Algorithms

(Parallel Divide-and-Conquer)

Pramod Ganapathi
Department of Computer Science
State University of New York at Stony Brook

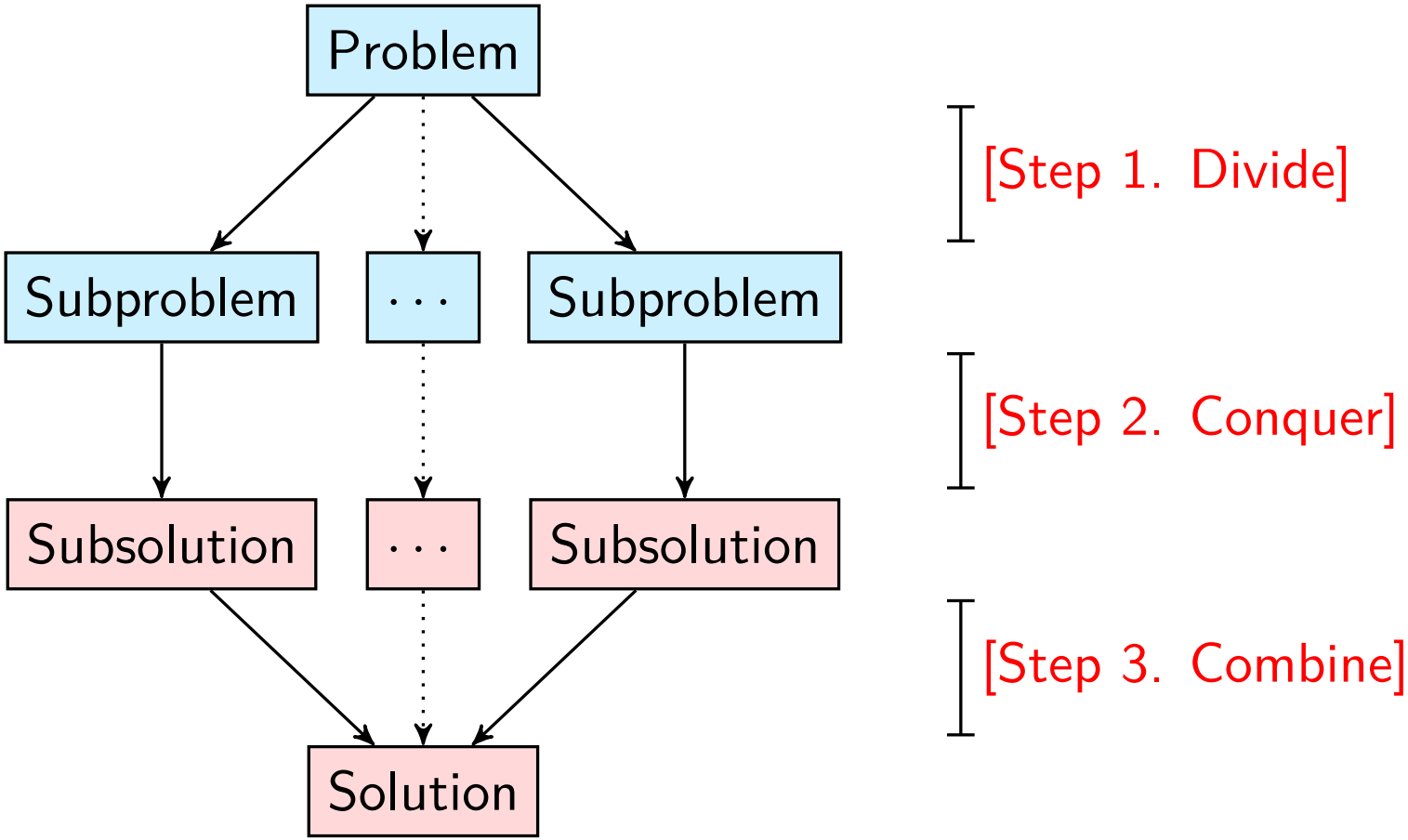
November 2, 2021

Contents

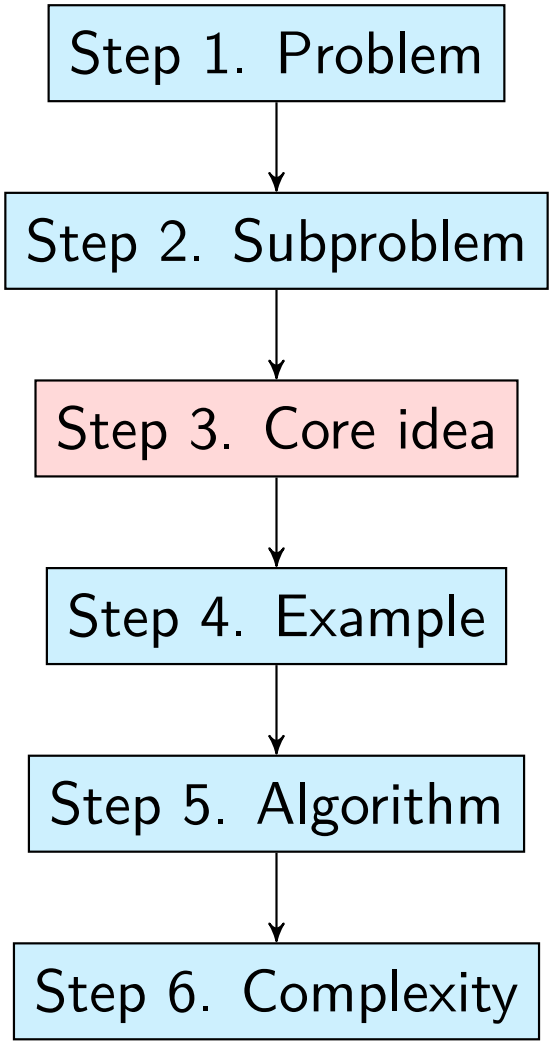
Contributors

- Ayush Sharma
- Sneh Amit Shah

Divide-and-conquer



D&C problem-solving template



Array Sum HOME

Step 1. Problem

Problem

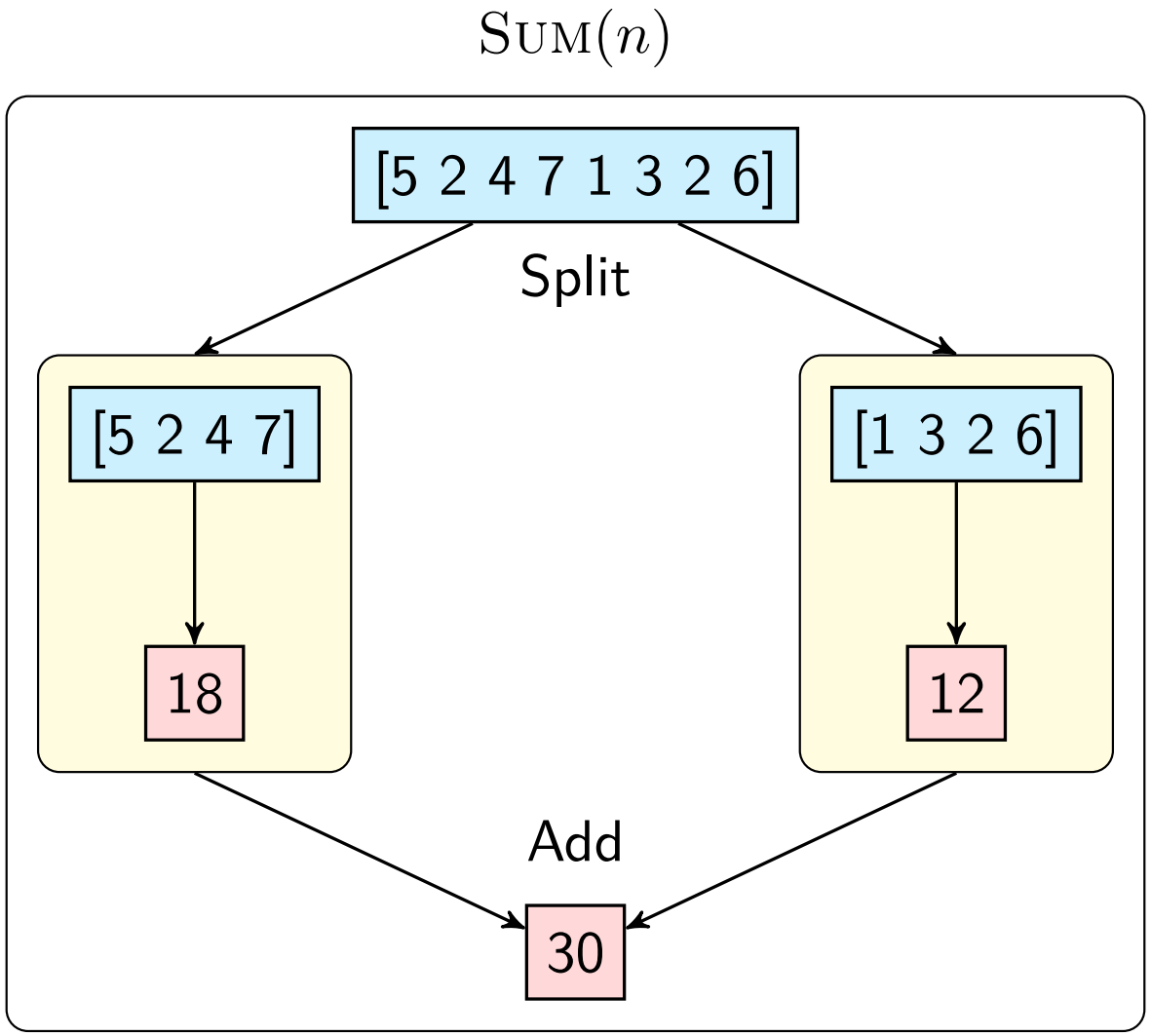
- Compute the sum of elements of a given array.

Step 2. Subproblem

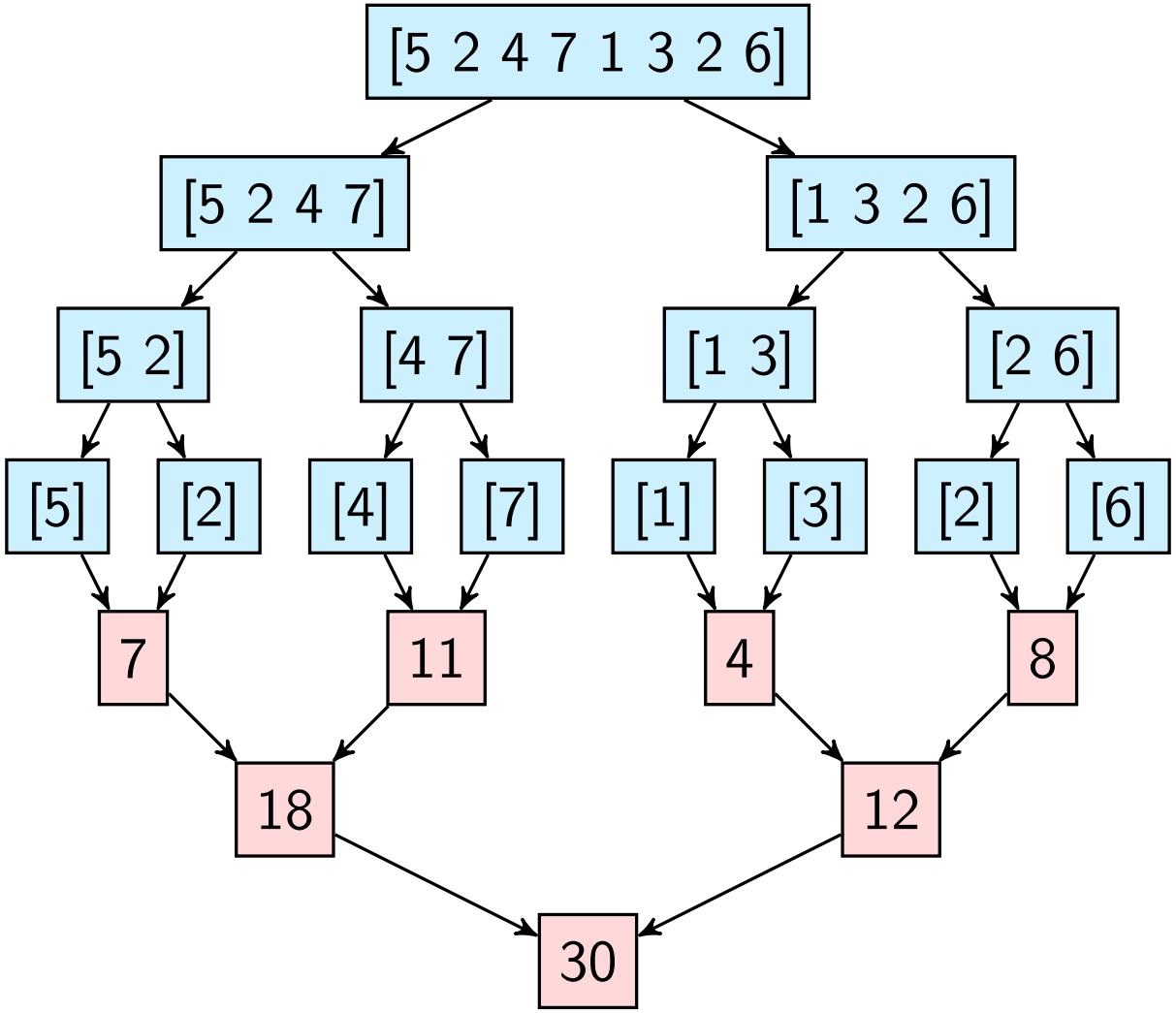
$$\textsc{Sum}(A[\ell..h]) = \text{Sum of all elements in subarray } A[\ell..h]$$

Compute $\textsc{Sum}(A[1..n])$.

Step 3. Core idea



Step 4. Example



Step 5. Algorithm

$\textsc{ArraySum}(A[low..high])$

```
if low = high then
  return A[mid]
else if low < high then
  mid ← (low + high)/2
  parallel: ℓsum ← ArraySum(A[low..mid])
            rsum ← ArraySum(A[(mid + 1)..high])
  return ℓsum + rsum
```

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(\log n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2S(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 2Q(n/2) + \Theta(1) & \text{if } n > \gamma M. \end{array} \right\} \in \Theta(n/B)$$

Area Estimation HOME

Step 1. Problem

Problem

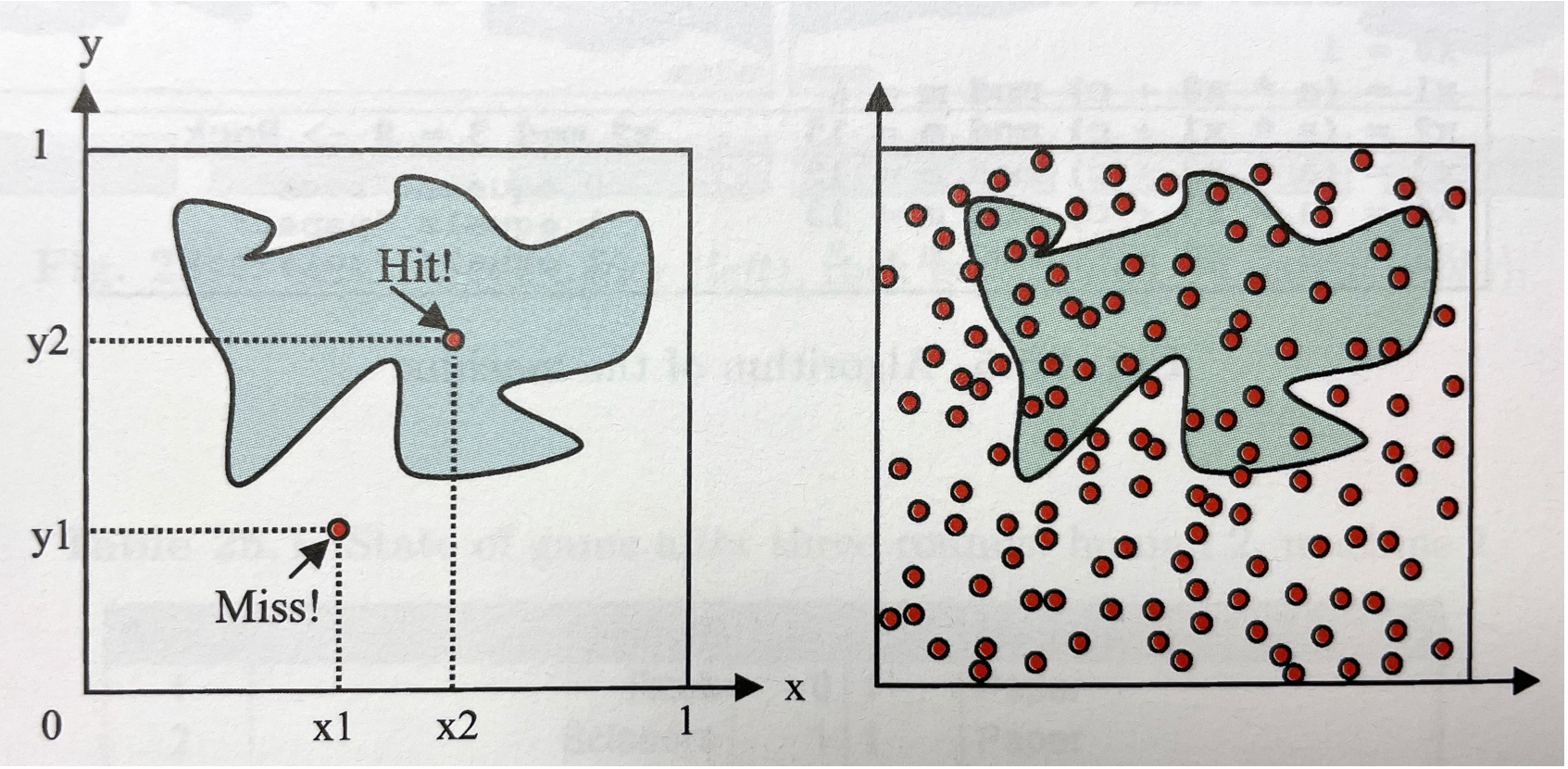
- Compute the area of a closed shape (in 2-D space) defined by function $f(x)$.

Step 2. Subproblem

$$\textsc{Hits}(k) = \#\text{Points falling inside the closed shape with } k \text{ attempts}$$

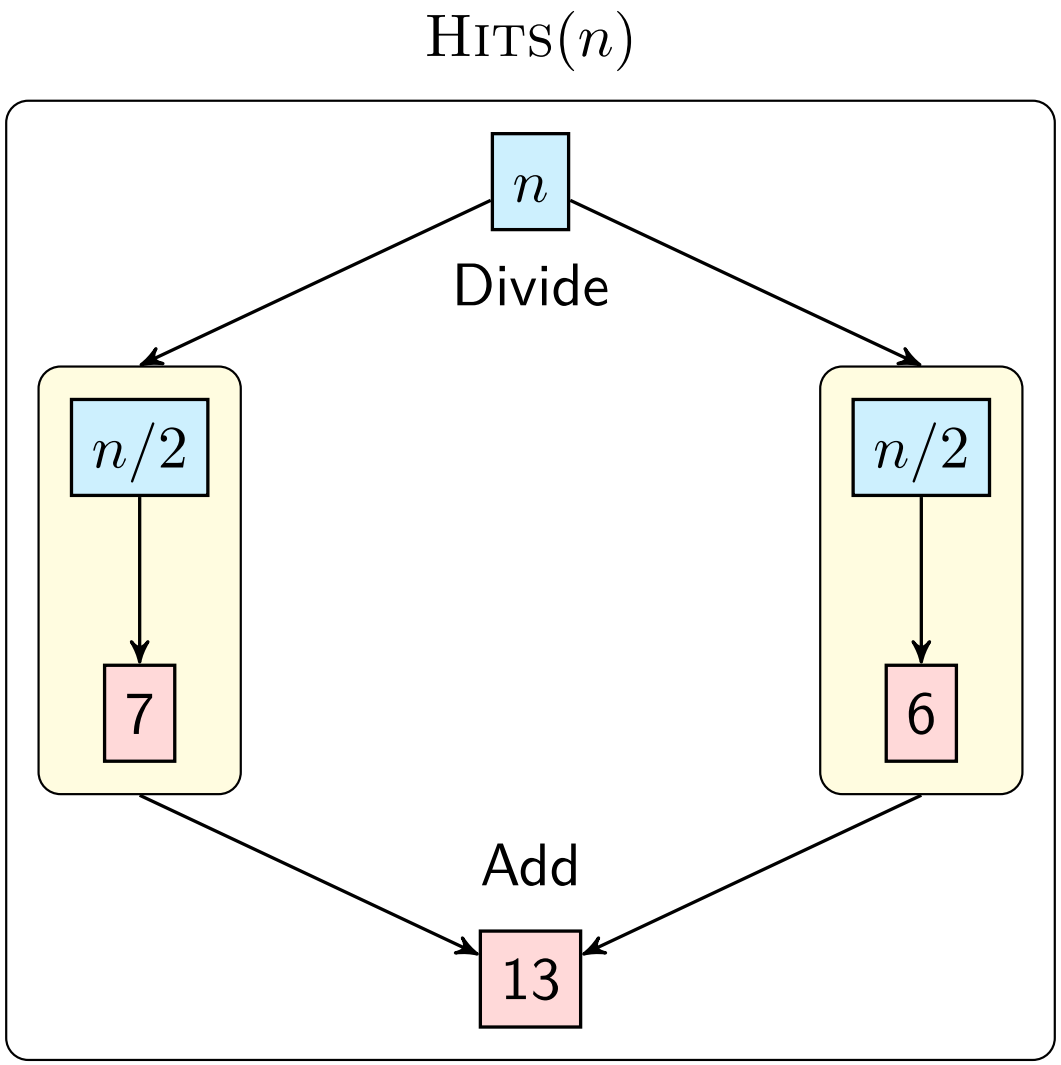
Compute $\textsc{Hits}(n)$.

