The Truncated Fourier Transform for Mixed Radices

Robin Larrieu

Laboratoire d'informatique de l'École polytechnique (LIX)
Palaiseau, France

ISSAC 2017
July 27, Kaiserslautern, Germany

Evaluation-interpolation

Example: polynomial multiplication. Let $A, B \in \mathbb{K}[X]$ with $\deg AB < n$.

Naive method

By definition, $AB = \sum_{i<n} \sum_{j<i} A_j B_{i-j} X^i$ (quadratic complexity).

Evaluation-interpolation

Example: polynomial multiplication. Let $A, B \in \mathbb{K}[X]$ with $\deg AB < n$.

Naive method

By definition, $AB = \sum_{i<n} \sum_{j<i} A_j B_{i-j} X^i$ (quadratic complexity).

Evaluation-Interpolation

- Evaluation

$$\begin{aligned} A &\rightarrow (A(x_0), A(x_1), \ldots, A(x_{n-1})) \\ B &\rightarrow (B(x_0), B(x_1), \ldots, B(x_{n-1})) \end{aligned}$$

- Interpolation

$$(AB(x_0), AB(x_1), \ldots, AB(x_{n-1})) \rightarrow AB$$

The Fast Fourier Transform

Assumptions

- Let $\omega \in \mathbb{K}$ be a primitive n-th root of unity.
- The order n is highly composite (ideally a power of 2)

The Fast Fourier Transform

Assumptions

- Let $\omega \in \mathbb{K}$ be a primitive n-th root of unity.
- The order n is highly composite (ideally a power of 2)

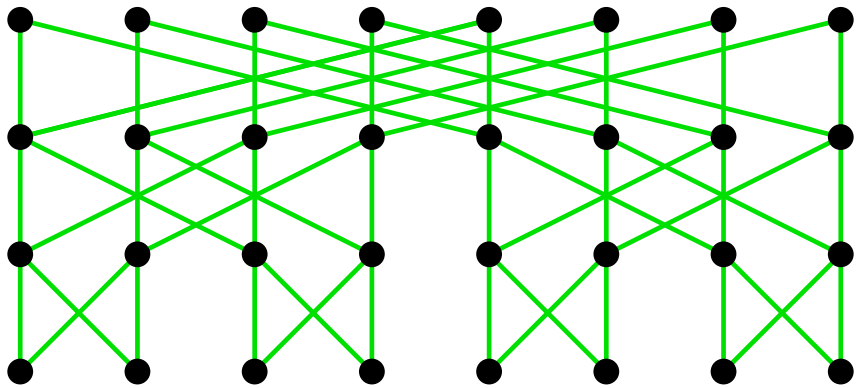
FFT

- Fast evaluation-interpolation scheme

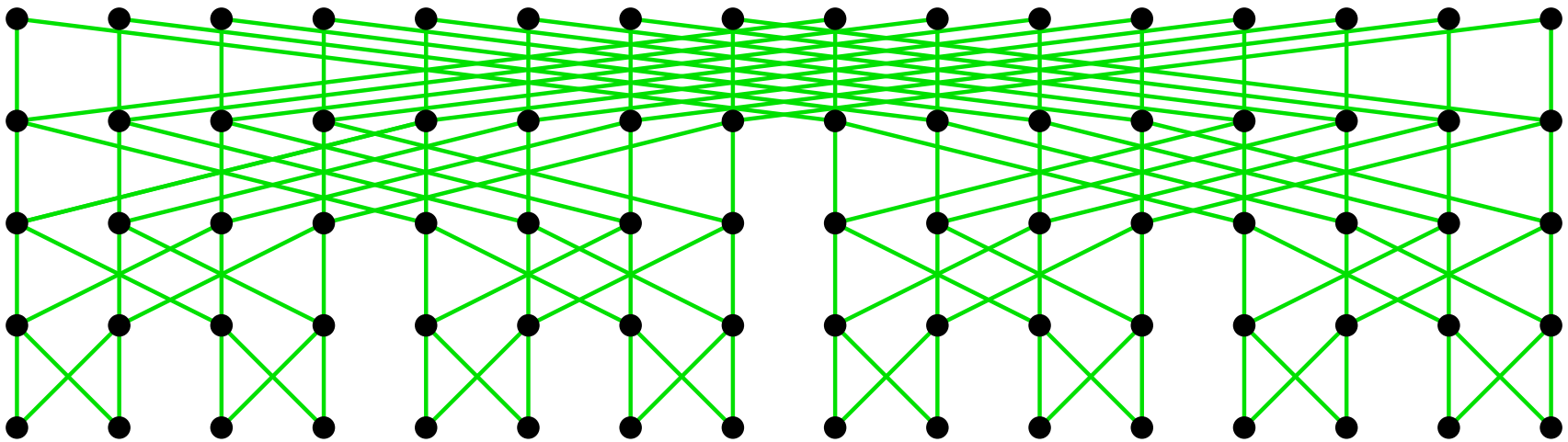
$$A \in \mathbb{K}[X]_{<n} \quad \overset{FFT}{\longleftrightarrow} \quad (A(1), A(\omega), \ldots, A(\omega^{n-1})) \in \mathbb{K}^n$$

- *Divide-and-conquer* strategy. Quasi-linear complexity.

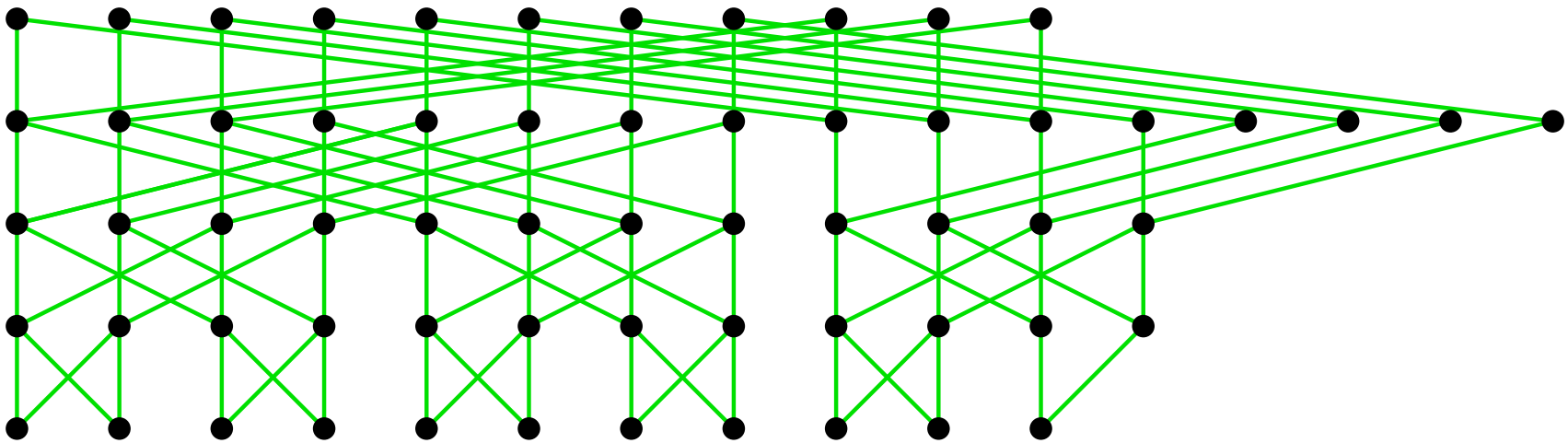
The Truncated Fourier Transform [van der Hoeven – 2004]



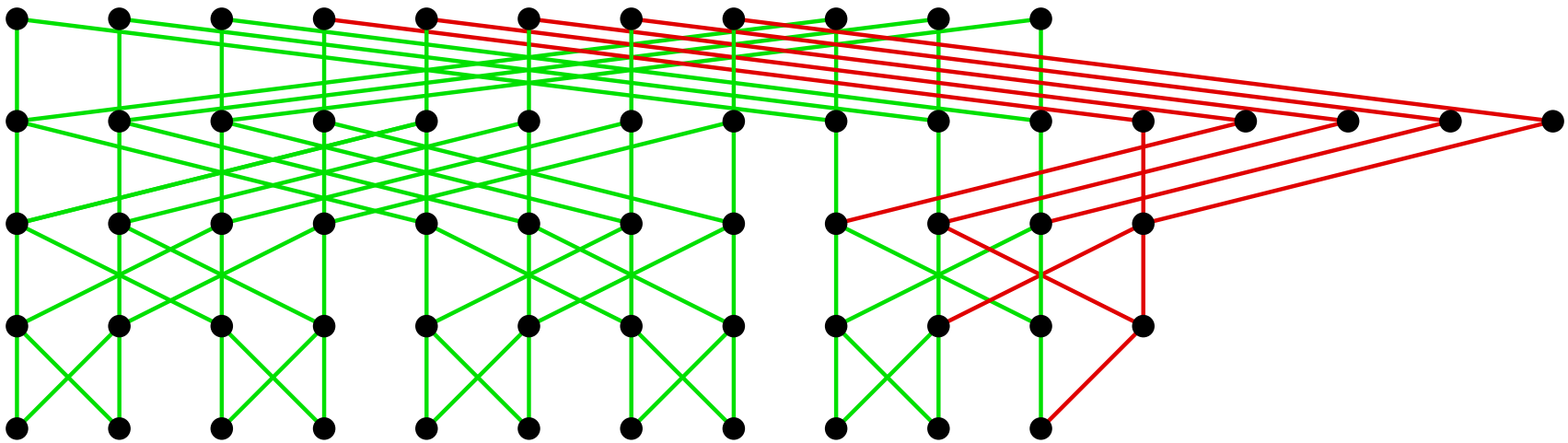
The Truncated Fourier Transform [van der Hoeven – 2004]



