CSE 548: Analysis of Algorithms

Lecture 6
(Divide-and-Conquer Algorithms: Some Applications of the Fourier Transform & the Master Theorem)

Rezaul A. Chowdhury

Department of Computer Science
SUNY Stony Brook
Fall 2012

Some Applications of Fourier Transform and FFT

- Signal processing
- Image processing
- Noise reduction
- Data compression
- Solving partial differential equation
- Multiplication of large integers
- Polynomial multiplication
- Molecular docking

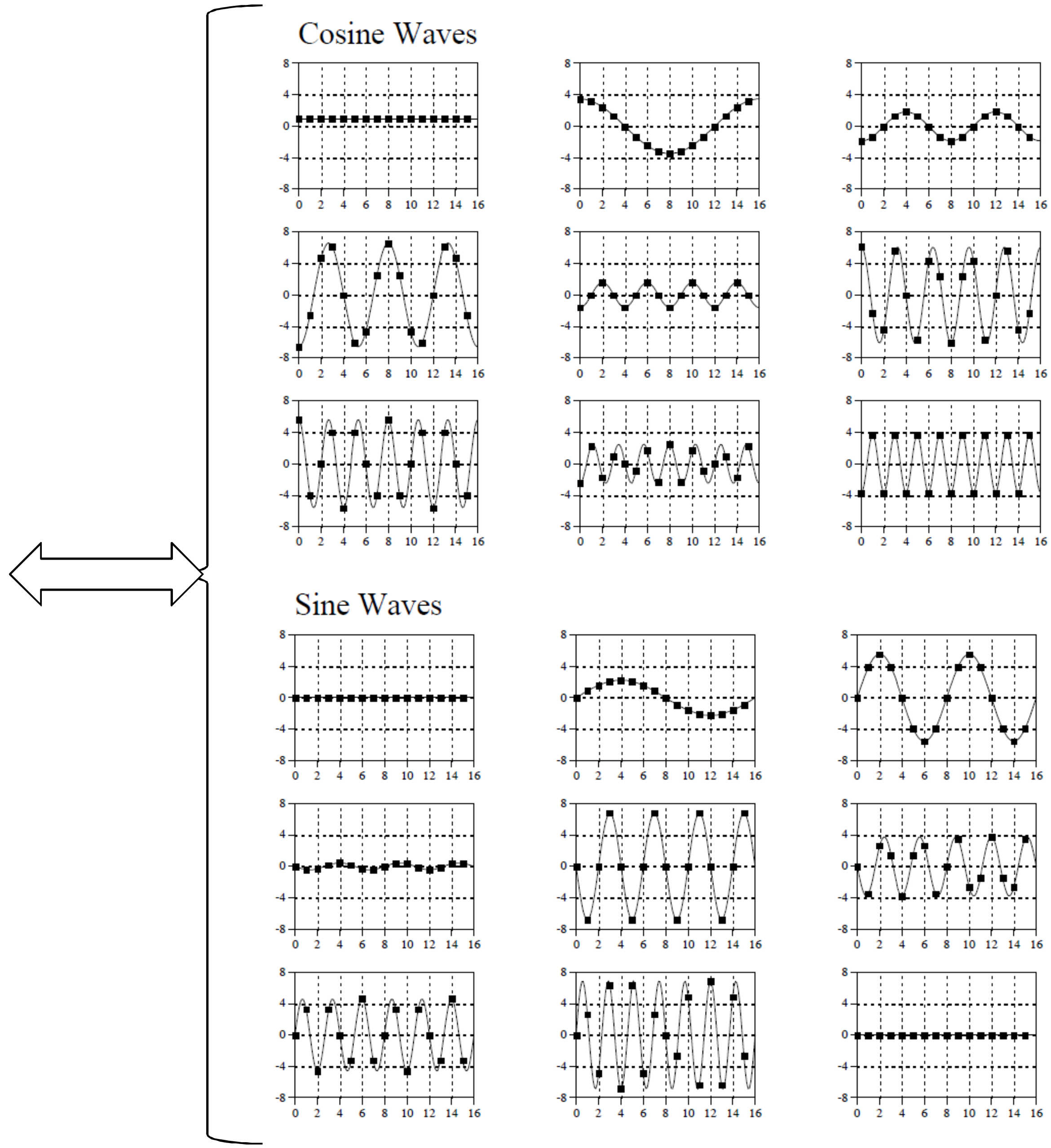
Some Applications of Fourier Transform and FFT

Jean Baptiste Joseph Fourier

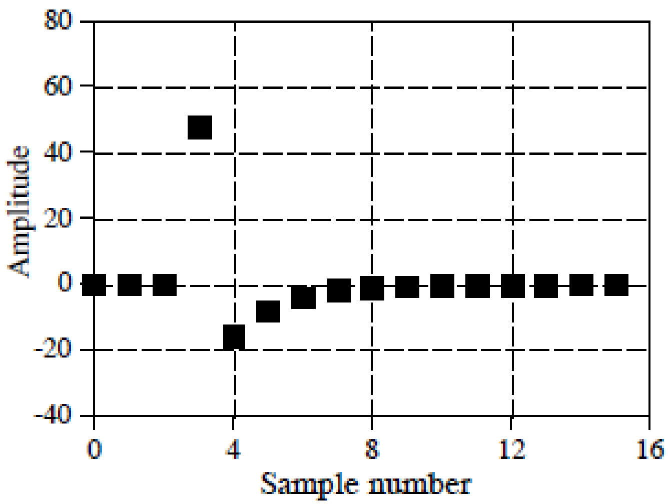
Any periodic signal can be represented as a sum of a series of sinusoidal (sine & cosine) waves. [1807]

Spatial (Time) Domain ⇔ Frequency Domain

Frequency Domain



Spatial (Time) Domain



Source: The Scientist and Engineer's Guide to Digital Signal Processing by Steven W. Smith

Spatial (Time) Domain $\Leftrightarrow$ Frequency Domain (Fourier Transforms)

Let $s(t)$ be a signal specified in the time domain.

The strength of $s(t)$ at frequency f is given by:

$$S(f) = \int_{-\infty}^{\infty} s(t) \cdot e^{-2\pi i f t}\, dt$$

Evaluating this integral for all values of f gives the frequency domain function.

Now $s(t)$ can be retrieved by summing up the signal strengths at all possible frequencies:

$$s(t) = \int_{-\infty}^{\infty} S(f) \cdot e^{2\pi i f t}\, df$$

Why do the Transforms Work?

Let's try to get a little intuition behind why the transforms work.

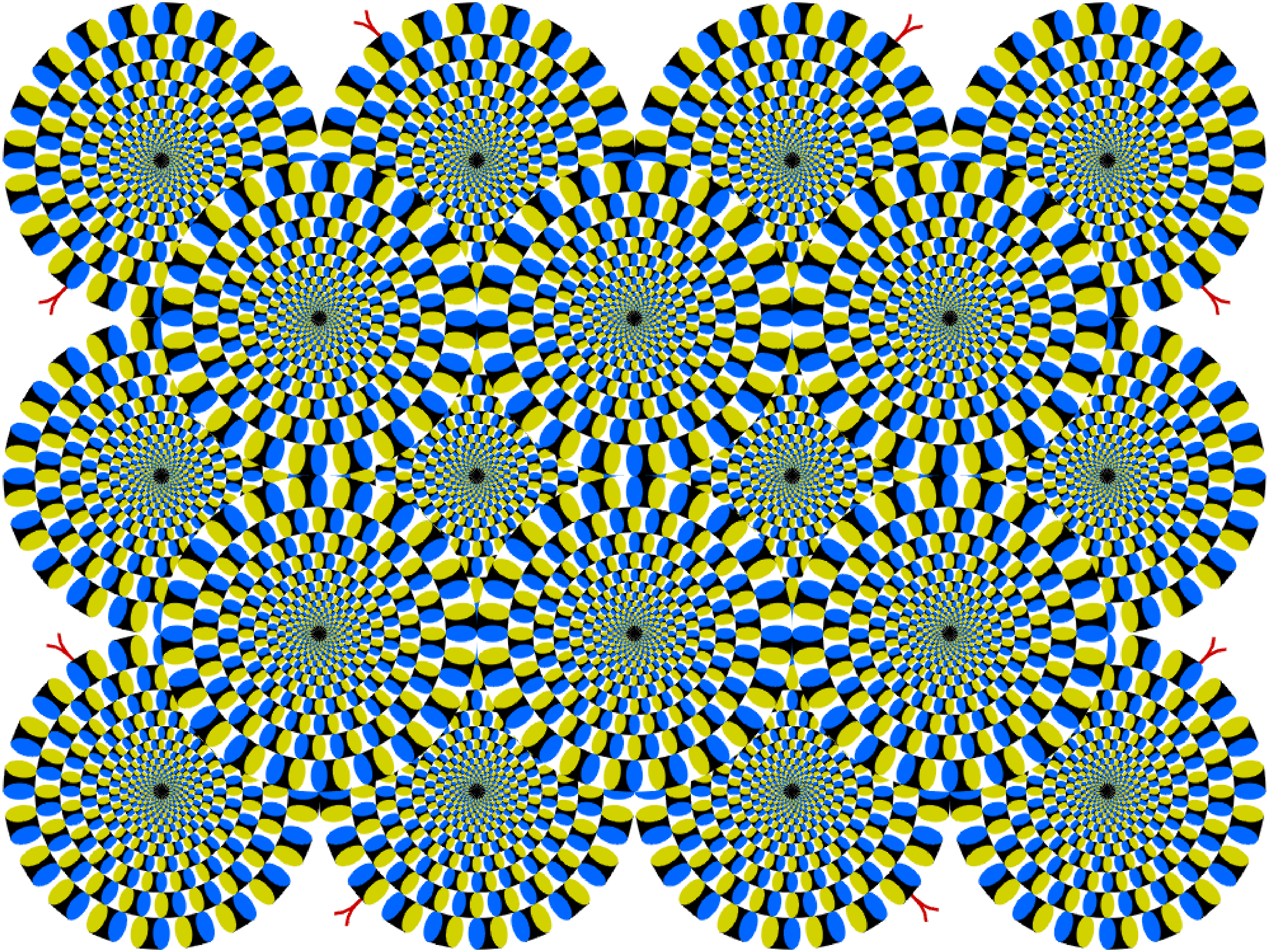
We will look at a very simple example.