Step 3. Core idea



Source: Algorithms Unplugged

Step 3. Core idea



Step 5. Algorithm

$\textsc{Area}(f(x), n)$

Input: Region $f(x)$, number of attempts n
Output: Area of region $f(x)$

1. $(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}) \leftarrow$ extremes of function $f(x)$
2. define a bounding box with boundaries $(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}})$
3. $boxarea \leftarrow (x_{\text{max}} - x_{\text{min}}) \times (y_{\text{max}} - y_{\text{min}})$
4. $hits \leftarrow 0$
5. **parallel: for** $i \leftarrow 1$ **to** n **do**
6. $\quad (x, y) \leftarrow$ random point in bounding box
7. $\quad$ **if** $\textsc{PointInsideRegion}(x, y)$ **then**
8. $\quad\quad hits \leftarrow hits + 1$
9. **return** $hits \times boxarea/n$

Step 5. Algorithm

$\textsc{Area}(f(x), n)$

Input: Region $f(x)$, number of attempts n
Output: Area of region $f(x)$

1. $(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}) \leftarrow$ extremes of function $f(x)$
2. define a bounding box with boundaries $(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}})$
3. $boxarea \leftarrow (x_{\text{max}} - x_{\text{min}}) \times (y_{\text{max}} - y_{\text{min}})$
4. $hits \leftarrow \textsc{Hits}(n)$
5. **return** $hits \times boxarea/n$

$\textsc{Hits}(n)$

Input: Number of attempts n; Global: $f(x)$ and bounding box
Output: Number of hits i.e., number of points present inside $f(x)$

1. **if** $n = 1$ **then**
2. $\quad (x, y) \leftarrow$ random point in bounding box
3. $\quad$ **if** $\textsc{PointInsideRegion}(x, y)$ **then return** 1
4. $\quad$ **else return** 0
5. **else**
6. $\quad$ **parallel:** $hits1 \leftarrow \textsc{Hits}(n/2)$
   $\qquad\qquad\quad hits2 \leftarrow \textsc{Hits}(n/2)$
7. $\quad$ **return** $hits1 + hits2$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(\log n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2S(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 2Q(n/2) + \Theta(1) & \text{if } n > \gamma M. \end{array} \right\} \in \Theta(n/B)$$

Prefix Sum HOME

Step 1. Problem

Problem

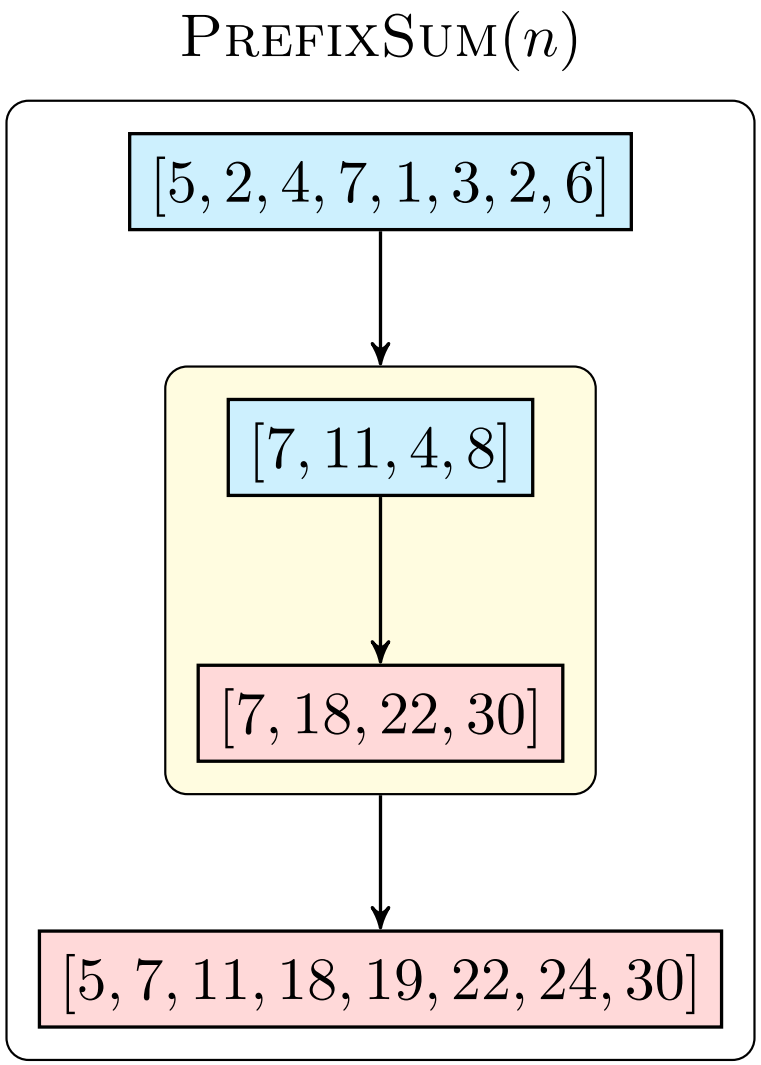
- Given an array $A[1..n]$ compute the partial sum array $S[1..n]$ such that $S[i] = A[1] + A[2] + \cdots + A[i]$.

Step 2. Subproblem

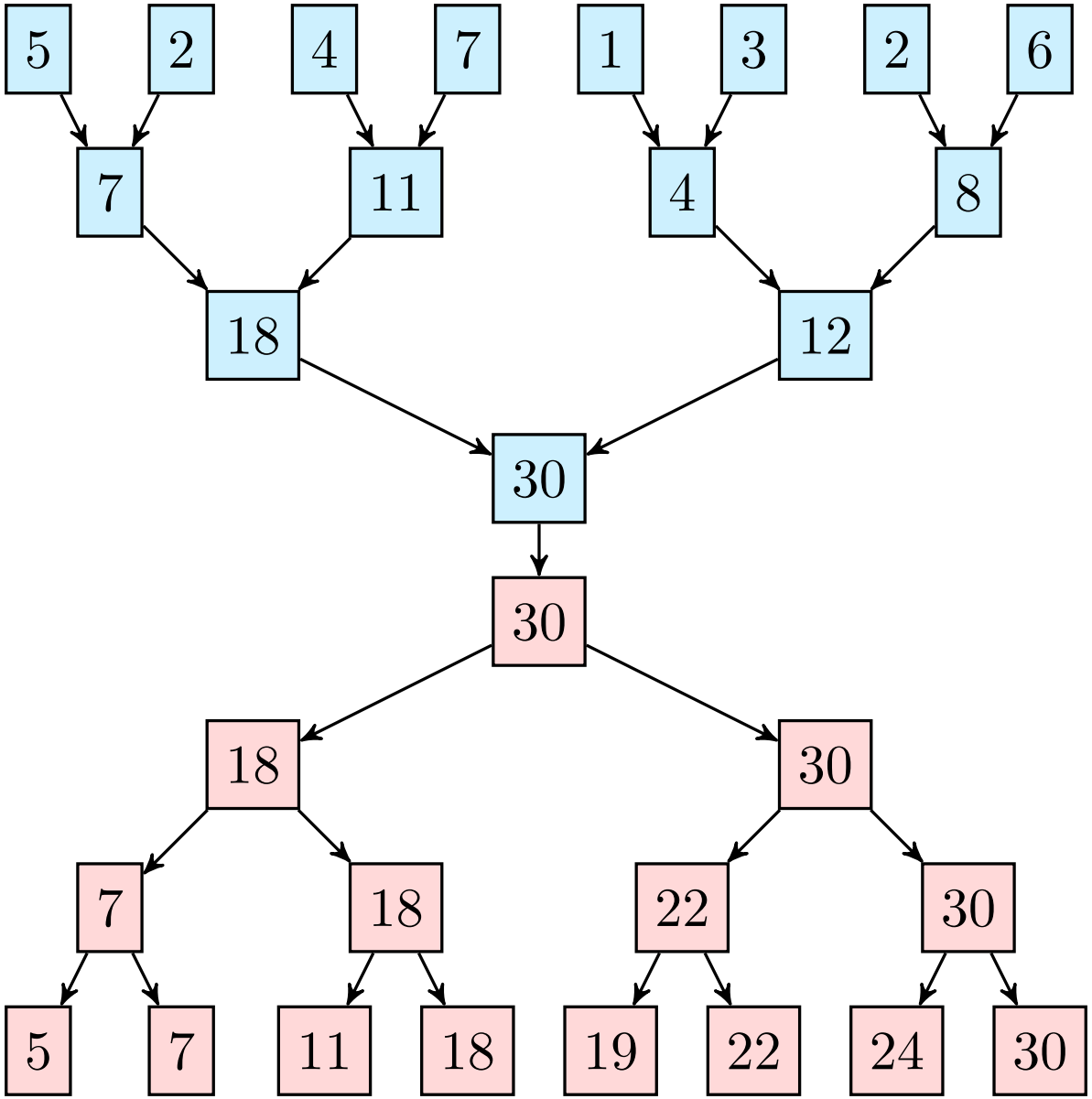
$$\textsc{PrefixSum}(A[1..m]) = \text{Prefix sum of } A[1..m]$$

Compute $\textsc{PrefixSum}(A[1..n])$.

Step 3. Core idea



Step 4. Example



Step 5. Algorithm

$\textsc{PrefixSum}(A[1..n])$

```
if n = 1 then
  return A[1]
[Stage 1. Decrease]
parallel: for i ← 1 to n/2 do
  B[i] ← A[2i − 1] + A[2i]
[Stage 2. Conquer]
C[1..n/2] ← PrefixSum(B[1..n/2])
[Stage 3. Combine]
parallel: for i ← 1 to n do
  if i = 1 then S[1] ← A[1]
  else if i is even then S[i] ← C[i/2]
  else if i is odd then S[i] ← C[(i − 1)/2] + A[i]
return S[1..n]
```

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ T(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(\log n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ Q(n/2) + \Theta(n/B) & \text{if } n > \gamma M. \end{array} \right\} \in \Theta(n/B)$$

Recurrences HOME

Method 1: Iteration and substitution

$$
\begin{aligned}
T(n) &= T\left(\frac{n}{2}\right) + 1 \\
&= T\left(\frac{n}{2^2}\right) + 1 + 1 \\
&= T\left(\frac{n}{2^3}\right) + 1 + 1 + 1 \\
&= T\left(\frac{n}{2^k}\right) + \underbrace{1 + 1 + \cdots + 1}_{k \text{ times}} \\
&= T\left(\frac{n}{2^k}\right) + k \qquad \left(\text{set the basecase } \frac{n}{2^k} = 1\right) \\
&= T(1) + \log n \\
&= 1 + \log n \\
&\in \Theta\left(\log n\right)
\end{aligned}
$$

Method 1: Iteration and substitution

$$
\begin{aligned}
T(n) &= 2T\left(\frac{n}{3}\right) + \Theta\left(n\right) \\
T(n) &\leq 2T\left(\frac{n}{3}\right) + cn \\
&\leq 2\left(2T\left(\frac{n}{3^2}\right) + \frac{cn}{3}\right) + cn \\
&= 2^2T\left(\frac{n}{3^2}\right) + \frac{2}{3}cn + cn \\
&\leq 2^2\left(2T\left(\frac{n}{3^3}\right) + \frac{cn}{3^2}\right) + \frac{2}{3}cn + cn \\
&= 2^3T\left(\frac{n}{3^3}\right) + \frac{2^2}{3^2}cn + \frac{2}{3}cn + cn \\
&= 2^3T\left(\frac{n}{3^3}\right) + cn\left(1 + \left(\frac{2}{3}\right) + \left(\frac{2}{3}\right)^2\right)
\end{aligned}
$$

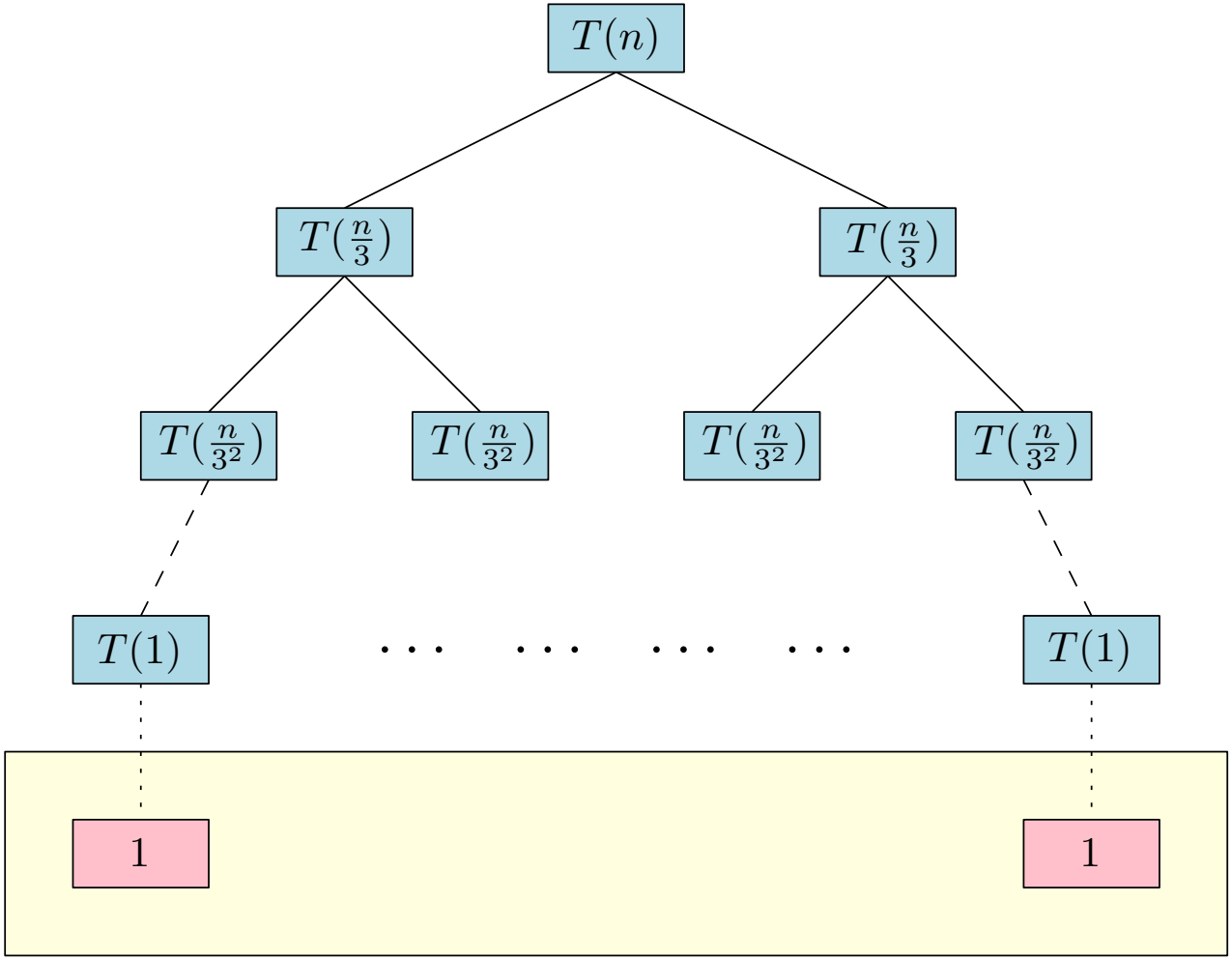
(will be continued on the next page...)

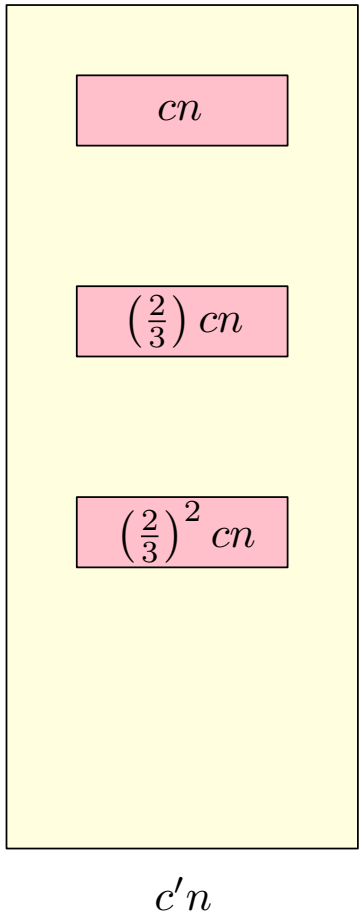
Method 1: Iteration and substitution

$$
\begin{aligned}
T(n) &\leq 2^k T\left(\frac{n}{3^k}\right) + cn\left(1 + \left(\frac{2}{3}\right) + \cdots + \left(\frac{2}{3}\right)^{k-1}\right) \\
&\leq 2^k T\left(\frac{n}{3^k}\right) + c'n \qquad \text{(geometrically decreasing series)} \\
&\left(\text{Set basecase } \frac{n}{3^k} = 1. \text{ This implies } 3^k = n \text{ and } 2^k = n^{\log_3 2}.\right) \\
&= n^{\log_3 2} \cdot T(1) + c'n \\
&\leq c''n \\
&\in \mathcal{O}(n)
\end{aligned}
$$

Method 2: Recursion tree

- $T(n) = 2T\left(\frac{n}{3}\right) + cn$





$$\underbrace{1 + 1 + \cdots + 1}_{2^{\log_3 n}} = n^{\log_3 2}$$

Method 3: Guess and test

- Given a recurrence, guess a function and test if the recurrence matches the function by induction.
- If the test fails, you might have to consider a fast-growing or slow-growing function and test it again.
- This is a pretty dangerous method because if a function does not match a recurrence, it does not necessarily imply that another one proportional to this one will not work.

