Lecture 4: FFT & Graphs

Announcements:

- Quizzes will now be for 10 mins!!
- Average for Quiz 2 was 0.9
- Assignment 1 is due Wednesday by 6:59 PM on Gradescope.
- Link to a Survey on Piazza

Recap of last lecture:

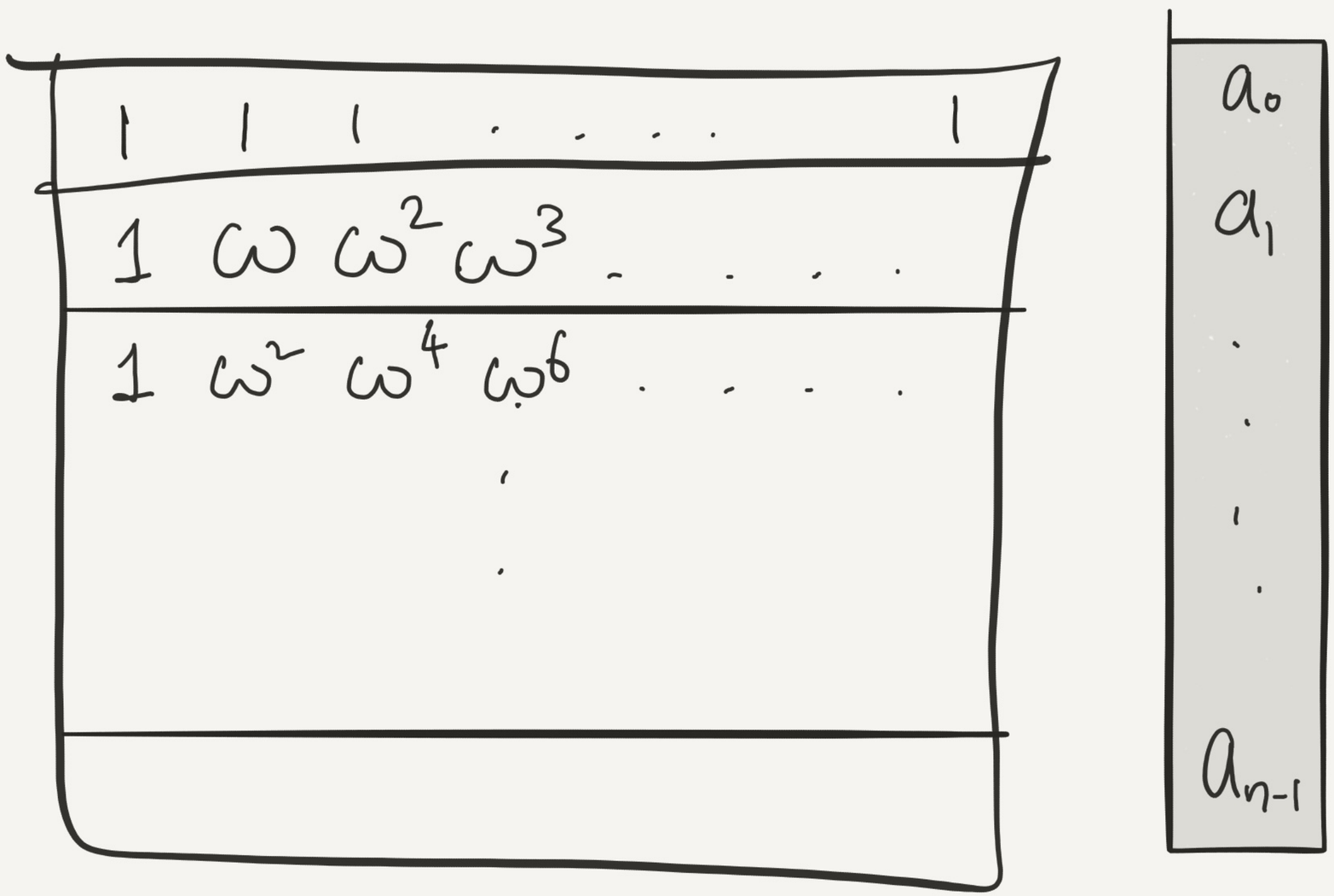
1. Defined the DFT
2. Naive algorithm to compute DFT
   → $O(n^2)$ time.

$$\bar{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} \xrightarrow{DFT_n} \bar{s} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix}$$

$$s_j = \sum_{k=0}^{n-1} \omega^{j \cdot k} \cdot a_k$$

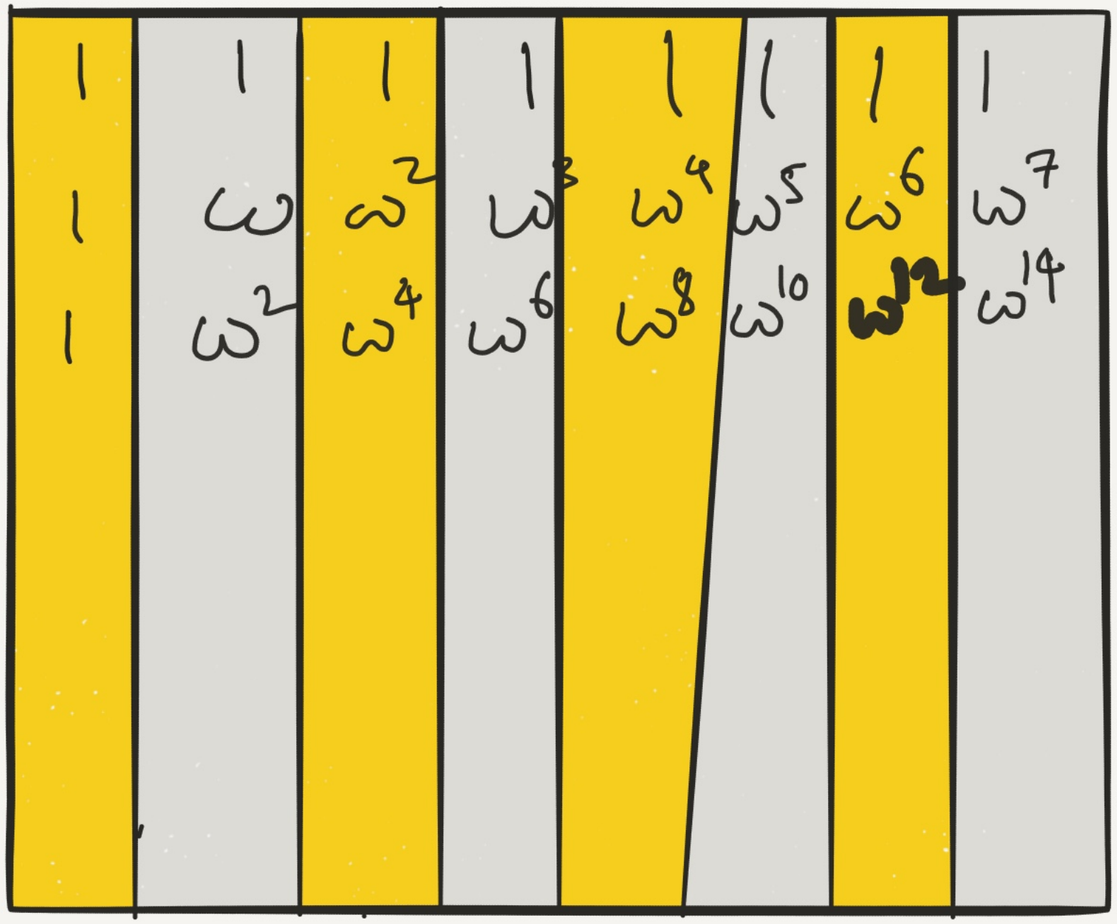
$$\omega = e^{\frac{2\pi i}{n}}.$$

Generally:



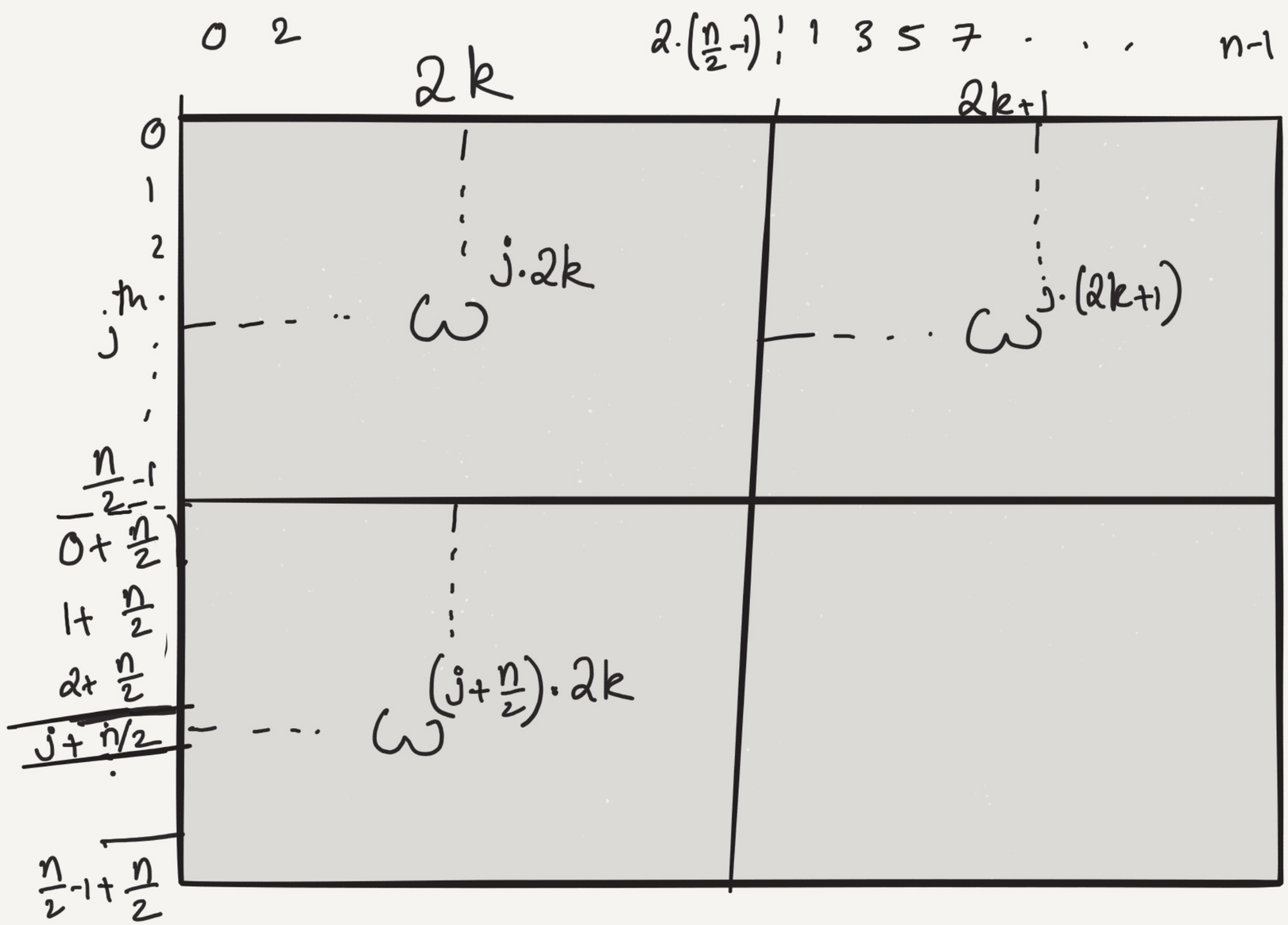
Let's Suppose n is an even number.

eg: $n = 8$



this for $n = 8$.

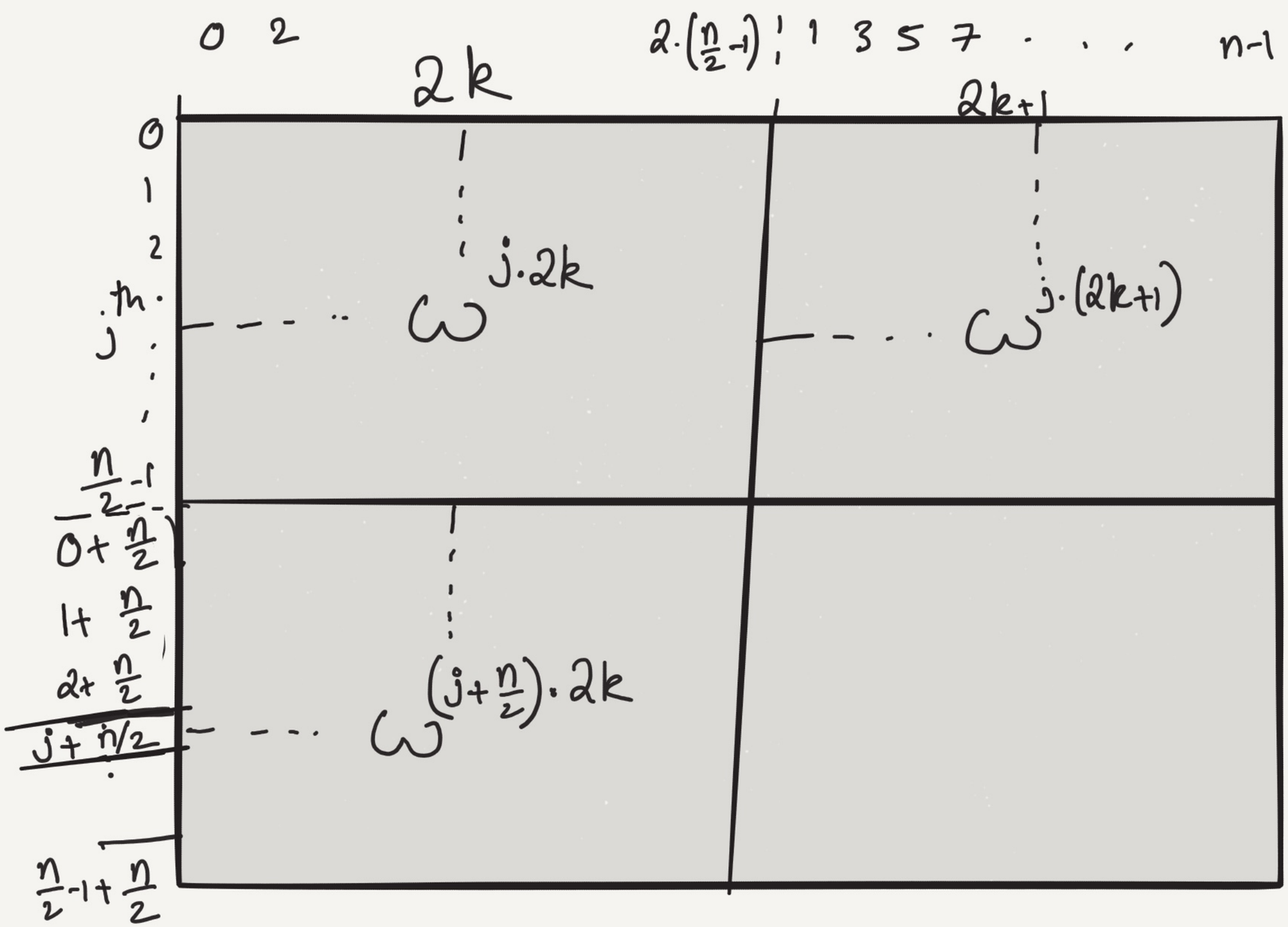
More generally we group all even columns first.



← DFT_n with even numbered columns first.

$$\omega^{(j+\frac{n}{2})\cdot 2k} = \omega^{j\cdot 2k + k\cdot n} = \omega^{(j\cdot 2k)} \cdot \omega^{k\cdot n}$$

$$= \omega^{(j\cdot 2k)} \left(\text{as } \omega^n = 1\right).$$

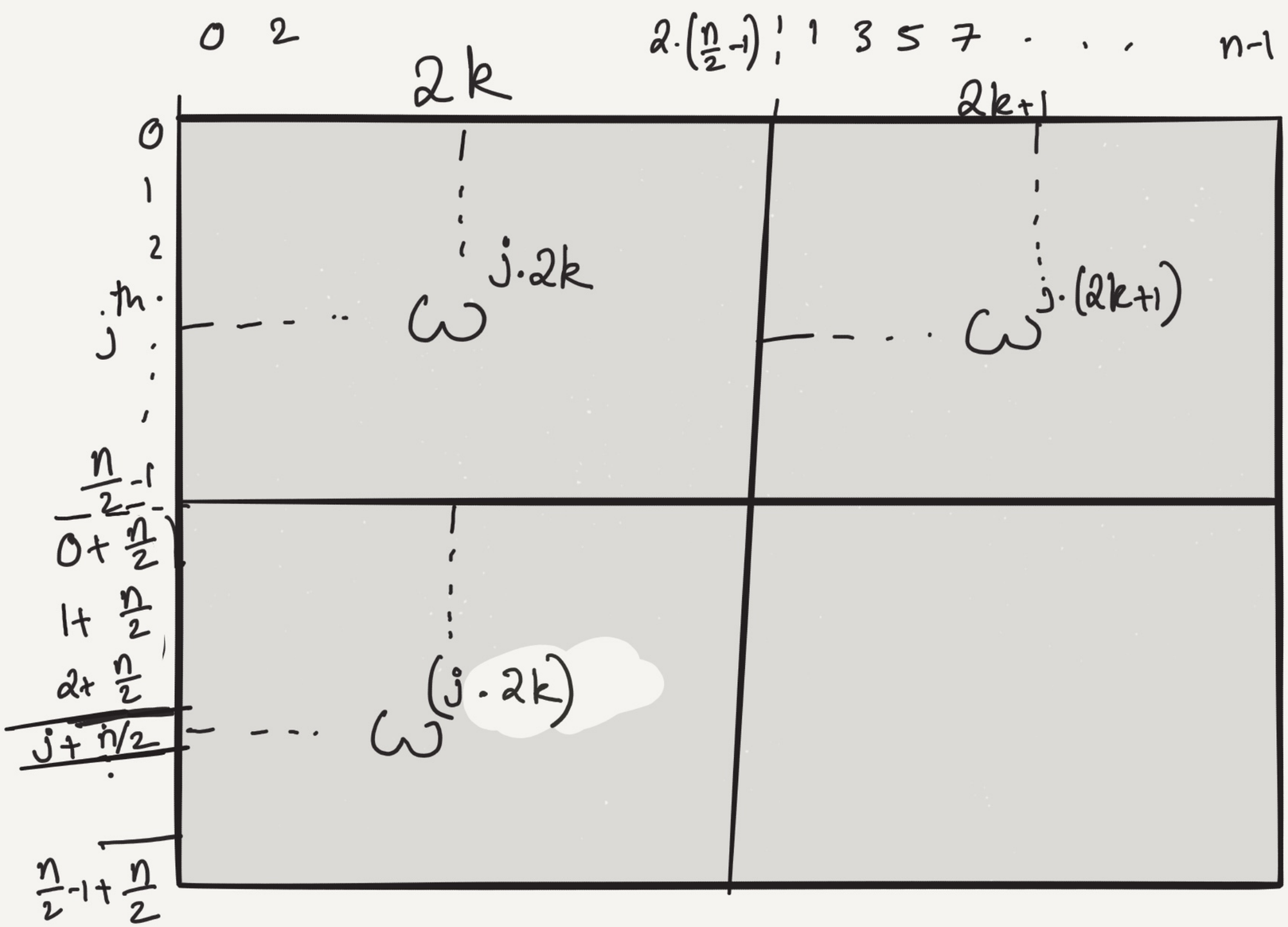


← DFT_n with even numbered columns first.