Suppose: $s(t) = \cos(2\pi h \cdot t)$

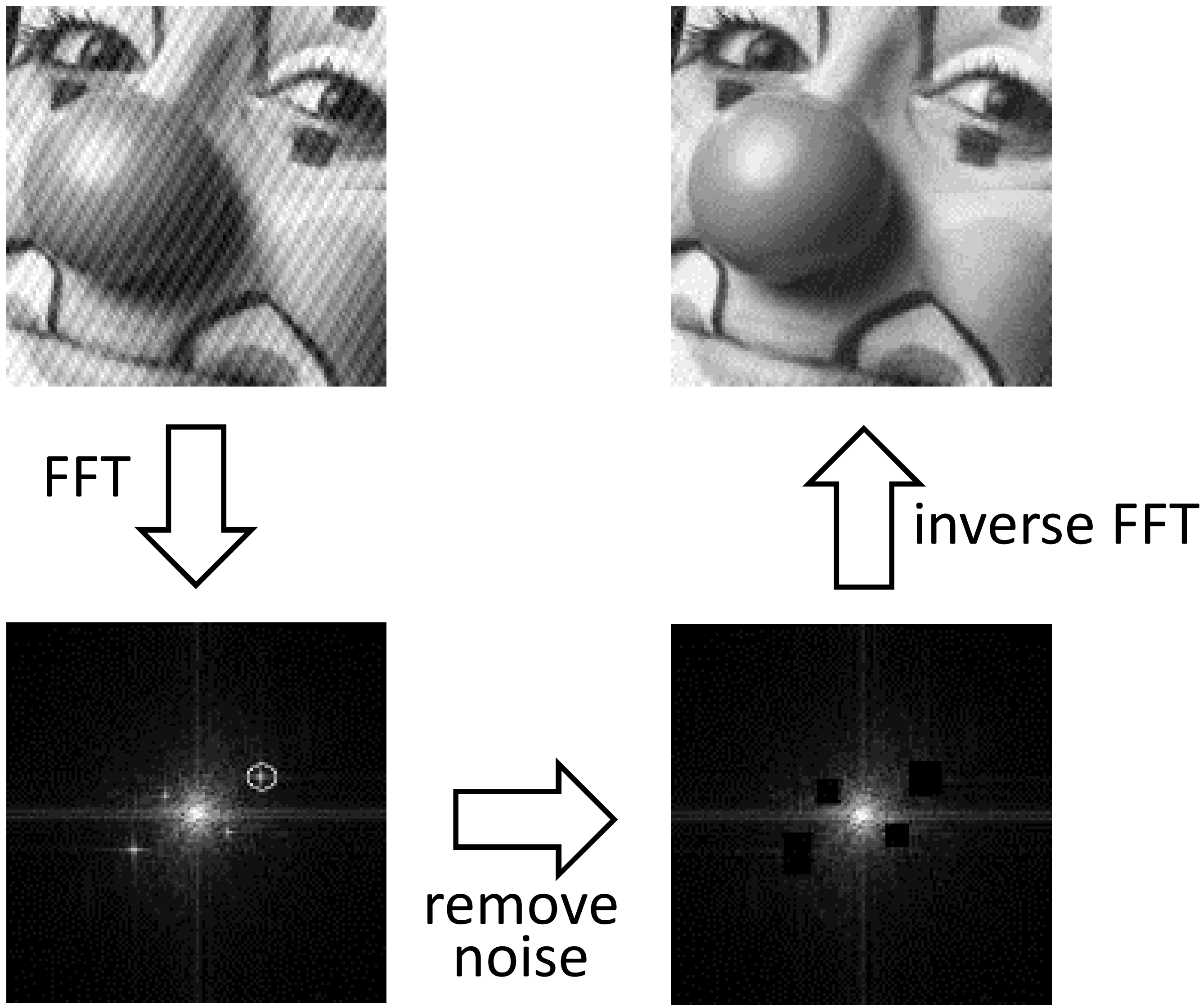
$$\frac{1}{T}\int_{-T}^{T} s(t) \cdot e^{-2\pi i f t}\, dt = \begin{cases} 1 + \dfrac{\sin(4\pi f T)}{4\pi f T}, & \text{if } f = h, \\ \dfrac{\sin(2\pi(h-f)T)}{2\pi(h-f)T} + \dfrac{\sin(2\pi(h+f)T)}{2\pi(h+f)T}, & \text{otherwise.} \end{cases}$$

$$\Rightarrow \lim_{T\to\infty}\left(\frac{1}{T}\int_{-T}^{T} s(t) \cdot e^{-2\pi i f t}\, dt\right) = \begin{cases} 1, & \text{if } f = h, \\ 0, & \text{otherwise.} \end{cases}$$

So, the transform can detect if $f = h$!



Noise Reduction



Source: http://www.mediacy.com/index.aspx?page=AH_FFTExample

Data Compression

- Discrete Cosine Transforms (DCT) are used for lossy data compression (e.g., MP3, JPEG, MPEG)
- DCT is a Fourier-related transform similar to DFT (Discrete Fourier Transform) but uses only real data (uses cosine waves only instead of both cosine and sine waves)
- Forward DCT transforms data from spatial to frequency domain
- Each frequency component is represented using a fewer number of bits (i.e., truncated / quantized)
- Low amplitude high frequency components are also removed
- Inverse DCT then transforms the data back to spatial domain
- The resulting image compresses better

Data Compression

Transformation to frequency domain using cosine transforms work in the same way as the Fourier transform.

Suppose: $s(t) = \cos(2\pi h \cdot t)$

$$\frac{1}{T}\int_{-T}^{T} s(t)\cdot\cos(2\pi f t)\,dt = \begin{cases} 1 + \dfrac{\sin(4\pi f T)}{4\pi f T}, & \text{if } f = h, \\ \dfrac{\sin(2\pi(h-f)T)}{2\pi(h-f)T} + \dfrac{\sin(2\pi(h+f)T)}{2\pi(h+f)T}, & \text{otherwise.} \end{cases}$$

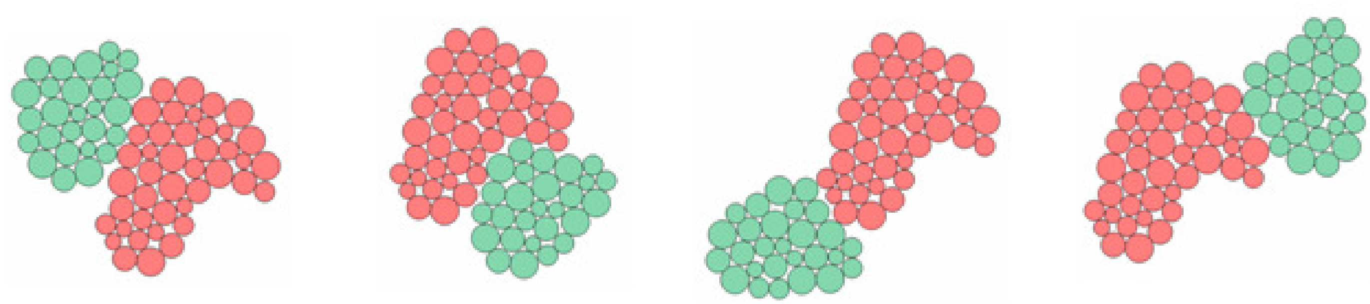
$$\Rightarrow \lim_{T\to\infty}\left(\frac{1}{T}\int_{-T}^{T} s(t)\cdot\cos(2\pi f t)\,dt\right) = \begin{cases} 1, & \text{if } f = h, \\ 0, & \text{otherwise.} \end{cases}$$

So, this transform can also detect if $f = h$.