Method 4: Master theorem

- Suppose $T(n)$ has the following recurrence:

$$\boxed{T(n) = \left\{ \begin{array}{ll} c & \text{if } n < d, \\ aT\left(\frac{n}{b}\right) + f(n) & \text{if } n \geq d. \end{array} \right\}}$$

  where $d \geq 1$ is a fixed natural number and $a \geq 1, c > 0, b > 1$ are real constants, and $f(n)$ is a positive function for $n \geq d$.
- Then, the complexity of $T(n)$ is

$$\boxed{T(n) \in \left\{ \begin{array}{ll} \Theta\left(n^{\Delta}\right) & \text{if } f(n) \in \mathcal{O}\left(n^{\Delta-\epsilon}\right) \text{ for } \epsilon > 0, \\ \Theta\left(f(n)\log n\right) & \text{if } f(n) \in \Theta\left(n^{\Delta}\log^{k} n\right), \\ \Theta\left(f(n)\right) & \text{if } f(n) \in \Omega\left(n^{\Delta+\epsilon}\right) \text{ for } \epsilon > 0 \\ & \text{and } f(n) > af(n/b). \end{array} \right\}}$$

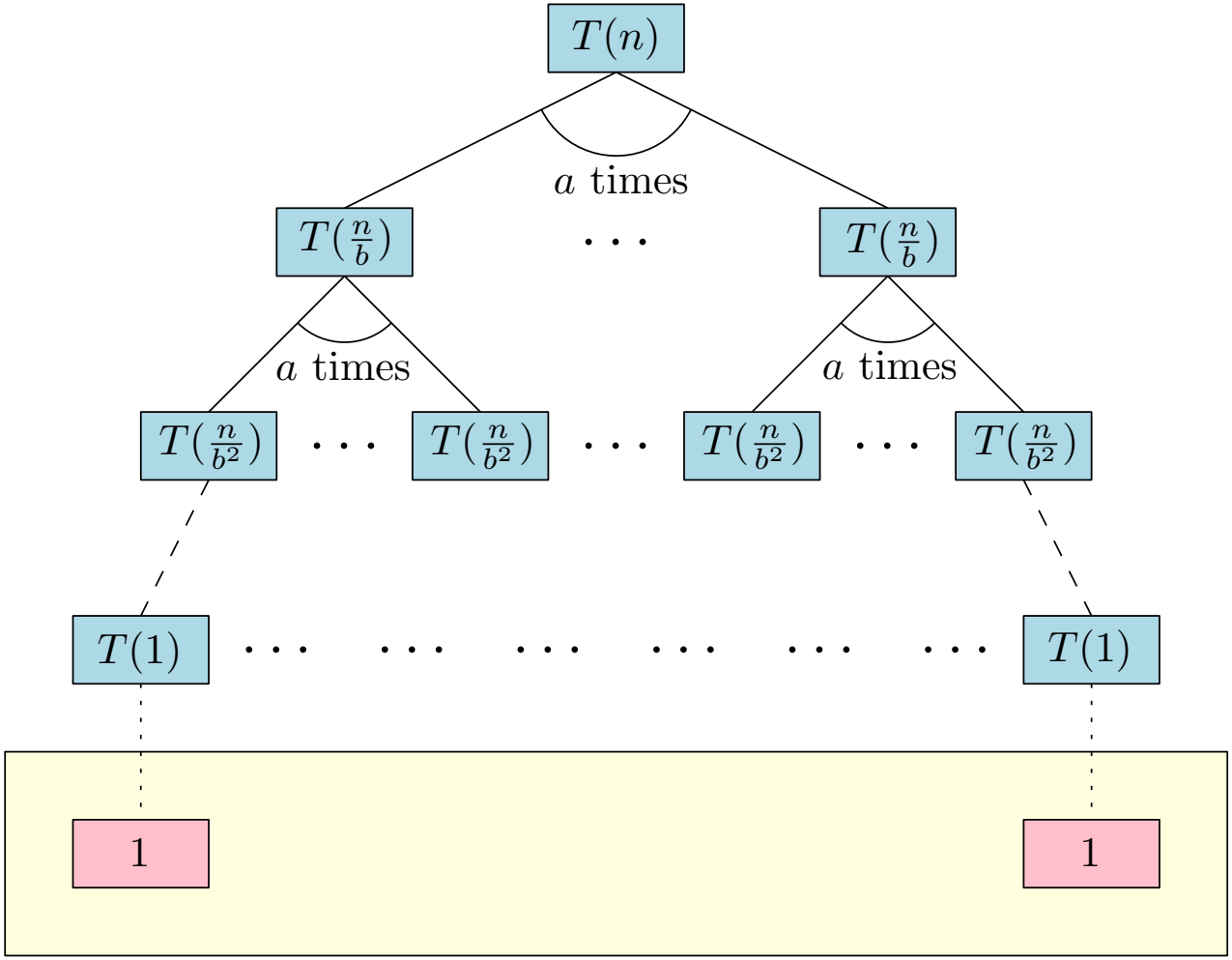
  where $\Delta = \log_b a$ is the discriminant of the recurrence.

Justification of master theorem

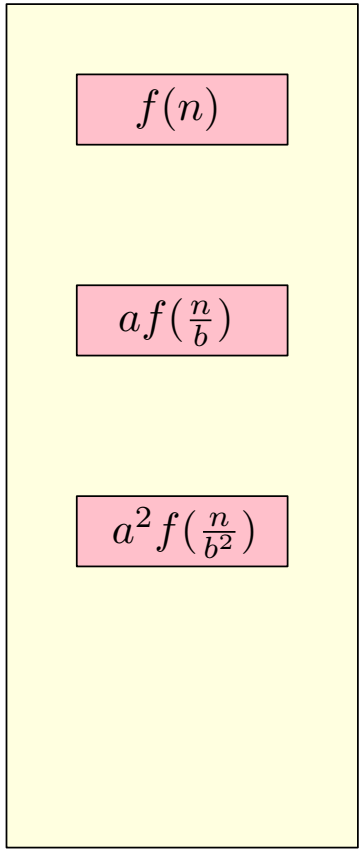
$$
\begin{aligned}
T(n) &= aT\left(\frac{n}{b}\right) + f(n) \\
&= a^2T\left(\frac{n}{b^2}\right) + af\left(\frac{n}{b}\right) + f(n) \\
&= a^3T\left(\frac{n}{b^3}\right) + a^2f\left(\frac{n}{b^2}\right) + af\left(\frac{n}{b}\right) + f(n) \\
&= a^{\log_b n}T(1) + \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) \\
&= n^{\log_b a} + \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right)
\end{aligned}
$$

1. $f(n)$ is polynomially smaller than $n^\Delta \implies T(n) \in \Theta\left(n^\Delta\right)$
2. $f(n)$ is asymptotically close to $n^\Delta \implies T(n) \in \Theta\left(f(n)\log n\right)$
3. $f(n)$ is polynomially larger than $n^\Delta \implies T(n) \in \Theta\left(f(n)\right)$

Justification of master theorem



$$\underbrace{1+1+\cdots+1}_{a^{\log_b n}} = n^{\log_b a}$$



$$\sum_{i=0}^{\log_b n-1} a^i f(\frac{n}{b^i})$$

Problems

Solve the following recurrences using the master theorem

- $T(n) = 4T(n/2) + n$
- $T(n) = 2T(n/2) + n \log n$
- $T(n) = T(n/3) + n$
- $T(n) = 2T(\sqrt{n}) + \log n$

Stooge Sort HOME

Step 1. Problem

Problem

- Sort a given n-sized array in nondecreasing order.

Step 2. Subproblem

$\textsc{Sort}(A[\ell..h])$ = Sort all elements in subarray $A[\ell..h]$ in nondecreasing order.

Compute $\textsc{Sort}(A[1..n])$.

Step 3. Core idea

9	3	8	6	7	1	5	2	4	The original array
9	3	8	6	7	1	5	2	4	First (2/3)rd of the array
1	3	6	7	8	9	5	2	4	Sort the first (2/3)rd of the array
1	3	6	7	8	9	5	2	4	Last (2/3)rd of the array
1	3	6	2	4	5	7	8	9	Sort the last (2/3)rd of the array
1	3	6	2	4	5	7	8	9	First (2/3)rd of the array
1	2	3	4	5	6	7	8	9	Sort the first (2/3)rd of the array
1	2	3	4	5	6	7	8	9	The original array is sorted

Step 5. Algorithm

$\textsc{StoogeSort}(A[\ell..h])$

Input: An array $A[\ell..h]$
Output: Array $A[\ell..h]$ sorted in nondecreasing order

```
size ← h − ℓ + 1
if size > 1 then
  if (A[ℓ] > A[h]) then Swap(A[ℓ], A[h])
  if (size > 2) then
    third ← size/3
    StoogeSort(A[ℓ..h − third])
    StoogeSort(A[ℓ + third..h])
    StoogeSort(A[ℓ..h − third])
```

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 2, \\ 3T(2n/3) + \Theta(n) & \text{if } n > 2. \end{array} \right\} \in \Theta\left(n^{\log_{1.5} 3}\right)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ S(2n/3) + \Theta(1) & \text{if } n > 1. \end{array} \right\} = \Theta(\log n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 3Q(2n/3) + \Theta(1) & \text{if } n > \gamma M. \end{array} \right\} = \mathcal{O}\left(\frac{n^{\log_{1.5} 3}}{MB}\right)$$

Merge Sort HOME

Step 1. Problem

Problem

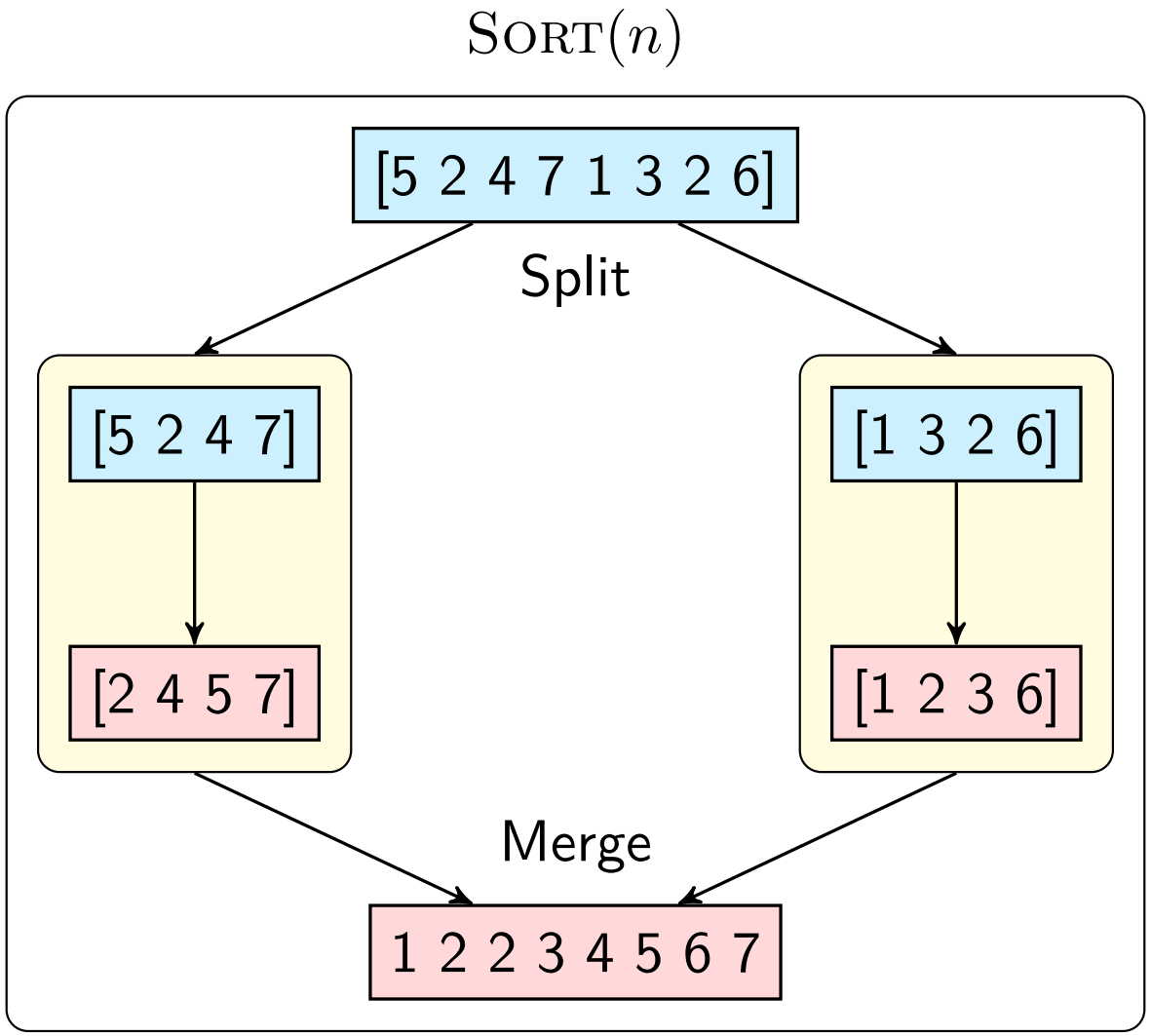
- Sort a given n-sized array in nondecreasing order.

Step 2. Subproblem

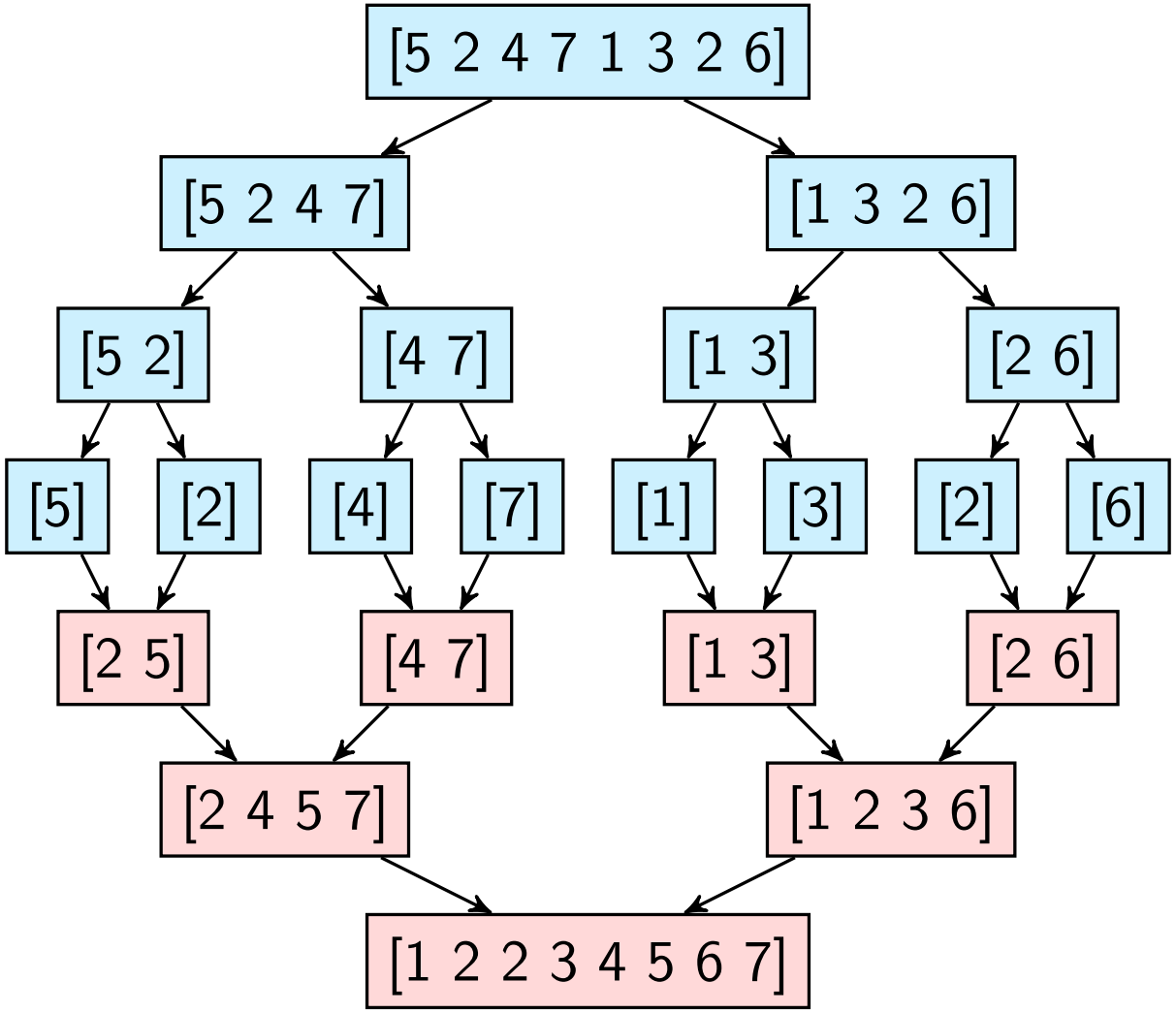
$\textsc{Sort}(A[\ell..h])$ = Sort all elements in subarray $A[\ell..h]$ in nondecreasing order.

Compute $\textsc{Sort}(A[1..n])$.

Step 3. Core idea



Step 4. Example



Step 4. Example

	p				q						$p+q$							
B	2	2	5	C	1	2	3	6	7	A								
B	2	2	5	C	1	2	3	6	7	A	1							
B	2	2	5	C	1	2	3	6	7	A	1	2						
B	2	2	5	C	1	2	3	6	7	A	1	2	2					
B	2	2	5	C	1	2	3	6	7	A	1	2	2	2				
B	2	2	5	C	1	2	3	6	7	A	1	2	2	2	3			
B	2	2	5	C	1	2	3	6	7	A	1	2	2	2	3	5		
B	2	2	5	C	1	2	3	6	7	A	1	2	2	2	3	5	6	7

Step 5. Algorithm

$\textsc{Sort}(A[0..(n-1)])$

Input: An array $A[0..(n-1)]$ of orderable elements
Output: Array $A[0..(n-1)]$ sorted in nondecreasing order

1. **if** $n > 1$ **then**
2. $\quad B[0..(\lfloor n/2 \rfloor - 1)] \leftarrow A[0..(\lfloor n/2 \rfloor - 1)]$
3. $\quad C[0..(\lceil n/2 \rceil - 1)] \leftarrow A[\lfloor n/2 \rfloor..(n-1)]$
4. $\quad$ **parallel:** $\textsc{Sort}(B[0..(\lfloor n/2 \rfloor - 1))$
   $\textsc{Sort}(C[0..(\lceil n/2 \rceil - 1))$
5. $\quad \textsc{Merge}(A, B, C)$

Step 5. Algorithm

$\textsc{Merge}(A[0..(p+q-1)], B[0..(p-1)], C[0..(q-1)])$

Input: Arrays $B[0..(p-1)]$ and $C[0..(q-1)]$ both sorted
Output: Sorted array $A[0..(p+q-1)]$ of the elements of B and C

```
i ← 0; j ← 0; k ← 0
while i < p and j < q do
  if B[i] ≤ C[j] then
    A[k] ← B[i]; i ← i + 1
  else
    A[k] ← C[j]; j ← j + 1
  k ← k + 1
if i = p then
  A[k..(p + q − 1)] ← C[j..(q − 1)]
else
  A[k..(p + q − 1)] ← B[i..(p − 1)]
```

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n \log n)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n \log n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 2Q(n/2) + \Theta(n/B) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n}{B} \log \frac{n}{M}\right)$$

Quicksort HOME

Step 1. Problem

Problem

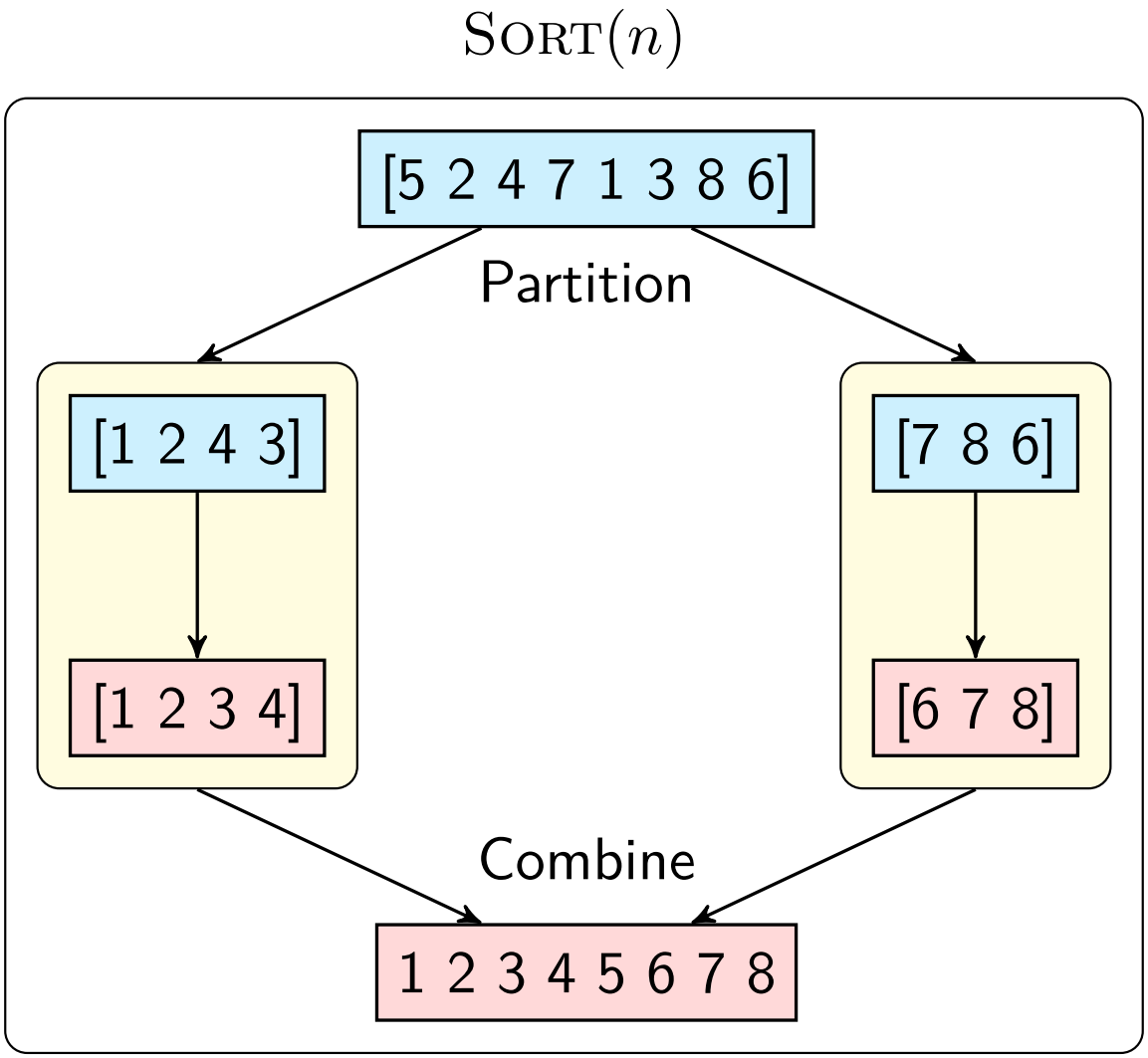
- Sort a given n-sized array in nondecreasing order.

