CSE 421

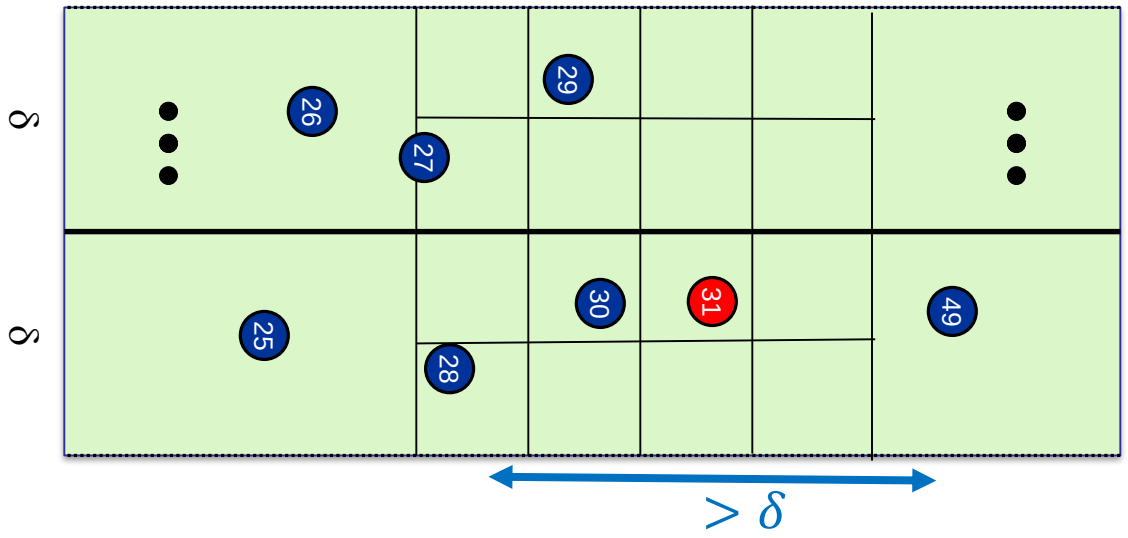
Master Theorem

Yin Tat Lee

Almost 1D Problem

Partition each side of L into $\frac{\delta}{2} \times \frac{\delta}{2}$ squares

Claim: No two points lie in the same $\frac{\delta}{2} \times \frac{\delta}{2}$ box.



Say we want to check if there is a point < δ close to point 31.

Only need to check points with indices roughly 31. (coz we sorted).

How many? Not too many coz every box only have 1 point.

Closest Pair (2 dimension)

```
Closest-Pair(p_1, p_2, ..., p_n) {
   if(n ≤ 2) return |p_1 − p_2|

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

   δ_1 = Closest-Pair(left half)
   δ_2 = Closest-Pair(right half)
   δ   = min(δ_1, δ_2)

   Delete all points further than δ from separation line L

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

   for i = 1, 2, ..., m
      for k = 1, 2, ..., 11
        if i + k ≤ m
            δ = min(δ, distance(p[i], p[i+k]));

   return δ.
}
```

Closest Pair Analysis

Let $D(n)$ be the number of pairwise distance calculations in the Closest-Pair Algorithm

$$D(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ 2D\left(\frac{n}{2}\right) + 11\,n & \text{o. w.} \end{cases} \Rightarrow D(n) = \mathrm{O}(n\log n)$$

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

What if we counted running time?

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

Closest Pair (2 dimension) Improved

```
Closest-Pair(p_1, p_2, ..., p_n) {
    if(n ≤ 2) return |p_1 − p_2|

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

   (δ_1, p_1) = Closest-Pair(left half)
   (δ_2, p_2) = Closest-Pair(right half)
    δ         = min(δ_1, δ_2)
    p_sorted  = merge(p_1, p_2)   (merge sort it by y-coordinate)

    Let q be points (ordered as p_sorted) that is δ from line L.

    for i = 1, 2, ..., m
        for k = 1, 2, ..., 11
          if i + k ≤ m
              δ = min(δ, distance(q[i], q[i+k]));

    return δ and p_sorted.
}
```