Protein-Protein Docking

- Knowledge of complexes is used in
  - Drug design
  - Studying molecular assemblies
  - Structure function analysis
  - Protein interactions

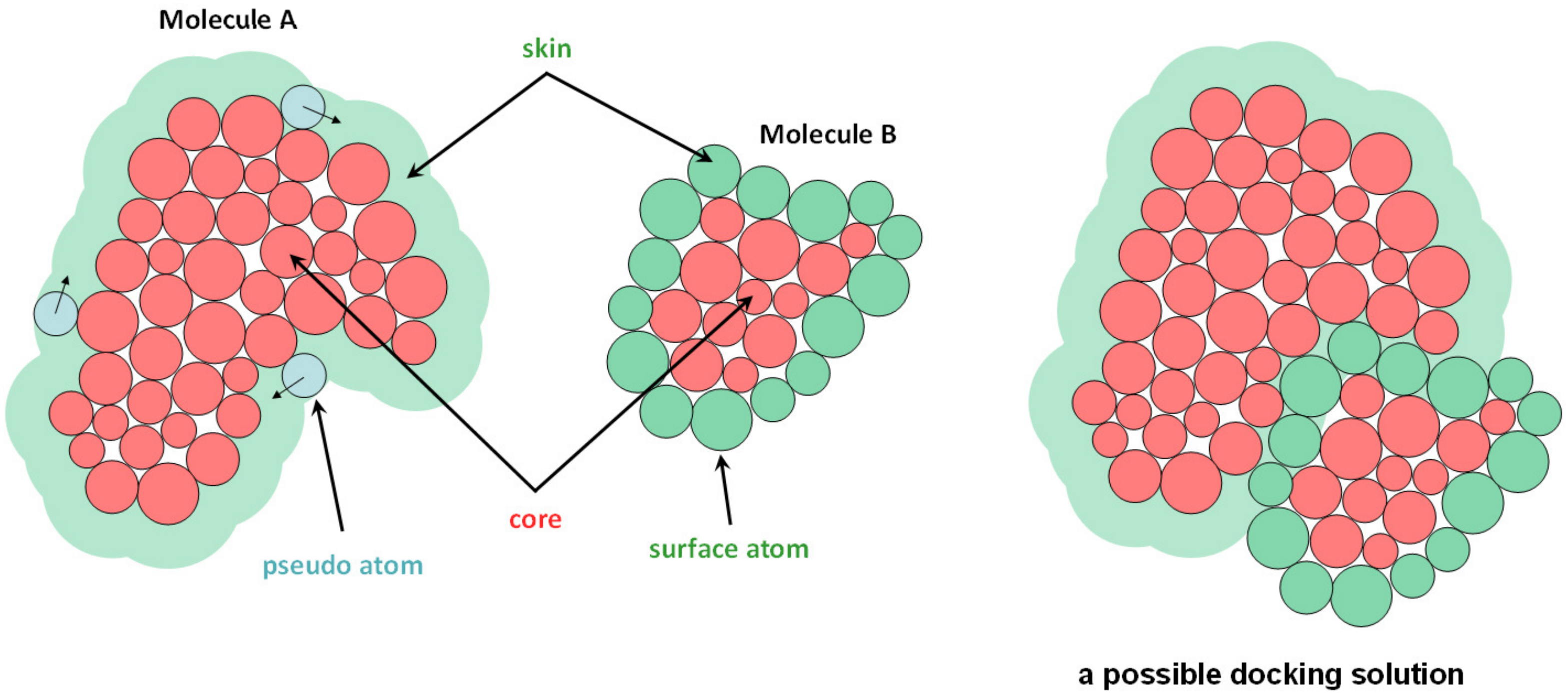
- Protein-Protein Docking: Given two proteins, find the best relative transformation and conformations to obtain a stable complex.



- Docking is a hard problem
  - Search space is huge (6D for rigid proteins)
  - Protein flexibility adds to the difficulty

Shape Complementarity

[Wang'91, Katchalski-Katzir et al.'92, Chen et al.'03]



To maximize skin-skin overlaps and minimize core-core overlaps

- assign positive real weights to skin atoms
- assign positive imaginary weights to core atoms

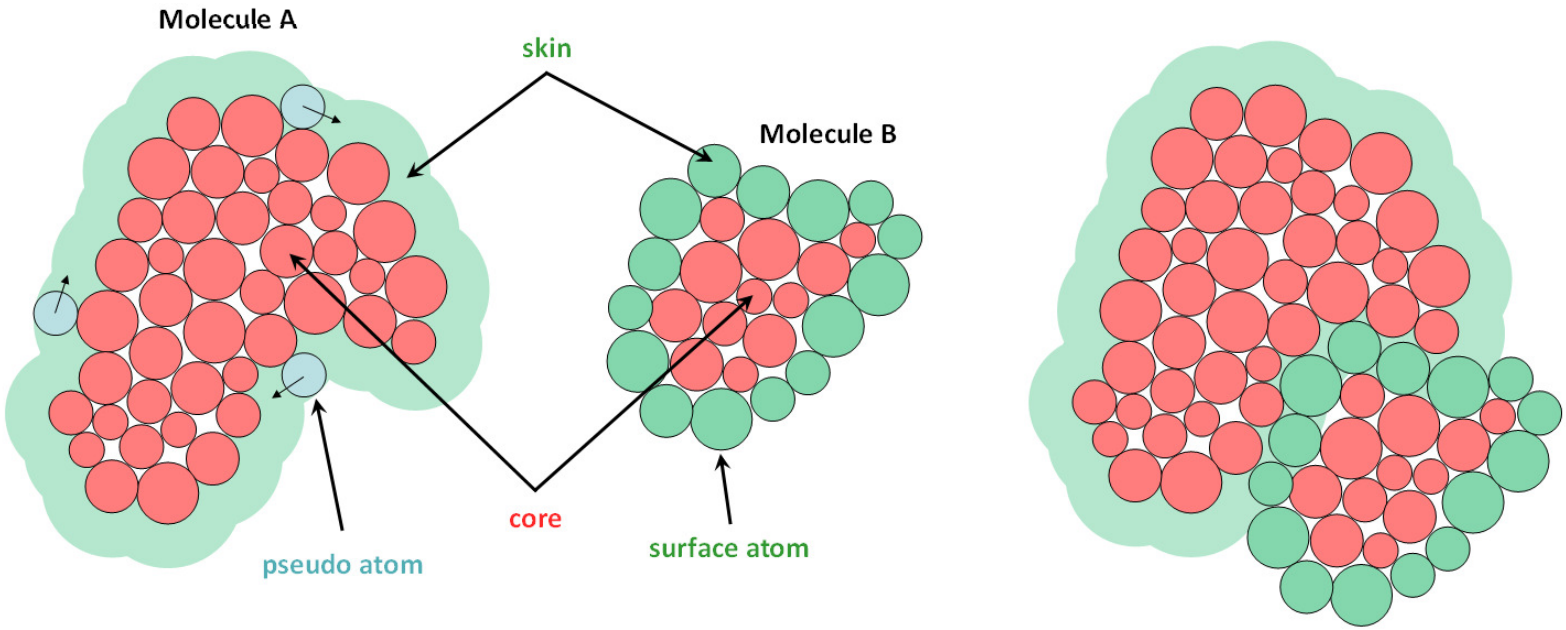
Let A' denote molecule A with the pseudo skin atoms.

For $P \in \{A', B\}$ with M_P atoms, affinity function: $f_P(x) = \sum_{k=1}^{M_P} w_k \cdot g_k(x)$

Here $g_k(x)$ is a Gaussian representation of atom k, and w_k its weight.

Shape Complementarity

[Wang'91, Katchalski-Katzir et al.'92, Chen et al.'03]



a possible docking solution

Let A' denote molecule A with the pseudo skin atoms.

For $P \in \{A', B\}$ with M_P atoms, affinity function:

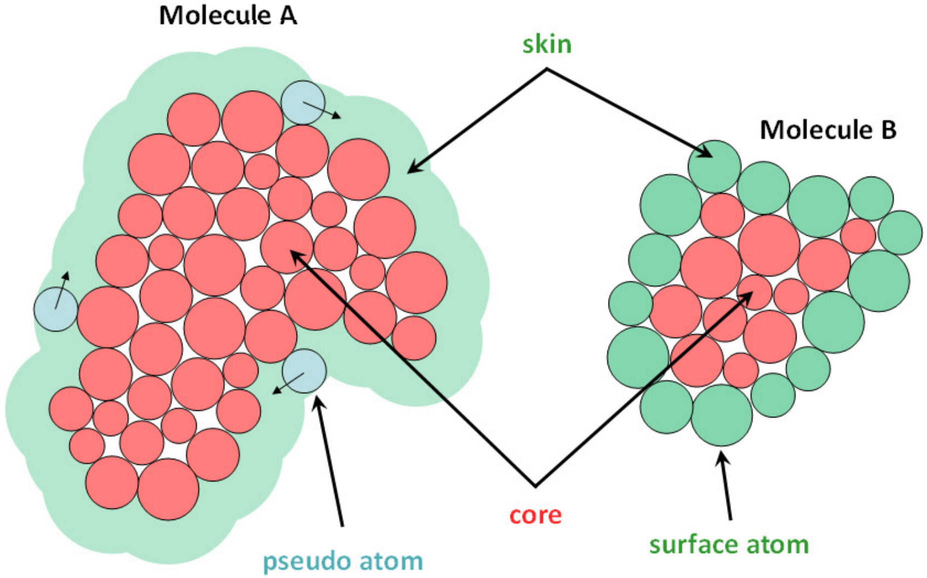
$$f_P(x) = \sum_{k=1}^{M_P} w_k \cdot g_k(x)$$