Step 2. Subproblem

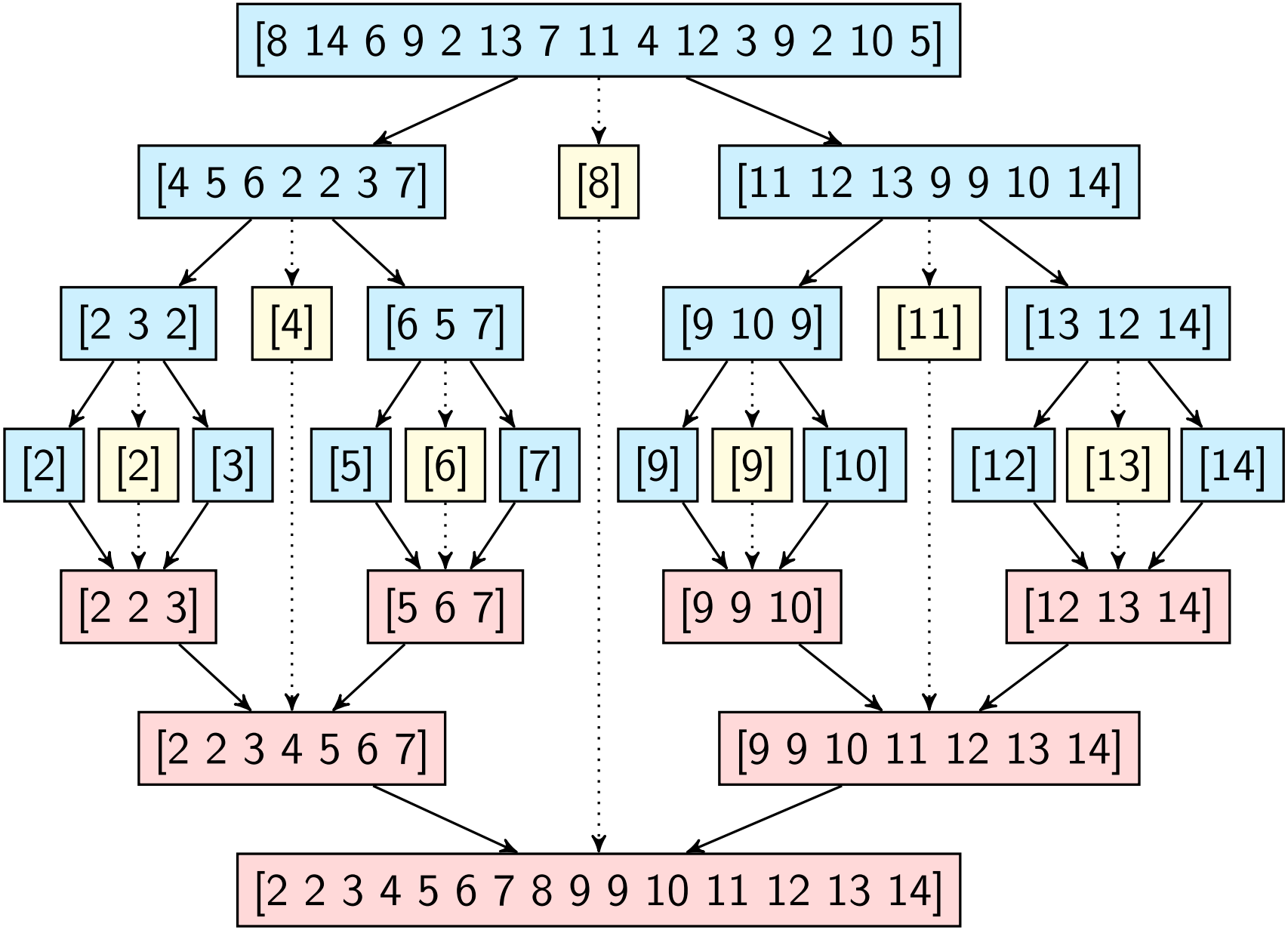
$\textsc{Sort}(A[\ell..h])$ = Sort all elements in subarray $A[\ell..h]$ in nondecreasing order.

Compute $\textsc{Sort}(A[1..n])$.

Step 3. Core idea



Step 4. Example



Step 5. Algorithm

$\textsc{Sort}(A[\ell..h])$

Input: An array $A[\ell..h]$ of orderable elements
Output: Array $A[\ell..h]$ sorted in nondecreasing order

1. **if** $\ell < h$ **then**
2. $\quad s \leftarrow \textsc{RandomizedPartition}(A[\ell..h])$ $\qquad \triangleright$ s is a split position
3. $\quad$ **parallel:** $\textsc{Sort}(A[\ell..s-1])$
   $\qquad\qquad\quad \textsc{Sort}(A[s+1..h])$

Step 5. Algorithm

$\textsc{RandomizedPartition}(A[\ell..h])$

1. $i \leftarrow \textsc{Random}(\{\ell, \ell+1, \ldots, h\})$
2. $\textsc{Swap}(A[\ell], A[i])$
3. $\textsc{HoarePartition}(A[\ell..h])$

$\textsc{HoarePartition}(A[\ell..h])$

1. $pivot \leftarrow A[\ell]$ ▷ first element is the pivot
2. $i \leftarrow \ell$; $j \leftarrow h+1$
3. **while** $true$ **do**
4. {
5. **while** $A[++i] < pivot$ **do**
6. **if** $i = h$ **then break**
7. **while** $pivot < A[--j]$ **do**
8. **if** $j = \ell$ **then break**
9. **if** $i \geq j$ **then break**
10. **else** $\textsc{Swap}(A[i], A[j])$
11. }
12. $\textsc{Swap}(pivot, A[j])$
13. **return** j

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ T(n-1) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^2\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n-1) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^2\right)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ S(n-1) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ Q(n-1) + \Theta(n/B) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n^2}{B}\right)$$

Bitonic Sort HOME

Step 1. Problem

Problem

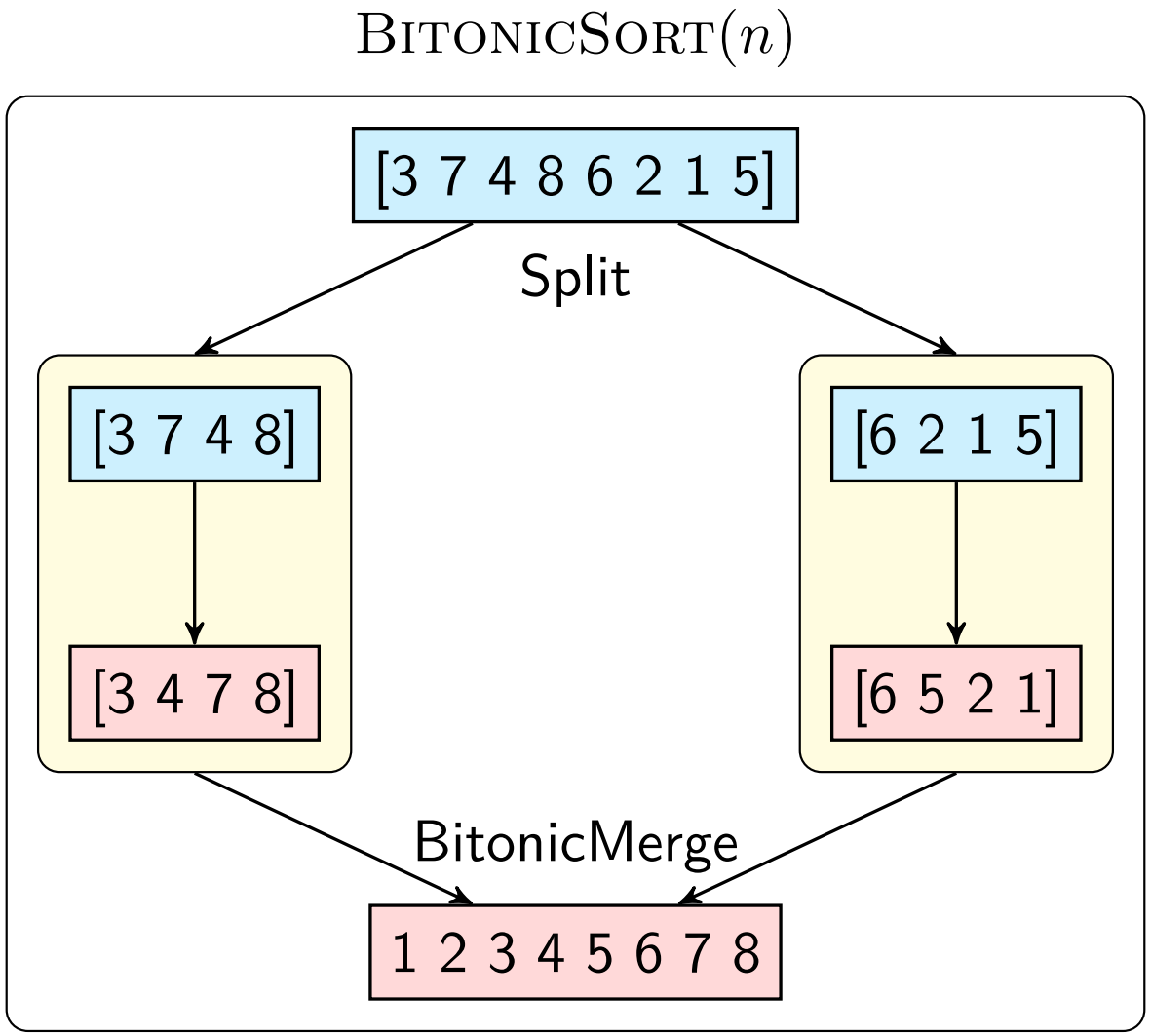
- Sort a given n-sized array in nondecreasing order.

Step 2. Subproblem

$\textsc{Sort}(A[\ell..h])$ = Sort all elements in subarray $A[\ell..h]$ in nondecreasing order.

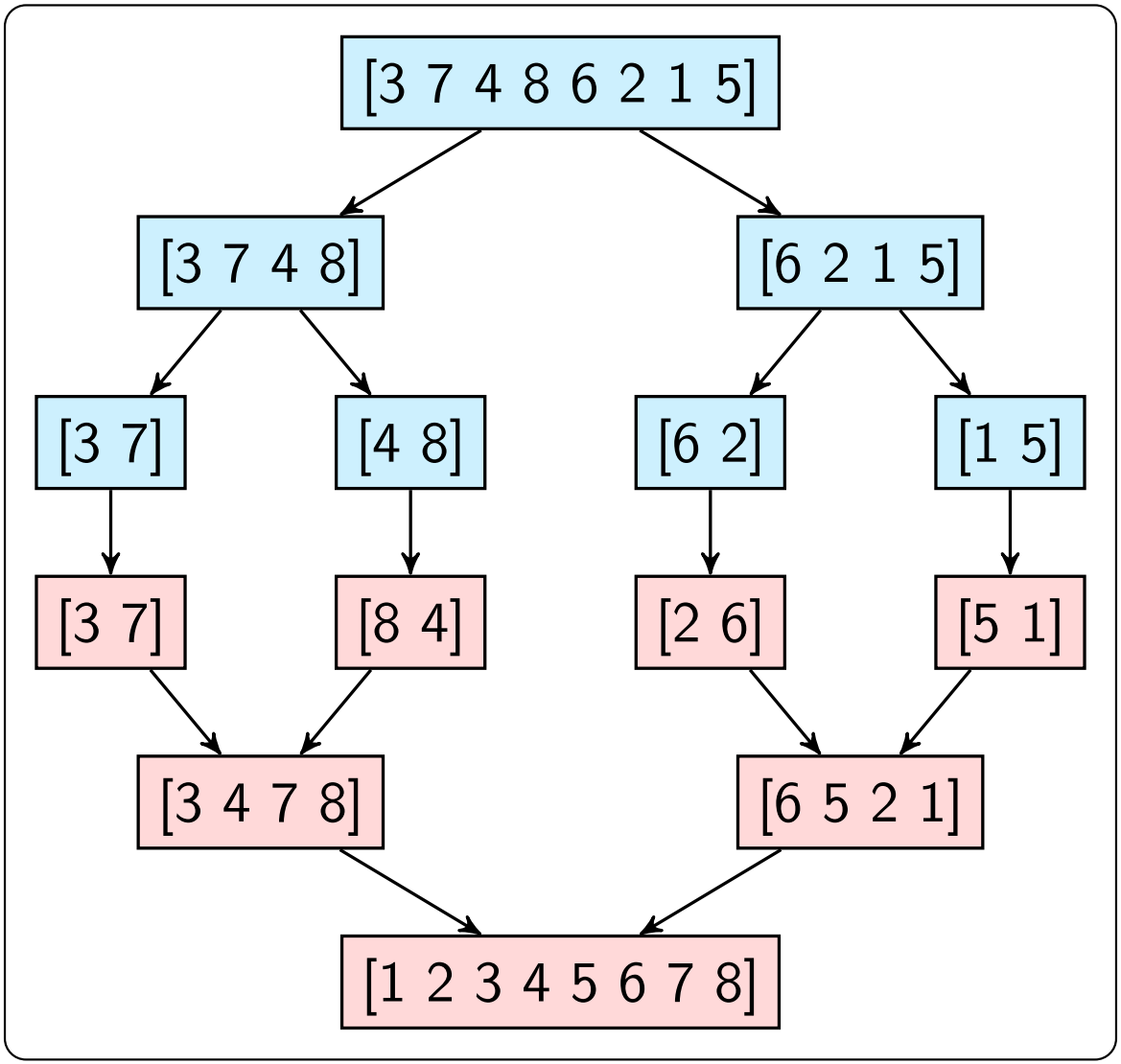
Compute $\textsc{Sort}(A[1..n])$.

Step 3. Core idea



Step 4. Example

BitonicSort(n)



Step 5. Algorithm

$\textsc{BitonicSort}(A[\ell..h], order)$

Input: An array $A[\ell..h]$, ascending/descending
Output: Array $A[\ell..h]$ sorted as per the given order
Invoke: $\textsc{BitonicSort}(A[0..n-1], ascending)$

1. $size \leftarrow h - \ell + 1$
2. **if** $size > 1$ **then**
3. $\quad m \leftarrow (\ell + h)/2$
4. $\quad \textsc{BitonicSort}(A[\ell..m], ascending)$
5. $\quad \textsc{BitonicSort}(A[m+1..h], descending)$
6. $\quad \textsc{BitonicMerge}(A[\ell..h], order)$

Step 5. Algorithm

$\textsc{BitonicMerge}(A[\ell..h], order)$

Input: Array $A[\ell..h]$, ascending/descending order
Output: Bitonic merge the array

1. $size \leftarrow h - \ell + 1$
2. **if** $size > 1$ **then**
3. $\quad m \leftarrow (\ell + h)/2$
4. $\quad \textsc{Compare\&Swap}(A[\ell..h], order)$
5. $\quad$ **parallel:** $\textsc{BitonicMerge}(A[\ell..m], order)$
$\qquad\qquad\qquad\qquad \textsc{BitonicMerge}(A[m+1..h], order)$

$\textsc{Compare\&Swap}(A[\ell..h], order)$

Input: Array $A[\ell..h]$, ascending/descending order
Output: Compare elements in left and right halves of $A[\ell..h]$ and order them

1. $size \leftarrow h - \ell + 1$
2. **for** $i \leftarrow \ell$ **to** $\ell + size/2 - 1$ **do**
3. $\quad j \leftarrow i + size/2$
4. $\quad$ **if** ($order$ is $ascending$ **and** $A[i] > A[j]$) **or**
$\qquad$ ($order$ is $descending$ **and** $A[i] < A[j]$) **then**
5. $\qquad \textsc{Swap}(A[i], A[j])$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(n \log n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n \log^2 n\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n \log n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 2Q(n/2) + \Theta\left(\frac{n}{B} \log \frac{n}{M}\right) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n}{B} \log^2 \frac{n}{M}\right)$$

Integer Multiplication HOME

Step 1. Problem

Problem

- Multiply two n-bit nonnegative binary numbers.
  For simplicity, we assume n is a power of 2.
- Formally, let $A[(n-1)..0]$ and $B[(n-1)..0]$ be n-bit binary numbers. Compute $C = C[(2n-1)..0]$ such that

$$\boxed{C[(2n-1)..0]} = \boxed{A[(n-1)..0]} \times \boxed{B[(n-1)..0]}$$

Step 2. Subproblem

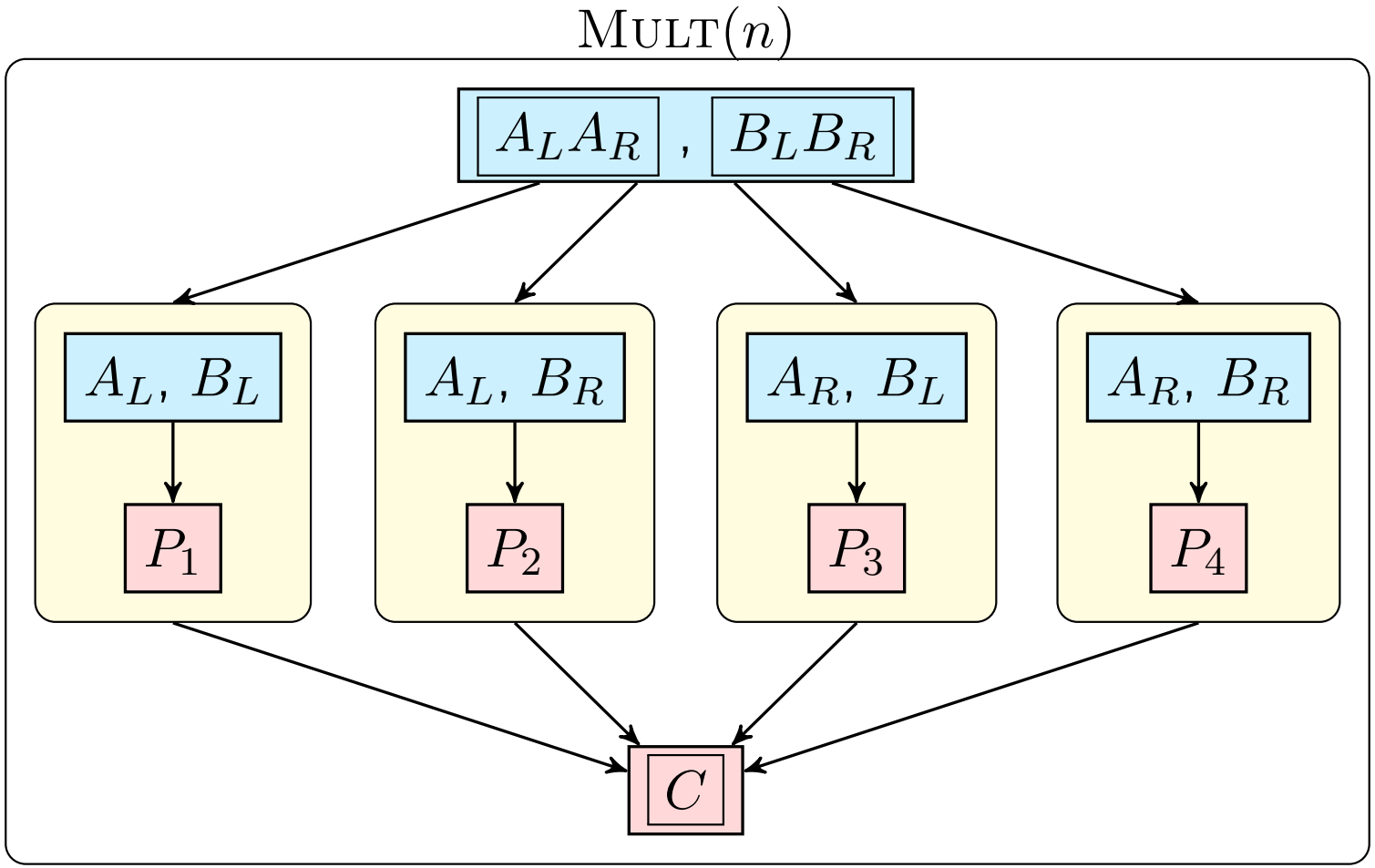
$\textsc{Mult}(A[h..\ell], B[h..\ell])$ = Multiply two nonnegative numbers $A[h..\ell]$ and $B[h..\ell]$.