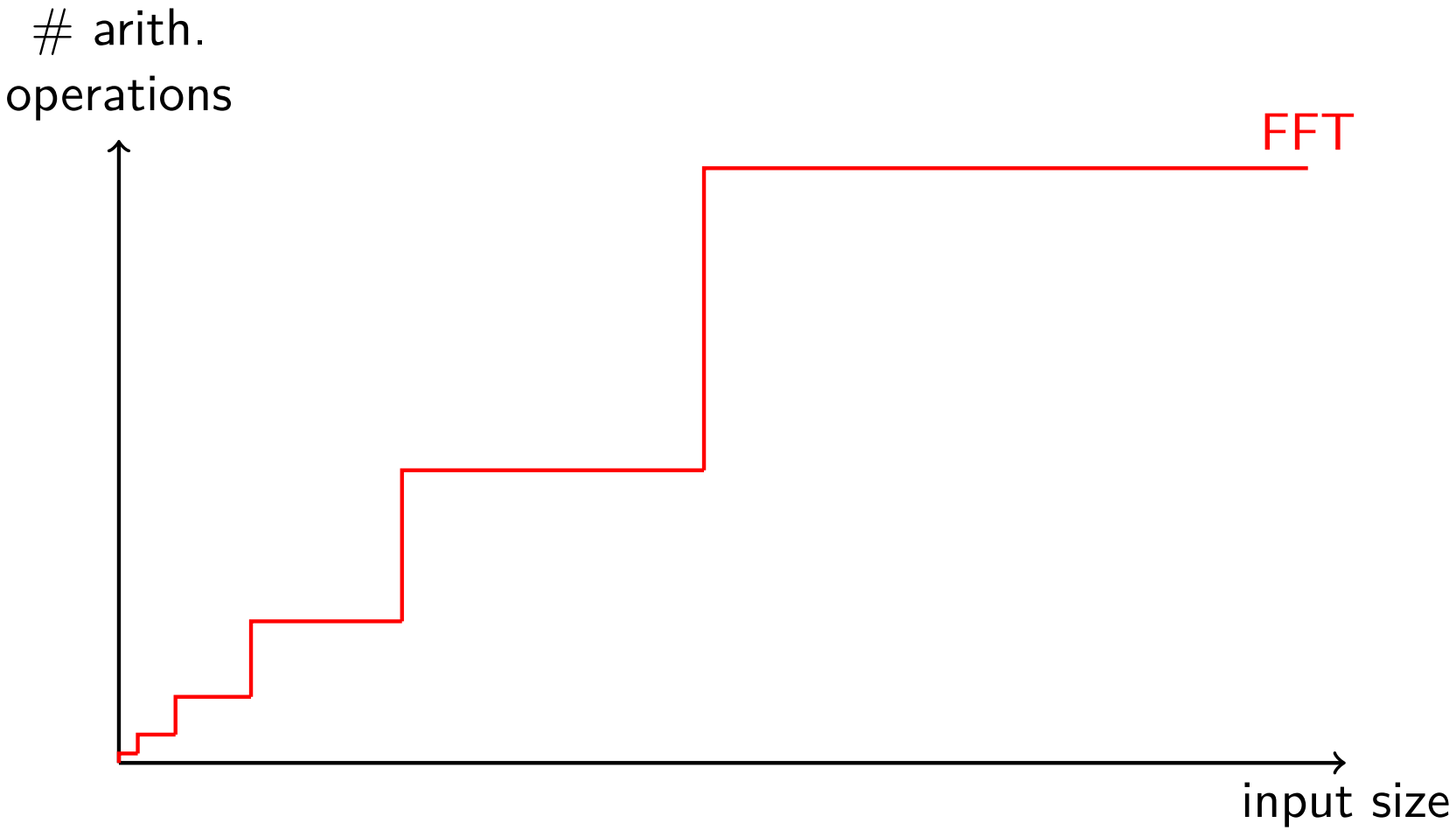
The Truncated Fourier Transform [van der Hoeven – 2004]



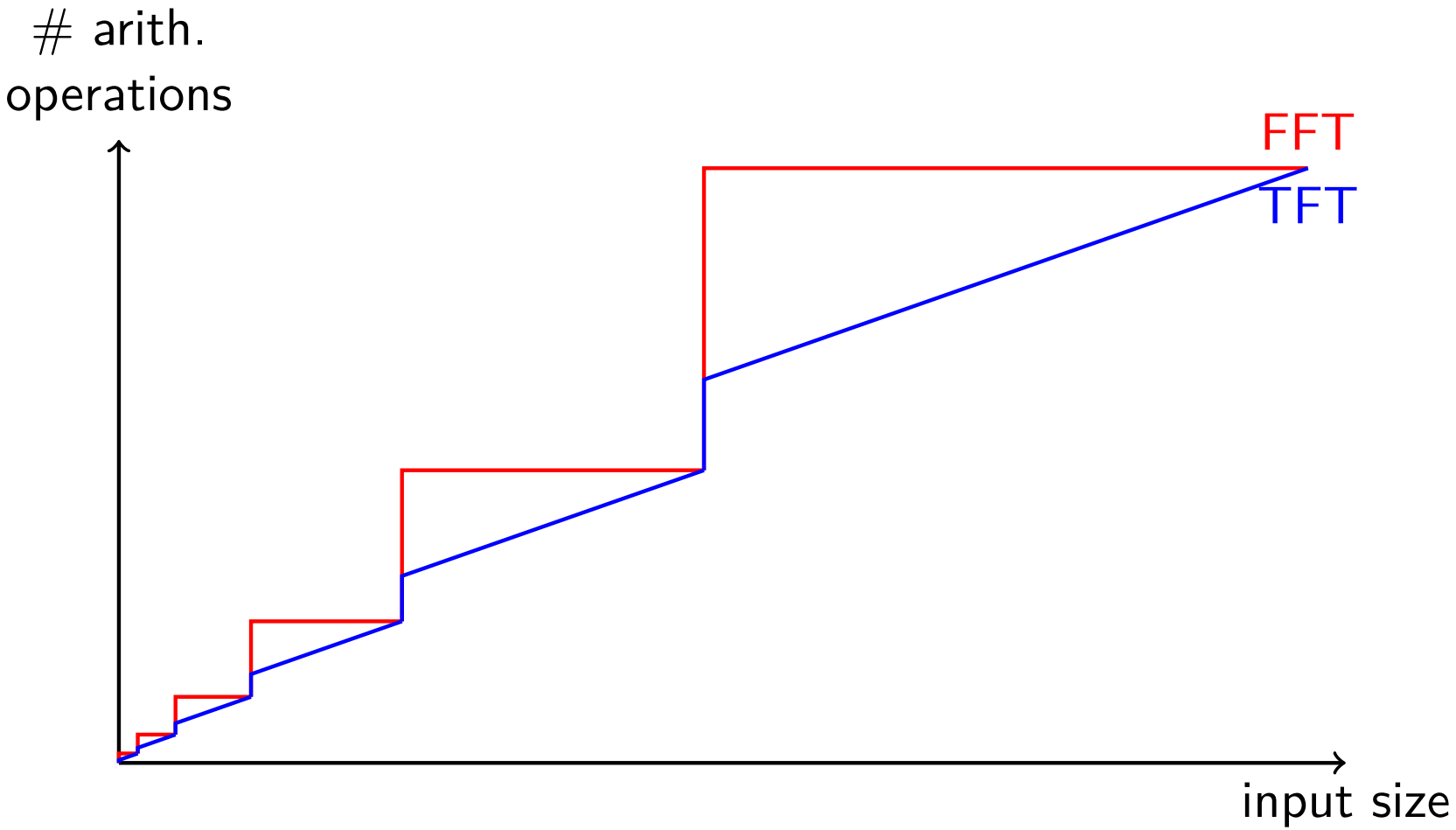
The Truncated Fourier Transform [van der Hoeven – 2004]



Jump phenomenon



Jump phenomenon



Restrictions for the TFT

Initial formulation [van der Hoeven – 2004]

- Size $n = 2^k$

Restrictions for the TFT

Initial formulation [van der Hoeven – 2004]

- Size $n = 2^k$

Alternative formulation [Mateer – 2008]

- Also compatible with additive FFT
- Can be extended for size $n = p^k$

Restrictions for the TFT

Initial formulation [van der Hoeven – 2004]

- Size $n = 2^k$

Alternative formulation [Mateer – 2008]

- Also compatible with additive FFT
- Can be extended for size $n = p^k$

This work

- Focus on multiplicative FFT
- Any (composite) n

Why generalize the TFT?

Choosing the size of the FFT

- Depends on the base field ($\omega \in \mathbb{K}$)
- Efficiency: n highly composite

Why generalize the TFT?

Choosing the size of the FFT

- Depends on the base field ($\omega \in \mathbb{K}$)
- Efficiency: n highly composite

- Jump phenomenon not specific to the size $n = 2^k$.
- Choosing the size $n = 2^k$ not always possible.

Why generalize the TFT?

Choosing the size of the FFT

- Depends on the base field ($\omega \in \mathbb{K}$)
- Efficiency: n highly composite

- Jump phenomenon not specific to the size $n = 2^k$.
- Choosing the size $n = 2^k$ not always possible.

Example: finite fields

- The field $\mathbb{F}_q$ has primitive roots of order $n | q - 1$

Why generalize the TFT?

Choosing the size of the FFT

- Depends on the base field ($\omega \in \mathbb{K}$)
- Efficiency: n highly composite

- Jump phenomenon not specific to the size $n = 2^k$.
- Choosing the size $n = 2^k$ not always possible.

Example: finite fields

- The field $\mathbb{F}_q$ has primitive roots of order $n | q - 1$
- No primitive square root in characteristic 2.