← We can simplify bottom matrix using

$$\omega^{(j+\frac{n}{2})\cdot 2k} = \omega^{j\cdot 2k + k\cdot n} = \omega^{(j\cdot 2k)} \cdot \omega^{k\cdot n}$$

$$= \omega^{(j\cdot 2k)} \quad (\text{as } \omega^n = 1).$$

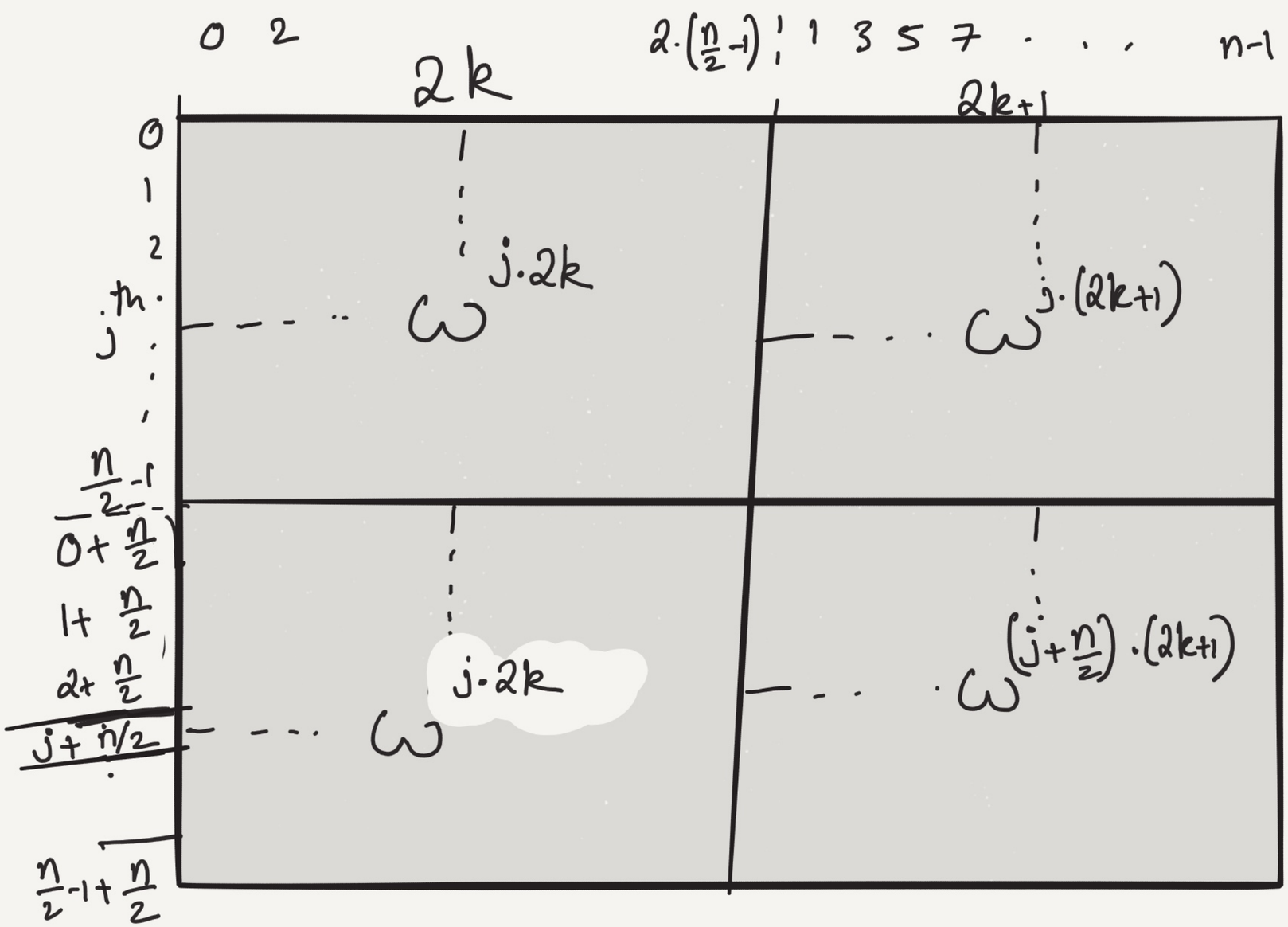


← DFT_n with even numbered columns first.

← We can simplify bottom matrix using

$$\omega^{(j+\frac{n}{2})\cdot 2k} = \omega^{j\cdot 2k + k\cdot n} = \omega^{(j\cdot 2k)} \cdot \omega^{k\cdot n}$$

$$= \omega^{(j\cdot 2k)} \quad (\text{as } \omega^n = 1).$$



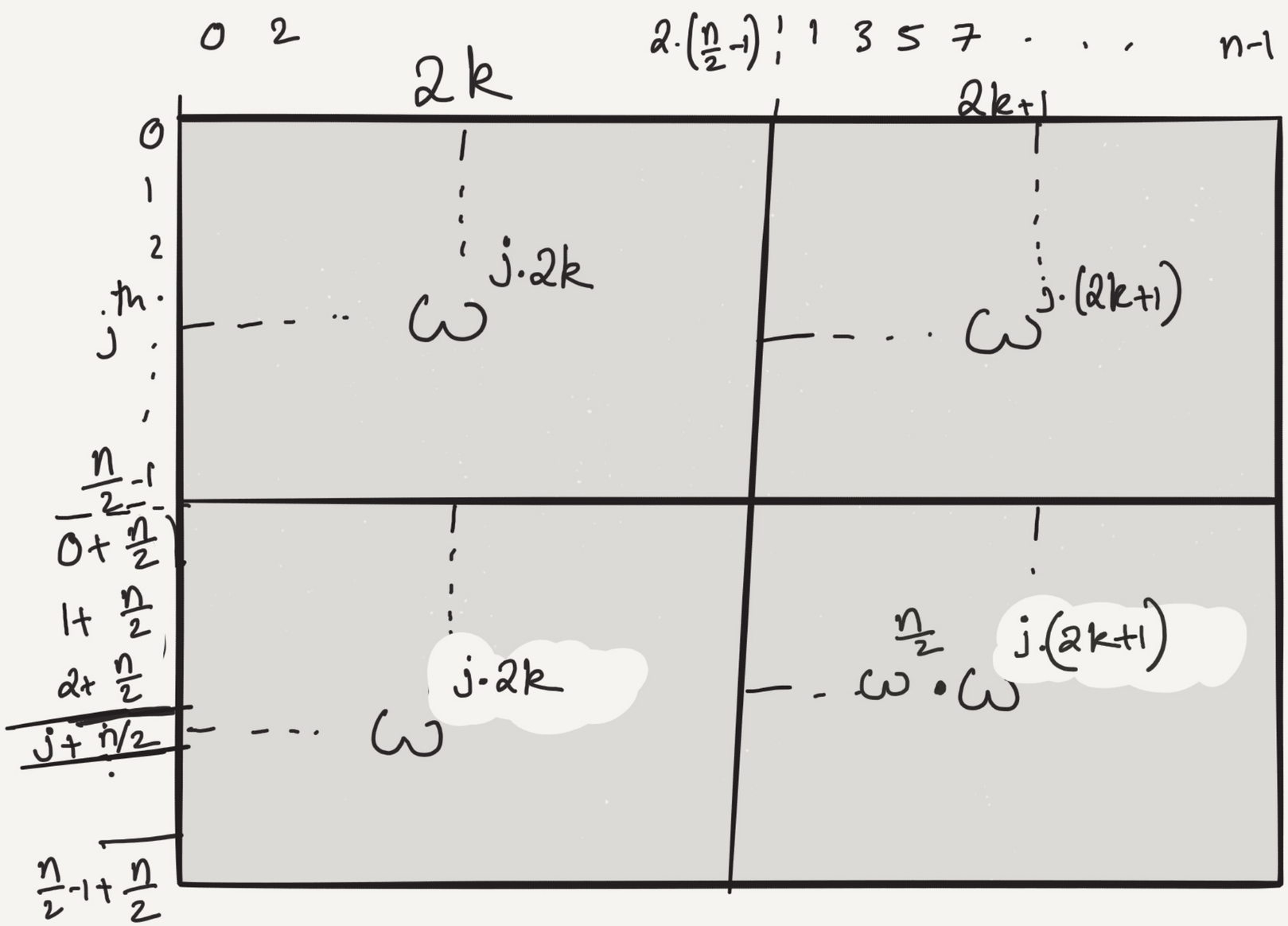
← DFT_n with even columns first

← We can also simplify bottom right matrix using

$$\omega^{(j+\frac{n}{2})(2k+1)} = \omega^{j\cdot(2k+1)+\frac{n}{2}(2k+1)} = \omega^{j\cdot(2k+1)} \cdot \omega^{\frac{n}{2}(2k+1)}$$