Compute $\textsc{Mult}(A[n-1..0], B[n-1..0])$.

Step 3. Core Idea

$$
\begin{aligned}
A \times B &= (A_L A_R) \times (B_L B_R) \\
&= (A_L \cdot 2^{n/2} + A_R) \times (B_L \cdot 2^{n/2} + B_R) \\
&= (A_L \times B_L) \cdot 2^n + (A_L \times B_R + A_R \times B_L) \cdot 2^{n/2} \\
&\quad + (A_R \times B_R)
\end{aligned}
$$

Step 3. Core idea



Step 4. Example

$$
\begin{aligned}
1100 \times 1001 &= (11)(00) \times (10)(01) \\
&= (11 \cdot 2^2 + 00) \times (10 \cdot 2^2 + 01) \\
&= (11 \times 10) \cdot 2^4 + (11 \times 01 + 00 \times 10) \cdot 2^2 + (00 \times 01)
\end{aligned}
$$

Step 5. Algorithm

$\textsc{Product}(A[h \ldots \ell], B[h \ldots \ell])$

Input: Two n-bit nonnegative binary numbers A and B, where h and ℓ are the higher and lower order bits and $n = h - \ell + 1$

Output: Product of nonnegative integers A and B

1. **if** $h = \ell$ **then**
2. **return** $A[h] \times B[h]$
3. **else**
4. $mid \leftarrow \lfloor (h + \ell)/2 \rfloor$; $n \leftarrow h - \ell + 1$
5. $A_L \leftarrow A[h \ldots mid]$, $A_R \leftarrow A[mid + 1 \ldots \ell]$
6. $B_L \leftarrow B[h \ldots mid]$, $B_R \leftarrow B[mid + 1 \ldots \ell]$
7. **parallel:** $P_1 \leftarrow \textsc{Product}(A_L, B_L)$
   $P_2 \leftarrow \textsc{Product}(A_L, B_R)$
   $P_3 \leftarrow \textsc{Product}(A_R, B_L)$
   $P_4 \leftarrow \textsc{Product}(A_R, B_R)$
8. **return** $(P_1 \cdot 2^n + (P_2 + P_3) \cdot 2^{n/2} + P_4)$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 4T(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^2\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 4S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^2\right)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 4Q(n/2) + \Theta(n/B) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n^2}{MB}\right)$$

Amazing idea

Problem

Is there is a strategy to perform multiplication of two complex numbers with only 3 multiplications?
$(a+ib)(c+id) = (ac-bd) + i(bc+ad)$

Solution 1

Let $x = bd$, $y = ac$, and $z = (a+b)(c+d)$.
Then, real part $= y - x$ and imaginary part $= z - x - y$.

Solution 2

Let $x = c(a+b)$, $y = a(d-c)$, and $z = b(c+d)$.
Then, real part $= x - z$ and imaginary part $= x + y$.

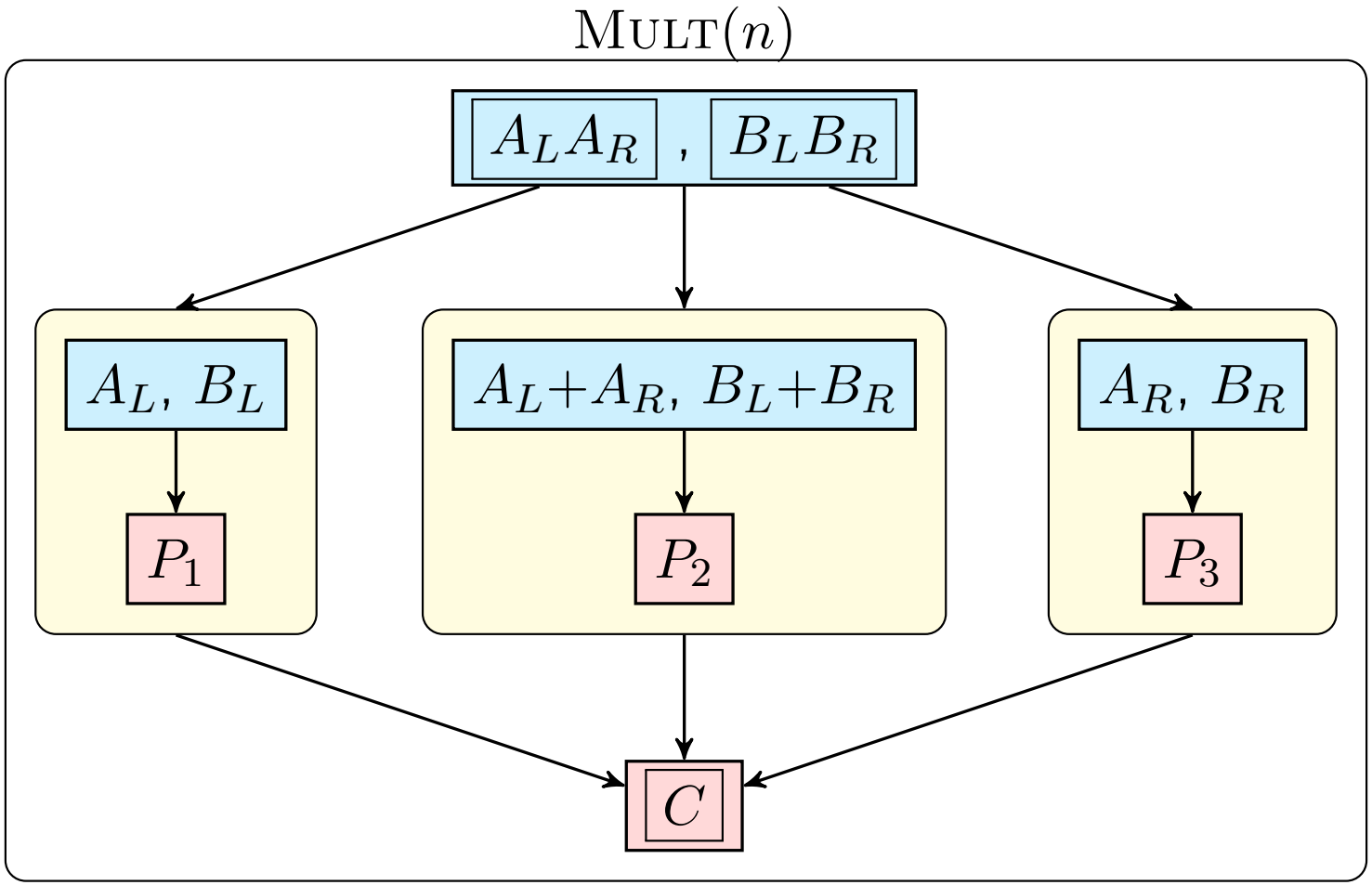
Solution 3

Let $x = c(a+b)$, $y = a(c-d)$, and $z = d(a-b)$.
Then, real part $= y + z$ and imaginary part $= x - y$.

Step 3. Core idea

$$
\begin{aligned}
A \times B &= (A_L A_R) \times (B_L B_R) \\
&= (A_L \cdot 2^{n/2} + A_R) \times (B_L \cdot 2^{n/2} + B_R) \\
&= (A_L \times B_L) \cdot 2^n + (A_L \times B_R + A_R \times B_L) \cdot 2^{n/2} \\
&\quad + (A_R \times B_R) \\
&= (A_L \times B_L) \cdot 2^n \\
&\quad + \left(\begin{array}{c} (A_L + A_R) \times (B_L + B_R) \\ -(A_L \times B_L) - (A_R \times B_R) \end{array} \right) \cdot 2^{n/2} \\
&\quad + (A_R \times B_R)
\end{aligned}
$$

Step 3. Core idea



Step 4. Example

$$
\begin{aligned}
1100 \times 1001 &= (11)(00) \times (10)(01) \\
&= (11 \cdot 2^2 + 00) \times (10 \cdot 2^2 + 01) \\
&= (11 \times 10) \cdot 2^4 + (11 \times 01 + 00 \times 10) \cdot 2^2 + (00 \times 01) \\
&= (11 \times 10) \cdot 2^4 + \left(\begin{array}{c} (11 + 00) \times (10 + 01) \\ -11 \times 10 - 00 \times 01 \end{array} \right) \cdot 2^2 \\
&\quad + (00 \times 01)
\end{aligned}
$$

Step 5. Algorithm

$\textsc{KaratsubaProduct}(A[h \ldots \ell], B[h \ldots \ell])$

Input: Two n-bit nonnegative binary numbers A and B, where h and ℓ are the higher and lower order bits and $n = h - \ell + 1$

Output: Product of nonnegative integers A and B

1. **if** $h = \ell$ **then**
2. **return** $A[h] \times B[h]$
3. **else**
4. $mid \leftarrow \lfloor (h + \ell)/2 \rfloor$; $n \leftarrow h - \ell + 1$
5. $A_L \leftarrow A[h \ldots mid]$, $A_R \leftarrow A[mid + 1 \ldots \ell]$
6. $B_L \leftarrow B[h \ldots mid]$, $B_R \leftarrow B[mid + 1 \ldots \ell]$
7. **parallel:** $P_1 \leftarrow \textsc{KaratsubaProduct}(A_L, B_L)$
   $P_2 \leftarrow \textsc{KaratsubaProduct}((A_L + A_R), (B_L + B_R))$
   $P_3 \leftarrow \textsc{KaratsubaProduct}(A_R, B_R)$
8. **return** $(P_1 \cdot 2^n + (P_2 - P_1 - P_3) \cdot 2^{n/2} + P_3)$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 3T(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^{\log_2 3}\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 3S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^{\log_2 3}\right)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 3Q(n/2) + \Theta(n/B) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n^{\log_2 3}}{MB}\right)$$

Matrix Multiplication HOME

Step 1. Problem

Example

$$\begin{bmatrix} 2 & 7 & 3 & 6 \\ 5 & 8 & 3 & 8 \\ 6 & 4 & 5 & 6 \\ 0 & 3 & 9 & 7 \end{bmatrix} \times \begin{bmatrix} 8 & 4 & 4 & 3 \\ 7 & 7 & 6 & 8 \\ 5 & 3 & 8 & 4 \\ 2 & 5 & 5 & 7 \end{bmatrix} = \begin{bmatrix} 92 & 96 & 104 & 116 \\ 127 & 125 & 132 & 147 \\ 113 & 97 & 118 & 112 \\ 80 & 83 & 125 & 109 \end{bmatrix}$$

- A's ith row $\times$ B's jth column $= C[i,j]$ cell
- E.g.: $5 \times 4 + 8 \times 6 + 3 \times 8 + 8 \times 5 = 132$

Definition

If A and B are $n \times n$ matrices consisting of real numbers, then the matrix product $C = A \times B$ is defined and computed as

$$C[i,j] = \sum_{k=1}^{n} A[i,k] \times B[k,j] \text{ for } i,j \in [1,n]$$

Step 2. Subproblem

$\mathrm{MM}(A, B) =$ Multiply two square submatrices A and B.

Compute $\mathrm{MM}(A, B)$.

Step 3. Core idea

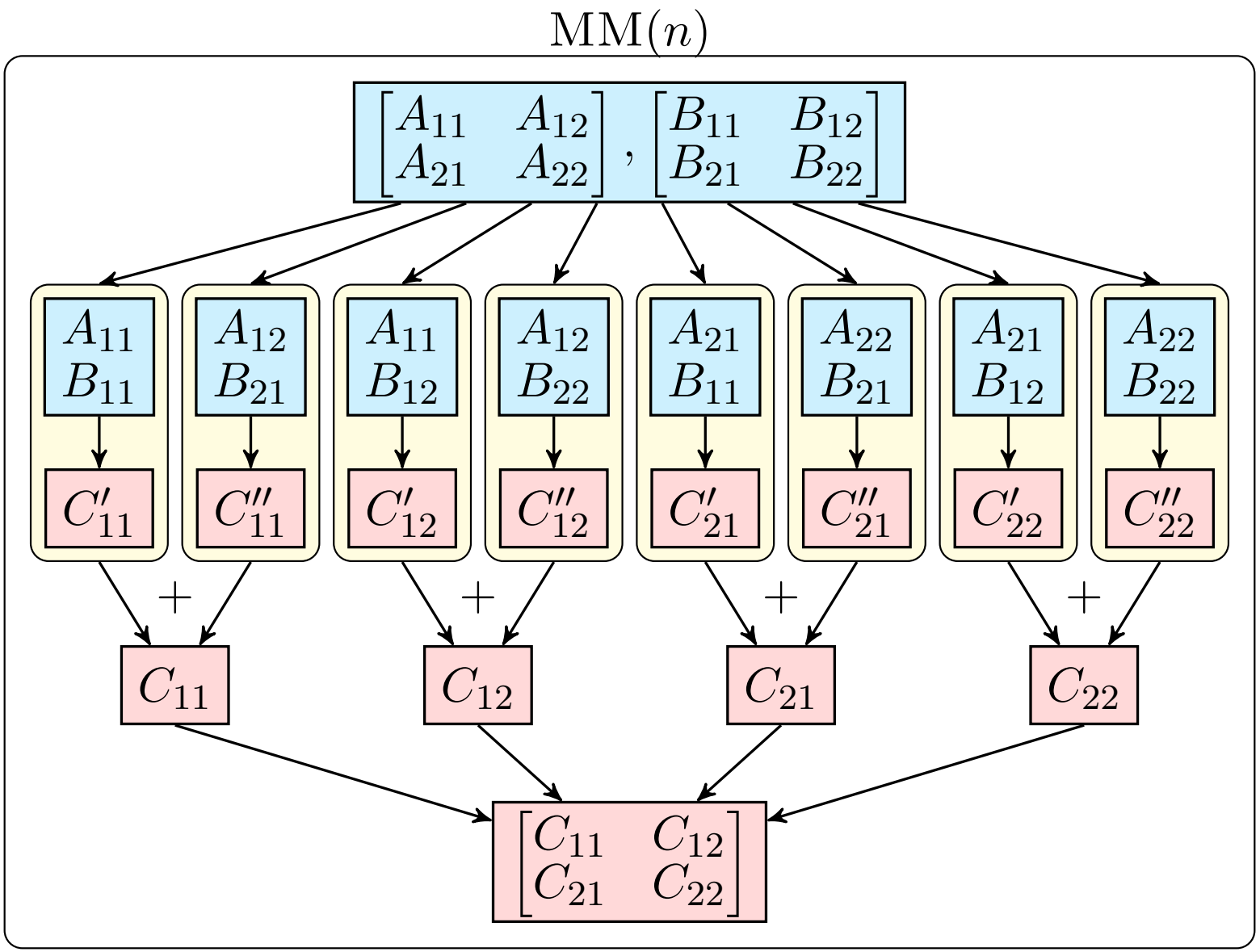
$$C = A \times B$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

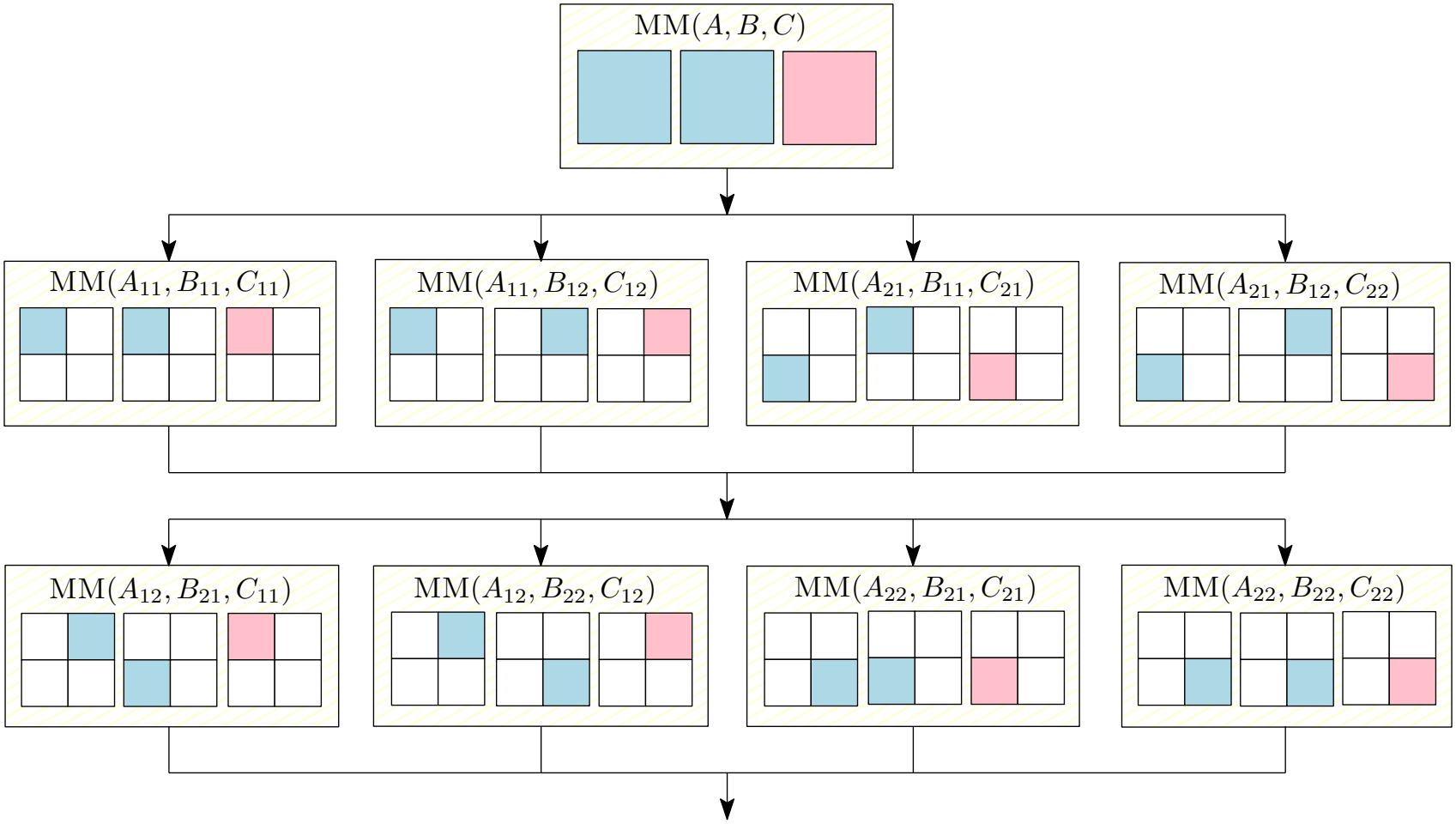
$$= \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11}B_{11} & A_{12}B_{21} \\ A_{21}B_{11} & A_{22}B_{21} \end{bmatrix} + \begin{bmatrix} A_{11}B_{12} & A_{12}B_{22} \\ A_{21}B_{12} & A_{22}B_{22} \end{bmatrix}$$

Step 3. Core idea



Step 3. Core idea



Step 5. Algorithm

$\mathrm{MM}(A, B, C, n)$

1. **if** $n = 1$ **then**
2. $\quad C = C + A \times B$
3. **else**
4. $\quad$ **parallel:** $\mathrm{MM}(A_{11}, B_{11}, C_{11}, n/2)$
   $\mathrm{MM}(A_{11}, B_{12}, C_{12}, n/2)$
   $\mathrm{MM}(A_{21}, B_{11}, C_{21}, n/2)$
   $\mathrm{MM}(A_{21}, B_{12}, C_{22}, n/2)$
5. $\quad$ **parallel:** $\mathrm{MM}(A_{12}, B_{21}, C_{11}, n/2)$
   $\mathrm{MM}(A_{12}, B_{22}, C_{12}, n/2)$
   $\mathrm{MM}(A_{22}, B_{21}, C_{21}, n/2)$
   $\mathrm{MM}(A_{22}, B_{22}, C_{22}, n/2)$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 8T(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^3\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2D(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 4S(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^2\right)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 8Q(n/2) + \Theta(1) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n^3}{\sqrt{M}B}\right)$$

Amazing idea

Problem

Is there is a strategy to perform multiplication of two complex numbers with only 3 multiplications?
$(a+ib)(c+id) = (ac-bd) + i(bc+ad)$

Solution 1

Let $x = bd$, $y = ac$, and $z = (a+b)(c+d)$.
Then, real part $= y - x$ and imaginary part $= z - x - y$.

Solution 2

Let $x = c(a+b)$, $y = a(d-c)$, and $z = b(c+d)$.
Then, real part $= x - z$ and imaginary part $= x + y$.

Solution 3

Let $x = c(a+b)$, $y = a(c-d)$, and $z = d(a-b)$.
Then, real part $= y + z$ and imaginary part $= x - y$.