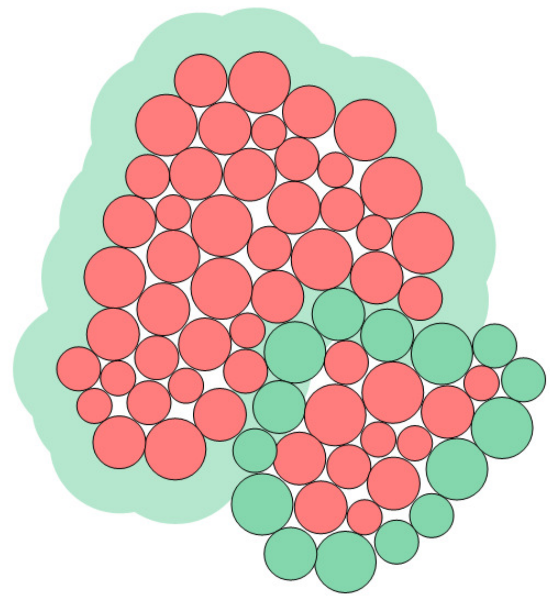
For rotation r and translation t of molecule B (i.e., $B_{t,r}$),

the interaction score, $F_{A,B}(t,r) = \int_x f_{A'}(x) f_{B_{t,r}}(x)\, dx$

Shape Complementarity

[Wang'91, Katchalski-Katzir et al.'92, Chen et al.'03]





a possible docking solution

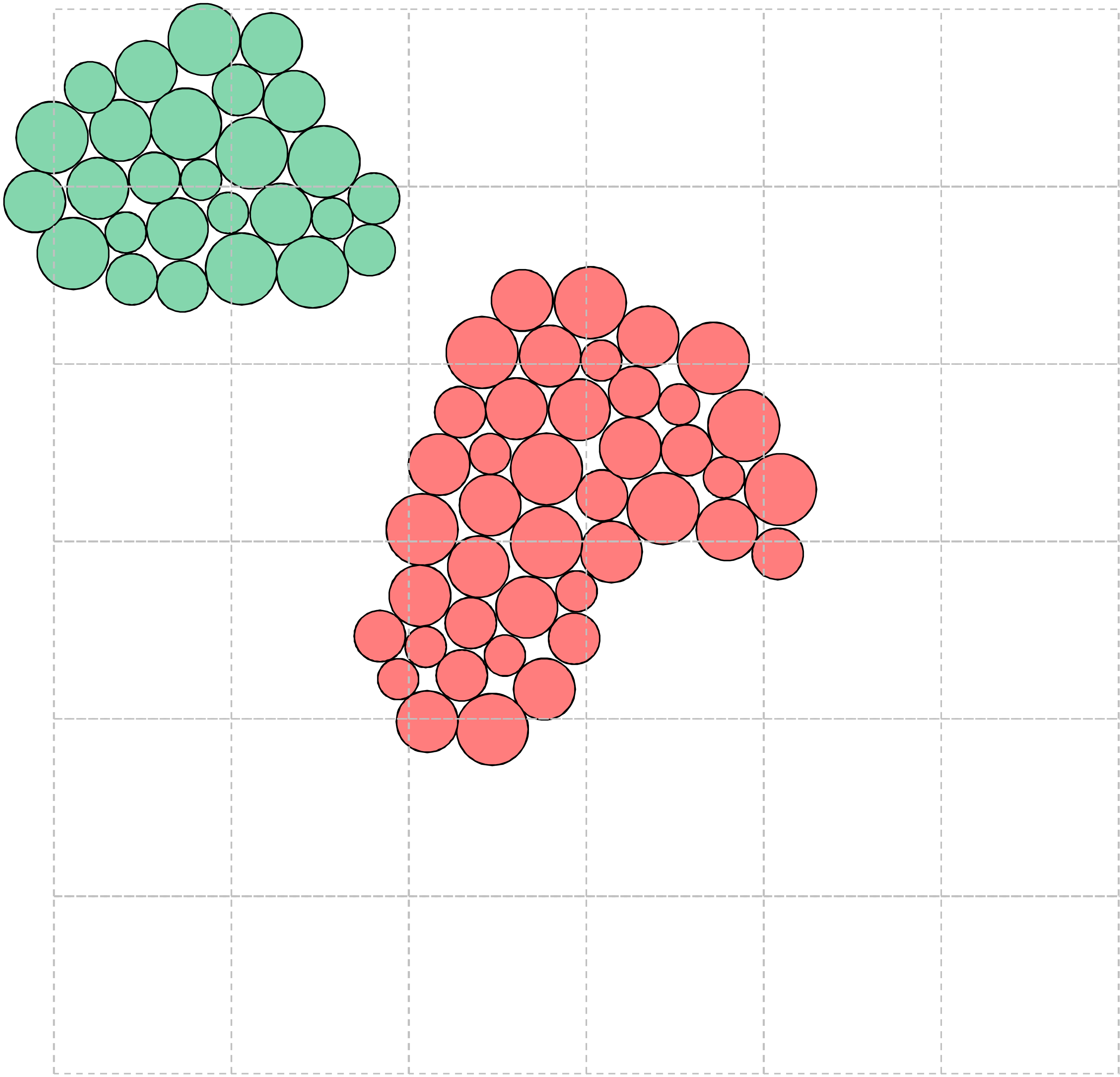
For rotation r and translation t of molecule B (i.e., $B_{t,r}$),

the interaction score, $F_{A,B}(t,r) = \int_x f_{A'}(x) f_{B_{t,r}}(x)\, dx$

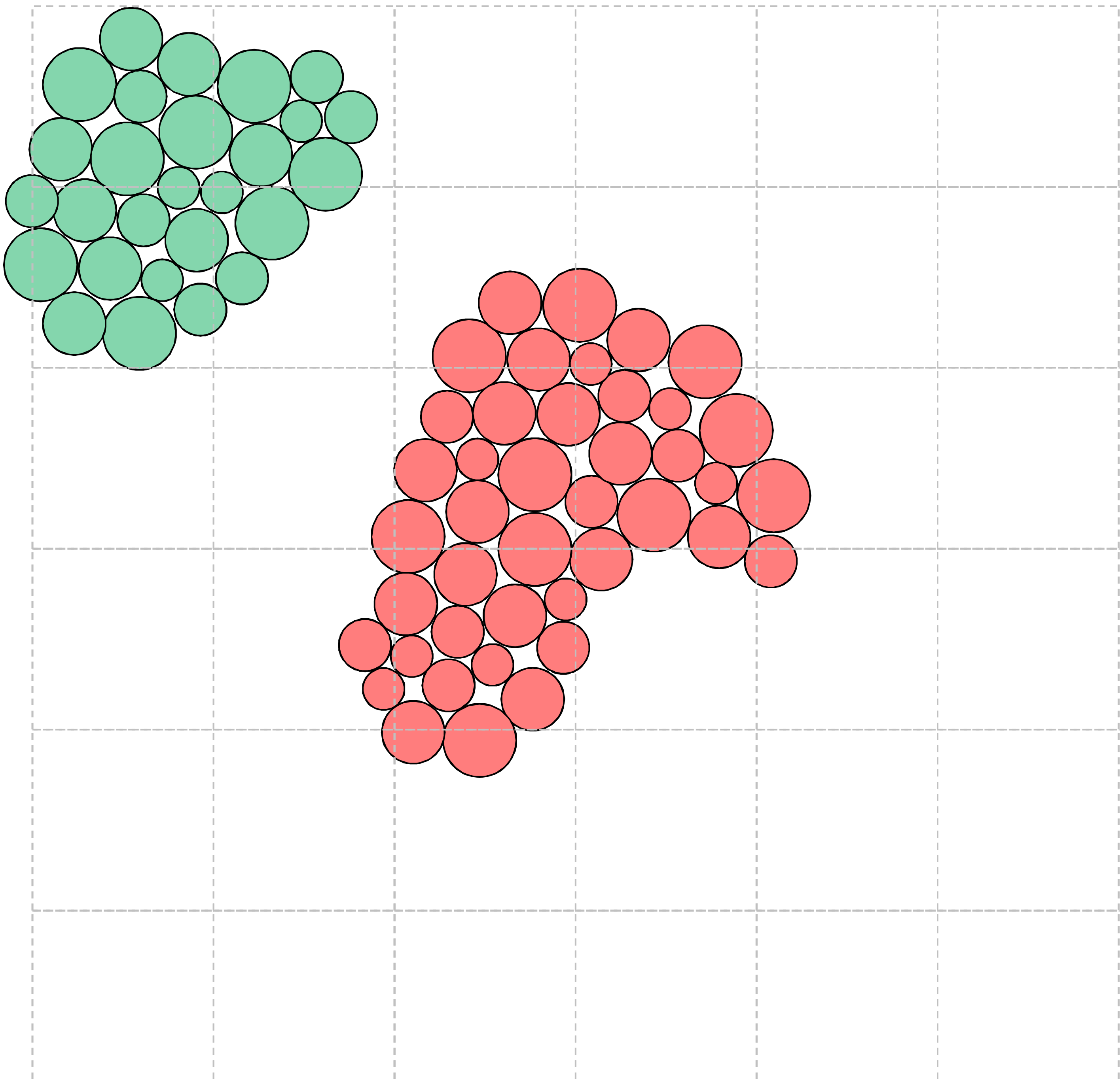
$Re\big(F_{A,B}(t,r)\big) =$ skin-skin overlap score $-$ core-core overlap score