$$= \omega^{j\cdot(2k+1)} \cdot \omega^{n\cdot k} \cdot \omega^{\frac{n}{2}}$$

$$= \omega^{j\cdot(2k+1)} \cdot \omega^{n/2} \cdot$$



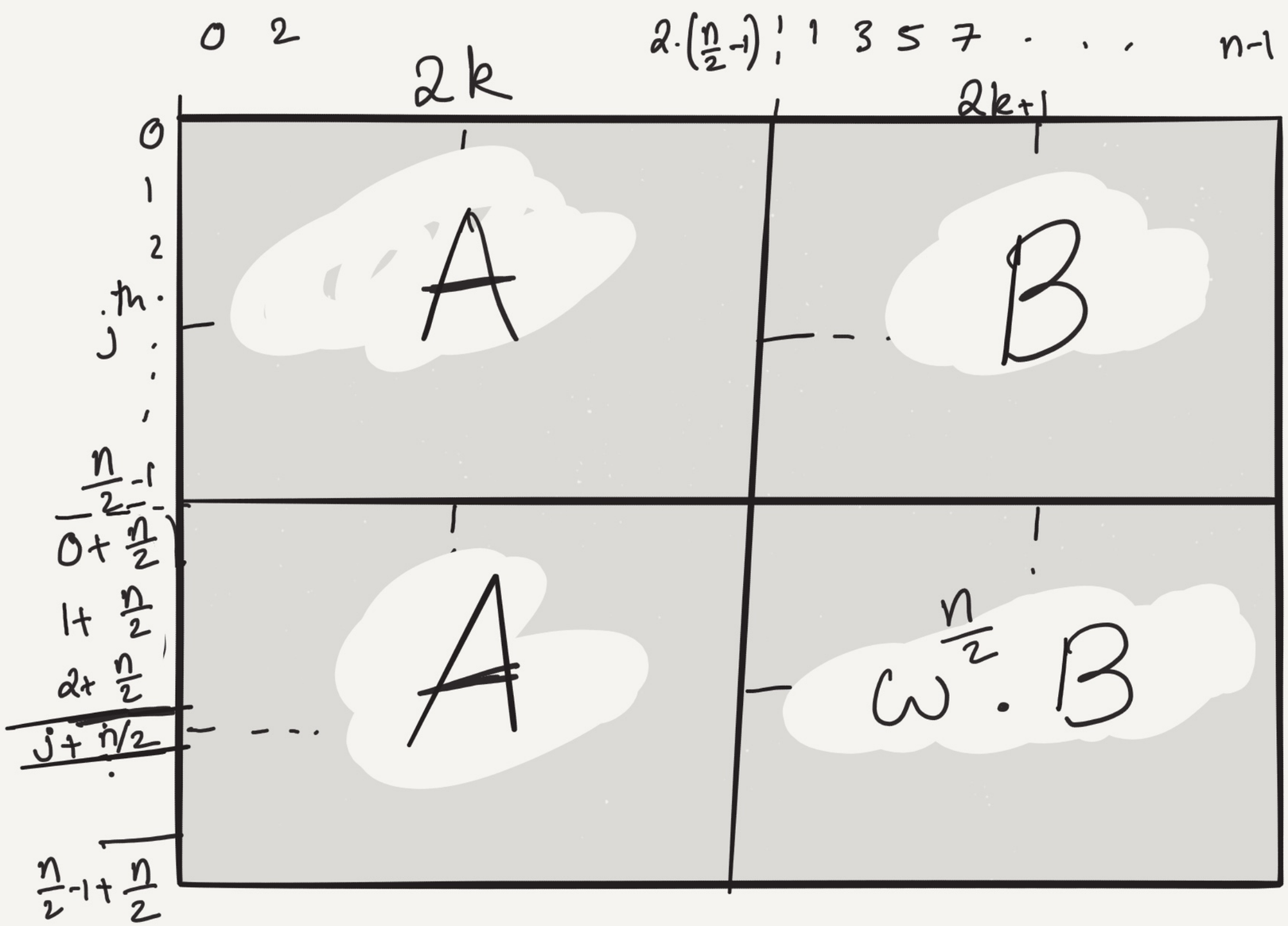
← DFT_n with even columns first

← We can also simplify bottom right matrix using

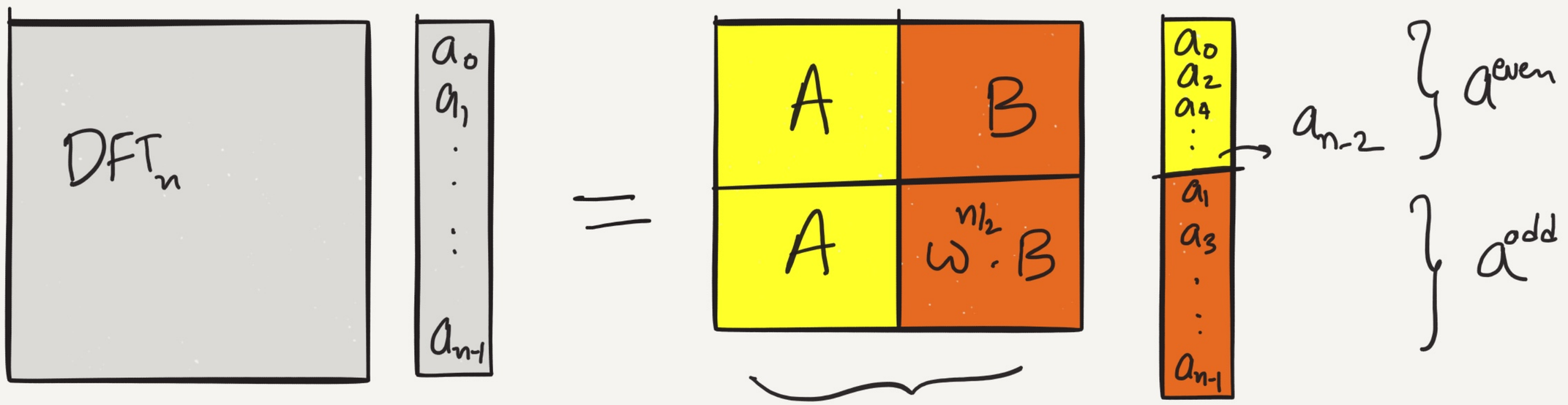
$$\omega^{(j+\frac{n}{2})(2k+1)} = \omega^{j\cdot(2k+1)+\frac{n}{2}(2k+1)} = \omega^{j\cdot(2k+1)} \cdot \omega^{\frac{n}{2}(2k+1)}$$

$$= \omega^{j\cdot(2k+1)} \cdot \omega^{n\cdot k} \cdot \omega^{\frac{n}{2}}$$

$$= \omega^{j\cdot(2k+1)} \cdot \omega^{n/2} \cdot$$



Thus, the matrix DFT_n with even numbered columns has the above nice recursive structure just like the example we saw for DFT_4.



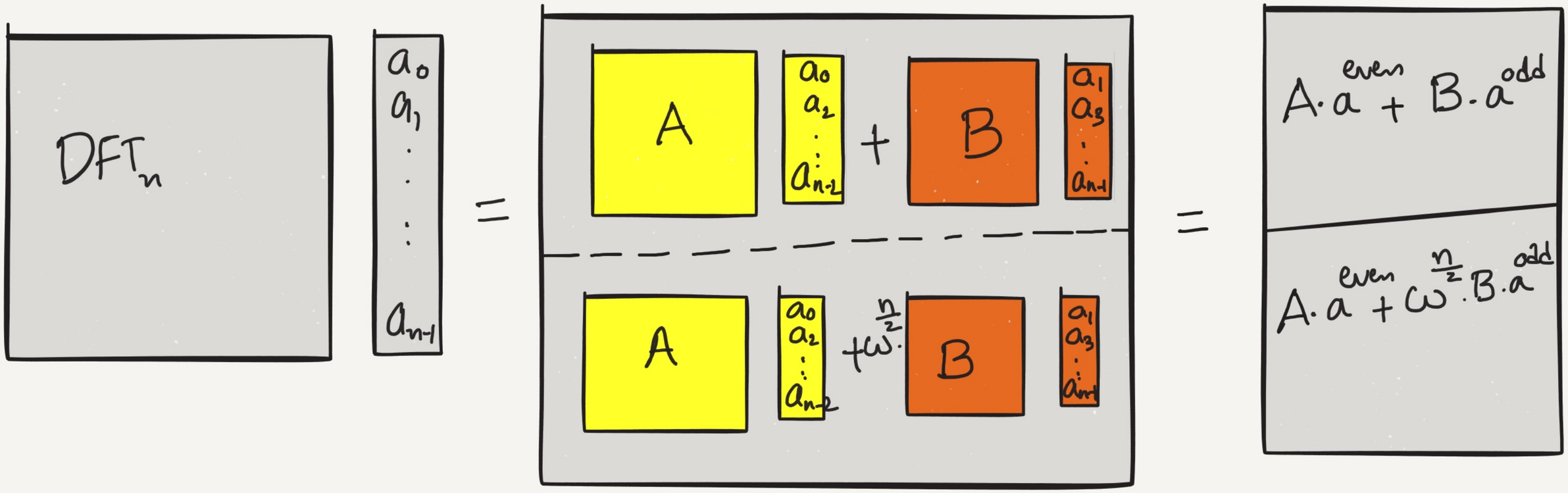
Now, to account for moving the even columns to the front, we can move the even numbered elements of a to the top to get the above nice decomposition.