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

$$\Rightarrow D(n) = O(n \log n)$$

Master Theorem

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

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

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

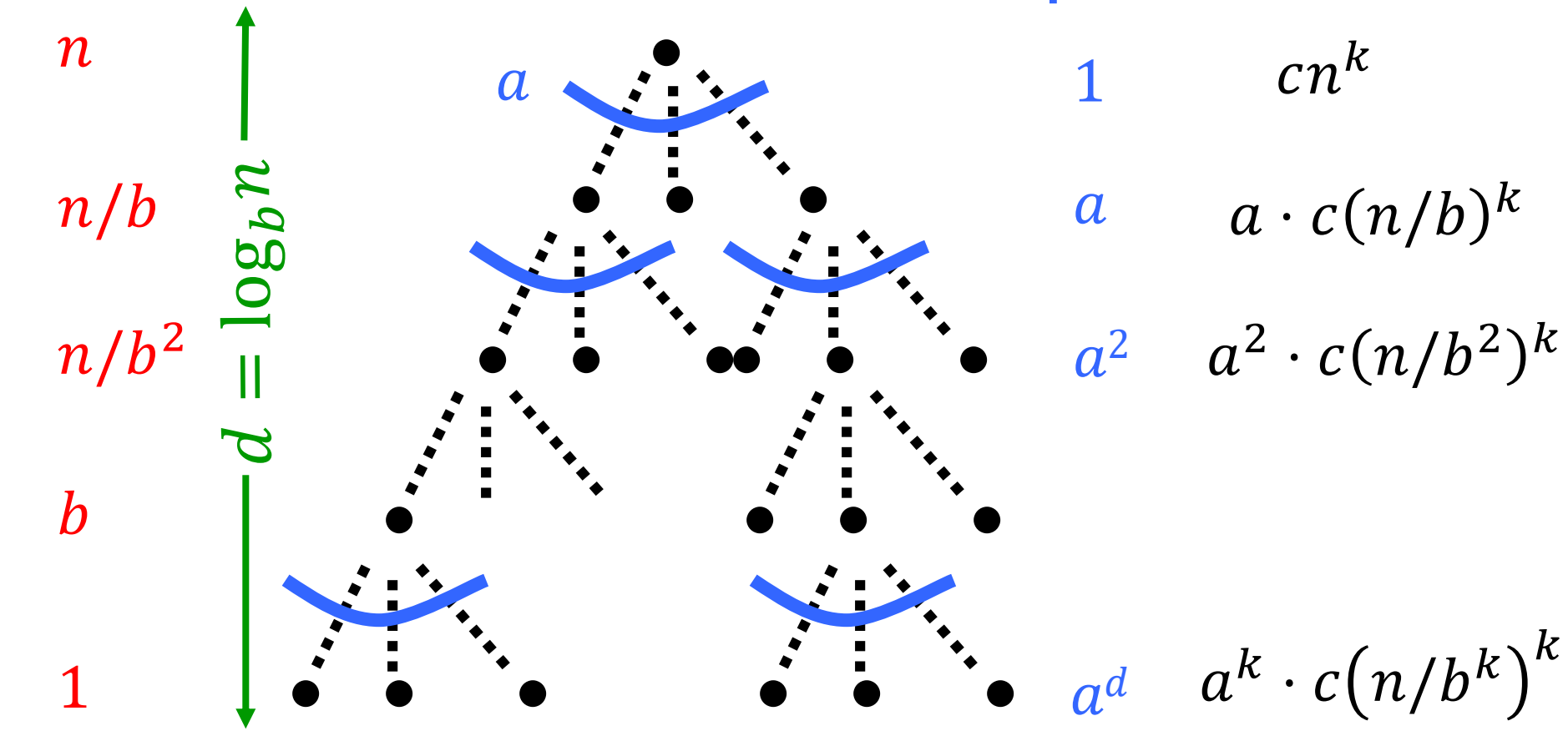
- If $a < b^k$ then $T(n) = \Theta\left(n^k\right)$
- If $a = b^k$ then $T(n) = \Theta\left(n^k \log n\right)$
- If $a > b^k$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

Example: For mergesort algorithm we have

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

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

Proving Master Theorem



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

Master Theorem

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

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

# of problems increases slower than the decreases of cost. First term dominates.

# of problems increases faster than the decreases of cost Last term dominates.

A Useful Identity

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

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

Corollary:

$$1 + x + x^2 + \cdots + x^d = \begin{cases} O_x(1) & \text{if } x < 1 \\ d+1 & \text{if } x = 1 \\ O_x\left(x^{d+1}\right) & \text{if } x > 1 \end{cases}$$

O_x means the hidden constant depends on x

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

Corollary:

$$1 + x + x^2 + \cdots + x^d = \begin{cases} \Theta_x(1) & \text{if } x < 1 \\ \Theta(d) & \text{if } x = 1 \\ \Theta_x\left(x^{d+1}\right) & \text{if } x > 1 \end{cases}$$

Going back, we have

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

Hence, we have

$$T(n) = \Theta\left(n^k\right) \begin{cases} 1 & \text{if } a < b^k \\ \log_b n & \text{if } a = b^k \\ \left(\frac{a}{b^k}\right)^{\log_b n} & \text{if } a > b^k \end{cases}$$

constant depends on a, b, c
When we can hide constant?

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

$$T(n) = \Theta\left(n^k\right)\begin{cases} 1 & \text{if } a < b^k \\ \log_b n & \text{if } a = b^k \\ \left(\frac{a}{b^k}\right)^{\log_b n} & \text{if } a > b^k \end{cases}$$

For $a < b^k$, we simply have $T(n) = \Theta\left(n^k\right)$.

For $a = b^k$, we have $T(n) = \Theta\left(n^k \log_b n\right) = \Theta(n^k \log n)$.

For $a > b^k$, we have $T(n) = \Theta\left(n^k \left(\frac{a}{b^k}\right)^{\log_b n}\right) = \Theta(n^{\log_b a})$.

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

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

Example