$Im\big(F_{A,B}(t,r)\big) =$ skin-core overlap score

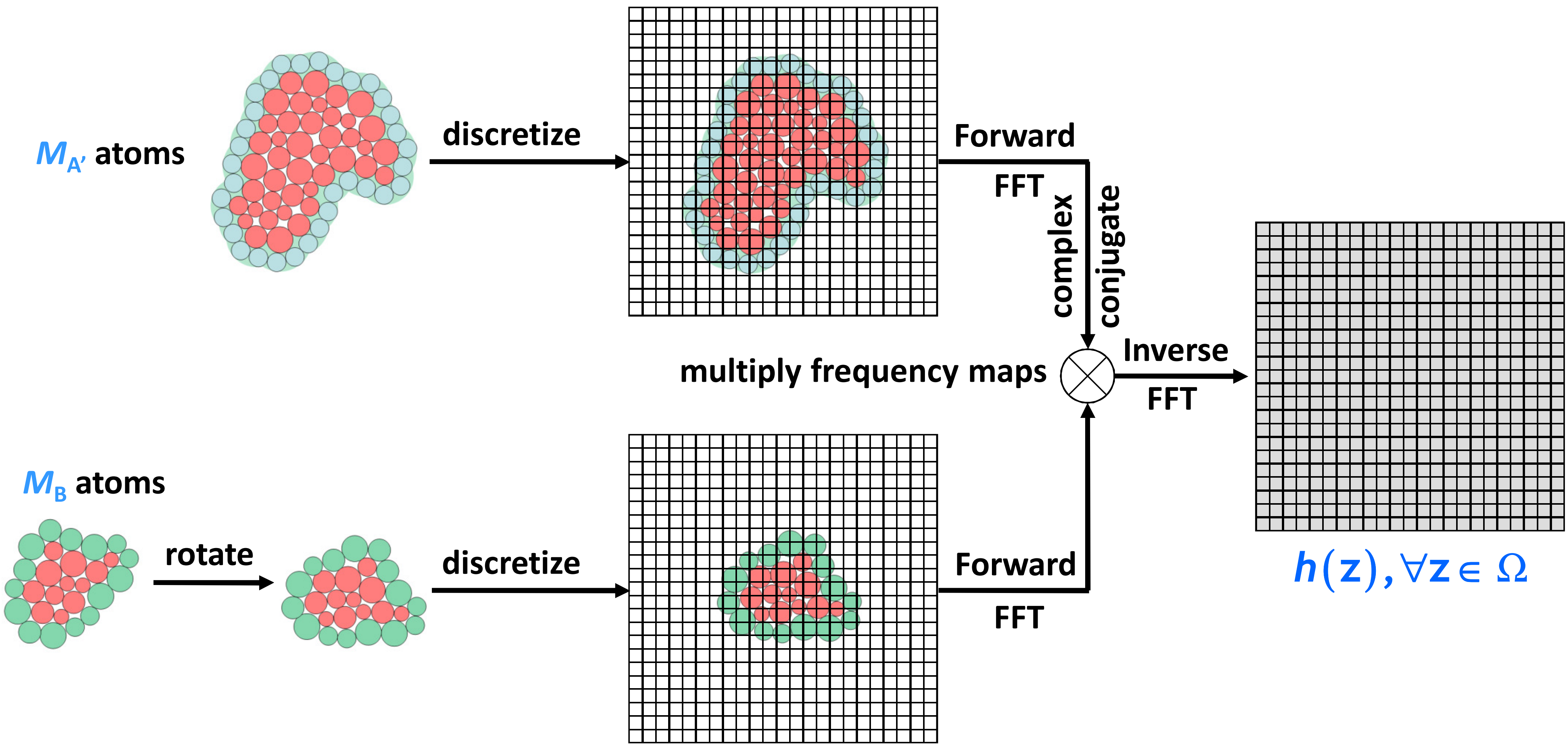
Docking: Rotational & Translational Search



Docking: Rotational & Translational Search



Translational Search using FFT



$$\forall z \in \Omega = [-n, n]^3, \qquad h(z) = \int_{x\in\Omega} f_{A'}(x) f_{B_r}(z-x)\,dx$$

The Master Theorem

A Useful Recurrence

Consider the following recurrence:

$$T(n) = \begin{cases} \Theta(1), & if\ n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise; \end{cases}$$

where, $a \geq 1$ and $b > 1$.

Arises frequently in the analyses of *divide-and-conquer* algorithms.

Recall the following from recurrences from earlier lectures.

Karatsuba's Algorithm: $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$

Strassen's Algorithm: $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n)$

Fast Fourier Transform: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & if\ n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise. \end{cases}$$

How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & if\ n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise. \end{cases}$$

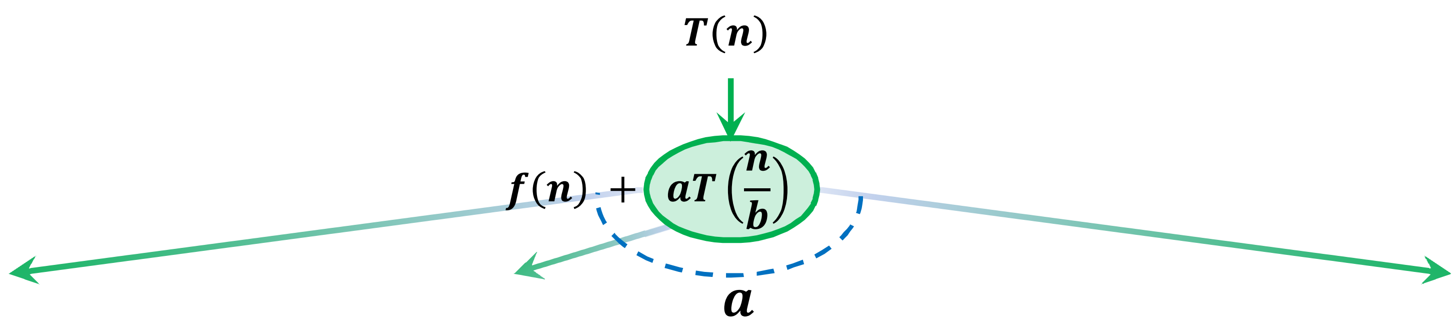
$$\boldsymbol{T(n)}$$

$$\downarrow$$

$$\boldsymbol{f(n)} \ + \ \boldsymbol{aT\left(\frac{n}{b}\right)}$$

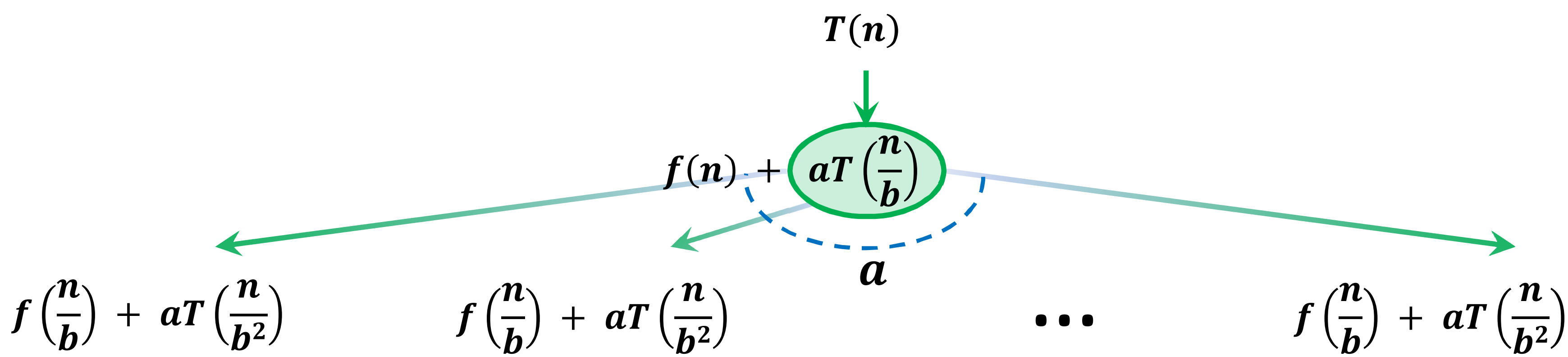
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & if\ n \le 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise. \end{cases}$$



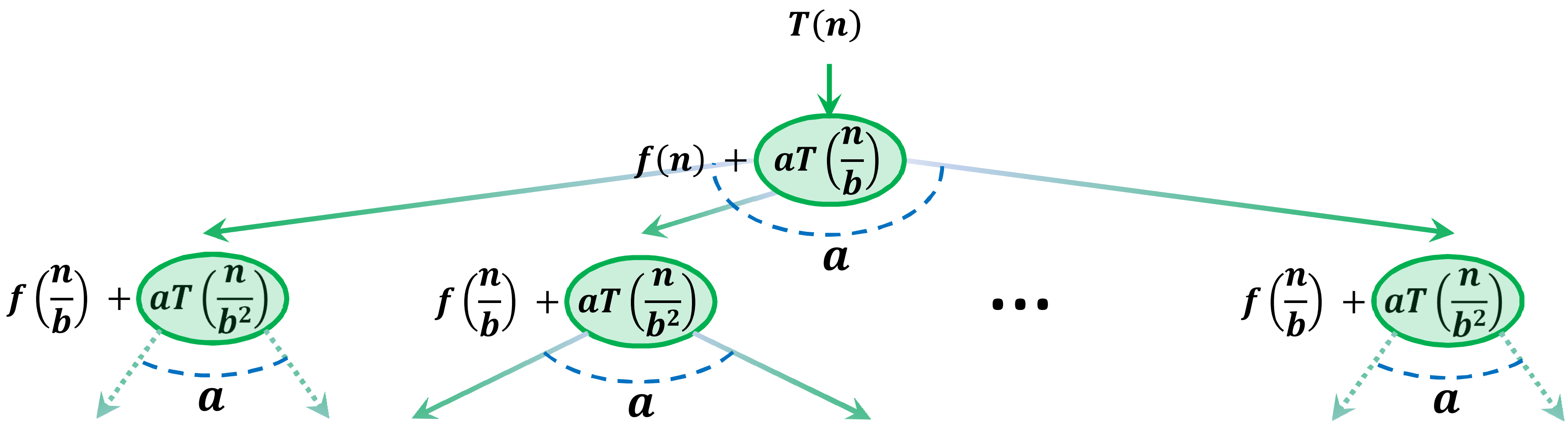
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & if\ n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise. \end{cases}$$