Practical use case: $\mathbb{F}_{2^{60}}$

[Harvey, van der Hoeven, Lecerf – 2016]

- Fast implementation of the FFT in $\mathbb{F}_{2^{60}}$
- Fast multiplication in $\mathbb{F}_2[X]$ and $\mathbb{F}_{2^k}[X]$

Practical use case: $\mathbb{F}_{2^{60}}$

[Harvey, van der Hoeven, Lecerf – 2016]

- Fast implementation of the FFT in $\mathbb{F}_{2^{60}}$
- Fast multiplication in $\mathbb{F}_2[X]$ and $\mathbb{F}_{2^k}[X]$

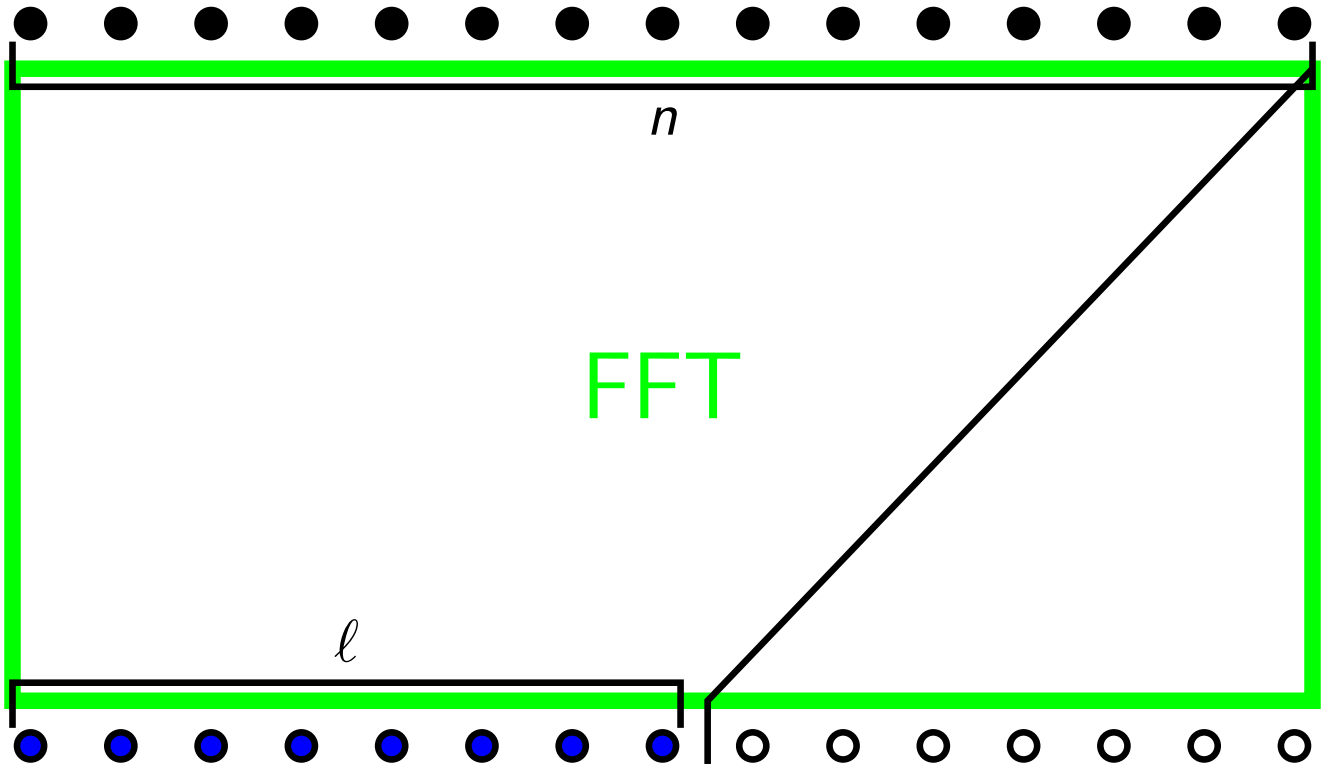
Why $\mathbb{F}_{2^{60}}$?

- Representation on a machine word
- $(X^{61}-1)/(X-1)$ irreducible over $\mathbb{F}_2$
- Highly composite order:

$$2^{60}-1 = 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 31 \cdot 41 \cdot 61 \cdot 151 \cdot 331 \cdot 1321$$

Outline

Objective



Reminder: FFT for any n

By definition, the DFT is given by

$$\hat{A}_k = A(\omega^k) = \sum_{i=0}^{n-1} A_i \omega^{ik}$$

Reminder: FFT for any n

By definition, the DFT is given by

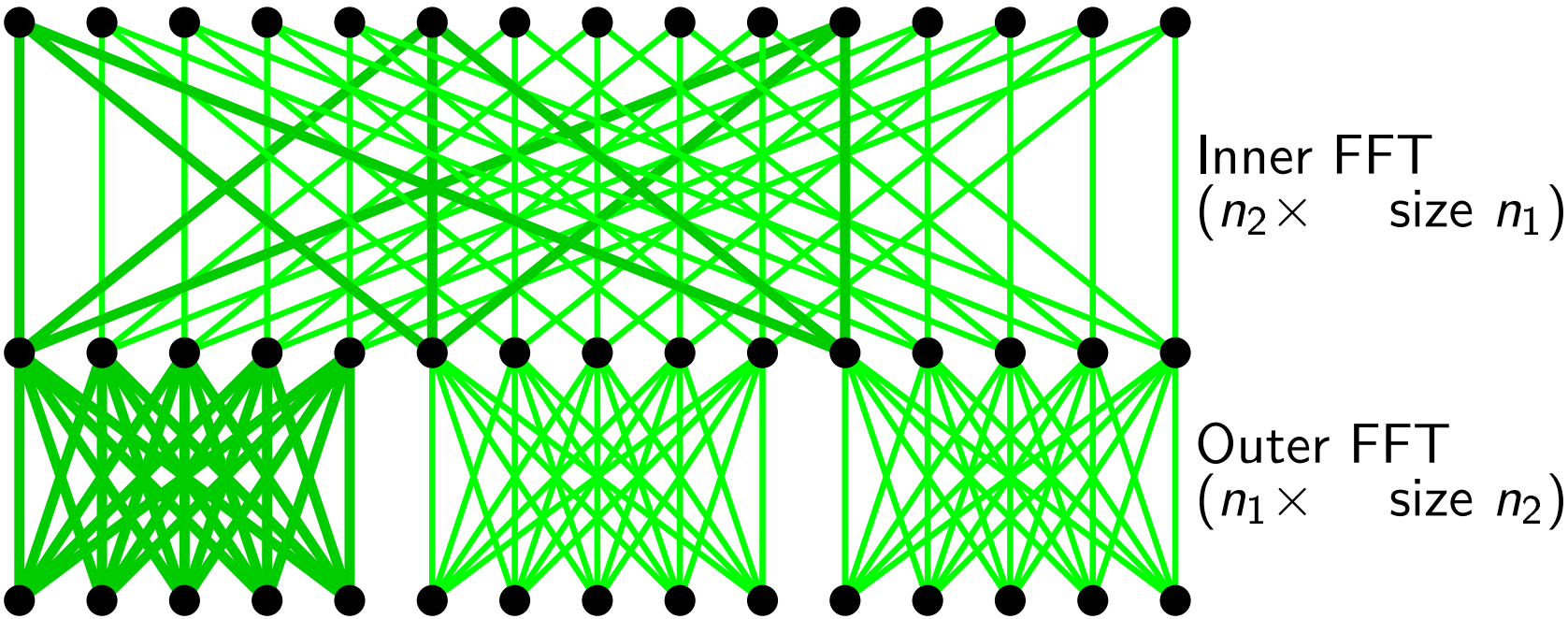
$$\hat{A}_k = A(\omega^k) = \sum_{i=0}^{n-1} A_i \omega^{ik}$$

The Cooley-Tukey FFT [Cooley, Tukey – 1965]

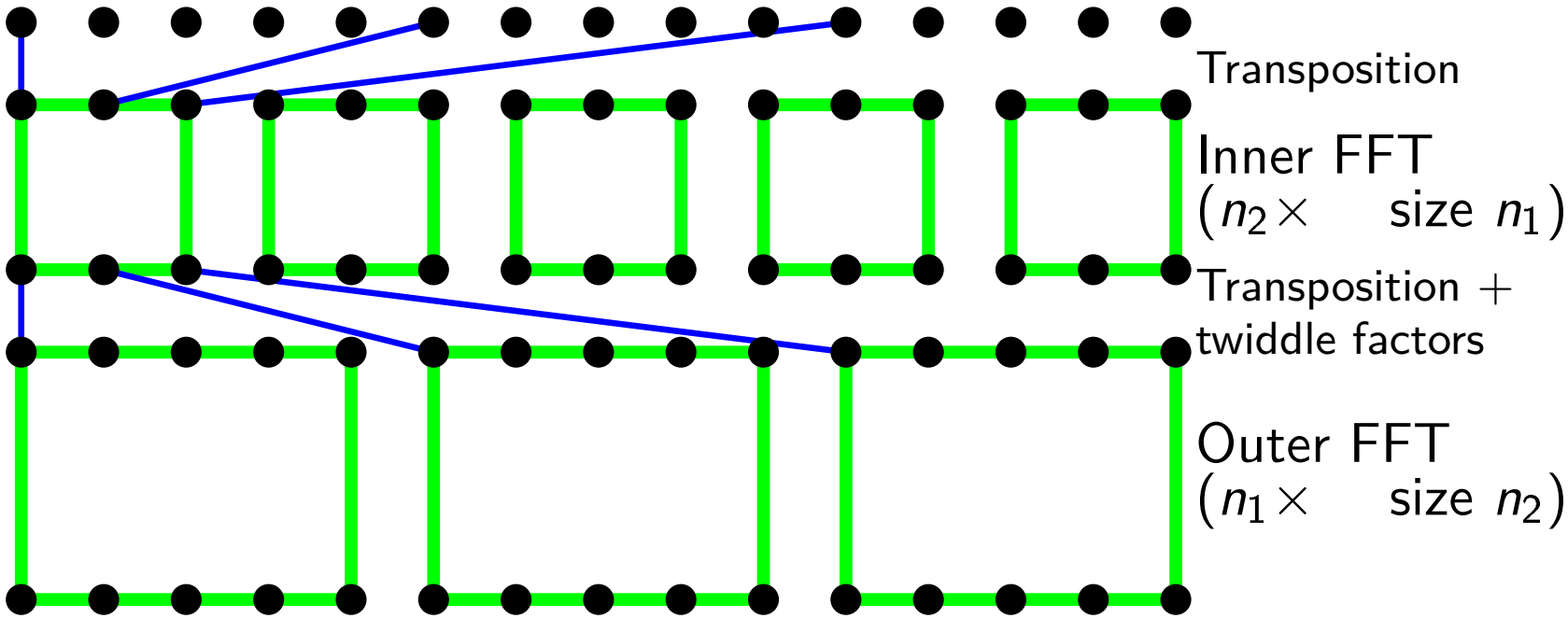
If $n = n_1 n_2$, we have:

$$\hat{A}_{k_1+n_1 k_2} = \sum_{i=0}^{n_2-1} \omega^{ik_1} \cdot \left(\sum_{j=0}^{n_1-1} A_{i+n_2 j} \cdot (\omega^{n_2})^{jk_1} \right) \cdot (\omega^{n_1})^{ik_2}$$