We can further refine it as follows ...

DFT_n $\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$ $=$ $\begin{pmatrix} A & B \\ A & \omega^{n/2} \cdot B \end{pmatrix}$ $\begin{pmatrix} a_0 \\ a_2 \\ a_4 \\ \vdots \\ a_{n-2} \\ a_1 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$ $\left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} a^{even}$ $\left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} a^{odd}$

Moved even Columns together

$=$ $\begin{pmatrix} A & B \\ A & \omega^{n/2} \cdot B \end{pmatrix}$ $\begin{pmatrix} a_0 \\ a_2 \\ \vdots \\ a_{n-2} \\ a_1 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$ $=$ $\begin{matrix} A \cdot a^{even} + B \cdot a^{odd} \\ \\ A \cdot a^{even} + \omega^{\frac{n}{2}} \cdot B \cdot a^{odd} \end{matrix}$



This suggests the following algorithm:

① Compute $A \cdot a^{even}$

② Compute $B \cdot a^{odd}$

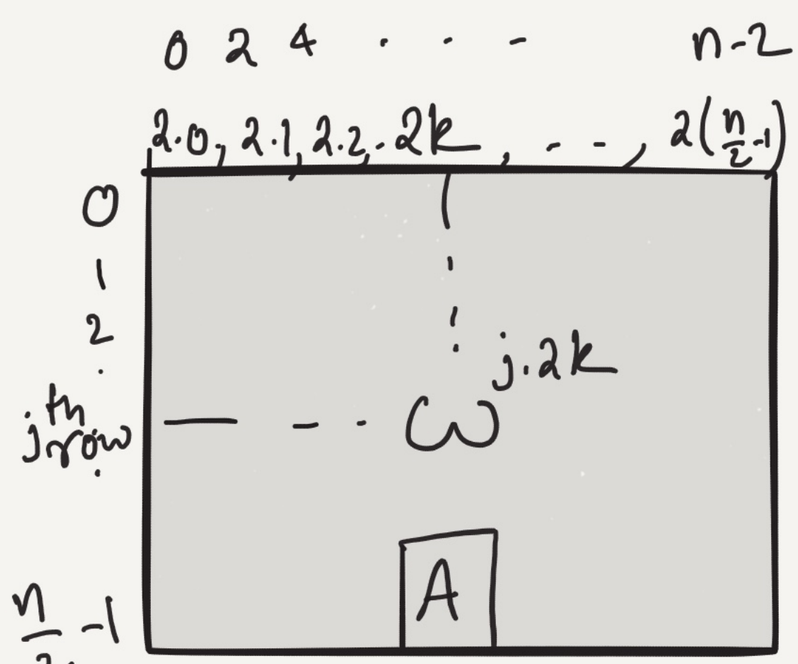
③ Compute $A \cdot a^{even} + B \cdot a^{odd}$

④ Compute $A \cdot a^{even} + \omega^{n/2} \cdot B \cdot a^{odd}$

} Can use ①, ② to do these two steps!

How to Compute $A \cdot a^{even}$??

What is the matrix A ??



$$\omega = e^{\frac{2\pi i}{n}}$$

$$\omega^{j \cdot 2k} = e^{\frac{2\pi i}{n}(j \cdot 2k)}$$

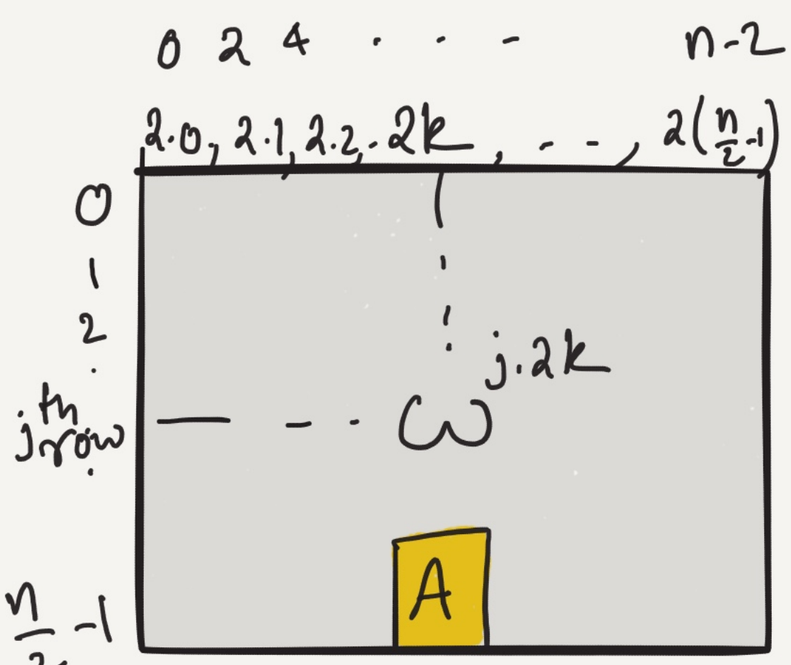
$$= e^{\frac{2\pi i \cdot (j \cdot k)}{(\frac{n}{2})}}$$

Let $\alpha = e^{\frac{2\pi i}{(\frac{n}{2})}}$

$$= \alpha^{j \cdot k}$$

How to Compute $A \cdot a^{even}$??

What is the matrix A ??



Recall: $\omega = e^{\frac{2\pi i}{n}}$

$\Rightarrow (j,k)^{th}$ entry of A $\left(\frac{2\pi i}{n/2}\right) \cdot j \cdot k$

$= \omega^{j \cdot 2k} = e^{\frac{2\pi i}{n} j \cdot 2k} = e$

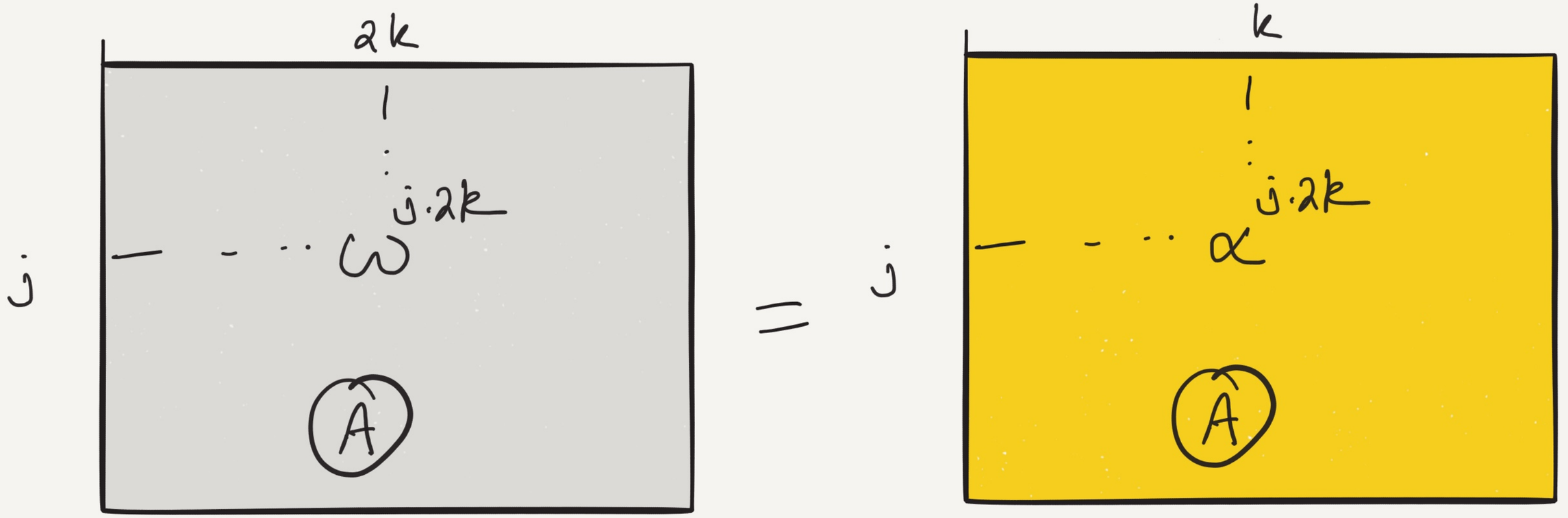
Let $\alpha = e^{2\pi i/(n/2)}$.

Then, $(j,k)^{th}$ entry of $A = \alpha^{j \cdot k}$.

Moreover, $\alpha = e^{2\pi i/(n/2)}$ is an $\left(\frac{n}{2}\right)^{th}$ root of unity!!

(As $\alpha^{n/2} = 1$).

So, we can write:



where $\alpha = e^{\frac{2\pi i}{(n/2)}}$

But this is the definition of $DFT_{n/2}$.

Therefore, $A = DFT_{n/2}$!!