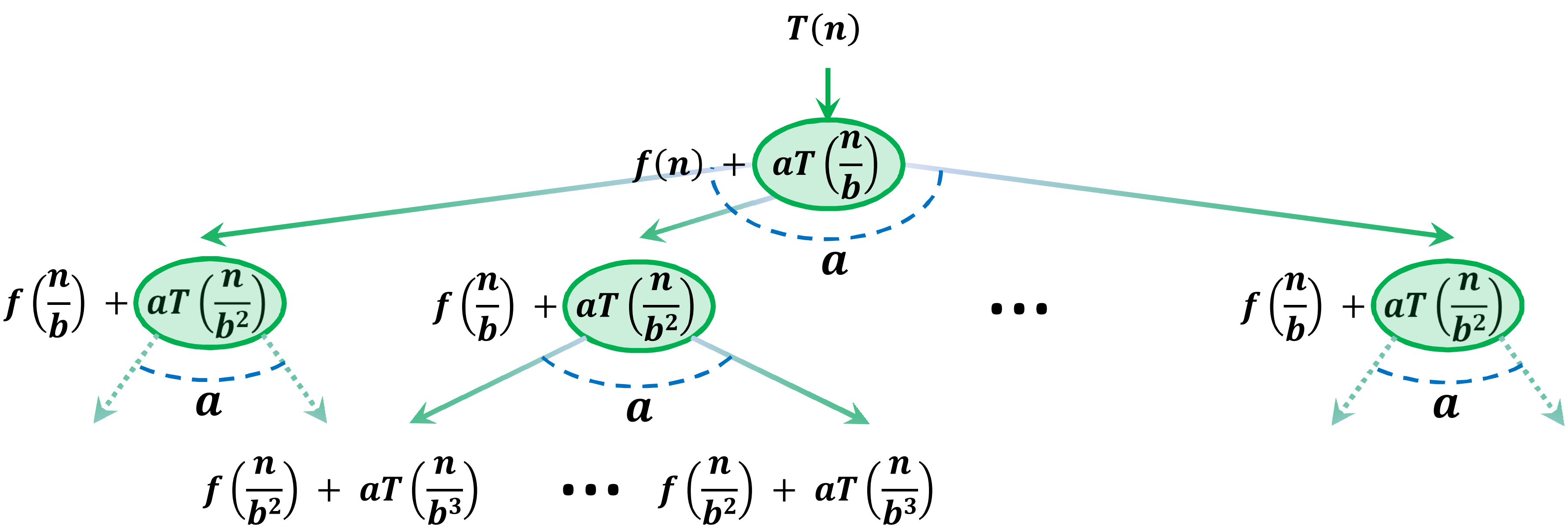
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & if\ n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise. \end{cases}$$



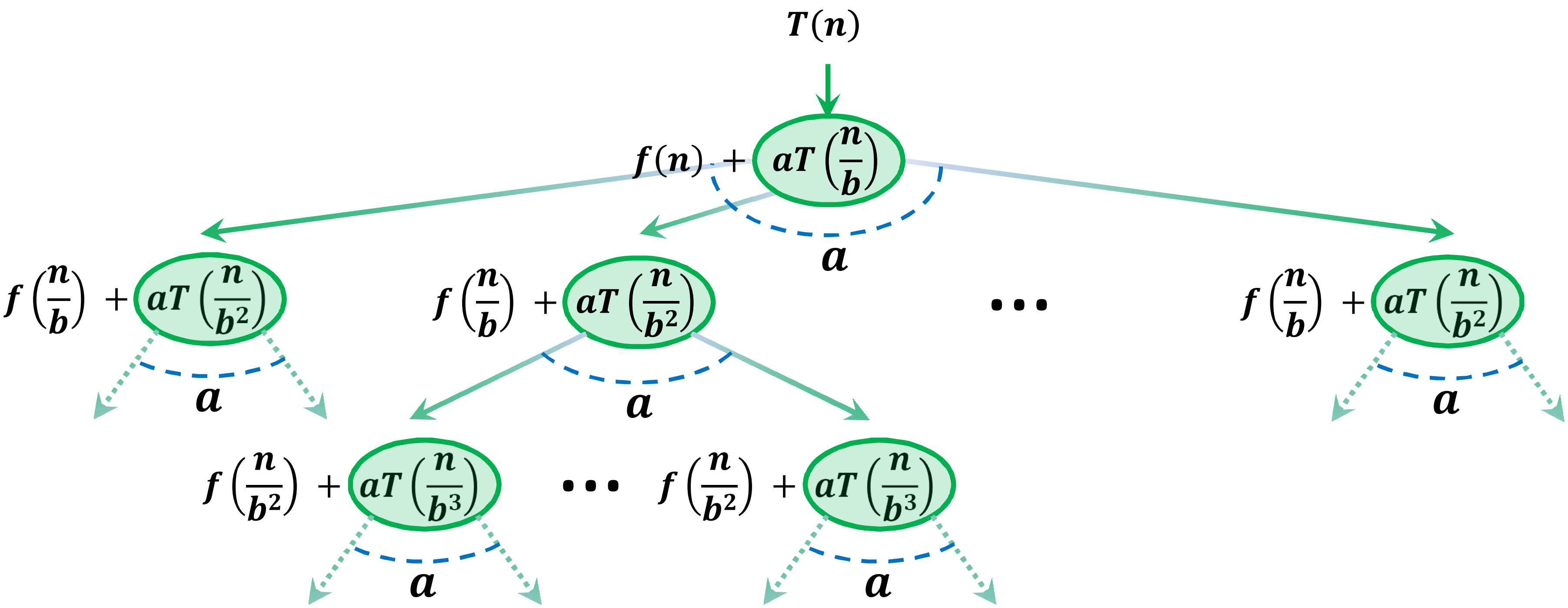
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & if\ n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise. \end{cases}$$



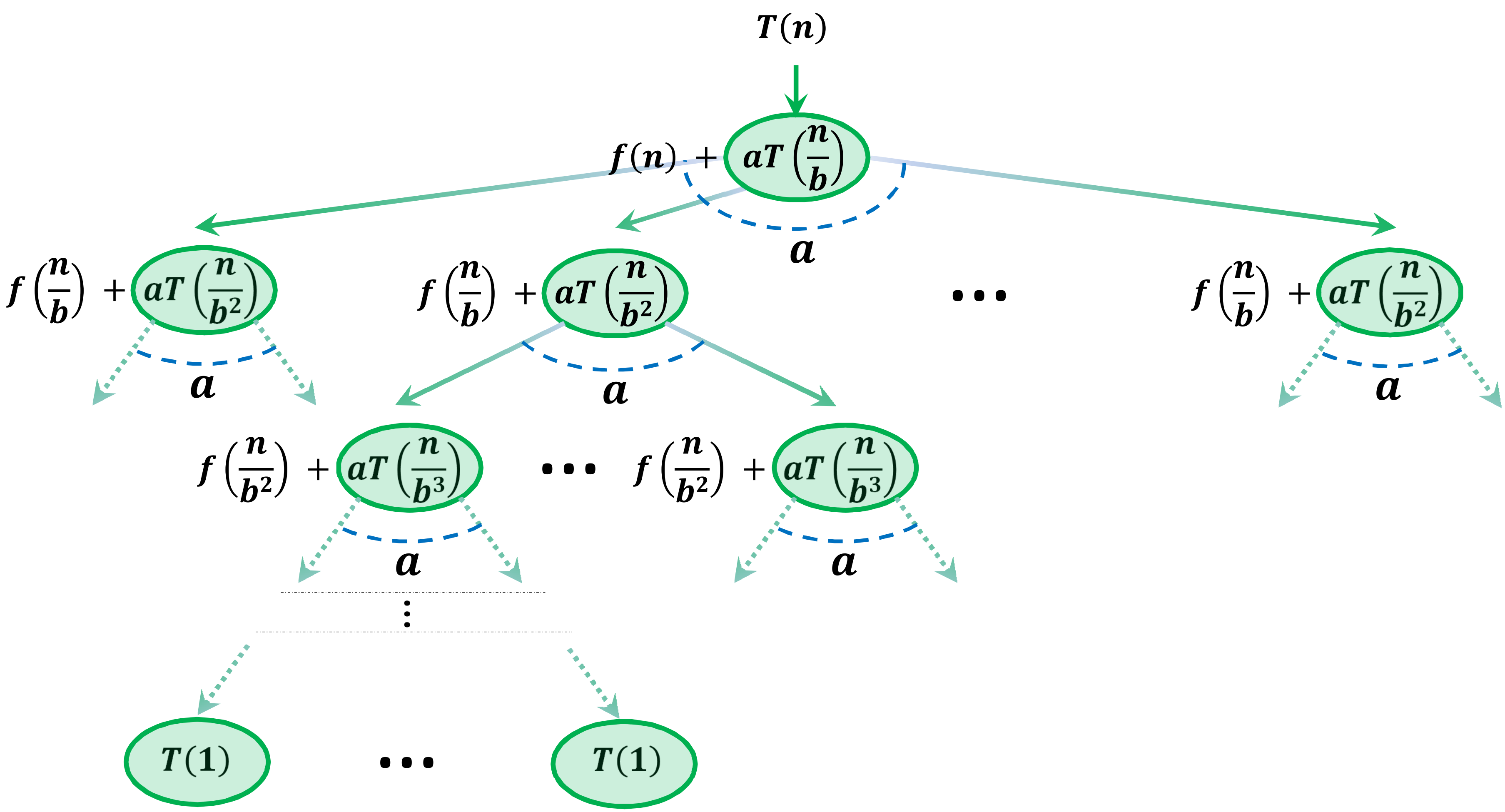
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \mathit{if}\ n \le 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \mathit{otherwise}. \end{cases}$$