Reminder: FFT for any n



Reminder: FFT for any n



Atomic transforms

TFT of prime order

- Direct computation (Horner's method)
  - Quadratic $\Rightarrow$ adapted only for small size
  - Can compute exactly the values we need
- Full FFT (Rader reduction)
  - Quasi-linear $\Rightarrow$ better for large sizes
  - Must compute all values and discard the unused ones

Atomic transforms

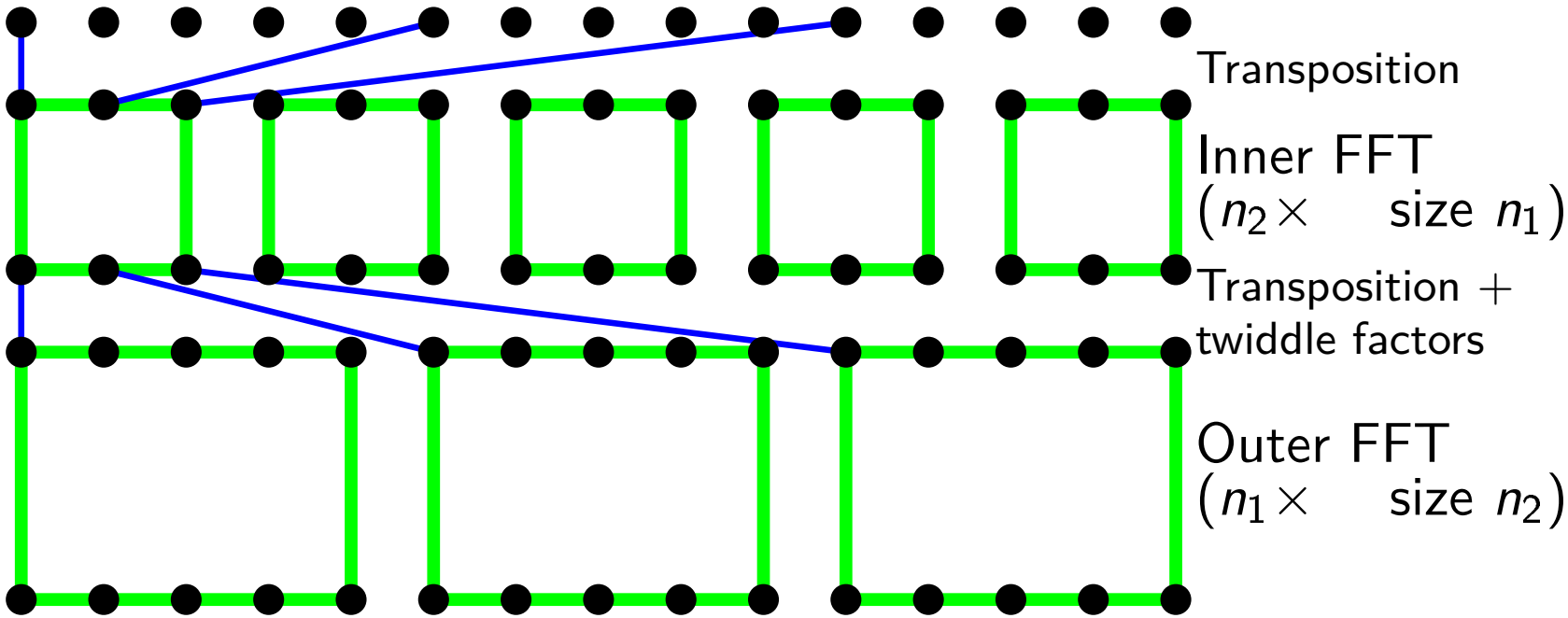
TFT of prime order

- Direct computation (Horner's method)
  - Quadratic $\Rightarrow$ adapted only for small size
  - Can compute exactly the values we need
- Full FFT (Rader reduction)
  - Quasi-linear $\Rightarrow$ better for large sizes
  - Must compute all values and discard the unused ones