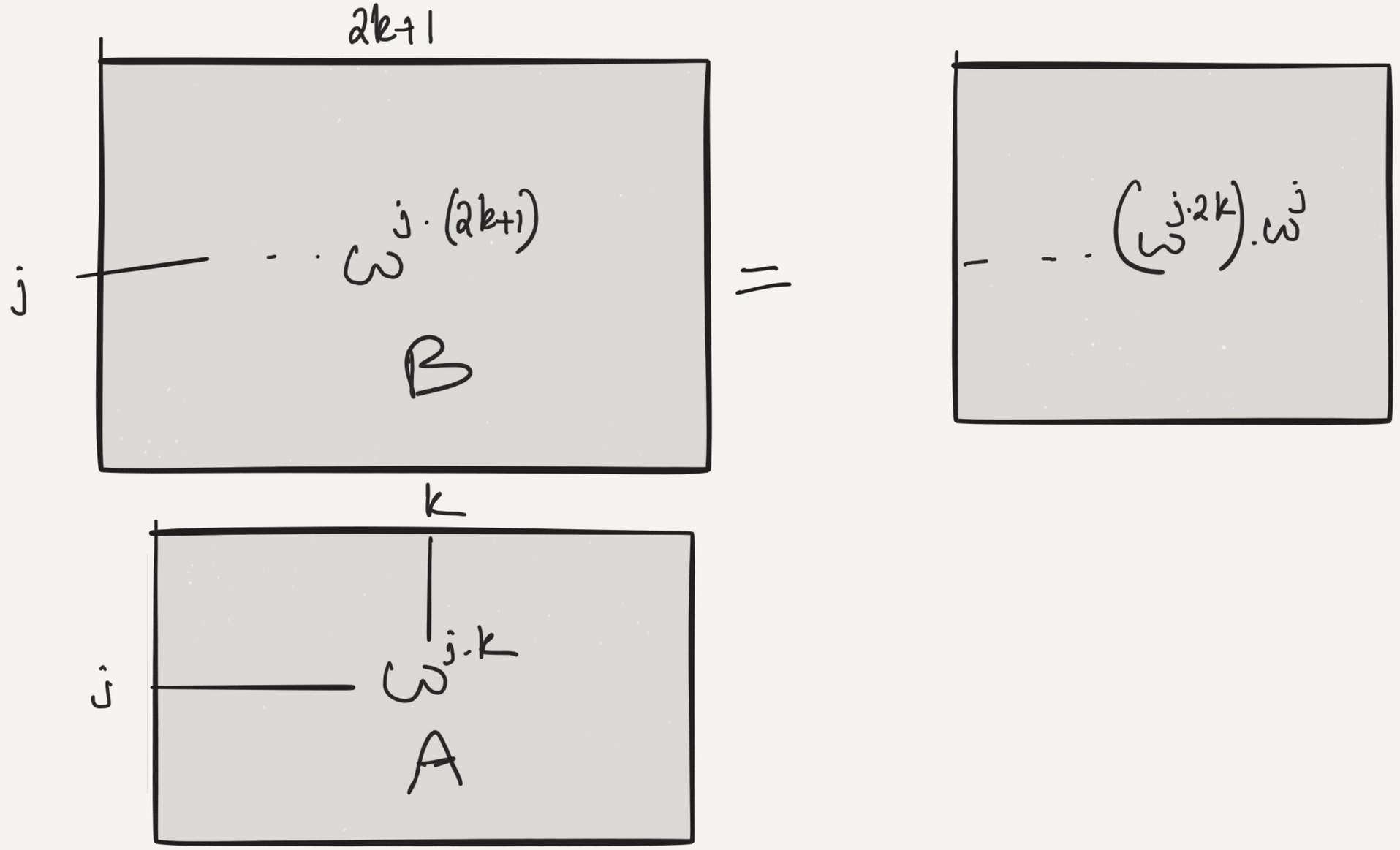
So, we can compute $A \cdot a^{even}$ by a recursive call:

$$A \cdot a^{even} = \left(DFT_{\frac{n}{2}}\right) \cdot a^{even}$$

$$= DFT_{\frac{n}{2}}(a_0, a_2, a_4, \ldots a_{n-2}).$$

How to Compute $B \cdot a^{odd}$??

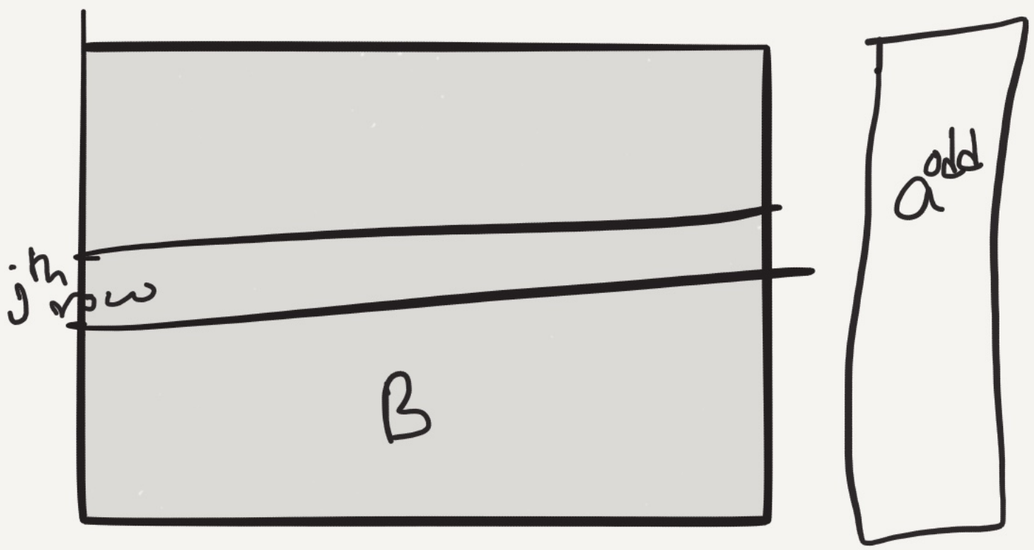
j^{th} row of $B = (j^{th}$ row of $A) \cdot \omega^j$

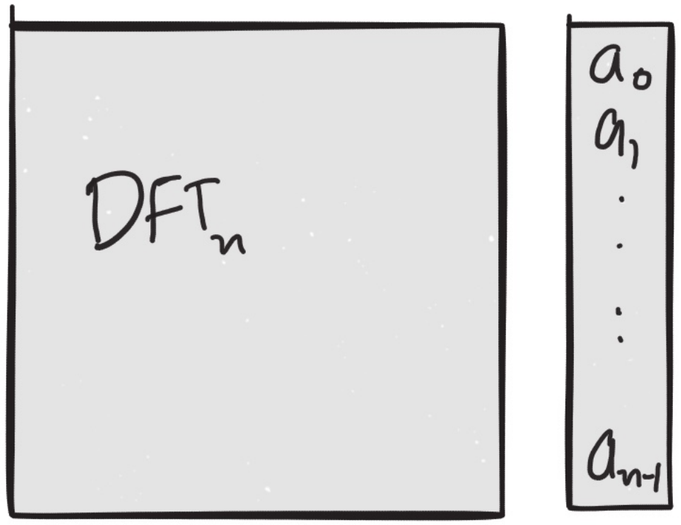


j^{th} entry of $\left(B \cdot a^{odd}\right)$

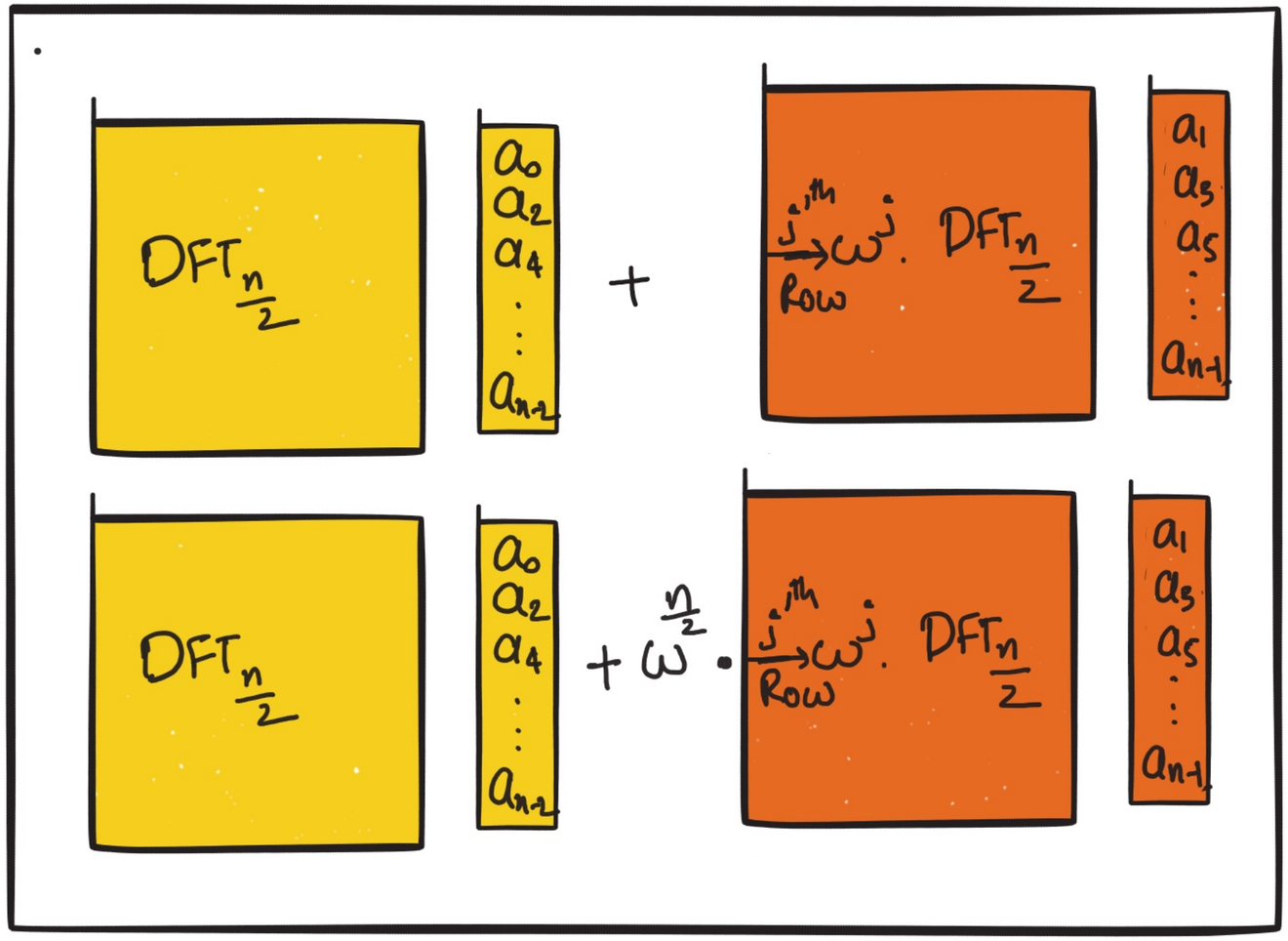
$= \left(j^{th} \text{ row of } B \cdot a^{odd}\right) = \omega^{j} \cdot \left(j^{th} \text{ row of } A \cdot a^{odd}\right)$

