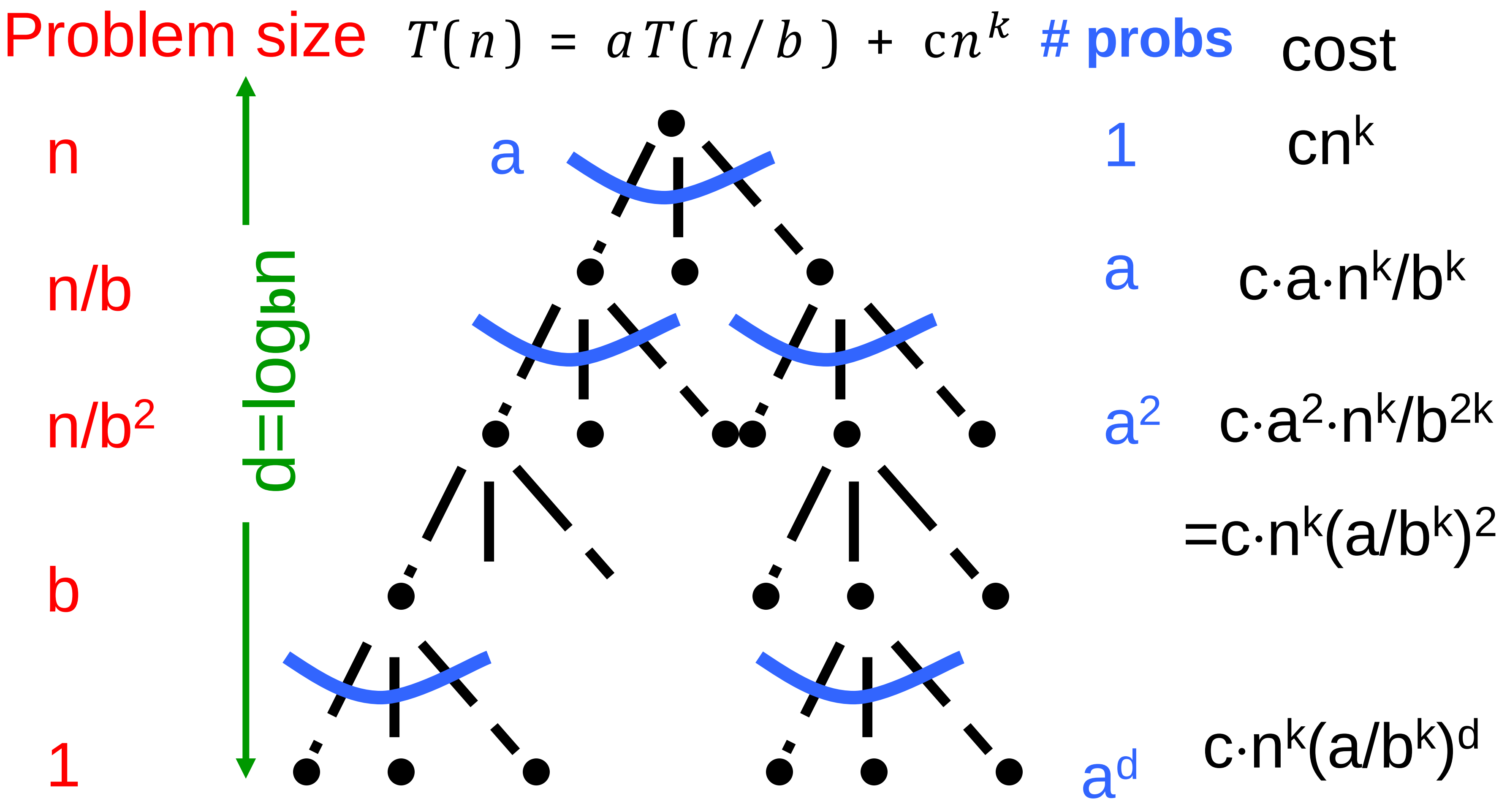
$$D(n) \le \begin{cases} 1 & \text{if } n = 1 \\ 2D\left(\frac{n}{2}\right) + 6n & \text{o.w.} \end{cases} \Rightarrow D(n) = O(n \log n)$$

BUT, that's only the number of *distance calculations*

What if we counted running time?

$$T(n) \le \begin{cases} 1 & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + O(n \log n) & \text{o.w.} \end{cases} \Rightarrow D(n) = O(n \log n)$$

Proving Master Theorem



$$T(n) = cn^k \sum_{i=0}^{d=\log_b n} \left(\frac{a}{b^k}\right)^i$$

A Useful Identity

Theorem: $1 + x + x^2 + \cdots + x^d = \frac{x^{d+1}-1}{x-1}$

Pf: Let $S = 1 + x + x^2 + \cdots + x^d$

Then, $xS = x + x^2 + \cdots + x^{d+1}$

So, $xS - S = x^{d+1} - 1$

i.e., $S(x - 1) = x^{d+1} - 1$

Therefore,

$$S = \frac{x^{d+1}-1}{x-1}$$

Solve: $T(n) = aT(\frac{n}{b}) + cn^k, a > b^k$

$$T(n) = cn^k \sum_{i=0}^{\log_b n} \left(\frac{a}{b^k}\right)^i$$

$$\frac{x^{d+1}-1}{x-1} \text{ for } x = \frac{a}{b^k}$$

$d = \log_b n$

using $x \neq 1$

$$= cn^k \frac{\left(\frac{a}{b^k}\right)^{\log_b n + 1} - 1}{\left(\frac{a}{b^k}\right) - 1}$$

$b^{k\log_b n}$

$= (b^{\log_b n})^k$

$= n^k$

$$\leq c\left(\frac{n^k}{b^{k\log_b n}}\right) \frac{\left(\frac{a}{b^k}\right)}{\left(\frac{a}{b^k}\right) - 1} a^{\log_b n}$$

$a^{\log_b n}$

$= (b^{\log_b a})^{\log_b n}$

$= (b^{\log_b n})^{\log_b a}$

$= n^{\log_b a}$

$$\leq ca^{\log_b n} = O(n^{\log_b a})$$

Solve: $T(n) = aT(\frac{n}{b}) + cn^k, a = b^k$

$$T(n) = cn^k \sum_{i=0}^{\log_b n} (\frac{a}{b^k})^i$$

$$= cn^k \log_b n$$

Master Theorem

Suppose $T(n) = aT(\frac{n}{b}) + cn^k$ for all $n > b$. Then,

- If $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$
- If $a < b^k$ then $T(n) = \Theta(n^k)$
- If $a = b^k$ then $T(n) = \Theta(n^k \log n)$

Works even if it is $\left\lceil \frac{n}{b} \right\rceil$ instead of $\frac{n}{b}$.

We also need $a \geq 1, b > 1$, $k \geq 0$ and $T(n) = O(1)$ for $n \leq b$.