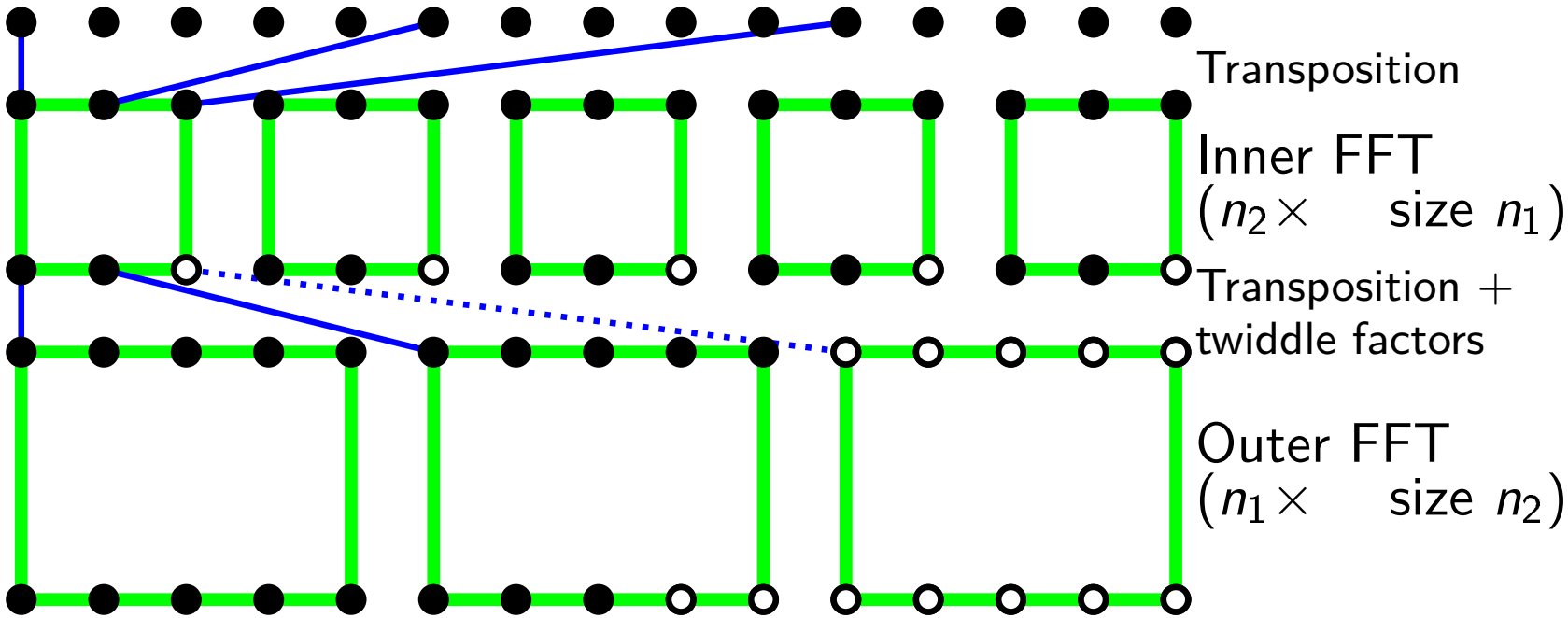
Small composite size

Direct computation

Recursive algorithm

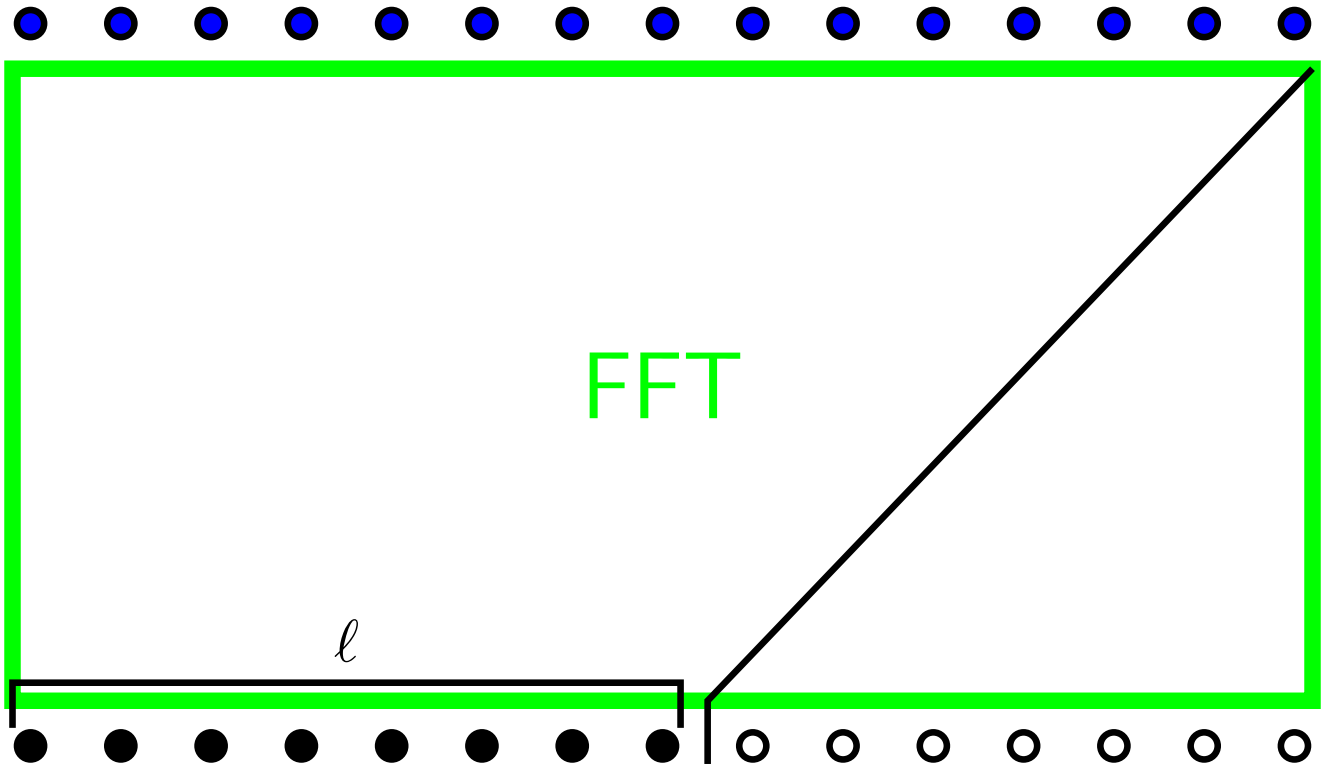


Recursive algorithm

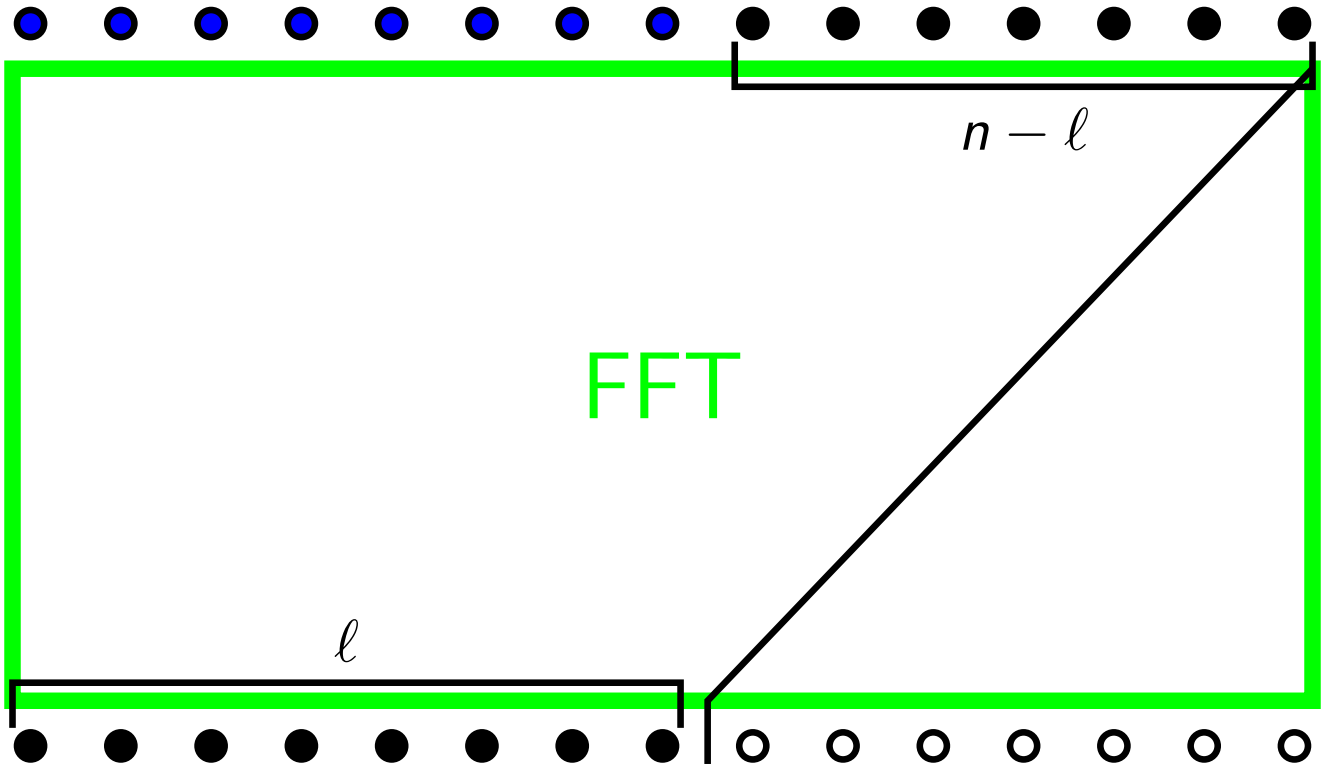


Outline

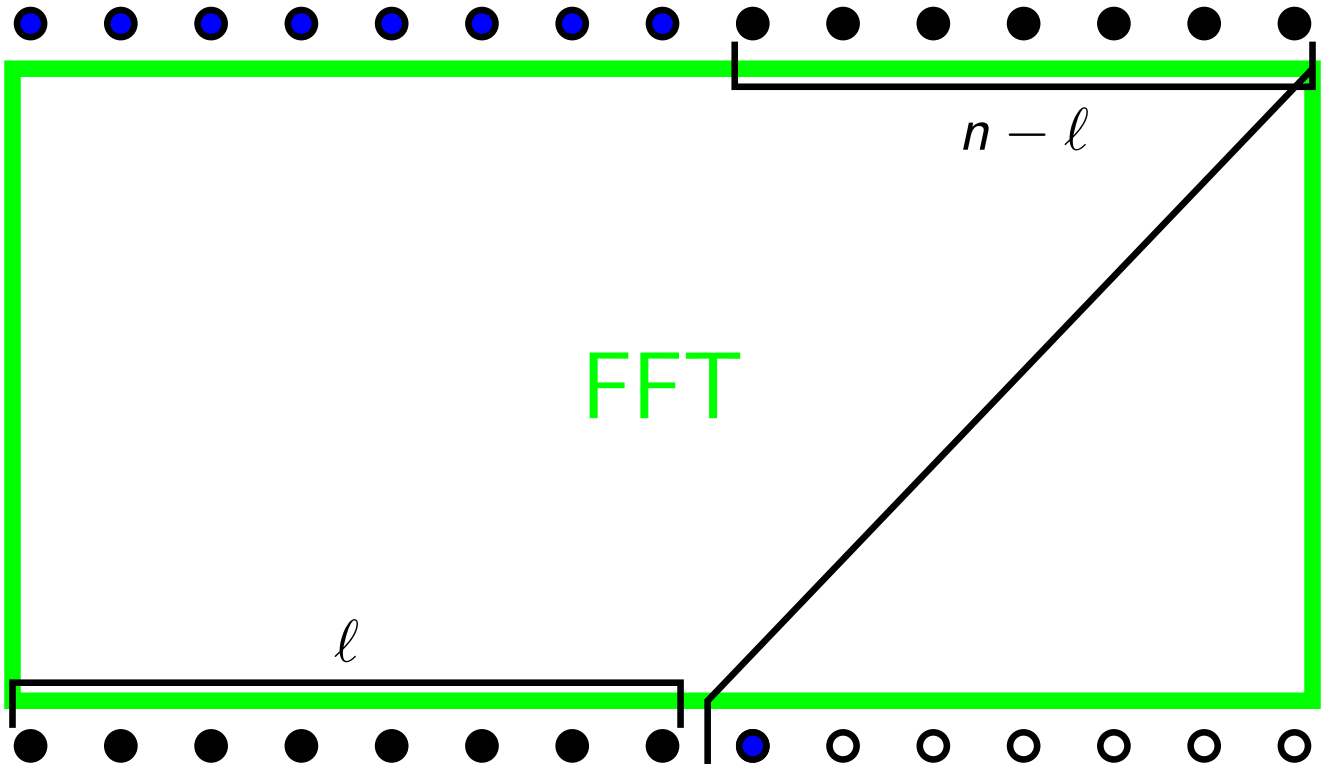
Objective



Objective



Objective



Atomic transformations

Problem: solve the matrix equation (e.g. $n = 5$, $\ell = 3$)

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3^{(?)} \\ b_4^{(?)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 \\ 1 & \omega^2 & \omega^4 & \omega & \omega^3 \\ 1 & \omega^3 & \omega & \omega^4 & \omega^2 \\ 1 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix} \begin{pmatrix} a_0^{(?)} \\ a_1^{(?)} \\ a_2^{(?)} \\ a_3 \\ a_4 \end{pmatrix}$$

Atomic transformations

Problem: solve the matrix equation (by blocks)

$$\begin{pmatrix} B_1 \\ B_2^{(?)} \end{pmatrix} = \begin{pmatrix} V & W \\ W^\top & \tilde{V} \end{pmatrix} \cdot \begin{pmatrix} A_1^{(?)} \\ A_2 \end{pmatrix}$$

Atomic transformations

Problem: solve the matrix equation (by blocks)

$$\begin{pmatrix} B_1 \\ B_2^{(?)} \end{pmatrix} = \begin{pmatrix} V & W \\ W^\top & \tilde{V} \end{pmatrix} \cdot \begin{pmatrix} A_1^{(?)} \\ A_2 \end{pmatrix}$$

Here V is a $\ell \times \ell$ Vandermonde matrix.