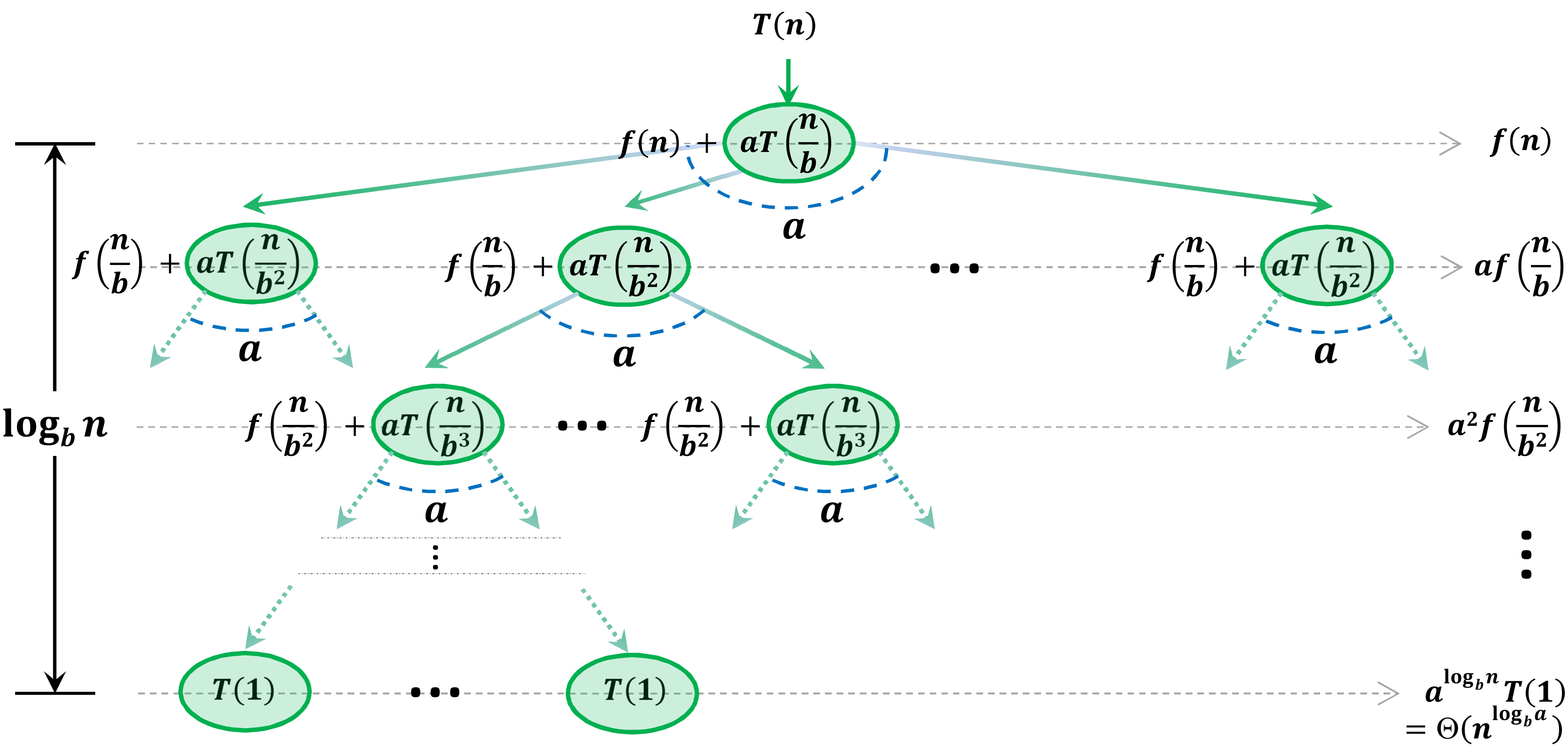
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & if\ n \le 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise. \end{cases}$$



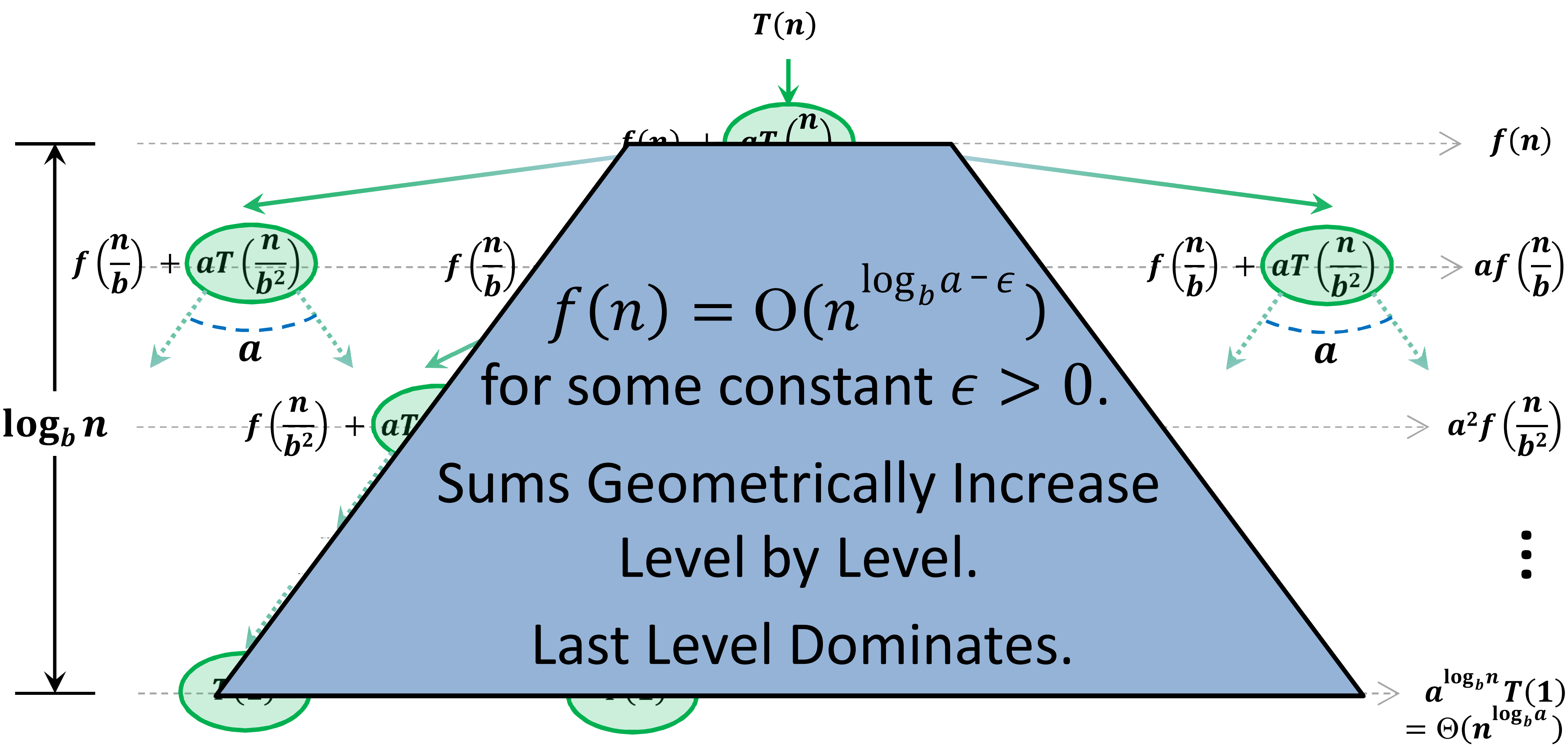
How the Recurrence Unfolds

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \le 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