Step 3. Core idea

Problem	Traditional	Solution 1	Solution 2	Solution 3
Complex number mult.	4 mults 2 adds	3 mults 5 adds	3 mults 5 adds	3 mults 5 adds

$$\begin{bmatrix} C \end{bmatrix} = \begin{bmatrix} A \end{bmatrix} \times \begin{bmatrix} B \end{bmatrix}$$

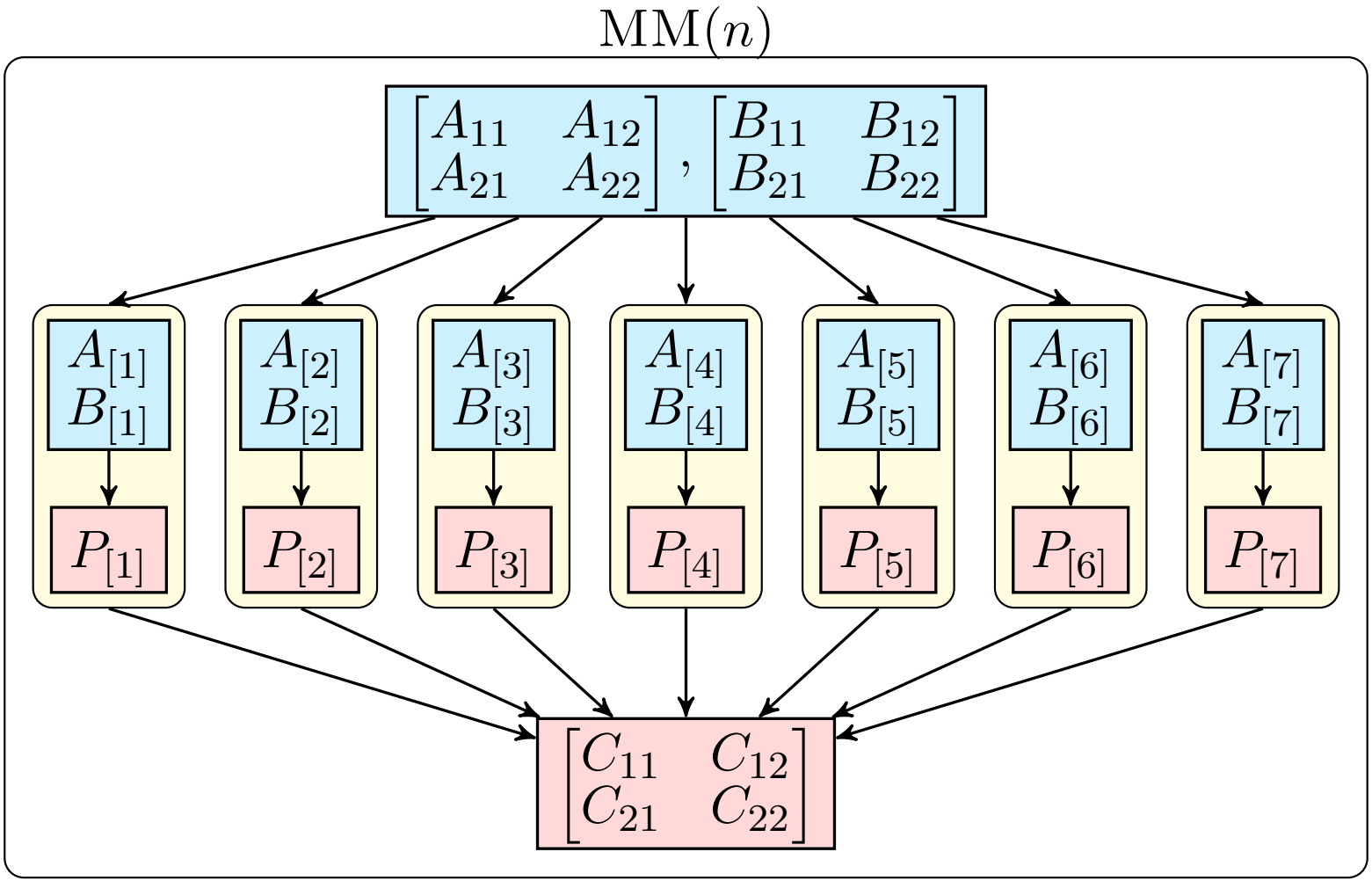
$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

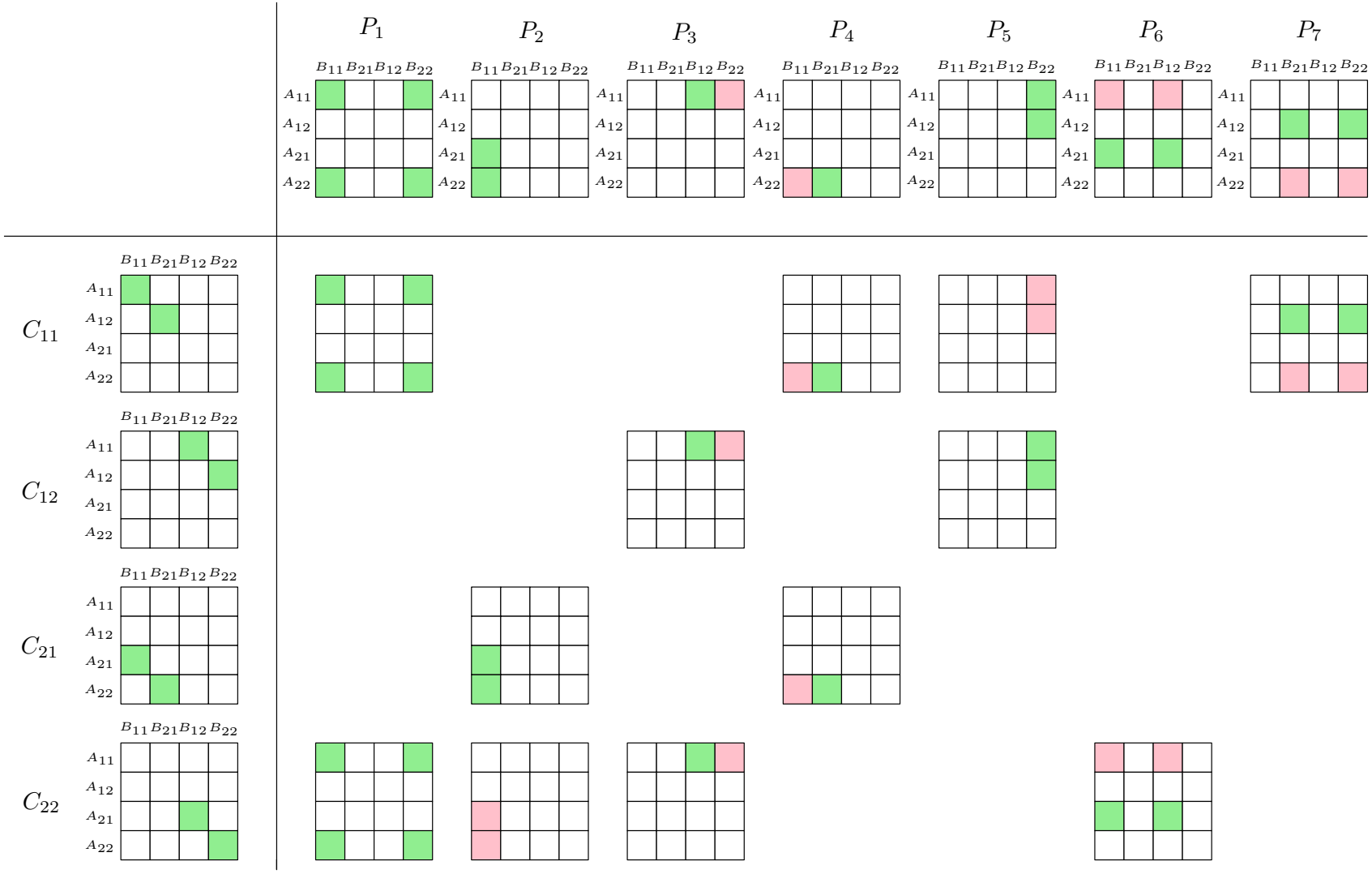
$$= \begin{bmatrix} A_{11}B_{11} & A_{12}B_{21} \\ A_{21}B_{11} & A_{22}B_{21} \end{bmatrix} + \begin{bmatrix} A_{11}B_{12} & A_{12}B_{22} \\ A_{21}B_{12} & A_{22}B_{22} \end{bmatrix}$$

Problem	Traditional	Strassen	Winograd
2×2 MM	8 mults 4 adds	7 mults 18 adds	7 mults 15 adds
$n \times n$ MM	n^3 mults $(n^3 - n^2)$ adds	$n^{\log_2 7}$ mults $(6n^{\log_2 7} - 6n^2)$ adds	$n^{\log_2 7}$ mults $(5n^{\log_2 7} - 5n^2)$ adds

Step 3. Core idea



Step 3. Core idea



Step 5. Algorithm

$\textsc{StrassenMM}(A, B, C, n)$

Input: $n \times n$ matrices A and B
Output: $C \leftarrow A \times B$

1. **if** $n = 1$ **then** $C \leftarrow A \times B$
2. **else**

   [Stage 1. Divide]
3. **parallel:**

$$
\begin{aligned}
&A_{[1]} \leftarrow A_{11} + A_{22}, && B_{[1]} \leftarrow B_{11} + B_{22},\\
&A_{[2]} \leftarrow A_{21} + A_{22}, && B_{[2]} \leftarrow B_{11},\\
&A_{[3]} \leftarrow A_{11}, && B_{[3]} \leftarrow B_{12} - B_{22},\\
&A_{[4]} \leftarrow A_{22}, && B_{[4]} \leftarrow B_{21} - B_{11},\\
&A_{[5]} \leftarrow A_{11} + A_{12}, && B_{[5]} \leftarrow B_{22},\\
&A_{[6]} \leftarrow A_{21} - A_{11}, && B_{[6]} \leftarrow B_{11} + B_{12},\\
&A_{[7]} \leftarrow A_{12} - A_{22}, && B_{[7]} \leftarrow B_{21} + B_{22}
\end{aligned}
$$

   [Stage 2. Conquer]
4. **parallel: for** $i \leftarrow 1$ **to** 7 **do**

   $\textsc{StrassenMM}(A_{[i]}, B_{[i]}, P_{[i]}, n/2)$

   [Stage 3. Combine]
5. **parallel:**

$$
\begin{aligned}
&C_{11} \leftarrow P_{[1]} + P_{[4]} - P_{[5]} + P_{[7]},\ C_{12} \leftarrow P_{[3]} + P_{[5]},\\
&C_{21} \leftarrow P_{[2]} + P_{[4]},\ C_{22} \leftarrow P_{[1]} - P_{[2]} + P_{[3]} + P_{[6]}
\end{aligned}
$$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 7T(n/2) + \Theta(n^2) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^{\log_2 7}\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2D(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 7S(n/2) + \Theta(1) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^{\log_2 7}\right)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 7Q(n/2) + \Theta(1) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n^{\log_2 7}}{\sqrt{M}B}\right)$$

Fast MM algorithms

Year	Discoverer	$T(n)$
–	Classical	$\mathcal{O}\left(n^3\right)$
1969	Volker Strassen	$\mathcal{O}\left(n^{2.808}\right)$
1973	Shmuel Winograd	$\mathcal{O}\left(n^{2.808}\right)$
1978	Victor Pan	$\mathcal{O}\left(n^{2.78017}\right)$
1979	Dario Bini et al.	$\mathcal{O}\left(n^{2.7799}\right)$
1979	Victor Pan	$\mathcal{O}\left(n^{2.6054}\right)$
1981	Arnold Schönhage	$\mathcal{O}\left(n^{2.5479}\right)$
1982	Don Coppersmith & Shmuel Winograd	$\mathcal{O}\left(n^{2.4955480}\right)$
1986	Volker Strassen	$\mathcal{O}\left(n^{2.4785}\right)$
1987	Don Coppersmith & Shmuel Winograd	$\mathcal{O}\left(n^{2.3754770}\right)$
2010	Andrew Stothers	$\mathcal{O}\left(n^{2.3737}\right)$
2014	Virginia Vassilevska Williams	$\mathcal{O}\left(n^{2.372873}\right)$
2014	François Le Gall	$\mathcal{O}\left(n^{2.3728639}\right)$
2020	Josh Alman & Virginia Vassilevska Williams	$\mathcal{O}\left(n^{2.3728596}\right)$

Polynomial Multiplication HOME

Step 1. Problem

Problem

- Multiply two $(n-1)$-degree polynomials.
  For simplicity, we assume n is a power of 2.
- Formally, let $A(x)$ and $B(x)$ be $(n-1)$-degree polynomials.
  Compute $(2n-2)$-degree polynomial $C(x)$ such that

$$\boxed{C(x)} = \boxed{A(x)} \times \boxed{B(x)} \qquad \text{where}$$

$$A(x) = a_0 + a_1 x^1 + \cdots + a_{n-1} x^{n-1}$$

$$B(x) = b_0 + b_1 x^1 + \cdots + b_{n-1} x^{n-1}$$

$$C(x) = c_0 + c_1 x^1 + \cdots + c_{2n-2} x^{2n-2}$$

Step 2. Subproblem

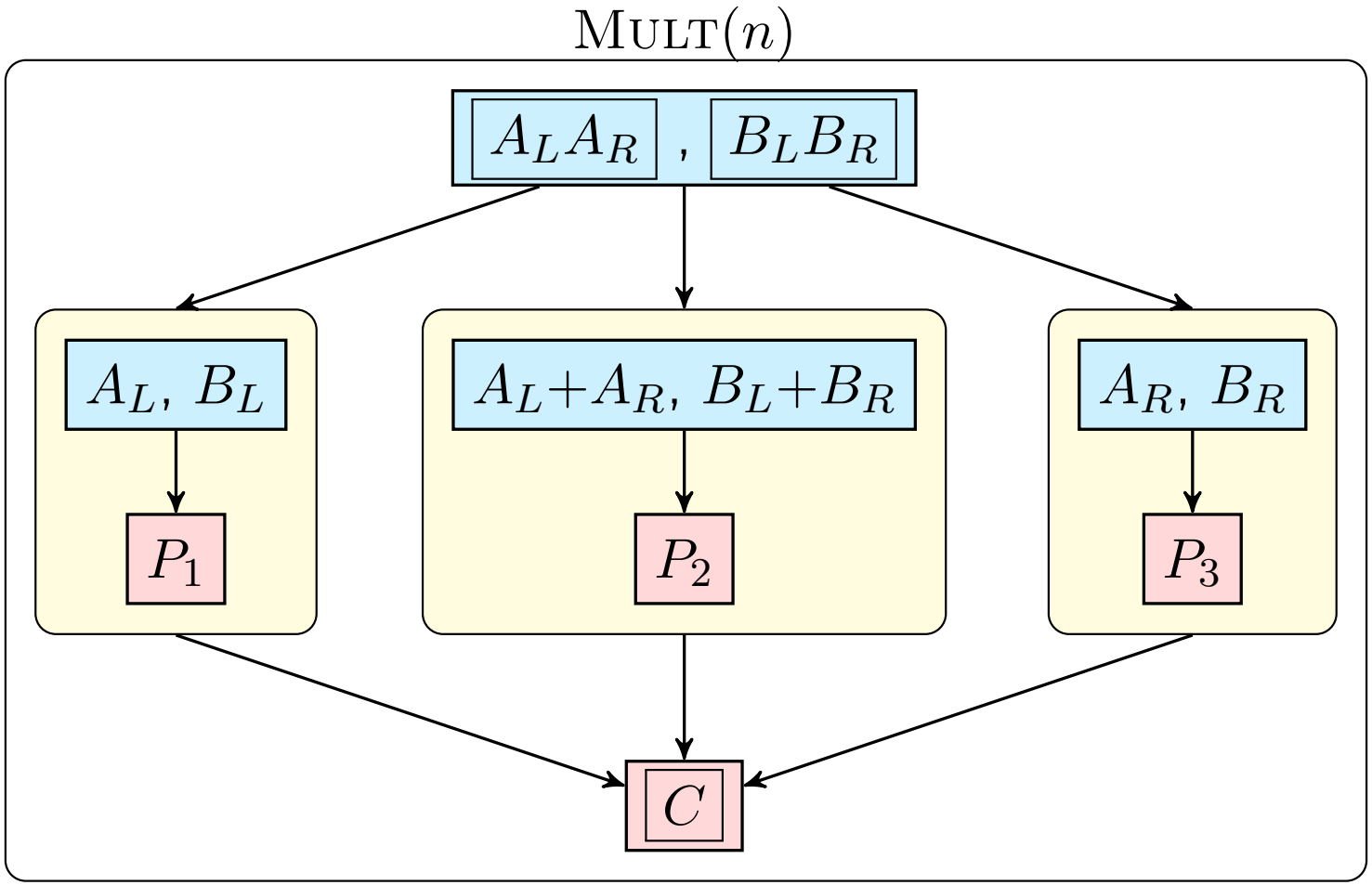
$\textsc{Mult}(A[\ell..h], B[\ell..h])$ = Multiply two $(h - \ell)$ degree polynomials $A[\ell..h]$ and $B[\ell..h]$.

Compute $\textsc{Mult}(A[0..n-1], B[0..n-1])$.

Step 3. Core idea

$$
\begin{aligned}
A \times B &= (A_L A_R) \times (B_L B_R) \\
&= (A_L + A_R \cdot x^{n/2}) \times (B_L + B_R \cdot x^{n/2}) \\
&= (A_L \times B_L) + (A_L \times B_R + A_R \times B_L) \cdot x^{n/2} \\
&\quad + (A_R \times B_R) \cdot x^n \\
&= (A_L \times B_L) \\
&\quad + \begin{pmatrix} (A_L + A_R) \times (B_L + B_R) \\ -(A_L \times B_L) - (A_R \times B_R) \end{pmatrix} \cdot x^{n/2} \\
&\quad + (A_R \times B_R) \cdot x^n
\end{aligned}
$$

Step 3. Core idea



Step 4. Example

Consider

$$A(x) = [-6, 11, -6, 1] = -6 + 11x - 6x^2 + x^3$$
$$B(x) = [-120, 74, -15, 1] = -120 + 74x - 15x^2 + x^3$$

Now consider $A(x) \cdot B(x)$:

$$\begin{aligned}
&[-6, 11, -6, 1] \times [-120, 74, -15, 1] \\
&= ([-6, 11] + [-6, 1]x^2) \times ([-120, 74] + [-15, 1]x^2) \\
&= [-6, 11] \times [-120, 74] \\
&\qquad + ([-6, 11] \times [-15, 1] + [-6, 1] \times [-120, 74])x^2 \\
&\qquad + ([-6, 1] \times [-15, 1])x^4 \\
&= [-6, 11] \times [-120, 74] + \\
&\qquad + \begin{pmatrix} ([-6, 11] + [-6, 1]) \times ([-120, 74] + [-15, 1]) \\ -([-6, 11] \times [-120, 74]) - ([-6, 1] \times [-15, 1]) \end{pmatrix} \cdot x^2 \\
&\qquad + ([-6, 1] \times [-15, 1])x^4
\end{aligned}$$

Step 5. Algorithm

$\textsc{KaratsubaProduct}(A[\ell \ldots h], B[\ell \ldots h])$

Input: Two $(h-\ell)$-degree polynomials A and B, where ℓ and h are the lower and higher order coefficients

Output: Product of polynomials A and B

1. **if** $\ell = h$ **then**
2. **return** $A[\ell] \times B[\ell]$
3. **else**
4. $mid \leftarrow \lfloor (h+\ell)/2 \rfloor$; $n \leftarrow h - \ell + 1$
5. $A_L \leftarrow A[\ell \ldots mid]$, $A_R \leftarrow A[mid+1 \ldots h]$
6. $B_L \leftarrow B[\ell \ldots mid]$, $B_R \leftarrow B[mid+1 \ldots h]$
7. **parallel:** $P_1 \leftarrow \textsc{KaratsubaProduct}(A_L, B_L)$
   $P_2 \leftarrow \textsc{KaratsubaProduct}((A_L + A_R), (B_L + B_R))$
   $P_3 \leftarrow \textsc{KaratsubaProduct}(A_R, B_R)$
8. **return** $(P_1 + (P_2 - P_1 - P_3) \cdot x^{n/2} + P_3 \cdot x^n)$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 3T(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^{\log_2 3}\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 3S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n^{\log_2 3}\right)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 3Q(n/2) + \Theta(n/B) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n^{\log_2 3}}{MB}\right)$$

Polynomial representation

1. Coefficient representation

- $(n-1)$-degree polynomial can be represented using n coefficients
- $A(x) = a_0 + a_1 x^1 + \cdots + a_{n-1} x^{n-1} = \sum_{i=0}^{n-1} a_i x^i$
- $A(x) = [a_0, a_1, \ldots, a_{n-1}]$ $\triangleright$ coefficient vector

2. Root representation

- $(n-1)$-degree polynomial can be represented using $n-1$ roots
- $A(x) = c(x - r_1)(x - r_1) \cdots (x - r_{n-1})$
- $A(x) = [c, \{r_1, r_1, \ldots, r_{n-1}\}]$ $\triangleright$ set of roots

