ALGORITHMS

Algorithm

- Dictionary definition
  - Procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation
  - A step-by-step method for accomplishing a task
- Informal description
  - An ordered sequence of instructions that is guaranteed to solve a specific problem

- **An algorithm is a list that looks like:**
  - **STEP 1: Do something.**
  - **STEP 2: Do something.**
  - **STEP 3: Do something.**
  - **. .**
  - **. .**
  - **. .**
  - **STEP N: Stop. You are finished.**

- Categories of operations used to construct algorithms:
  - Sequential operations
    - Carry out a single well-defined task; when that task is finished, the algorithm moves on to the next operation
  - Conditional operations
    - Ask a question and then select the next operation to be executed on the basis of the answer to that question
  - Iterative operations
    - Tell us to go back and repeat the execution of a previous block of instructions

- Algorithm
  - A well-ordered collection of unambiguous and effectively computable operations that, when executed, produces a result and halts in a finite amount of time
- Unambiguous operation
  - An operation that can be understood and carried out directly by the computing agent without needing to be further simplified or explained
- Computing agent
  - The machine, robot or person automatically carrying out the steps of the algorithm
  - Does not need to understand the concepts or ideas underlying the solution

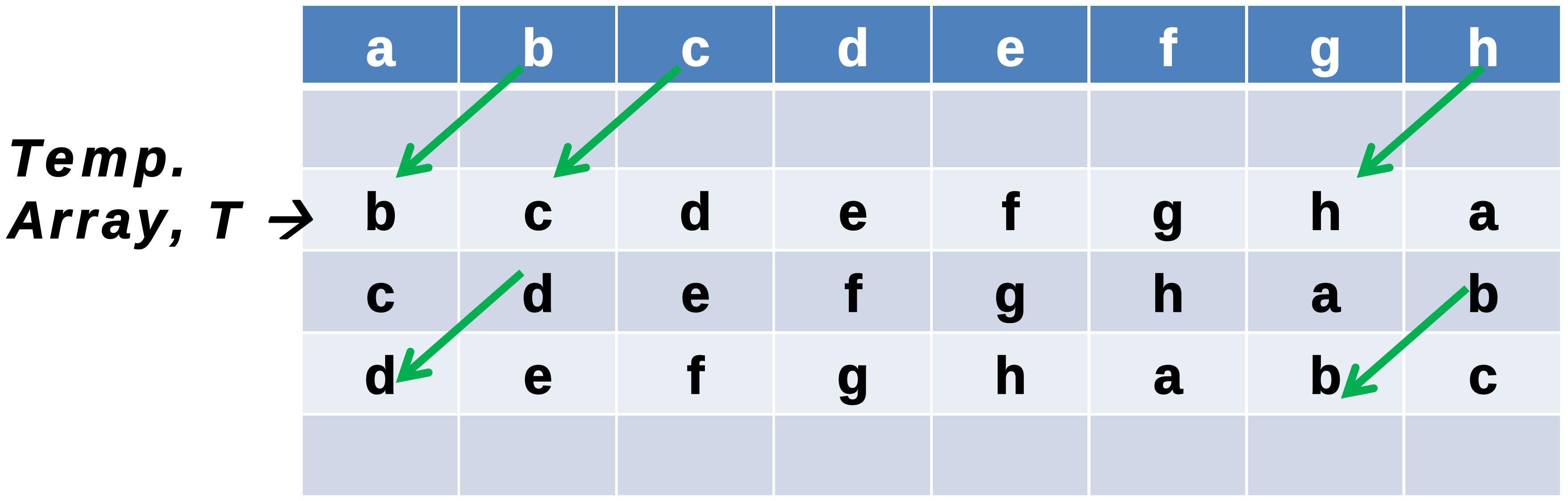
Example problem with STRINGS

Problem:

Rotate a 1-D vector of n elements left by j positions.

e.g. abcdefgh n = 8;

If j = 3, output: defghabc



	a	b	c	d	e	f	g	h
Temp. Array, T →	b	c	d	e	f	g	h	a
	c	d	e	f	g	h	a	b
	d	e	f	g	h	a	b	c

No. of operations : O(j*n);

Space required : n element intermediate vector

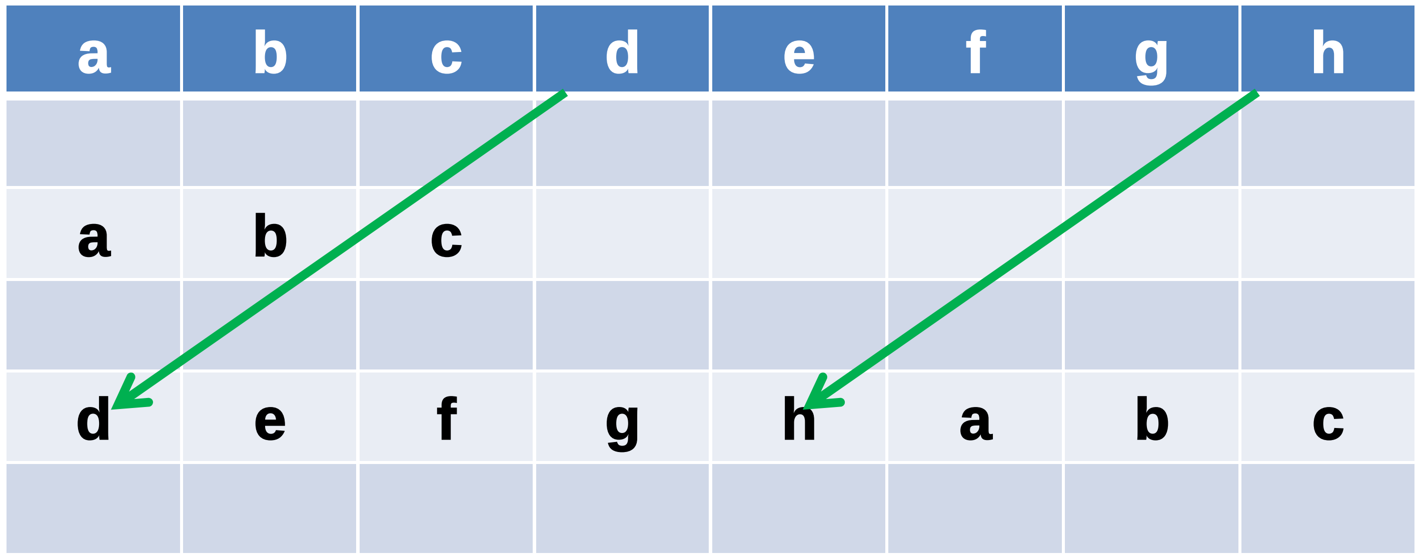
Problem:

Rotate a 1-D vector of n elements left by j positions.

e.g. abcdefgh n = 8;

If j = 3, output: defghabc

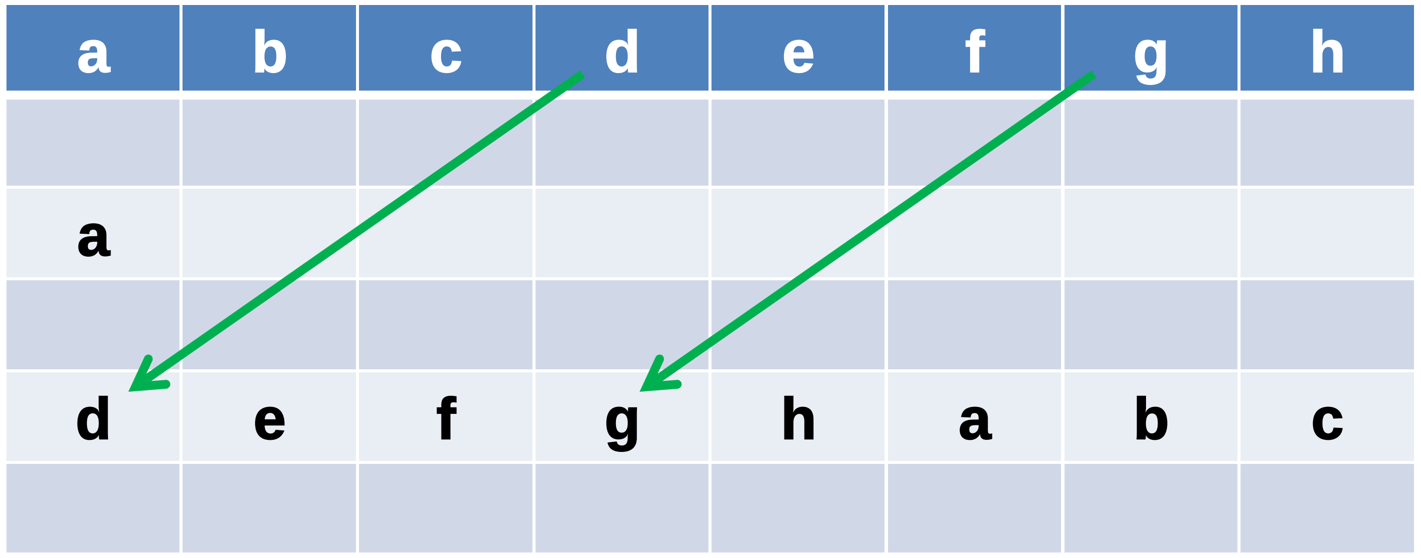
Solution (?) for -
No. of operations : O(n);
Space required : n element intermediate vector



	a	b	c	d	e	f	g	h
Temp. Array, T →	a	b	c					
	d	e	f	g	h	a	b	c

Solution (?) for -
No. of operations : O(kn);
Space required : m (<n) element intermediate vector

Temp. Array, T →



a	b	c	d	e	f	g	h
a							
d	e	f	g	h	a	b	c

Watch Steps:

- **Move x[0] to t;**
- **Move x[j] to x[0]; x[2j] to x[j];……… / * all indices are {x mod n} * /**

  The sequence of movement is*:*

  d, g, b, e, h, c, f, a

  So finally, U come back to x[0] → (a);

- **for x[0], copy from T {single element space}**
- **STOP when x[0] or T is touched**

Take, n = 8; j = 3. Solve it now, using previous algo.

a	b	c	d	e	f	g	h	i	j	k	l
a											
d			g			j					

The sequence of movement is:

d, g, j, a ---- OOPS !!!!

Process HALTED – WHY ??

Soln. ??

Re-Start from next element x[1],…… till over.

- a complex code results, compared to the earlier version (but U got O(n) time and space, j = 1).

Can U still be more elegant with idea / algo. and code, and get same run / space complexity ??

Problem:

Rotate a 1-D vector of n elements left by j positions.

e.g. **abcdefgh** n = 8;

If j = 3, output: **defghabc**

Let P = abc ; Q = degfh.

Thus with Input → PQ; Output → QP.

OK? How do U extrapolate this idea for a good and neat implementation

Look at this transformation:

(P' Q')' → QP;

We need a function which can reverse all the elements in an array A';

Use the same to reverse the elements in a specified portion of an array (Ac)'.

e.g. abcdefgh n = 8;

If j = 3, output: defghabc

Look at this transformation:

(P’ Q’)’ → QP;

Algo:

Reverse (0, j-1): cbadefgh;

Reverse (j, n-1): cbahgfed;

Reverse (o, n-1): defghabc;

Reversing a string sequence is the most easiest program/function, you need to write.