$$A_1 = V^{-1}(B_1 - WA_2)$$

Atomic transformations

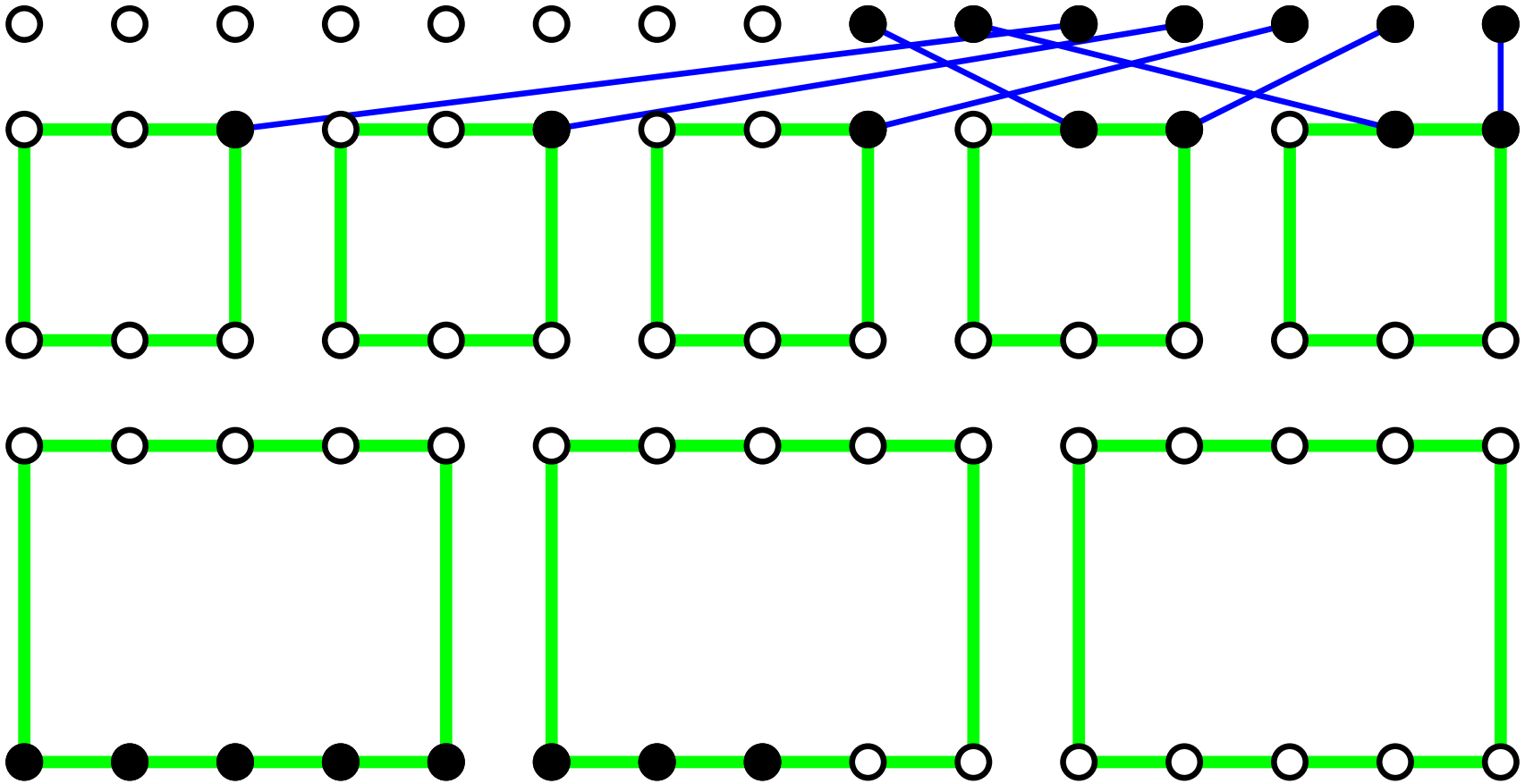
Problem: solve the matrix equation (by blocks)

$$\begin{pmatrix} B_1 \\ B_2^{(?)} \end{pmatrix} = \begin{pmatrix} V & W \\ W^\top & \tilde{V} \end{pmatrix} \cdot \begin{pmatrix} A_1^{(?)} \\ A_2 \end{pmatrix}$$

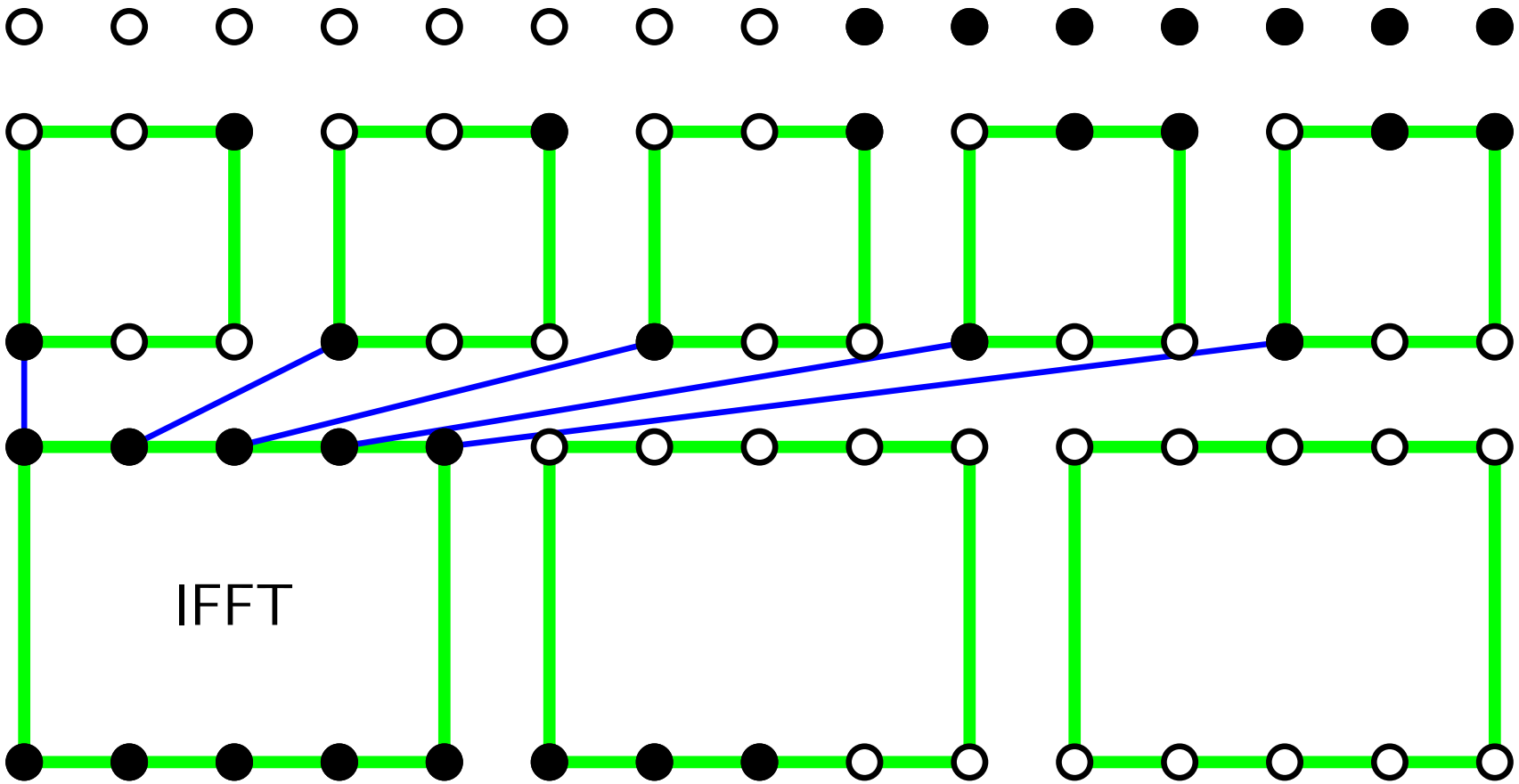
Here V is a $\ell \times \ell$ Vandermonde matrix.