3. Point representation

- $(n-1)$-degree polynomial can be represented using n points
- $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$ where $y_i = A(x_i)$
- $A(x)$ is the set of these sample points $\triangleright$ set of samples

Polynomial representation

1. Coefficient representation

- 3-degree polynomial can be represented using 4 coefficients
- $A(x) = -6 + 11x - 6x^2 + x^3$
- $A(x) = [-6, 11, -6, 1]$ $\triangleright$ coefficient vector

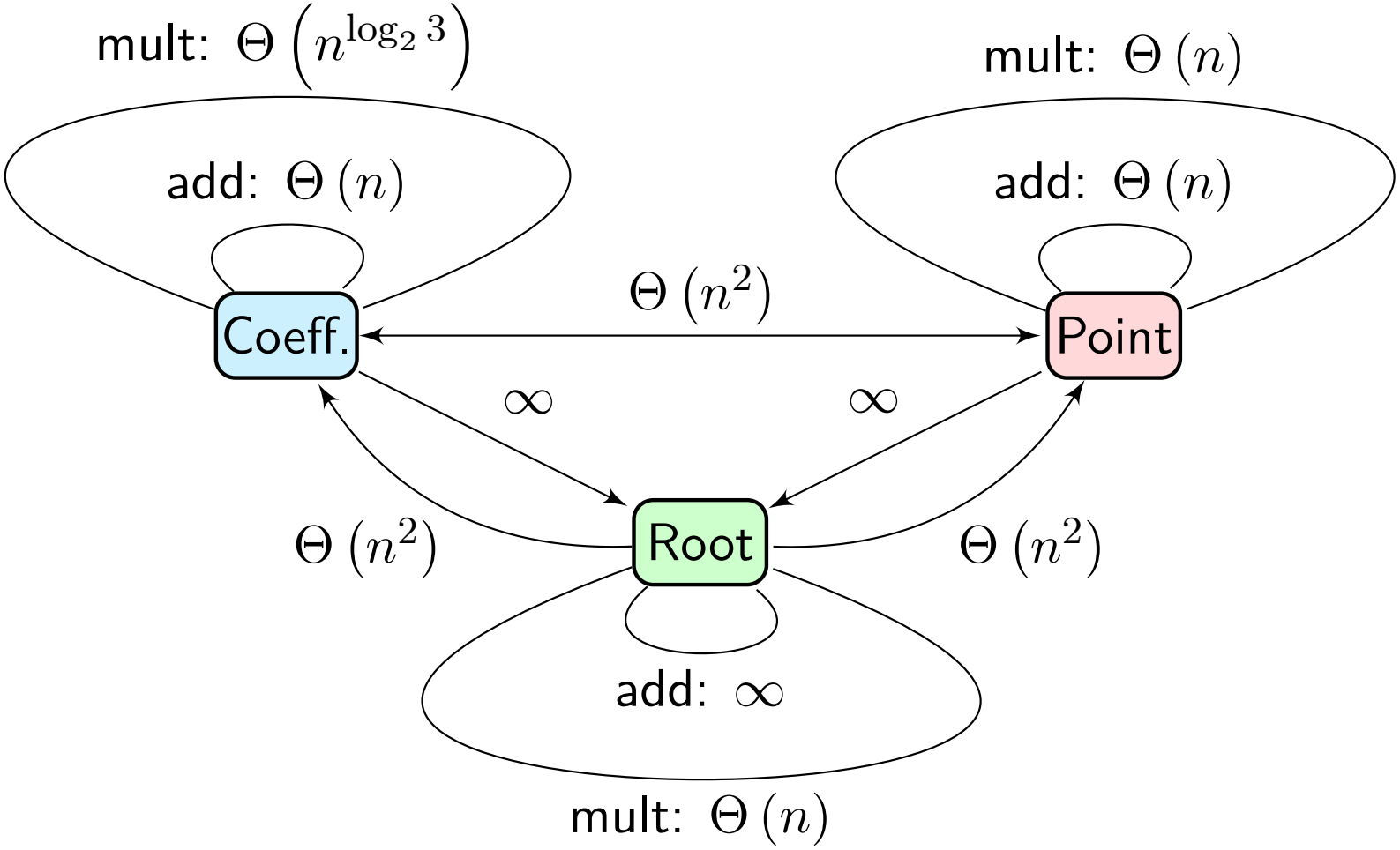
2. Root representation

- 3-degree polynomial can be represented using 3 roots
- $A(x) = 1(x-1)(x-2)(x-3)$
- $A(x) = [1, \{1, 2, 3\}]$ $\triangleright$ set of roots

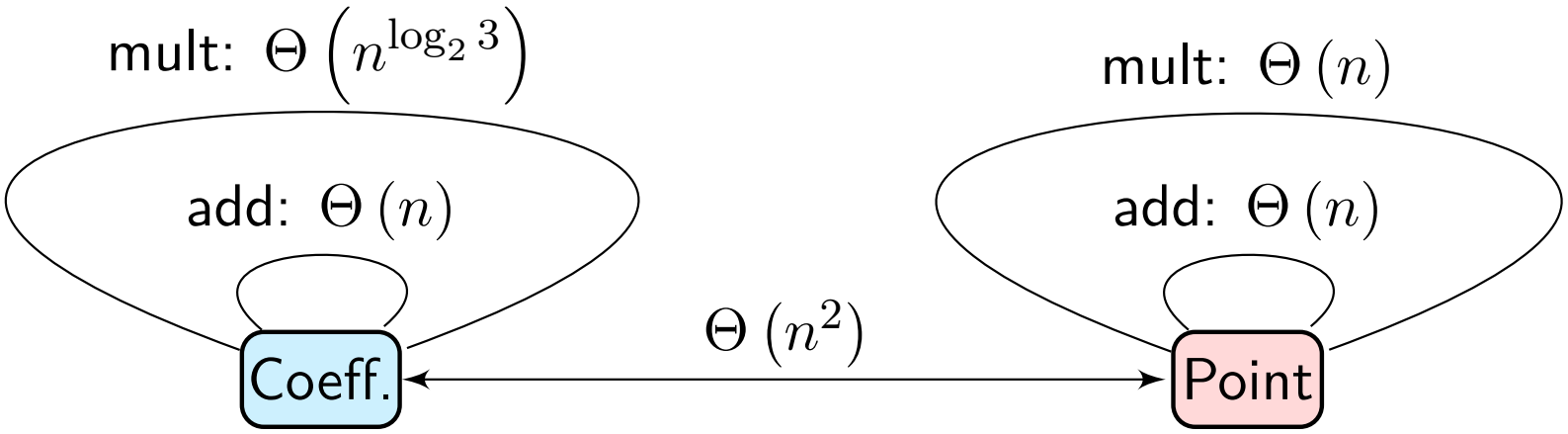
3. Point representation

- 4-degree polynomial can be represented using 4 points
- $\{(0, -6), (10, 504), (20, 5814), (30, 21924)\}$
- $A(x)$ is the set of these sample points $\triangleright$ set of samples

Operations on polynomials



Operations on polynomials



- Root representation is not very useful. Let's remove it.
- Polynomial multiplication can be done in two different ways:
  1. Multiply in coefficient representation using Karatsuba's idea
  2. Convert coefficient to point representation
     Multiply in point representation
     Convert point to coefficient representation

Evaluation and interpolation

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ A(x_2) \\ \vdots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

$\boxed{X = \{x_0, x_1, \ldots, x_{n-1}\}}$ and $\boxed{Y = V_X A}$

- Evaluation.
  Convert coefficient to point representation.
  A is known, X is chosen, Y is computed.
  Y can be computed in $\Theta\left(n^2\right)$ time using Horner's formula.
- Interpolation.
  Convert point to coefficient representation.
  X and Y are known, A is computed.
  A can be computed in $\Theta\left(n^2\right)$ time using Lagrange's formula.

Evaluation and interpolation

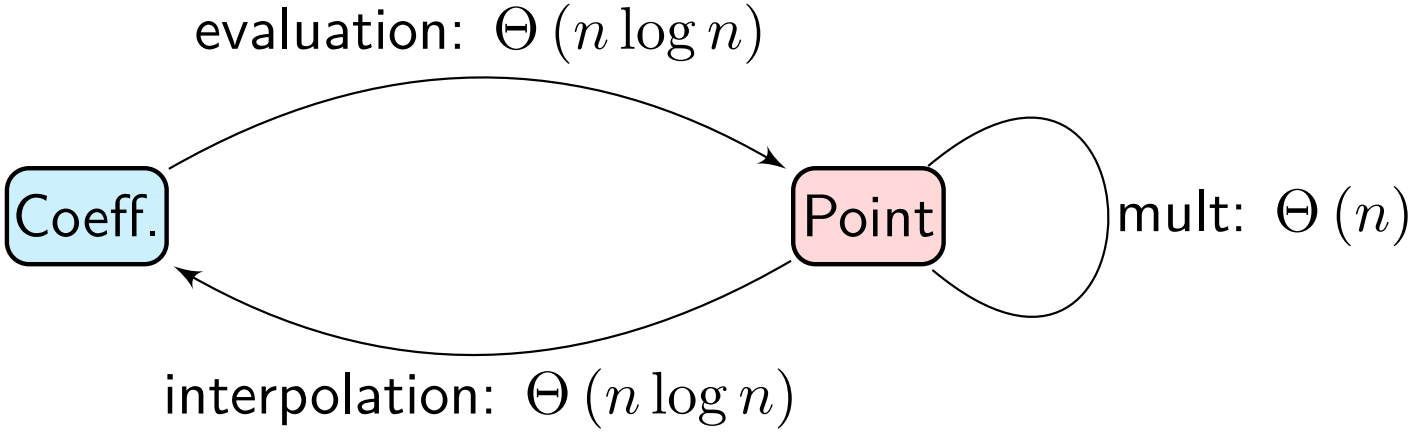
Problem

- Can we perform evaluation & interpolation better than $\Theta\left(n^2\right)$?

Great idea

- We can perform evaluation and interpolation in $\Theta\left(n \log n\right)$ using roots of 1 and divide-and-conquer.
- Evaluation of $(n-1)$-degree polynomial $A(x)$ at n roots of unity can be done in $\Theta\left(n \log n\right)$ using divide-and-conquer. Interpolation of n roots of unity to an $(n-1)$-degree polynomial can be done in $\Theta\left(n \log n\right)$ using divide-and-conquer.

Step 3. Core idea



Step 3. Core idea

- Core idea.

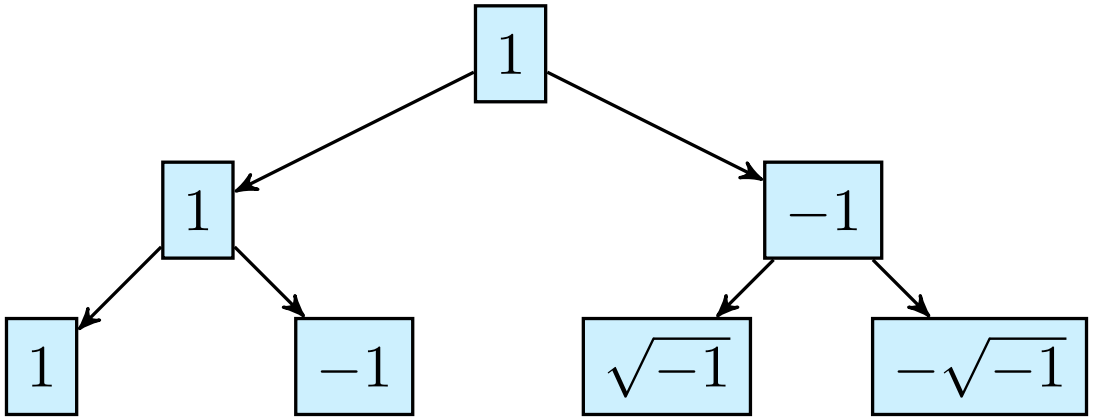
$$\boxed{\begin{aligned} A(x) &= A^{\text{even}}(x^2) + xA^{\text{odd}}(x^2) \\ A(-x) &= A^{\text{even}}(x^2) - xA^{\text{odd}}(x^2) \end{aligned}}$$

- Example.

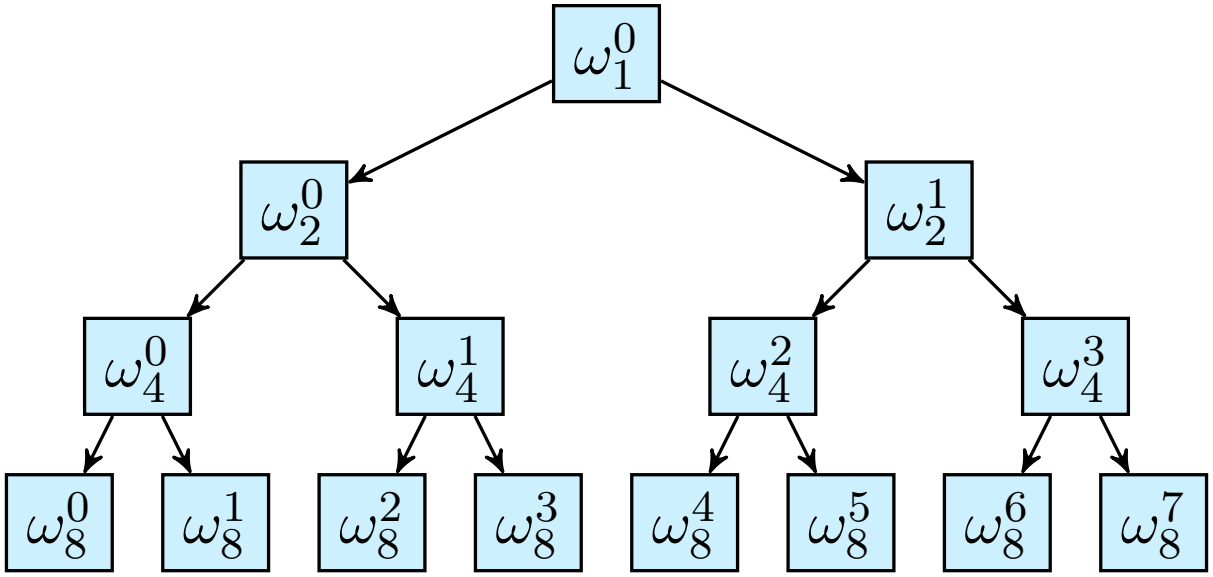
$$\begin{aligned} 7 + 3x + 2x^2 + 6x^3 &= (7 + 2x^2) + x(3 + 6x^2) \\ 7 + 3(-x) + 2(-x)^2 + 6(-x)^3 &= (7 + 2x^2) - x(3 + 6x^2) \end{aligned}$$

- Interpretation.
  If we have the results of $A^{\text{even}}(x^2)$ and $A^{\text{odd}}(x^2)$,
  we can compute $A(x)$ and $A(-x)$ in constant time.
  Because we take square roots repeatedly, we use roots of unity.
  This leads to the amalgamation of ideas from mathematics (roots of unity) and computation (divide-and-conquer).

Roots of unity



- n roots of unity are the n solutions to equation $x^n = 1$.
- n roots of unity are $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$, where $\omega_n^k = e^{k(2\pi i)/n} = \cos(k(2\pi/n)) + i\sin(k(2\pi/n))$ and $i = \sqrt{-1}$.



Step 3. Core idea

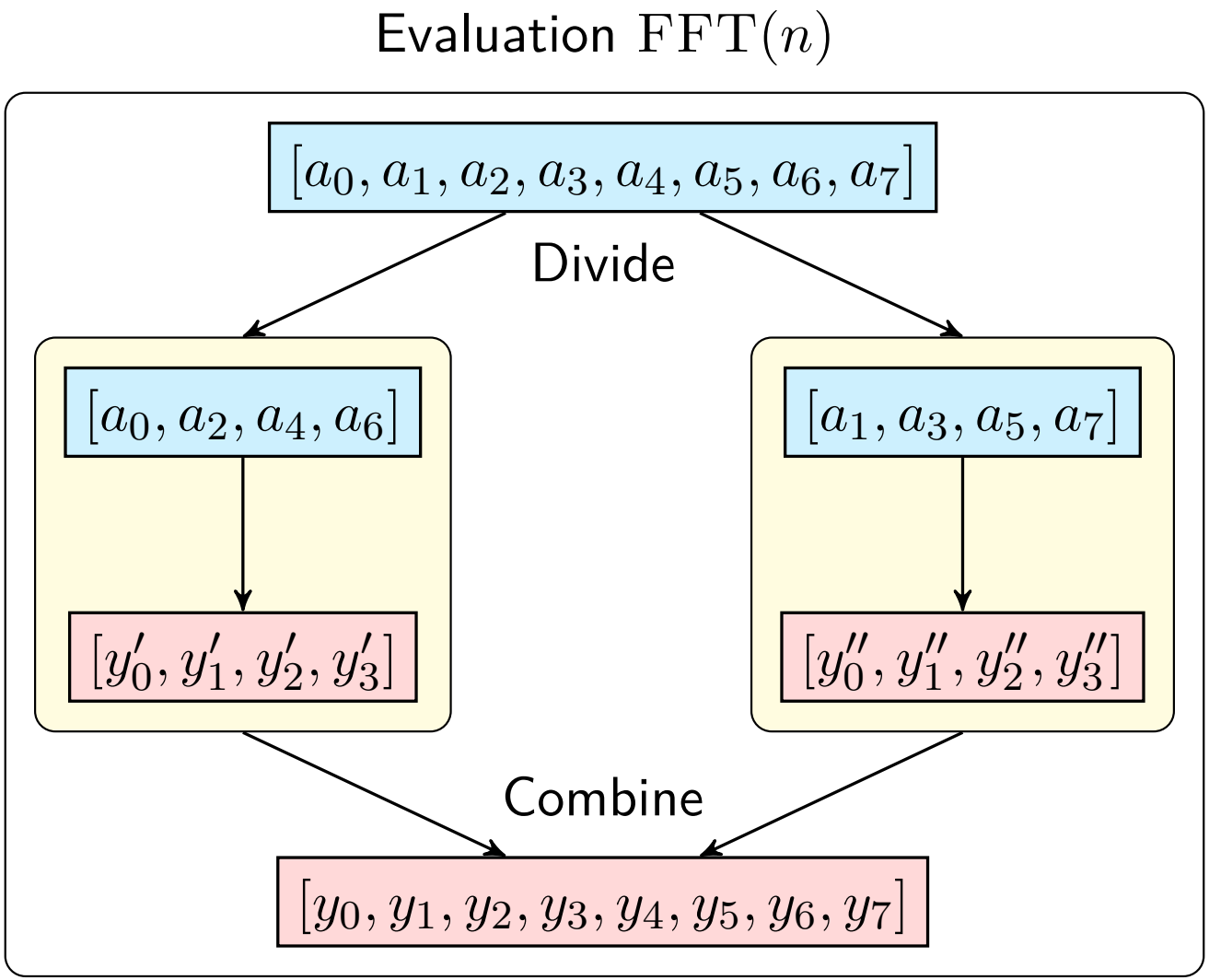
- $\Theta(n \log n)$ evaluation: Use $\boxed{X = \{\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}\}}$.

$$\begin{bmatrix} A(\omega_n^1) \\ A(\omega_n^1) \\ A(\omega_n^2) \\ \vdots \\ A(\omega_n^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & \omega_n^0 & (\omega_n^0)^2 & \cdots & (\omega_n^0)^{n-1} \\ 1 & \omega_n^1 & (\omega_n^1)^2 & \cdots & (\omega_n^1)^{n-1} \\ 1 & \omega_n^2 & (\omega_n^2)^2 & \cdots & (\omega_n^2)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & (\omega_n^{n-1})^2 & \cdots & (\omega_n^{n-1})^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

- $\Theta(n \log n)$ interpolation: Use $\boxed{X = \frac{1}{n}\{\omega_n^{-0}, \omega_n^{-1}, \ldots, \omega_n^{-(n-1)}\}}$.