$= \omega^{j} \cdot \left(j^{th} \text{ entry of } \left(A \cdot a^{odd}\right)\right)$





=



→ Clean full algorithm ...

FFT Full algorithm

INPUT: $\bar{a} = (a_0, a_1, \ldots, a_{n-1})$; n is a power of 2.

(1) If $n=1$, return $\bar{a}$

(2) Compute $\bar{s} = DFT_{\frac{n}{2}}(a_0, a_2, \ldots, a_{n-2})$

(3) Compute $\bar{s}' = DFT_{\frac{n}{2}}(a_1, a_3, \ldots, a_{n-1})$

(4) For $j=0$ to $\frac{n}{2}-1$

$$y_j = \bar{s}_j + \omega^j \cdot \bar{s}'_j$$

$$y_{j+\frac{n}{2}} = \bar{s}_j + \omega^{\frac{n}{2}} \cdot \omega^j \cdot \bar{s}'_j$$

(5) RETURN y.

FFT Full algorithm

INPUT: $\bar{a} = (a_0, a_1, \ldots, a_{n-1})$; n is a power of 2.

(1) If $n=1$, return $\bar{a}$

(2) Compute $\bar{s} = DFT_{\frac{n}{2}}(a_0, a_2, \ldots, a_{n-2})$ $\equiv A \cdot a^{even}$

(3) Compute $\bar{s}' = DFT_{\frac{n}{2}}(a_1, a_3, \ldots, a_{n-1})$ $\equiv A \cdot a^{odd}$

(4) For $j = 0$ to $\frac{n}{2} - 1$

$$\gamma_j = \bar{s}_j + \omega^j \cdot \bar{s}'_j$$

$$\gamma_{j+\frac{n}{2}} = \bar{s}_j + \omega^{\frac{n}{2}} \cdot \omega^j \cdot \bar{s}'_j$$

} → Adding/multiplying complex numbers (not vectors).

(5) RETURN γ.

TIME COMPLEXITY ??

INPUT: $\bar{a} = (a_0, a_1, \ldots, a_{n-1})$; n is a power of 2.

① If $n=1$, return $\bar{a}$

② Compute $\bar{s} = DFT_{\frac{n}{2}}(a_0, a_2, \ldots, a_{n-2})$ → $\equiv T(n/2)$

③ Compute $\bar{s}' = DFT_{\frac{n}{2}}(a_1, a_3, \ldots, a_{n-1})$ → $\equiv T(n/2)$

④ For $j = 0$ to $\frac{n}{2} - 1$

$$r_j = \bar{s}_j + \omega^j \cdot \bar{s}'_j$$

$$r_{j+\frac{n}{2}} = \bar{s}_j + \omega^{\frac{n}{2}} \cdot \omega^j \cdot \bar{s}'_j$$

→ Adding/multiplying complex numbers (not vectors).

Each iteration takes $O(1)$ time.

⑤ RETURN r.

Let $T(n)$ = time complexity on instances of size n.

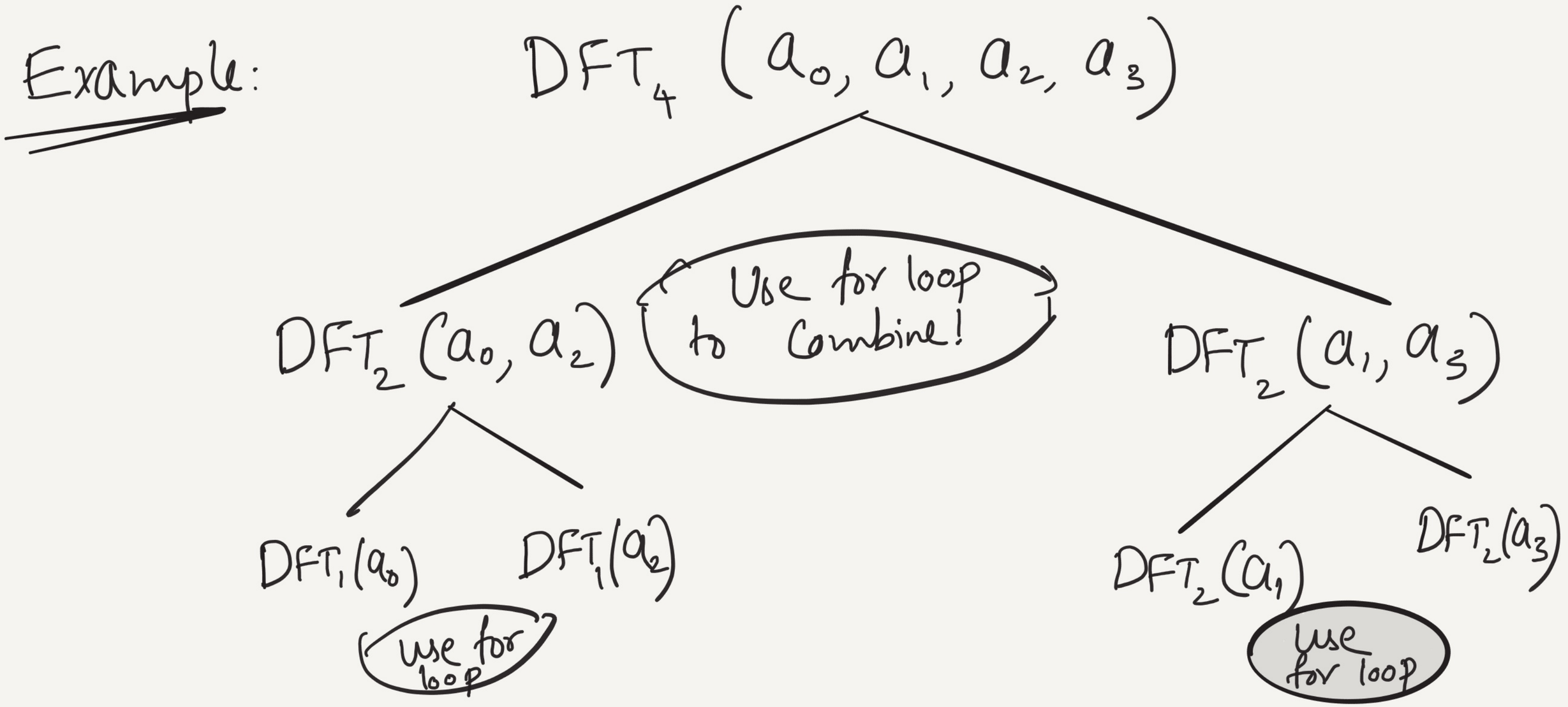
Time Complexity of FFT

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

$$\Rightarrow \boxed{T(n) = \bigcirc(n \cdot \log n).}$$

(by Master theorem, as we saw for Mergesort).

Example:


$DFT_4(a_0, a_1, a_2, a_3)$
$DFT_2(a_0, a_2)$
Use for loop to Combine!
$DFT_2(a_1, a_3)$
$DFT_1(a_0)$
$DFT_1(a_2)$
$DFT_2(a_1)$
$DFT_2(a_3)$
use for loop
use for loop

Summary of Divide & Conquer

→ Divide the problem into Sub problems

→ Solve the subproblems recursively

→ Combine/Merge the solutions.

GRAPHS

→ Basic definitions of graphs

→ Two important sub-routines ("BFS/DFS")
. .