$$A_1 = V^{-1}(B_1 - WA_2)$$
$$B_2 = W^\top A_1 + \tilde{V}A_2$$

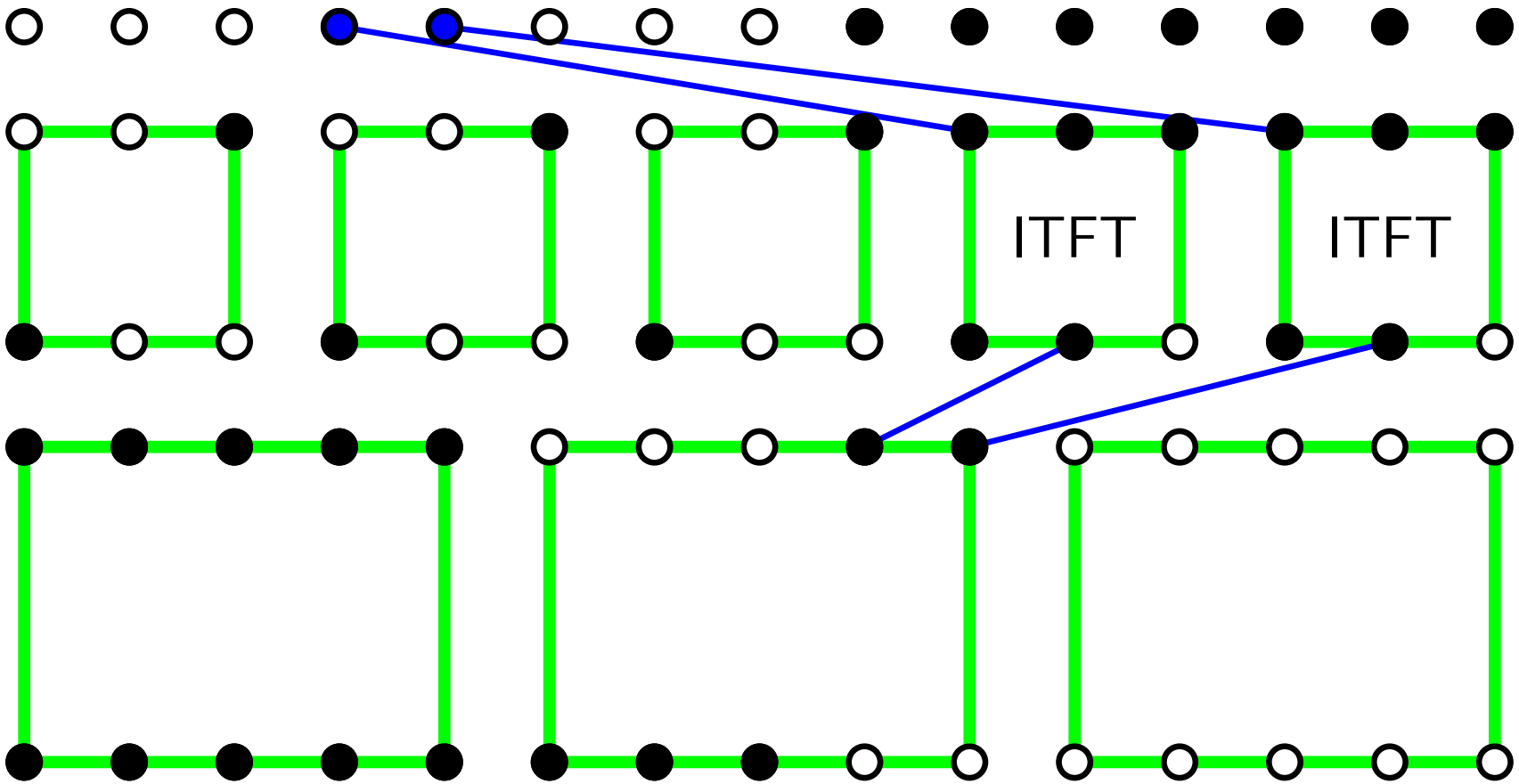
Recursive algorithm



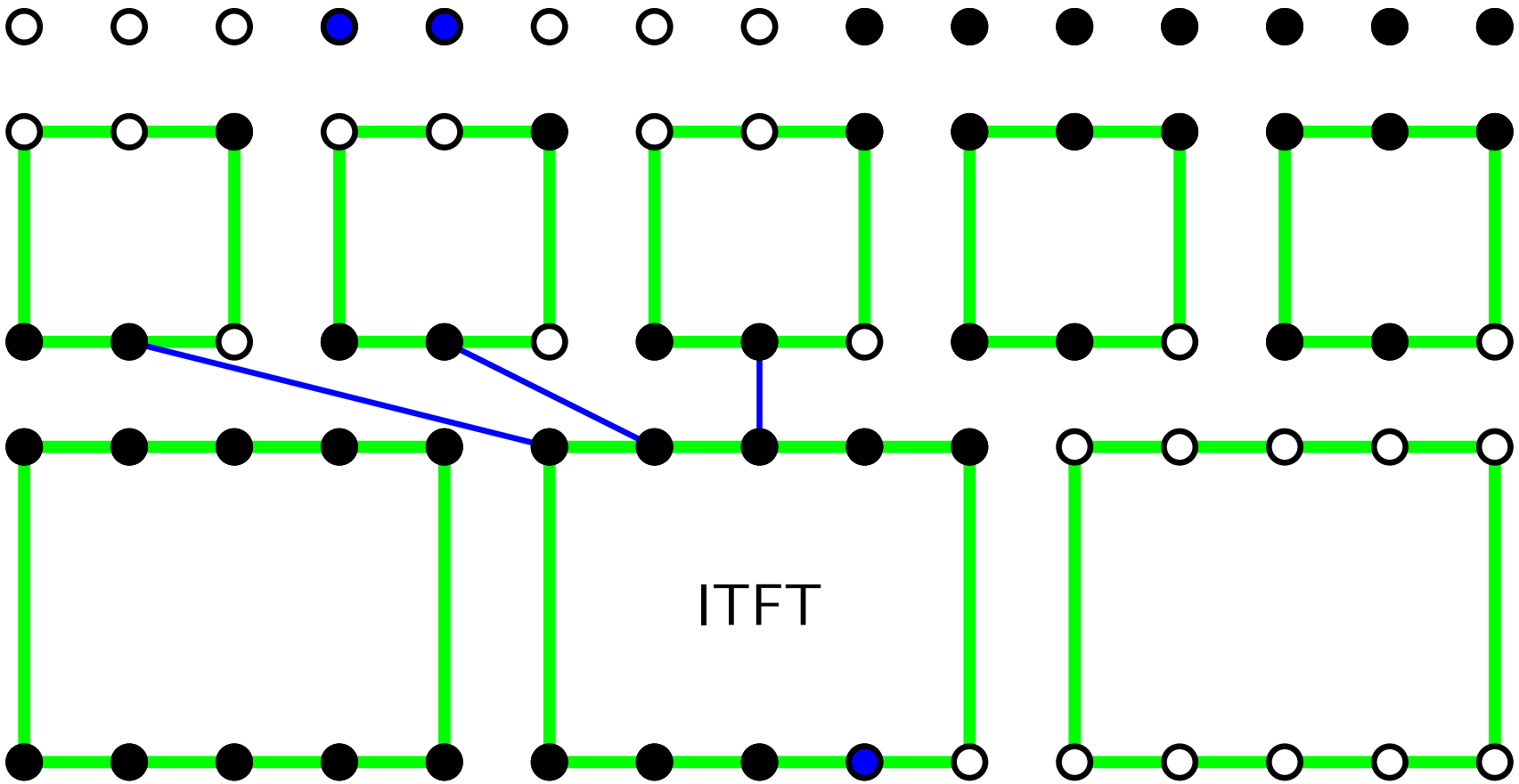
Recursive algorithm



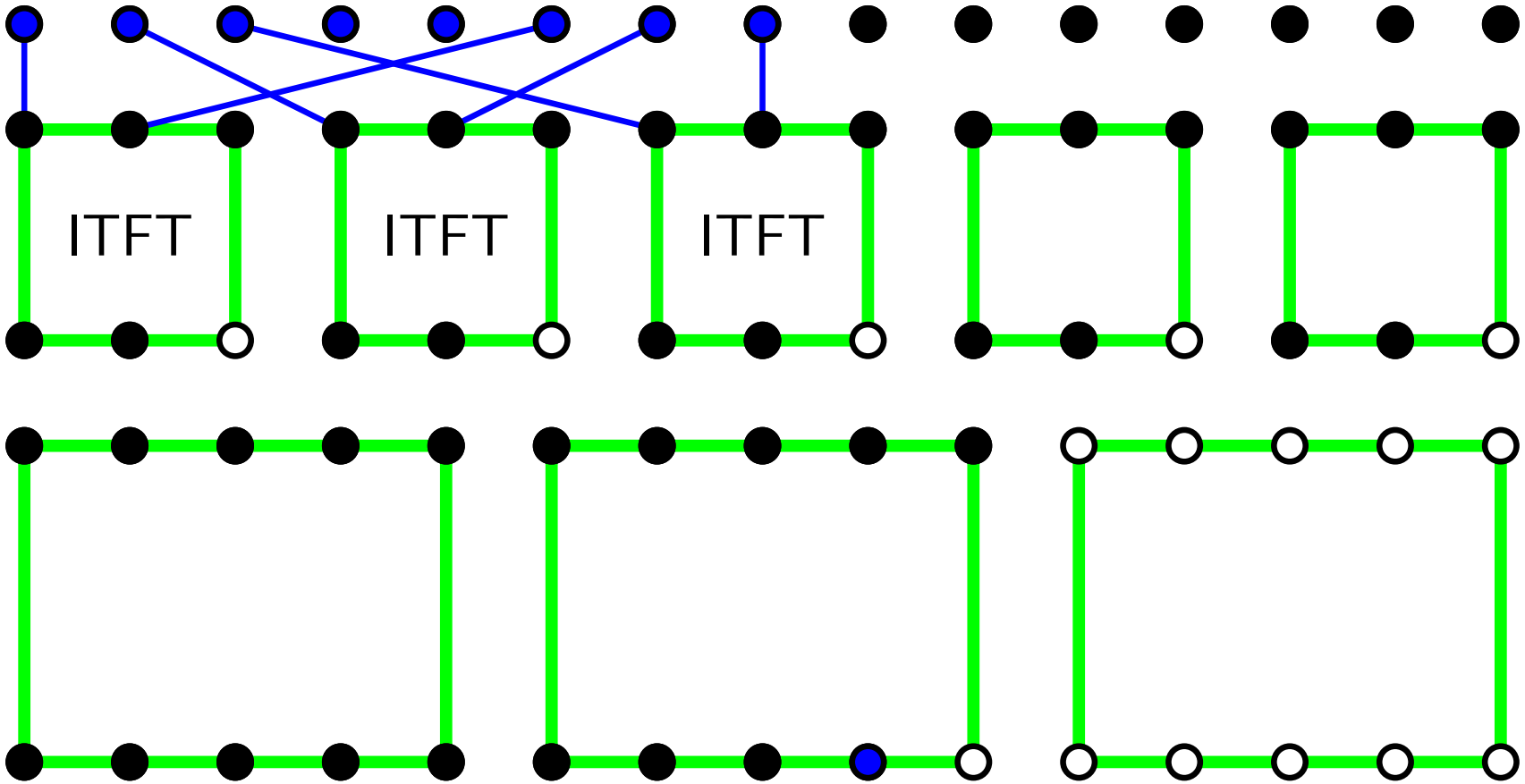
Recursive algorithm



Recursive algorithm



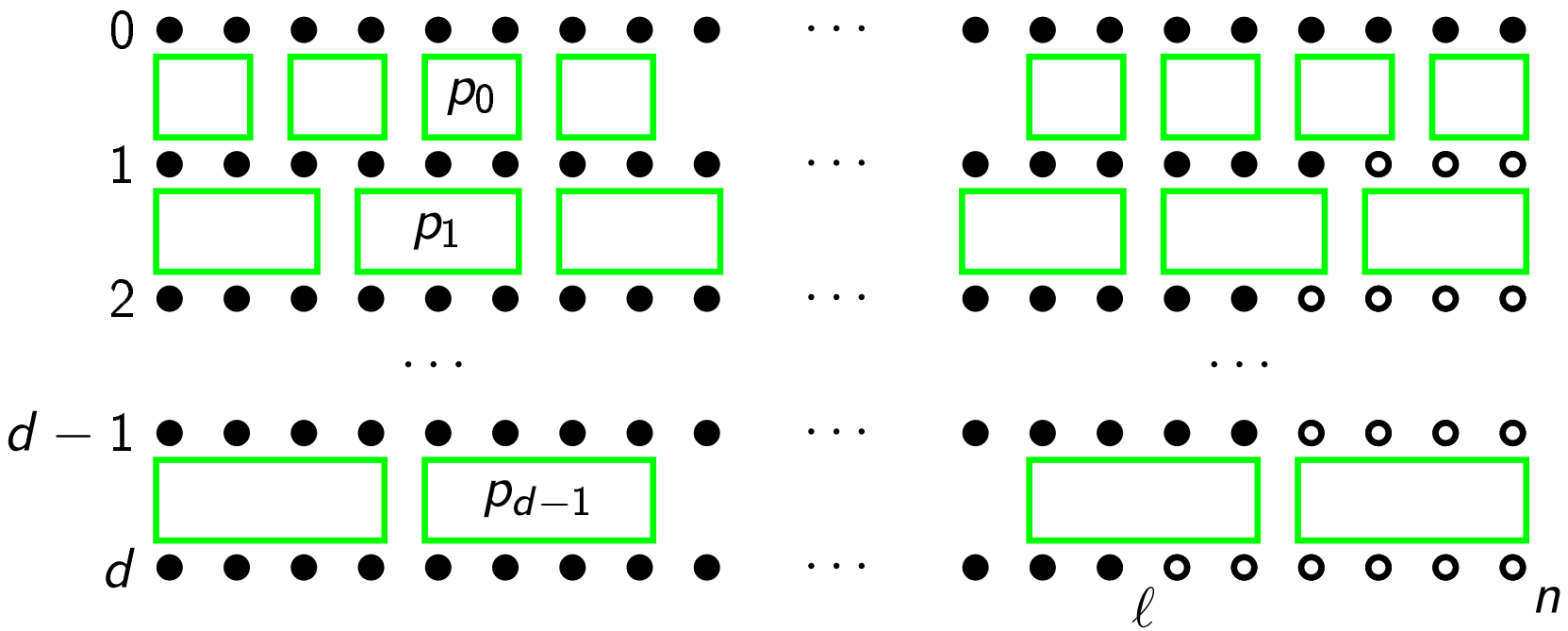