How the Recurrence Unfolds: Case 1

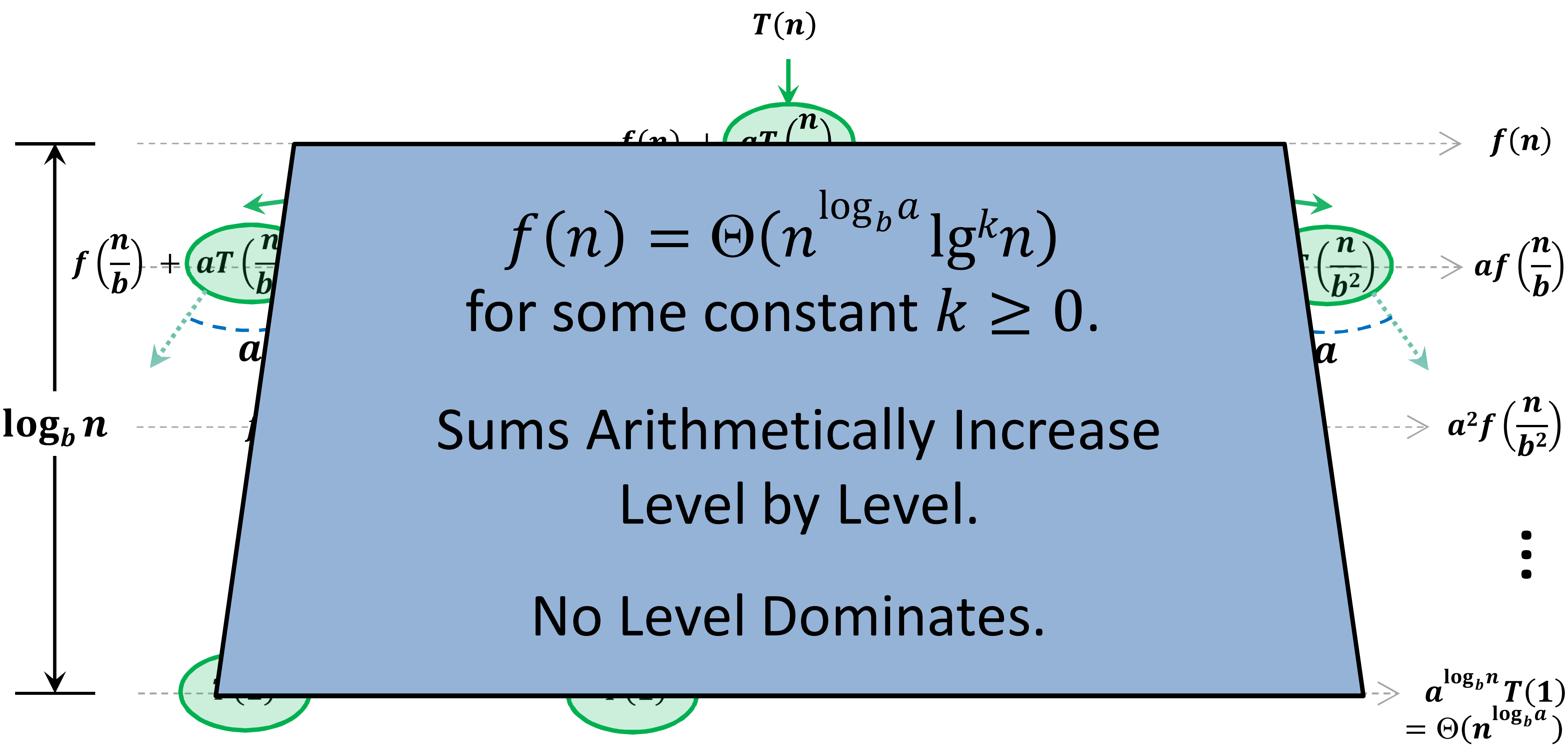
$$T(n) = \begin{cases} \Theta(1), & \text{if } n \le 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



$$\boxed{T(n) = \Theta(n^{\log_b a})}$$

How the Recurrence Unfolds: Case 2

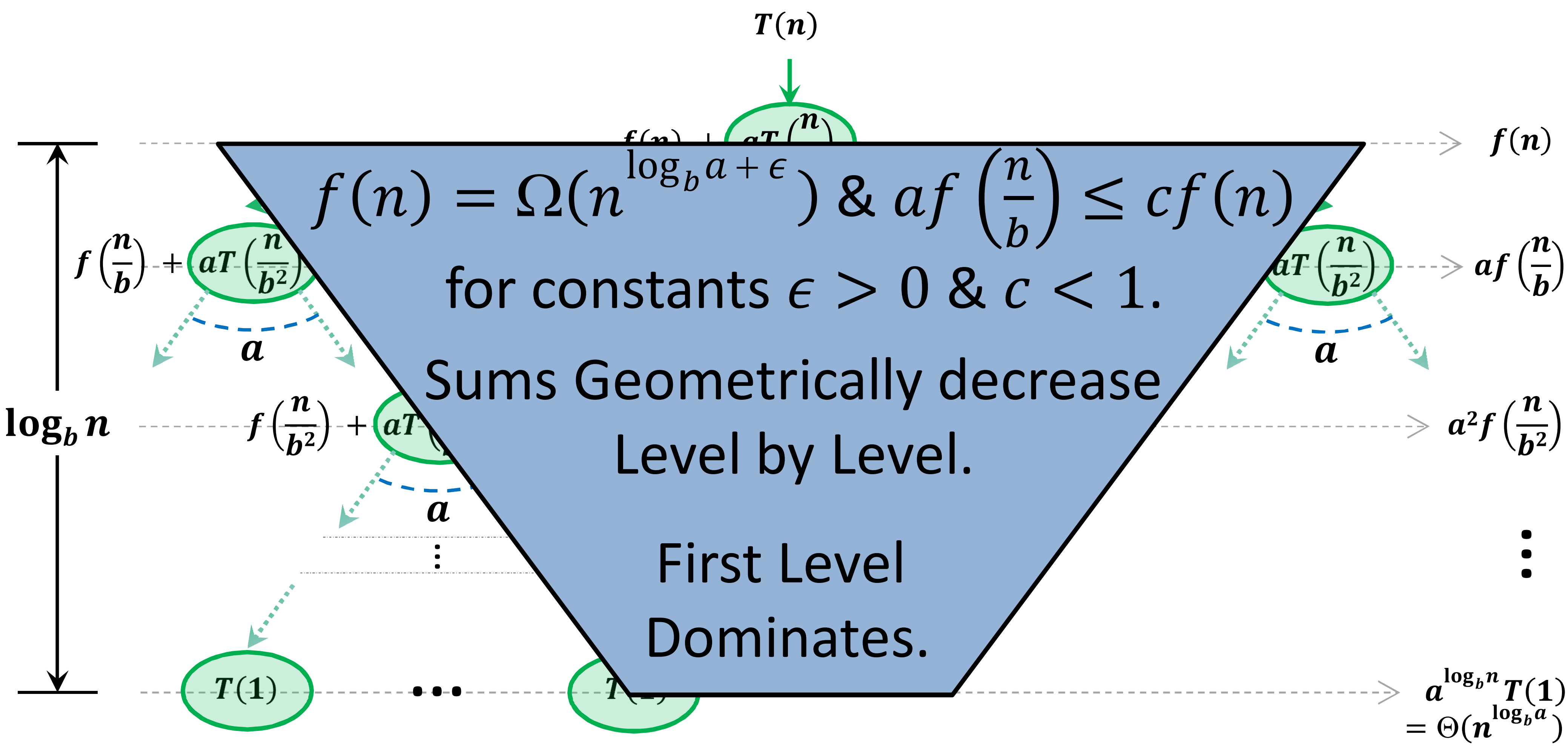
$$T(n) = \begin{cases} \Theta(1), & \text{if } n \le 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



$$\boxed{T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)}$$

How the Recurrence Unfolds: Case 3

$$T(n) = \begin{cases} \Theta(1), & \text{if } n \le 1, \\ aT\left(\frac{n}{b}\right) + f(n), & \text{otherwise.} \end{cases}$$



$f(n) = \Omega(n^{\log_b a + \epsilon})$ & $af\left(\frac{n}{b}\right) \le cf(n)$

for constants $\epsilon > 0$ & $c < 1$.

Sums Geometrically decrease Level by Level.

First Level Dominates.

$$\boxed{T(n) = \Theta(f(n))}$$

The Master Theorem

$$T(n) = \begin{cases} \Theta(1), & if\ n \leq 1, \\ aT\left(\frac{n}{b}\right) + f(n), & otherwise\ (a \geq 1, b > 1). \end{cases}$$

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$

$$\boxed{T(n) = \Theta(n^{\log_b a})}$$

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant $k \geq 0$.

$$\boxed{T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)}$$

Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$ and $af\left(\frac{n}{b}\right) \leq cf(n)$
for constants $\epsilon > 0$ and $c < 1$.

$$\boxed{T(n) = \Theta\big(f(n)\big)}$$

Example Applications of Master Theorem

Example 1: $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 1: $T(n) = \Theta\left(n^{\log_2 3}\right)$

Example 2: $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2)$

Master Theorem Case 1: $T(n) = \Theta\left(n^{\log_2 7}\right)$

Example 3: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$

Master Theorem Case 2: $T(n) = \Theta(n\log n)$

Assuming that we have an infinite number of processors, and each recursive call in example 2 above can be executed in parallel:

Example 4: $T(n) = T\left(\frac{n}{2}\right) + \Theta(n^2)$

Master Theorem Case 3: $T(n) = \Theta(n^2)$