Graphs: "Relations between Objects"

Examples: → Roads & Buildings

Graph:

OBJECTS	RELATIONSHIPS
Buildings	Roads
People	Friendships ("Facebook" graph).
Points in space	$(x, f(x))$
Events	Relations between events
Computers / Terminals	Network
People	Family relations
Genes	Dependency between genes

Formal definition of Graphs:

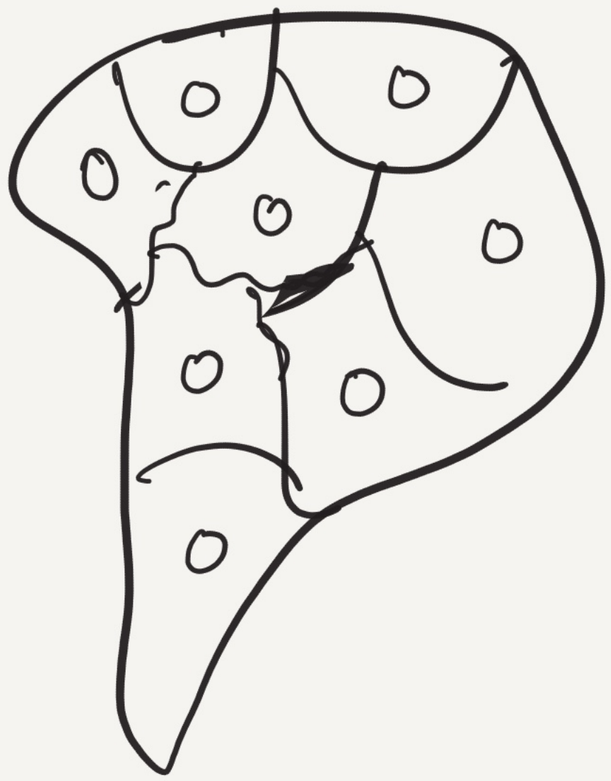
→ "Vertices": $V = \{v_1, v_2, \ldots, v_n\}$

→ Edges: $E = \{$ pairs of vertices $\}$

(e.g.: $= \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_3, v_5\}, \ldots\}$)

$G = (V, E)$

"What is relevant"



→ Color the map so that Counties with Common boundaries get different colors.

Coloring a political map.

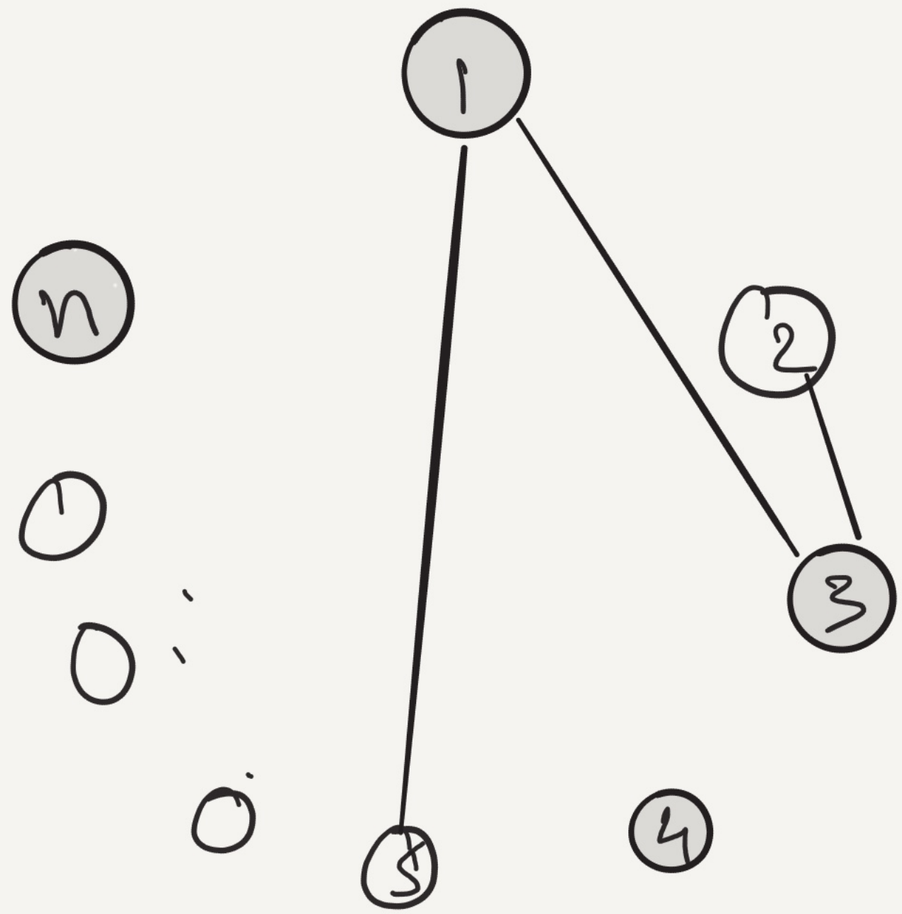
→ Make a vertex/node/object for each country.

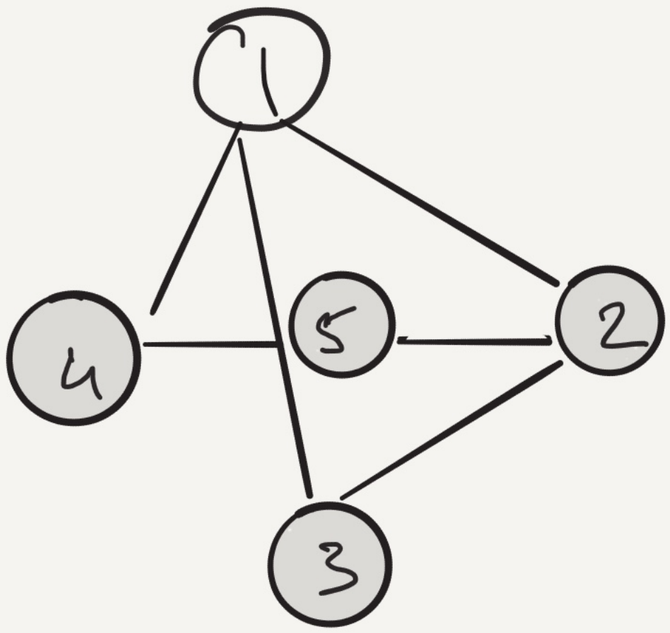
Edges: { all pairs of countries that share a boundary }.

Visualise relationships

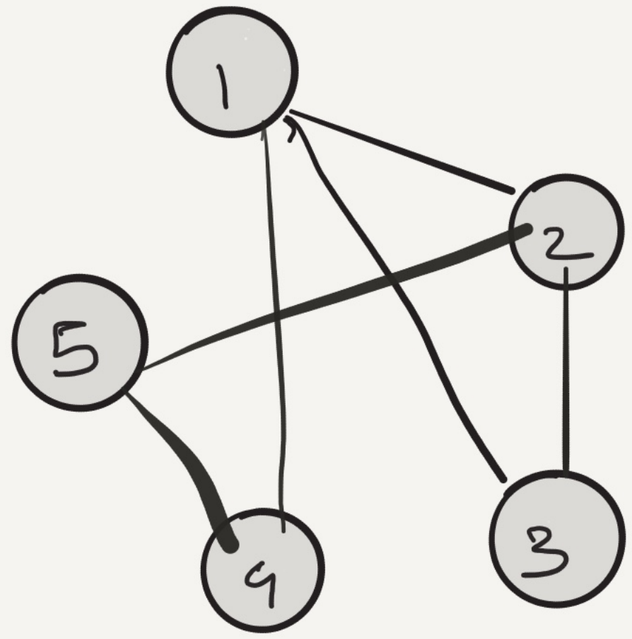
$$G = (V, E)$$

$$V = \{1, 2, \ldots, n\}; \quad E = \{\{1,3\}, \{1,5\}, \{2,3\}, \ldots\}$$





≡



→ Drawing is "unimportant".

Algorithmic Questions about Graphs:

→ What is the fastest way to get from one vertex to the other.

→ Are two graphs the "same" graph.

→ Bipartition of graphs

→ Finding a tree of least weight ...

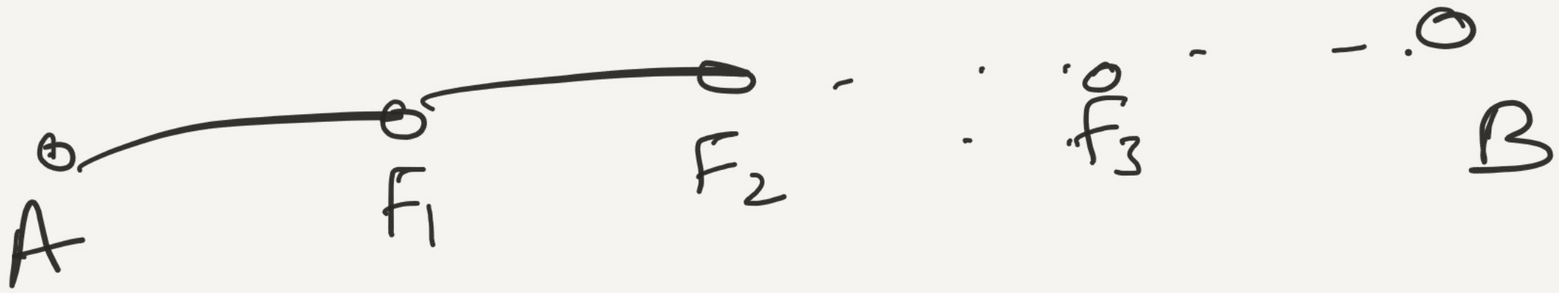
→ Traveling salesperson problem

→ "Reachability".

"Reachability" / "Connectivity"

Example: Facebook graph

two people.



Path

Given $G = (V, E)$

Path is a sequence of vertices $v_1, v_2, v_3, \ldots, v_k$ such that $\{v_l, v_{l+1}\} \in E$ for $l = 1, 2, \ldots, k-1$.