Recursive algorithm



Outline

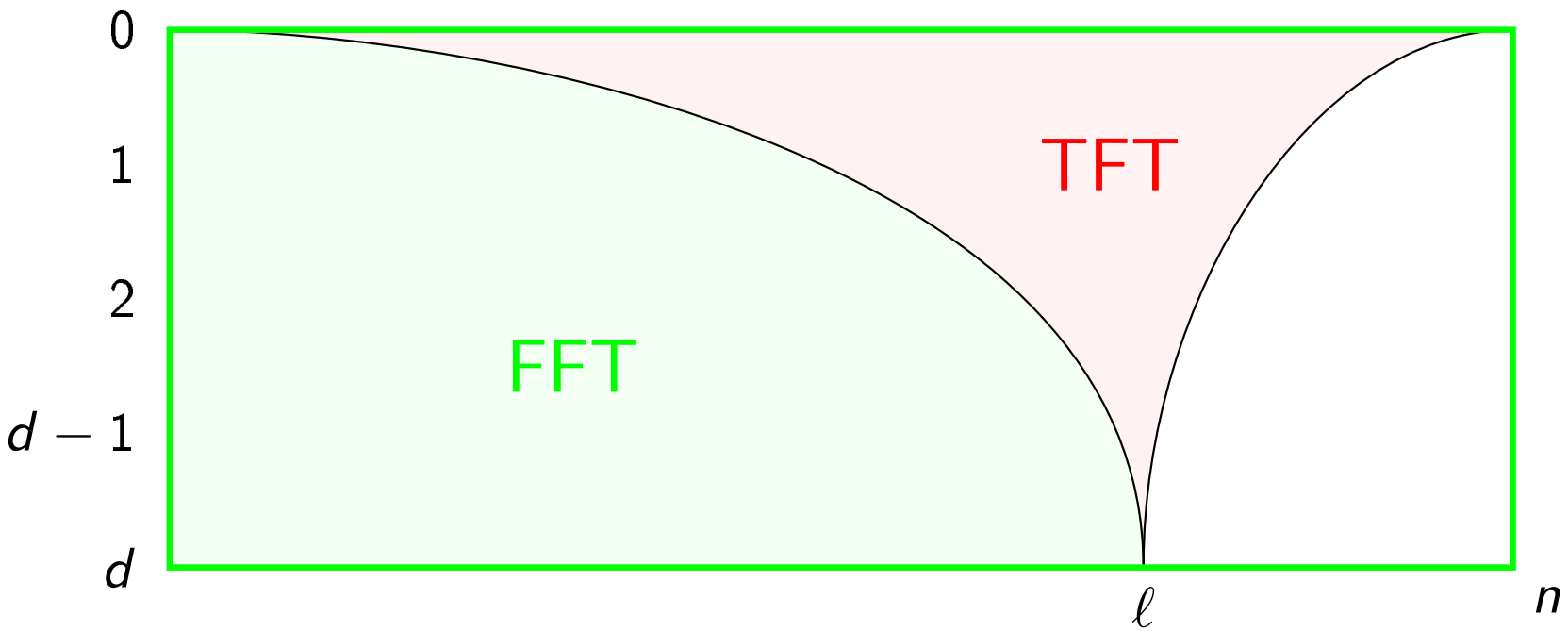
Complexity

$$n = p_0 p_1 \cdots p_{d-1}$$



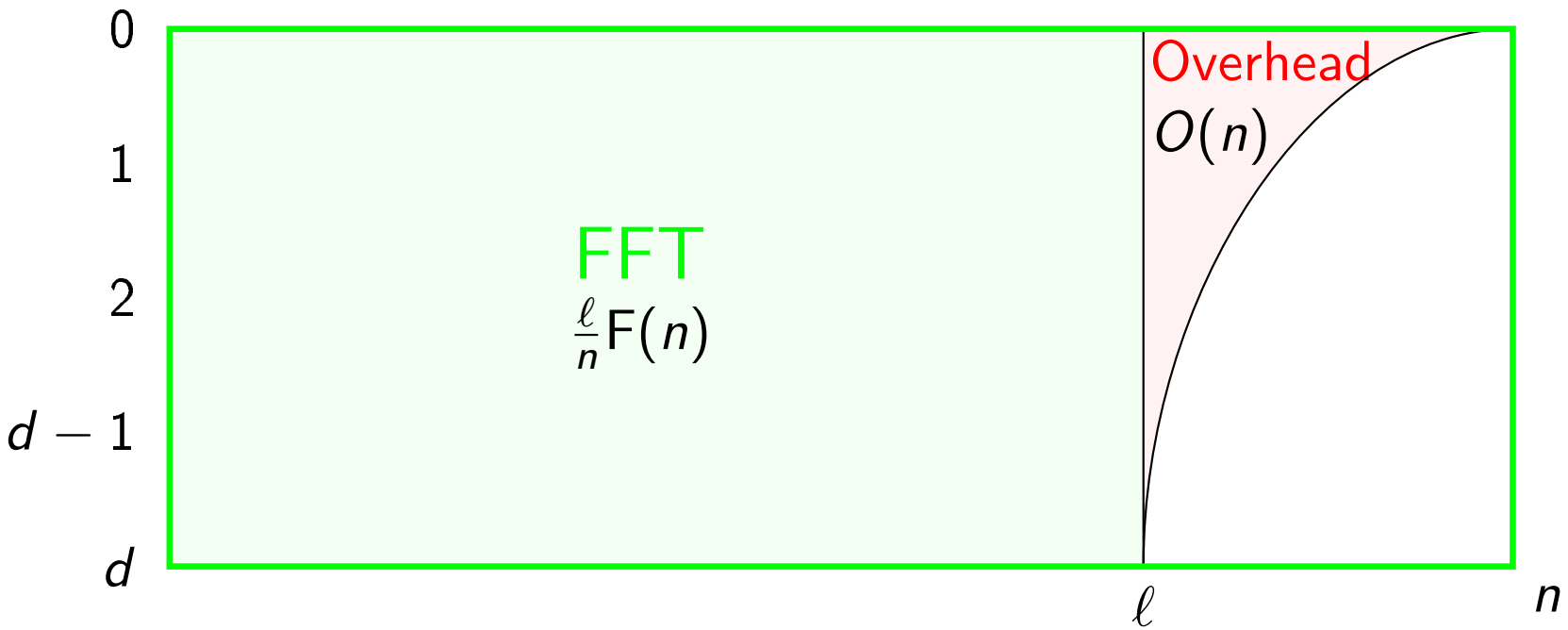
Complexity

$$n = p_0 p_1 \cdots p_{d-1}$$



Complexity

$$n = p_0 p_1 \cdots p_{d-1}$$



Questions?

Thank you for your attention