# The Master Method

If $T(n) \leq aT\left(\frac{n}{b}\right) + O(n^d)$

then

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \quad \text{(Case 1)} \\ O(n^d) & \text{if } a < b^d \quad \text{(Case 2)} \\ O(n^{\log_b a}) & \text{if } a > b^d \quad \text{(Case 3)} \end{cases}$$

# Preamble

**Assume:** recurrence is

(i) $T(1) \leq c$

(ii) $T(n) \leq aT\left(\frac{n}{b}\right) + cn^d$

(for some constant $c$)

**and** $n$ is a power of $b$.

(general case is similar, but more tedious)

**Idea:** generalize MergeSort analysis.

(i.e., use a recursion tree)

# How To Think About (*)

Our upper bound on the work at level j:

$$cn^d \times (\frac{a}{b^d})^j$$

Interpretation

$a$ = rate of subproblem proliferation (RSP)

$b^d$ = rate of work shrinkage (RWS)
(per subproblem)

# Intuition for the 3 Cases

Upper bound for level j: $cn^d \times (\frac{a}{b^d})^j$

(1) RSP = RWS => Same amount of work each level (like Merge Sort) [expect $O(n^d \log n)$]

(2) RSP < RWS => less work each level => most work at the root [might expect $O(n^d)$]

(3) RSP > RWS => more work each level => most work at the leaves [might expect O(# leaves)]

# The Story So Far/Case 1

Total work: $\leq \underline{cn^d} \times \underbrace{\sum_{j=0}^{\log_b n} \left(\frac{a}{b^d}\right)^j}_{= (\log_b n + 1)}$ (*)

$\frac{a}{b^d} = 1$

$\left(\frac{a}{b^d}\right)^j = 1$ for all $j$

If $a = b^d$, then

$(*) = cn^d(\log_b n + 1)$

$= O(n^d \log n)$

Tim Roughgarden

# Case 2

Total work: $\leq cn^d \times \sum_{j=0}^{\log_b n} (\frac{a}{b^d})^j$ (*)

$\frac{a}{b^d} := r$

$\leq$ a constant (independent of n)

[by basic sums fact]

If $a < b^d$ [RSP < RWS]

$= O(n^d)$

[total work dominated by top level]

# Case 3

Total work: $\leq cn^d \times \sum_{j=0}^{\log_b n} \left(\frac{a}{b^d}\right)^j$ (*)

$\left(\frac{a}{b^d}\right) := r > 1$

$\hookrightarrow \leq$ constant * largest term

If $a > b^d$ [RSP > RWS]

then $(*) = O\left(n^d \cdot \left(\frac{a}{b^d}\right)^{\log_b n}\right)$

Note: $b^{-d\log_b n} = \left(\cancel{b^{\log_b n}}\right)^{-d} = n^{-d}$

So: $(*) = O\left(a^{\log_b n}\right)$