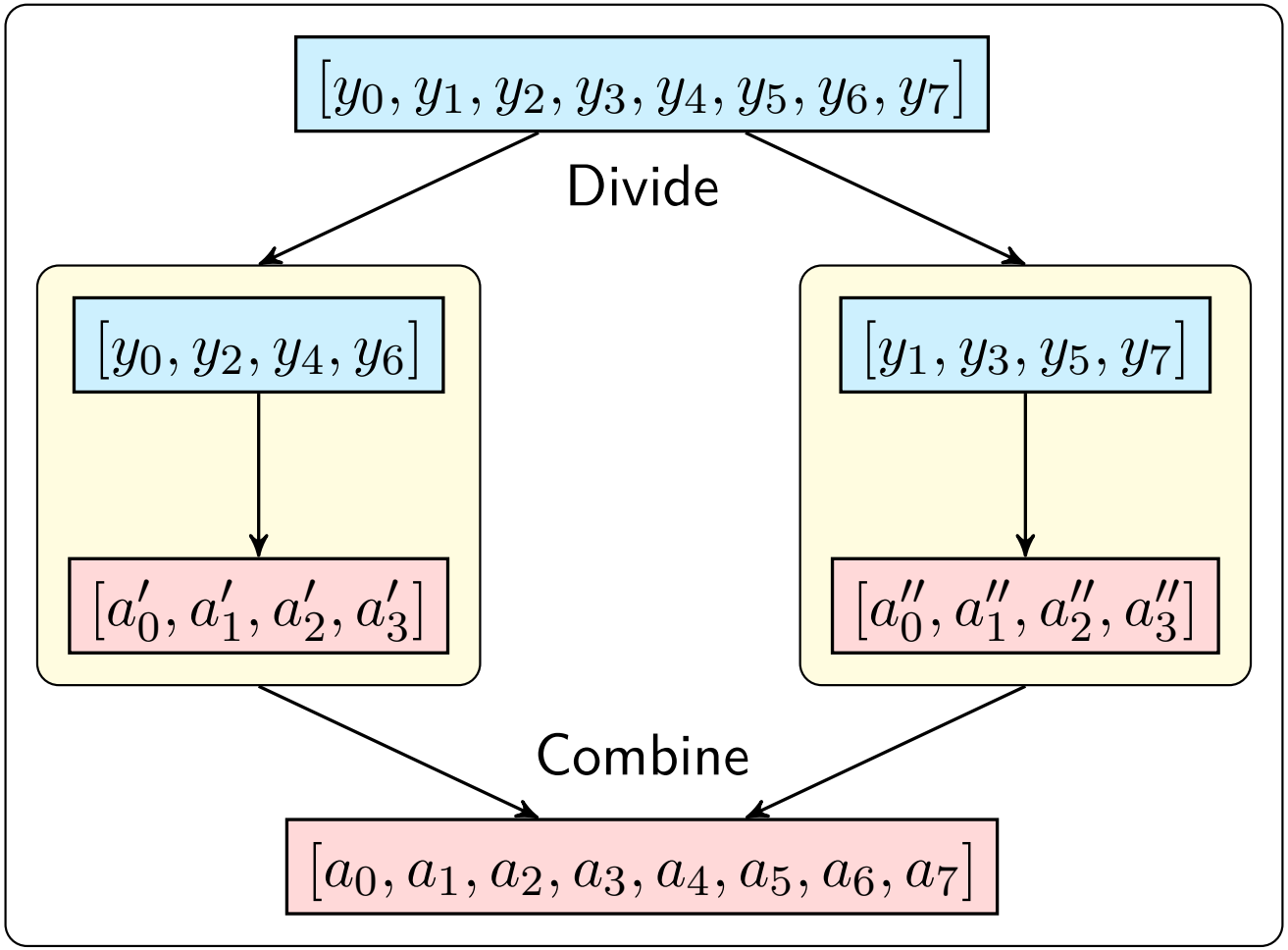
$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 1 & \omega_n^{-0} & (\omega_n^{-0})^2 & \cdots & (\omega_n^{-0})^{n-1} \\ 1 & \omega_n^{-1} & (\omega_n^{-1})^2 & \cdots & (\omega_n^{-1})^{n-1} \\ 1 & \omega_n^{-2} & (\omega_n^{-2})^2 & \cdots & (\omega_n^{-2})^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{-(n-1)} & (\omega_n^{-(n-1)})^2 & \cdots & (\omega_n^{-(n-1)})^{n-1} \end{bmatrix} \begin{bmatrix} A(\omega_n^1) \\ A(\omega_n^1) \\ A(\omega_n^2) \\ \vdots \\ A(\omega_n^{n-1}) \end{bmatrix}$$

Step 3. Core idea



Step 3. Core idea

Interpolation $\textsc{InverseFFT}(n)$



Step 4. Example (Evaluation)

$$[a_0, a_1, a_2, a_3] = [7, 3, 2, 6]$$

$$[a_0, a_1] = [7, 2] \qquad [a_0, a_1] = [3, 6]$$

$$a_0 = 7 \qquad a_0 = 2 \qquad a_0 = 3 \qquad a_0 = 6$$

$$y_0 = 7 \qquad y_0 = 2 \qquad y_0 = 3 \qquad y_0 = 6$$

$$\begin{aligned} y_0 &= 7 + 2\omega_2^0 \\ y_1 &= 7 - 2\omega_2^0 \end{aligned} \qquad \begin{aligned} y_0 &= 3 + 6\omega_2^0 \\ y_1 &= 3 - 6\omega_2^0 \end{aligned}$$

$$\begin{aligned}
y_0 &= (7 + 2\omega_2^0) + (3 + 6\omega_2^0)\omega_4^0 &&= 7 + 3\omega_4^0 + 2(\omega_4^0)^2 + 6(\omega_4^0)^3 \\
y_1 &= (7 - 2\omega_2^0) + (3 - 6\omega_2^0)\omega_4^1 &&= 7 + 3\omega_4^1 + 2(\omega_4^1)^2 + 6(\omega_4^1)^3 \\
y_2 &= (7 + 2\omega_2^0) - (3 + 6\omega_2^0)\omega_4^0 &&= 7 + 3\omega_4^2 + 2(\omega_4^2)^2 + 6(\omega_4^2)^3 \\
y_3 &= (7 - 2\omega_2^0) - (3 - 6\omega_2^0)\omega_4^1 &&= 7 + 3\omega_4^3 + 2(\omega_4^3)^2 + 6(\omega_4^3)^3
\end{aligned}$$

Step 5. Algorithm

$\mathrm{FFT}([a_0, a_1, \ldots, a_{n-1}])$ ▷ Evaluation

Input: Coefficients of polynomial $A(x)$: $[a_0, a_1, \ldots, a_{n-1}]$
Output: Point values vector Y for X values $[\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}]$

1. **if** $n = 1$ **then return** a_0
2. $\omega_n \leftarrow e^{2\pi i/n}$
3. $\omega \leftarrow 1$

   [Stage 1. Divide]
4. $A^{\text{even}} \leftarrow [a_0, a_2, \ldots, a_{n-2}]$
5. $A^{\text{odd}} \leftarrow [a_1, a_3, \ldots, a_{n-1}]$

   [Stage 2. Conquer]
6. **parallel:** $Y^{\text{even}} \leftarrow \mathrm{FFT}(A^{\text{even}})$
   $Y^{\text{odd}} \leftarrow \mathrm{FFT}(A^{\text{odd}})$

   [Stage 3. Combine]
7. **for** $k \leftarrow 0$ **to** $n/2 - 1$ **do**
8. $\quad y_k \leftarrow Y_k^{\text{even}} + \omega Y_k^{\text{odd}}$
9. $\quad y_{n/2+k} \leftarrow Y_k^{\text{even}} - \omega Y_k^{\text{odd}}$
10. $\quad \omega \leftarrow \omega\omega_n$
11. **return** $[y_0, y_1, \ldots, y_{n-1}]$

Step 5. Algorithm

$\textsc{InverseFFT}([y_0, y_1, \ldots, y_{n-1}])$ ▷ Interpolation

Input: Point values vector Y for X values $[\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}]$
Output: Coefficients of polynomial $A(x)$: $[a_0, a_1, \ldots, a_{n-1}]$

1. **if** $n = 1$ **then return** y_0
2. $\omega_n \leftarrow (1/n)e^{-2\pi i/n}$
3. $\omega \leftarrow 1$

[Stage 1. Divide]

4. $Y^{\text{even}} \leftarrow [y_0, y_2, \ldots, y_{n-2}]$
5. $Y^{\text{odd}} \leftarrow [y_1, y_3, \ldots, y_{n-1}]$

[Stage 2. Conquer]

6. **parallel:** $A^{\text{even}} \leftarrow \textsc{InverseFFT}(Y^{\text{even}})$
   $A^{\text{odd}} \leftarrow \textsc{InverseFFT}(Y^{\text{odd}})$

[Stage 3. Combine]

7. **for** $k \leftarrow 0$ **to** $n/2 - 1$ **do**
8. $\quad a_k \leftarrow A_k^{\text{even}} + \omega A_k^{\text{odd}}$
9. $\quad a_{n/2+k} \leftarrow A_k^{\text{even}} - \omega A_k^{\text{odd}}$
10. $\quad \omega \leftarrow \omega\omega_n$
11. **return** $[a_0, a_1, \ldots, a_{n-1}]$

Step 5. Algorithm

$\textsc{CooleyTukeyProduct}(A(x), B(x))$

Input: Polynomials $A(x)$ and $B(x)$ of same degree
Output: Polynomial product $C(x) = A(x) \times B(x)$

1. $[a_0, a_1, \ldots, a_{n-1}] \leftarrow \textsc{Coefficients}(A(x))$
2. $[b_0, b_1, \ldots, b_{n-1}] \leftarrow \textsc{Coefficients}(B(x))$

[Stage 1. Add high-order coefficients]

3. $[a_n, a_{n+1}, \ldots, a_{2n-1}] \leftarrow [0, 0, \ldots, 0]$
4. $[b_n, b_{n+1}, \ldots, b_{2n-1}] \leftarrow [0, 0, \ldots, 0]$

[Stage 2. Evaluate]

5. **parallel:** $[y_0^A, y_1^A, \ldots, y_{2n-1}^A] \leftarrow \mathrm{FFT}([a_0, a_1, \ldots, a_{2n-1}])$
   $[y_0^B, y_1^B, \ldots, y_{2n-1}^B] \leftarrow \mathrm{FFT}([b_0, b_1, \ldots, b_{2n-1}])$

[Stage 3. Pointwise multiply]

6. **parallel:** **for** $k \leftarrow 0$ **to** $2n - 1$ **do**
7. $\quad y_k^C \leftarrow y_k^A \times y_k^B$

[Stage 4. Interpolate]

8. $[c_0, c_1, \ldots, c_{2n-1}] \leftarrow \textsc{InverseFFT}([y_0^C, y_1^C, \ldots, y_{2n-1}^C])$
9. $C(x) \leftarrow [c_0, c_1, \ldots, c_{2n-1}]$
10. **return** $C(x)$

Step 6. Complexity

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n \log n)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n \log n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 2Q(n/2) + \Theta(n/B) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n}{B} \log \frac{n}{M}\right)$$

Coin Toss HOME

Step 1. Problem

- [Link] Given n biased coins such that the probability of getting head from coin $i \in [1, n]$ is $p[i]$, compute the probability of getting exactly k heads when these n coins are tossed.

Step 2. Subproblem

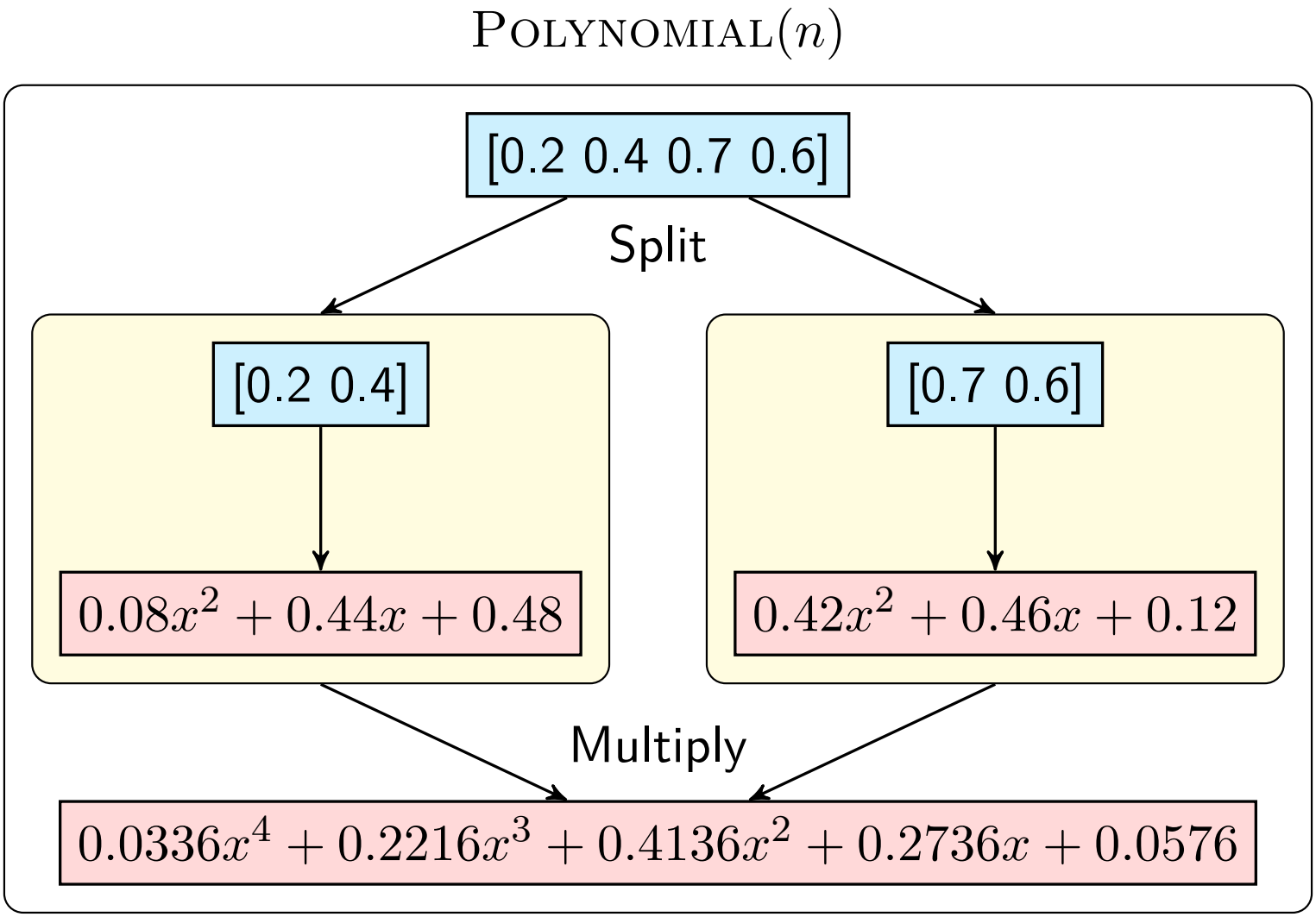
$P(i, j, k) =$ Probability of getting k heads when coins in the range $[i, j]$ are tossed.

(We set $P(i, j, k) = 0$ if $k > j - i + 1$.)

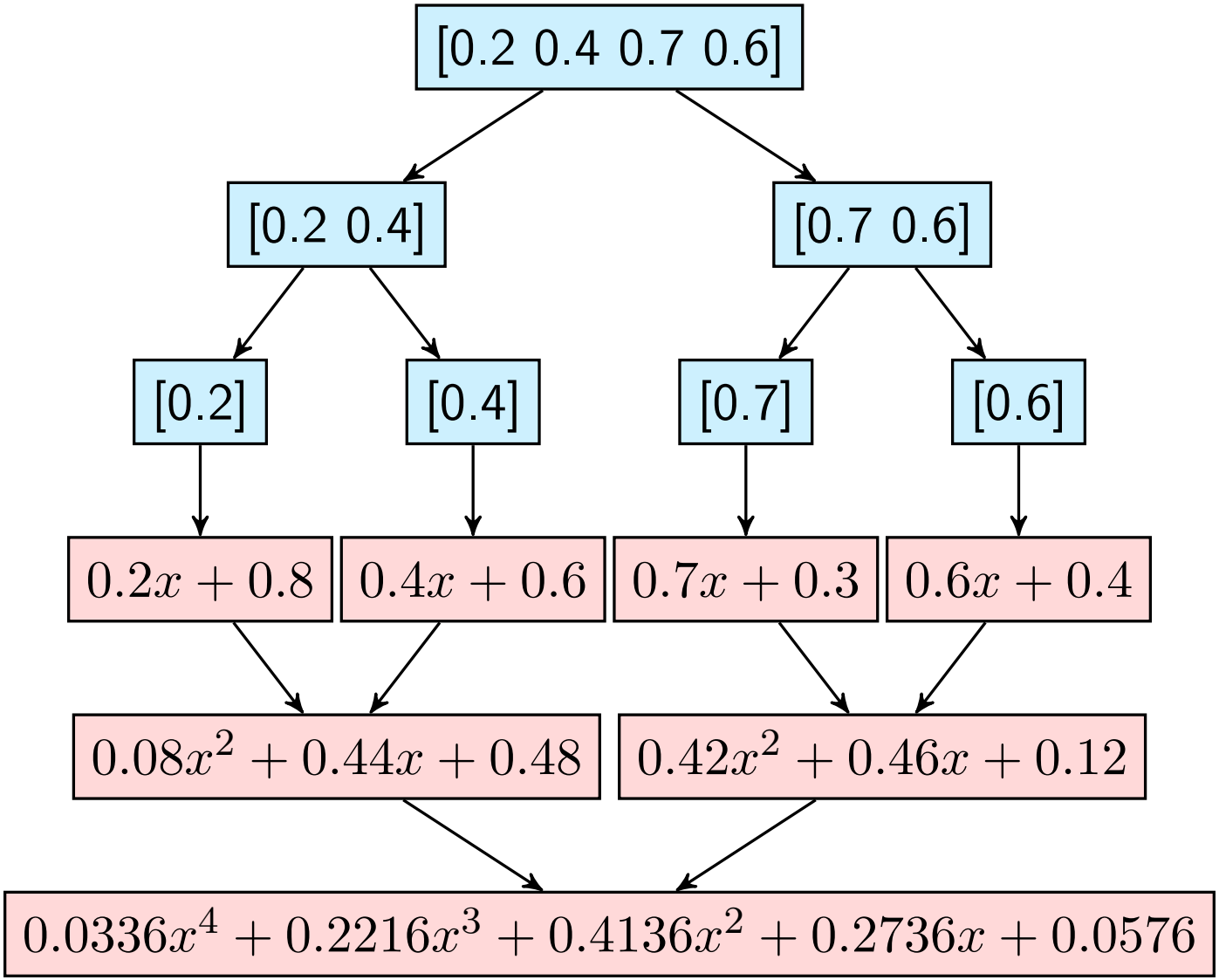
$$
\begin{aligned}
F_{i,j}(x) &= \sum_{k=0}^{j-i+1} P(i, j, k) x^k \\
&= P(i, j, j-i+1) x^{j-i+1} + P(i, j, j-i) x^{j-i} + \cdots \\
&\quad + P(i, j, 2) x^2 + P(i, j, 1) x^1 + P(i, j, 0) x^0
\end{aligned}
$$

Compute the coefficient of x^k in the polynomial $F_{1,n}(x)$.

Step 3. Core idea



Step 4. Example



Step 5. Algorithm

$\textsc{CountingHeads}(p[1..n], k)$

Input: Array $p[1..n]$ of success probabilities for coins $[1..n]$ and k representing the number of heads required.
Output: Probability of getting k heads when n coins are tossed.

1. $polynomial \leftarrow \textsc{Polynomial}(p[1..n])$
2. $result \leftarrow$ coefficient of x^k in $polynomial$
3. **return** $result$

$\textsc{Polynomial}(p[\ell..h])$

Input: Array $p[\ell..h]$ of success probabilities for coins $[\ell..h]$.
Output: Polynomial $F_{\ell,h}(x)$.

1. **if** $\ell = h$ **then**
2. **return** $p[\ell] \times x + (1 - p[\ell])$
3. **else**
4. $m \leftarrow (\ell + h)/2$
5. **parallel:** $lpolynomial \leftarrow \textsc{Polynomial}(p[\ell..m])$
   $rpolynomial \leftarrow \textsc{Polynomial}(p[m+1..h])$
6. $polynomial \leftarrow \textsc{Multiply}(lpolynomial, rpolynomial)$
7. **return** $polynomial$

Step 6. Complexity

- We use FFT as the polynomial multiplication algorithm.

$$\text{Work } T(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2T(n/2) + \Theta(n \log n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(n \log^2 n\right)$$

$$\text{Depth } D(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ D(n/2) + \Theta(\log^n) & \text{if } n > 1. \end{array} \right\} \in \Theta\left(\log^3 n\right)$$

$$\text{Space } S(n) = \left\{ \begin{array}{ll} \Theta(1) & \text{if } n = 1, \\ 2S(n/2) + \Theta(n) & \text{if } n > 1. \end{array} \right\} \in \Theta(n \log n)$$

$$\text{Cache } Q(n) = \left\{ \begin{array}{ll} \mathcal{O}(M/B) & \text{if } n \leq \gamma M, \\ 2Q(n/2) + \Theta\left(\frac{n}{B} \log \frac{n}{M}\right) & \text{if } n > \gamma M. \end{array} \right\} \in \mathcal{O}\left(\frac{n}{B} \log^2 \frac{n}{M}\right)$$